The Sigmoidal Growth of Operating System Security Vulnerabilities: An Empirical Revisit

Jukka Ruohonen[a,*], Sami Hyrynsalmi[a], Ville Leppänen[a]

[a] *Department of Information Technology, University of Turku, FI-20014 Turun yliopisto, Finland*

Abstract

Purpose. Motivated by the calls for more replications, this paper evaluates a theoretical model for the sigmoidal growth of operating system security vulnerabilities by replicating and extending the existing empirical evidence. *Approach.* The paper investigates the growth of software security vulnerabilities by fitting the linear, logistic, and Gompertz growth models with nonlinear least squares to time series data that covers a number of operating system products from Red Hat and Microsoft. *Results.* Although the fitted models are not free of statistical problems, the empirical results show that a sigmoidal growth function can be used for descriptive purposes. The paper further shows that a sigmoidal trend applies also to the number of software faults that were fixed in the Red Hat products. *Conclusion.* The paper supports the contested theoretical growth model. The few discussed theoretical problems can be used to develop the model further.

Keywords: software vulnerability, growth curve, Gompertz, replication, operating system, technology diffusion

1. Introduction

This paper evaluates and extends the existing empirical results concerning the presumed sigmoidal growth of software security vulnerabilities in operating system software products. The original theoretical model and the initial empirical results were presented by Alhazmi and associates (Alhazmi et al., 2005, 2007; Alhazmi and Malaiya, 2008). The model is an important contribution to the ongoing efforts to understand the discovery of software vulnerabilities by means of time series analysis. This provides the underlying rationale for the replication: a solid empirical foundation is a prerequisite for further theoretical advances and practical applications.

The empirical target in the model relates to the long-run software life cycles that are typical to operating systems. If the cumulation of vulnerabilities tends to follow a systematic sigmoidal pattern across these cycles, this information alone can be valuable when different software and release engineering decisions are made regarding further cycles. The presumed theoretical explanation is not explicitly related to software engineering, however.

The basic theoretical argument is that sigmoidal vulnerability growth trends tend to follow the popularity of operating systems: once the popularity reaches an inflection point, decreasing rate of adoption makes the given operating system less lucrative for exploitation development, which leads to fewer and fewer vulnerabilities (Alhazmi et al., 2005). Eventually a saturation point is reached; the operating system product has been substituted by a new product, and a new sigmoidal growth function takes over.

This theoretical background places the model into an interesting intersection of empirical software engineering research (Massacci and Nguyen, 2014). At the methodological front the model largely falls to the domain of software reliability modeling, but the theoretical argument is distinctively different. Accordingly, the amount of discovered vulnerabilities does not slow down because security issues would be harder to find as time passes, but rather because there is a decreasing interest to find vulnerabilities from old software products. Consequently, security bugs are different from normal bugs, and, by implication, vulnerability modeling is different from reliability modeling. Besides these software engineering aspects, the model contains also theoretical presumptions that can be related particularly to the different sigmoidal models for the progress of technology. This wider theoretical scope raises the relevance of the replication: provided that the foundation holds, further opportunities exist for interdisciplinary research.

The paper proceeds by first discussing the larger theoretical background behind sigmoidal growth models, connecting the vulnerability model to technology diffusion and software life cycle models. Three hypotheses are also postulated for contesting the model empirically: (a) vulnerability discovery follows a sigmoidal trend; (b) the same applies to fixed software faults; and (c) major and minor operating system releases do not systematically differ with respect to the presumed growth trends.

These three hypotheses are tested in the empirical part of the paper. The primary empirical dataset contains time

*Corresponding author.
Email addresses: juanruo@utu.fi (Jukka Ruohonen), sthyry@utu.fi (Sami Hyrynsalmi), ville.leppanen@utu.fi (Ville Leppänen)

series from 29 deprecated Red Hat operating system products. In order to assess the generalizability of the results, the presence of a sigmoidal trend is further evaluated with a secondary dataset that contains 40 operating system products from Microsoft. The time period covered spans from circa 2000 to 2014. Estimation is carried out by testing the linear, logistic, and Gompertz growth functions by nonlinear least squares. Model comparisons are carried out with an information criterion measure and different statistical tests. Given the general need for sensitivity (Höök et al., 2011; Suominen and Seppänen, 2014) and model assessments (López et al., 2004; Meade and Islam, 2006; Wang and Bushman, 2006) in growth curve modeling, the estimated models are exposed to a few conventional statistical tests over the basic time series characteristics. In general, however, the paper adopts a viewpoint that growth curves are essentially stylized and descriptive characterizations of the underlying time series trends.

Finally, a few brief remarks should be made to clarify the replication approach itself. Like any replication, the paper ultimately tests an existing theory and verifies existing empirical results. In contrast to the term reproduction, the term replication is, consequently, defined in this paper to mean that "*independent researchers are able to reach the same qualitative conclusions by repeating the original study using the same methods on different data* (Boylan et al., 2015, p. 81)". This fundamental goal can be narrowed with the different replication functions that were recently described by Gómez et al. (2014). Thus, the replication seeks: (1) to validate hypotheses; (2) to control potential sampling errors in the original empirical experiment; (3) to understand potential population limits regarding generalizability; and (4) to affirm that researcher bias (Shepperd et al., 2014) is not present. The four replication functions are equally important.

2. Theoretical Background

The contested theoretical model is simple. In essence, the model states that (a) the logistic growth curve can be used to empirically describe the cumulative amount of security vulnerabilities that have affected a software product, and that (b) the observed sigmoidal growth pattern can be theoretically related to the popularity of the examined product (Alhazmi et al., 2005). These simple but fundamental presumptions are easy to relate to the general growth curve modeling literature and to a few specific topics in software engineering.

2.1. Sigmoidal Growth

If growth is taken to follow a S-shaped curve that is determined by a given sigmoid function, there are only a limited number of theoretically meaningful parameters that can be derived from the curve. After a certain *lag phase* λ, the growth rate eventually leads to the upper *asymptote* α. The used functional forms imply also a certain point at which the growth attains its *maximum slope* μ. In essence, besides the actual statistical fit, these three parameters are the primary interest in many empirical growth curve modeling applications. The delivered theoretical message is consequently simple: although growth starts slow (λ), it accelerates rapidly until the maximum growth rate is reached (μ), after which it slows down but still approaches a saturation point asymptotically (α). The growth is thus bounded and the maximum growth rate separates two growth phases.

The interpretation given for the two phases varies from an application domain to another. It is still possible to reach the fundamental theoretical trait by considering the first phase as a replication process during which growth is proportional to an existing population, while maintaining that the second phase signifies an inhibiting process during which the population reaches its stable carrying capacity (Cunningham and Kwakkel, 2014). These dual premises were common in the early models for the growth of human populations. In other words, at any instant of time, the population growth rate was taken to be proportional not only to the already attained growth – the already existing population, but also to the unutilized potential in a given limited area to support the existing population (Pearl and Reed, 1920). The latter premise can be located also from many models for the progress of technology.

In particular, it has been presumed that advancements occur through competitive substitution of different technologies; coal for wood, mobile telephones for landlines, IPv6 for IPv4, and so forth. Once a substitution process has initially taken off, it is assumed to always reach the saturation point, but the rate of substitution of new for old is proportional to the remaining old technology to be substituted (Fisher and Pry, 1971). When a few more steps are taken along this theoretical path, more general technology life cycle models quickly emerge in the horizon. In these models a given S-shaped curve is theoretically divided into different stages such as emergence, growth, maturity, and eventually the saturation stage at which another substitution process may again begin (Gao et al., 2013). A comparable fivefold theoretical construct would be the hypothetical progress of a technology through basic research, applied research, development, application, and eventually the wider social impact (Suominen and Tuominen, 2010). These general models for the progress and adoption of technology fall under the label of technology diffusion models, which can be seen to differ from the noted substitution models in that the latter require an assumption of existing markets; there must be something old to be substituted for something new (Norton and Bass, 1987). These theoretical models for technology are easy to relate to the realm of software (technology) in general, and software security and vulnerability trends in particular.

2.2. Sigmoidal Models for Software

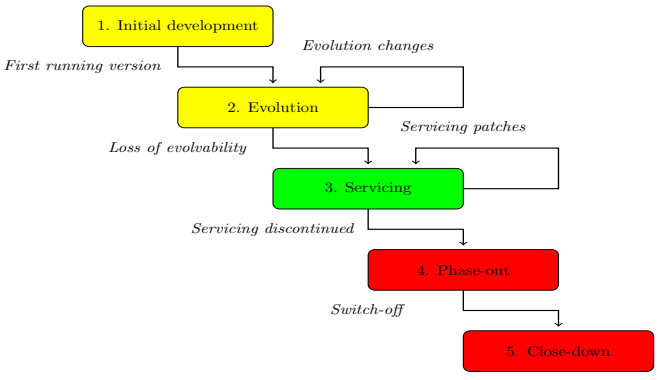
The S-shaped curves and the associated staged models have been also used to describe different characteristics of

software. Two typical application domains can be mentioned from the field of software engineering.

The first has examined the long-run growth trends of software particularly in terms software size; it tends to grow continuously during the active life cycle. While S-shaped growth functions have been proposed (Lehman and Fernández-Ramil, 2006), also numerous other functions have appeared in this extensive branch of empirical research. The theoretical origins in the branch were related to the potential limits for the observed growth (Lehman and Belady, 1985), which is easy to relate to the more general growth modeling literature in that only bounded growth functions are often argued to be plausible and valid (Höök et al., 2011; Meade and Islam, 2006). The other example comes from reliability modeling within which the central question is how reliability of software increases over time. As recently surveyed by Wang et al. (2014), numerous different S-shaped curves have been used in this equivalently extensive branch of empirical research. If no new faults are assumed to arrive when detected faults are removed, the so-called perfect debugging models are often based on the familiar assumptions: the number of corrected faults is proportional to the existing faults, while the upper asymptote implies a satisfactory level of quality improvements. A software product thus exhibits a decreasing rate of faults (either in the short-run or in the long-run), although the size of the product tends to grow in the long-run.

These two software engineering branches differ in one fundamental respect: the former approach observes the long-run trends after the initial software development has already been completed, whereas the reliability models typically (but not necessarily) observe the rate of faults during the (short-run) development phase. Typical applications thus operate at different stages through which software products can be theorized to live through. The model of Rajlich and Bennett (2000) is a good theoretical example of the potential stages. This software life cycle model is shown in Fig. 1.

Figure 1: A Staged Life Cycle Model (Rajlich and Bennett, 2000)



It should be emphasized that the model is not a reinvention of the so-called waterfall model for software development; rather, the intention is to illustrate the entire life cycle within which the completion of initial technical deliveries is only a relatively short time period. The model also assumes that practical reliability assessments are done during the evolution stage during which also concepts such as time-to-market and time-to-delivery are relevant (Rajlich and Bennett, 2000). The timing of the first release available to customers can be assumed to occur during the start of the servicing period, although different alpha and beta releases may be delivered to customers during the evolution stage or – keeping the current agile software development practices in mind – already during the initial development stage.

The staged life cycle model implies a decision over whether a given S-shaped growth pattern is taken to apply across the stages or within the stages. If the whole life cycle is observed, the asymptotes for the observed quantity would be located at the initial development and close-down stages. On the other hand, different growth curves may apply at different stages. Given the latter assumption, the comparable staged models for technology diffusion, for instance, consider different sources of empirical data for each stage (Gao et al., 2013; Suominen and Seppänen, 2014; Suominen and Tuominen, 2010). The same applies to software reliability modeling. Although software fault tracking is likely to be applied during the whole life cycle, other data sources imply a more explicit stage.

For instance, fault data from customer support services (Jalote et al., 2008) imply that the given product is already in widespread use. This would point to the servicing stage in Fig. 1. Since active debugging activities have presumably been completed before this stage, the specific context would be post-release reliability modeling. If a S-shaped curve is assumed, the term product stabilization time (Jalote et al., 2008) could be taken to refer to the lag phase λ, that is, once μ is reached, the growth rate of faults would start to decline.

Another possibility is to consider some market-based mechanism such that the observed S-shaped quantity would be dependent on the number of customers and their preferences. It could be, for instance, that μ marks a point at which customer preferences change from mere functionality to such factors as usability and reliability (Miranda and Lima, 2013). This would slow down the purchases of new units, but increase the demand for updates and enhancements. When the software life cycle model is considered, the inflection point could be taken to refer even more generally to a hypothetical point at which customers of a long-lived software product release are increasingly switching to a new release. Although it is possible to consider both adoption and replacement processes (Meade and Islam, 2006), the context is general enough to allow an assumption that these new products may include both new releases from the same company as well as the discussed technology substitution mechanism. These market-based assumptions apply also to the considered S-shaped growth

model for security vulnerabilities.

2.3. The Growth of Security Vulnerabilities

The model of Alhazmi et al. (2005, 2007) assumes three stages, given a logistic growth function. During the lag phase λ, the given operating system is gaining momentum in the market, which affects the preferences of the so-called crackers who are studying the system for exploitation. The second phase ends to the maximum growth rate μ, which denotes the peak of popularity, while the third and final stage refers to a substitution process between different operating system releases (Alhazmi et al., 2005). The general assumption is that the market share is among the most important factors affecting the long-run growth of security vulnerabilities. Consequently, the model has been further extended by explicitly including the number of installations as an explanatory variable (Woo et al., 2011). A few restrictive theoretical points are worth remarking about this simple S-shaped growth model.

First, the theoretical background could be postulated also in terms of reliability, provided that security vulnerability patches are considered to be a subclass of reliability patches. By assumption, the overall reliability and, hence, security are both bounded and increasing functions of time. Second, the model operates in the post-release context, using a so-called postmortem variable (Hein and Saiedian, 2009) to evaluate the trends during the servicing stage in Fig. 1. That is to say, no assumptions are possible to make regarding the earlier stages, even though the observed post-release vulnerability trends are presumably affected by the pre-release processes that are not observed.

Third, the model is specifically aimed to describe the long-run trends in the amount of operating system vulnerabilities. The operating system context implies that the applicable software products have a long life cycle and a mature code base. In fact, the truly long-run security trends could be examined also in terms of the technology life cycle stages. That is, the operating system security landscape includes also extensive research and development efforts that span several decades. The same applies to the number of different historical substitution processes that have occurred in the markets for operating systems.

However, fourth, the post-release context assumes a univariate per-release time series context, which implies a strictly fixed life cycle for each release. This also implies that the model does not consider the evolution of code that is shared between successive releases (Alhazmi and Malaiya, 2008). It is worth noting that the literature contains also some attempts to examine longer trends by combining the individual vulnerability product time series (Kim et al., 2007), although practical challenges have been noted to exist (Woo et al., 2011). It is worth noting, moreover, that analogous generalizations have been made with technology diffusion models (Meade and Islam, 2006). When interpreted against the staged life cycle model, such combined approaches imply that the growth would be observed across different stages of multiple subsequent (or parallel) products.

Given the market-based assumptions and the substitution processes, several interesting theoretical questions arise. From a customer perspective, for instance, an existing installation should never reach the phase-out stage because this is the stage during which all updates have already been halted – even though the given vendor may still try to generate revenue from the product (Rajlich and Bennett, 2000). An installed operating system release in the phase-out stage is obviously a security risk. The prime example of a large-scale transition process between products would likely be the ongoing phase-out of Microsoft Windows XP.[1] However, besides the early Windows family of operating systems, the model was originally tested also with two early (Red Hat) Linux distributions (Alhazmi et al., 2007). The fifth consequent remark is that the openness of the source code should not be a factor that significantly shapes the long-run operating system vulnerability trends.

Last but not least, the original model is strictly a vulnerability growth model. While the original model was framed also by considering scaling in terms of software size (Alhazmi et al., 2007), the actual growth was still measured in terms of the cumulative amount of known vulnerabilities. This separates the model from the extensive branch of empirical research that have examined the vulnerability discovery process in terms of the time lags between discovery and availability of patches (Arora et al., 2010; Jones, 2007; Wright et al., 2013). All in all, under these six restrictive assumptions, the following three hypotheses (H) can be formulated for contesting the model empirically.

- H_1 Given the discussed theoretical restrictions, a commonly used, well-defined, sigmoidal growth function describes the cumulative amount of known security vulnerabilities in a statistically satisfactory manner.
- H_2 Provided that H_1 holds, the same sigmoidal function sufficiently describes also the cumulative amount of fixed software faults.
- H_3 Provided that H_1 holds, the statistically satisfactory results apply to major operating system releases as well as to minor releases that are assumed to extend the life cycle of the major releases.

Regardless whether the first Hypothesis H_1 holds, nothing is implied about the plausible possibility that some other non-sigmoidal function provides a good fit – or a better fit. The latter corollary presumption, H_2, is formulated to address the overall construct validity behind the model. If H_2

[1] The case would likely require also alterations to Fig. 1, given the small legion of different product variants and their different extended support periods. In particular, the phase-out stage would perhaps refer to the varying transition periods, while the support would end in the close-down stage. These notes apply also to the products that are examined in this paper.

holds, it is arguably unclear whether the model is unique to software vulnerabilities. In this case the context would perhaps point to more general reliability modeling either theoretically or due to the nature and operationalization of empirical data. The final Hypothesis H_3 is based on the assumption that separate sigmoidal growth curves can be used at the different life cycle stages. In the forthcoming empirical experiment a major operating system release lives through all of the stages in Fig. 1. Once the support ends in the phase-out stage, it is assumed that a minor release can be made to extend the life cycle for a relatively short period until the final close-down stage is reached.

3. Empirical Approach

A fundamental distinction in growth modeling can be set between empirical and mechanical modeling (López et al., 2004). Although both modeling approaches can be either descriptive or predictive (Höök et al., 2011), the distinction rather underlines the difference between statistical (empirical) and mathematical (mechanical) models; between stochastic and deterministic models – in essence, between statistical time series models and differential equations. The empirical approach follows the latter course, but not without statistical assessments over the plausibility of the mechanical models in a context that is not mechanical (for the other course see Roumani et al. 2015). The approach is subsequently elaborated with an introduction of the empirical data, a brief discussion of two growth functions, and an outline of computational details.

3.1. Data

Two datasets were collected for the empirical evaluation. The primary dataset contains most operating system products from the Red Hat incorporation that have reached the end-of-life (EOL) stage by November, 2014. This EOL stage is equivalent to the phase-out stage (in Fig. 1) in the sense that both refer to a stage at which all updates have already been halted. Based on the records kept by the company (Red Hat, Inc., 2014), the dataset contains the daily aggregated sum of security advisories that were released for the included products before the EOL stage; the last included security advisory is typically the EOL announcement itself. In order to evaluate Hypothesis H_2, the primary dataset was also augmented with all updates that were classified as bug fixes for the corresponding products.

The secondary dataset contains security advisories that have been published for a sample of Microsoft Windows operating systems (Microsoft, Inc., 2015). Unlike with the Red Hat products, many of the included Windows operating systems have not yet reached the EOL stage, meaning that the products are still eligible for security updates from Microsoft. Due to data retrieval limitations, moreover, the secondary dataset does not contain bug fixes or other updates for the products.

The included Red Hat and Microsoft products are enumerated in Appendix (Tables A.1 and A.2, respectively). Only those products were included that had a sufficient amount of observations. The excluded operating system products mostly include special purpose product variants. Each product is measured by a time series for which the calendar time endpoints are defined in terms of the first and the last issued security advisory or bug fix. All daily arrivals were further cumulated to daily sums. The resulting time series are visualized in Fig. 2.

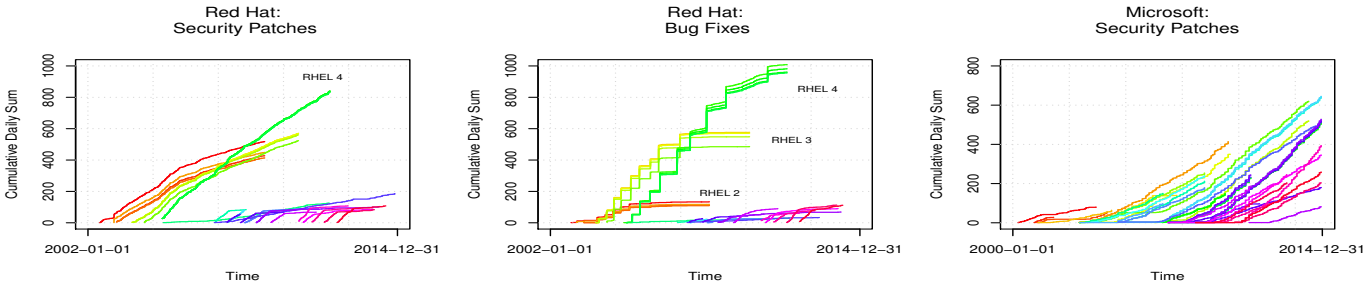
A few important remarks should be made about the empirical data. First and foremost, the known and disclosed historical security vulnerabilities that have affected the products are measured indirectly by observing the timing of the released security advisories. It is also assumed that all released advisories have corresponded with the availability of security patches, although, in theory, security advisories may be issued before updates are available, or the other way around. The empirical approach, consequently, differs from the original evaluation that used the National Vulnerabilities Database (NVD) and the Common Vulnerabilities and Exposures (CVEs) database (Alhazmi and Malaiya, 2008). This implies that the timings are not exactly comparable in terms of different vulnerability life cycle models (cf. Wright et al., 2013, Fig. 1). When viewed against the software life cycle model, in contrast, the empirical effect could be assumed to be negligible because the observed time lines may cover even more than a decade. Therefore, it is more important to stress that the original evaluation observed the cumulation of individual vulnerabilities, while this paper observes the cumulation of security patches.[2] As a rule rather than an exception, the security patches from both Red Hat and Microsoft have covered multiple vulnerabilities. Although these counting issues are known to affect estimates (Massacci and Nguyen, 2014), the use of security advisories does not invalidate the theoretical premises as such; regardless whether the known vulnerabilities are observed individually or in groups, the resulting trends should follow a sigmoidal growth trend.

Second, as both companies support multiple parallel products, many of the patched security vulnerabilities have affected multiple products. This implies that the observed growth trends are highly similar particularly in terms of the common product lines such as Red Hat Enterprise Linux (RHEL). The trends are identical, however.

Third, all growth curve modeling applications should explicitly articulate the assumptions that surround the lower and upper asymptotes. Since the first released security patches (or bug fixes) mark the first observations, the lower asymptotes do not correspond with the product launches. This is an important theoretical restriction that is implicitly present also in the original model; the initial time period without known vulnerabilities is excluded. Since all included Red Hat products have reached the EOL stage, the theoretical upper asymptote is explicitly known

[2] Another point worth emphasizing is that primary security advisory data from software vendors is presumably more robust than data from the noted secondary, third-party, vulnerability databases.

Figure 2: The Red Hat and Microsoft Datasets



for these products. This does not apply in the secondary dataset of Microsoft products, however. While the growth remains bounded as also these products will be eventually deprecated, it should be still emphasized that only partial software life cycles are empirically observed with many of the Windows operating systems.

Last, all comparisons should be done with care. Since the number of observations (days) is different for the majority of observed products, no direct comparisons can be made between products or product lines. The same applies to direct comparisons across the two datasets, or across security patches and bug fixes for the Red Hat products.

3.2. Growth Curves

The literature contains an abundant amount of sigmoid functions for S-shaped growth patterns (Höök et al., 2011; López et al., 2004; Massacci and Nguyen, 2014; Meade and Islam, 2006; Wang et al., 2014; Zwietering et al., 1990). A classical example is the famous function that Gompertz (1825) formulated to determine the rate of mortality. Since the inception almost two centuries ago, the function has been used to model growth in numerous different fields, including software reliability (Ohishi et al., 2005; Yamada et al., 1983). In growth modeling applications the Gompertz function is typically written as

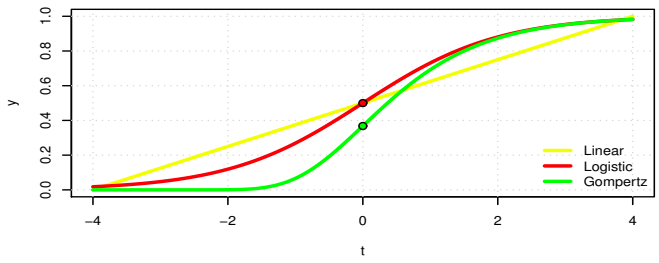
$$y = \alpha \exp[-\exp(\beta - \zeta\, t)], \tag{1}$$

where α, β, and ζ are constants, while t denotes the used unit of time (Annadurai et al., 2000; Winsor, 1932). Since growth is modeled, α and ζ are positive. It is a straightforward mathematical exercise to derive the three parameters of theoretical interest from (1). The maximum slope μ can be obtained by calculating the first derivative at the inflection point at which the second derivative is zero; the lag phase λ is given by the t-axis intercept of the tangent through the inflection point; and finally, the higher asymptote α is reached when t approaches (positive) infinity (Zwietering et al., 1990). Like most sigmoid functions, the Gompertz curve is rather similar to the conventional logistic growth function:

$$y = \frac{\alpha}{1 + \exp(\beta - \zeta\, t)}, \tag{2}$$

as written by Annadurai et al. (2000) and Winsor (1932). In both functions the growth rate parameter ζ controls how rapidly the growth occurs from the lower asymptotes to the upper asymptotes. These are zero and α in both functions, respectively. The logistic curve is symmetric about its central point of inflection, $y = \alpha\ /\ 2$. The first derivative takes the familiar bell-shaped form. Given the representation in (2), the t-axis is crossed at $\beta\ /\ \zeta$ when the first half of the asymptotic growth has been reached (see Fig. 3). The Gompertz model, in contrast, is asymmetric about its point of inflection. The first derivative is skewed to the right. Since the inflection point is at $t = \beta\ /\ \zeta$, that is, $y = \alpha\ /\ \exp(1)$, it was well-recognized already in the early 1930s that the Gompertz function is likely to give a better fit on data that shows an inflection early on, when about 37 per cent of the total growth has been reached (Winsor, 1932). The maximum growth rate, μ, is $\alpha\,\zeta\ /\ 4$ and $\alpha\,\zeta\ /\ \exp(1)$ in the logistic and Gompertz functions, respectively. The inflection point is strictly fixed in both functions.

Figure 3: Three Growth Functions ($\alpha = \zeta = 1,\ \beta = 0$)



In many applications it is desirable to allow a more flexible inflection point. The midpoint peak in the logistic curve, in particular, has been a frequent target of criticism, regardless whether the modeling context is diffusion of technology (Miranda and Lima, 2013) or available resources in oil fields (Höök et al., 2011). The message from the 1930s was that a fourth constant was required for a variable degree of skewness (Winsor, 1932). The challenge was answered by Richards (1959) who developed a sigmoid function that allows an inflection anywhere between the lower and upper asymptotes. This flexibility has ensured the popularity of the function in biology and related

sciences ever since the inception, although not without criticism (Annadurai et al., 2000; Birch, 1999; Zwietering et al., 1990). The function has recently been used also to model the diffusion of technology (Marinakis, 2012). It is worth noting, moreover, that also Bass' (1969) classical model for the adoption of technology allows a flexible inflection point. While this model has likewise remained popular in its own application domain (Cunningham and Kwakkel, 2014; Meade and Islam, 2006), Richards' function is explicitly an empirical generalization of the logistic and related functions. In fact, it is sometimes known as the generalized logistic function.

The forthcoming empirical analysis is carried by using only (1) and (2), however. The primary reason for this choice is that the considered theoretical vulnerability growth model does not allow to postulate any specific inflection point. In other words, the points offered by the logistic and Gompertz functions are as good *hypotheses* as any other inflection point that would be available by manipulating the fourth parameter in the Richards' function. Even when some theory is available to guide the analysis, it is far from being a straightforward endeavor to use the Richards' function to discriminate between different growth patterns (Marinakis, 2012). Moreover, in some applications it is difficult or even impossible to model growth with the function without violating theoretical premises (Birch, 1999), or to interpret the fourth parameter theoretically (Annadurai et al., 2000; Zwietering et al., 1990). In other words, the flexibility comes at a price. Many of these points could be restated with respect to general curve fitting techniques such as polynomial regressions and smoothing splines. In general, as the examined hypotheses postulate sigmoidal growth, this must be also the tested growth pattern.

3.3. Computation

There are two main contemporary alternatives for typical curve fitting estimation: nonlinear least squares (NLS) and maximum likelihood (Meade and Islam, 2006). This excludes linear regression with transformed variables. As is well-known, these linearized approaches are constrained by the requirement of a known asymptote (Suominen and Seppänen, 2014; Young, 1993), which is readily estimated from the data by using nonlinear least squares.

The estimation is carried out by using NLS in which the residual sum of squares (RSS) is minimized, given the single predictor, the time t, and the parameter vector $(\alpha, \beta, \zeta)'$. In other words, the expected growth follows

$$y_t = \mathrm{E}(y_t \mid t) + \varepsilon_t = f(t, \alpha, \beta, \zeta) + \varepsilon_t, \tag{3}$$

where $f(\cdot)$ denotes either the Gompertz or the logistic function. Instead of the Levenberg-Marquardt algorithm that is frequently used for curve fitting (Höök et al., 2011; Zwietering et al., 1990), the utilized $nls(\cdot)$ function in R optimizes

$$\mathrm{RSS}(\alpha, \beta, \zeta) = \min_{\alpha,\beta,\zeta} \sum_{t=1}^{T} [y_t - f(t, \alpha, \beta, \zeta)]^2 \tag{4}$$

via the standard Gauss-Newton method. Following the ideas of Fox and Weisberg (2010), the starting values for the constant β and the rate parameter ζ were set to $-\widehat{\beta}_0$ and $\widehat{\zeta}_0$ that were obtained from a simple least squares regression of

$$g(y_t \ / \ \alpha_0) = \beta_0 + \zeta_0 \, t + \varepsilon_t, \tag{5}$$

where $g(\cdot)$ is either

$$\mathrm{logit}(z) = \ln [z \ / \ (1 - z)] \tag{6}$$

or

$$\mathrm{gompit}(z) = -\ln[-\ln(z)], \tag{7}$$

depending on whether (2) or (1) is estimated, respectively. The term "gompit" is credited to Berger (1981). The starting value for the asymptote, α_0, which is used also in (5), was set to one plus the maximum value of the cumulated time series. No convergence problems were observed with these starting values.[3]

A simple linear growth curve can be specified as the most general alternative for the two sigmoidal growth functions (Alhazmi and Malaiya, 2008). Also this curve was computed via $nls(\cdot)$, although estimating (5) with $lm(\cdot)$ would yield identical results when $g(\cdot)$ is defined to be an identity function and α_0 is restricted to unity. As the number of parameters is different, RSS alone is not adequate to compare the three models for a given product. The traditional F-test has been, therefore, a common option to compare the model fits (López et al., 2004; Zwietering et al., 1990). If RSS_2 denotes the linear fit with two parameters and RSS_3 is taken to come from the three-parameter sigmoidal growth curves, the test value is

$$F = \frac{\mathrm{RSS}_2 - \mathrm{RSS}_3}{\mathrm{RSS}_3 \ / \ \mathrm{DF}_3}, \tag{8}$$

where DF_3 is the degrees of freedom in the sigmoidal models, that is, $\mathrm{DF}_3 = T - k$, given the number of observations (days) and a constant $k = |\{\alpha, \beta, \zeta\}| = 3$. The null hypothesis is that the linear model is preferable; the sigmoidal functions do not provide a significantly better fit. The value in (8) can be taken to approximately follow the F-distribution (with one and DF_3 degrees of freedom) also

[3] Two alternatives were considered. First, the iterative algorithm of Kahm et al. (2010) obtains the starting values from smooth spline fits, and then uses these to fit analytical growth curves, using Akaike's information criterion to select the best fit. Second, R contains also a number of so-called self-starting values for common functions, including the logistic and the Gompertz function (Fox and Weisberg, 2010). Since no problems were detected with the discussed approach, both of these considerations were ruled out.

in the considered nonlinear models, although the word approximately should be underlined for small samples (Zwietering et al., 1990). The same applies to other distributive assumptions such as the approximate normality for the ratios of parameter estimates to their standard errors (Fox and Weisberg, 2010). In general, large-sample asymptotic assumptions are required.

Information criteria measures have provided another common strategy for model comparisons also in the curve fitting context. Akaike's information criterion (AIC), which is frequently used in the context (Alhazmi and Malaiya, 2008; López et al., 2004; Ohishi et al., 2005; Wang et al., 2014), is as a good alternative for coarse approximate comparisons as any other readily available criterion.[4] The AIC values and F-tests provide a decent comparative toolbox for the initial analysis. The comparable AIC comparison algorithm of Kahm et al. (2010) has been also used to approach much more complex biological datasets with thousands of curves to be fitted (Vaas et al., 2012). The small analytical toolbox is still needed also for the present empirical experiment that fits $(3 \times 29 \times 2) + (3 \times 40) = 294$ growth curves; three for each of the 29 products in the primary twofold Red Hat dataset, augmented with the evaluation of the three growth curves for the forty examined Microsoft operating system products.

4. Estimation

Estimation is carried out by focusing on the primary dataset of Red Hat products. The estimation proceeds in three steps: the nonlinear growth curve fits are first evaluated, after which statistical assessments and tentative forecasts follow. In order to evaluate the generalizability of the results, the same procedure is finally repeated for the Microsoft products in the secondary dataset.

4.1. Estimates

Parameter estimates, AICs, and results from F-tests are shown in Appendix (Table A.3) for the security patches released for the examined Red Hat products. The Gompertz function is the clear winner in terms of row-wise AIC comparisons: the linear curve and the logistic function both only win the race to the smallest AIC twice. The same applies to the F-tests against the linear model: the Gompertz models rejects ($p < 0.05$) the null hypothesis (linear model) a couple of times more than the logistic function. The same holds for the bug fixes: the Gompertz function has the smallest AIC for all but five products. In these two respects the samples are similar. In the bug fix sample, however, there tends to be a higher range between the row-wise AIC values. This is illustrated graphically in Fig. 4. As can be seen, the difference is specifically located between the linear model and the two sigmoidal functions.

[4] It is worth noting that the $AIC(\cdot)$ function in R applies directly also to all $nls(\cdot)$ objects, but the number of parameters used in the function are defined to be $k + 1$ rather than k.

Figure 4: AICs for Bug Fixes (Red Hat)

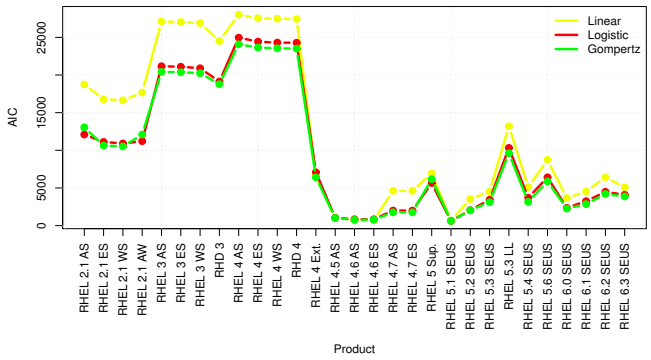
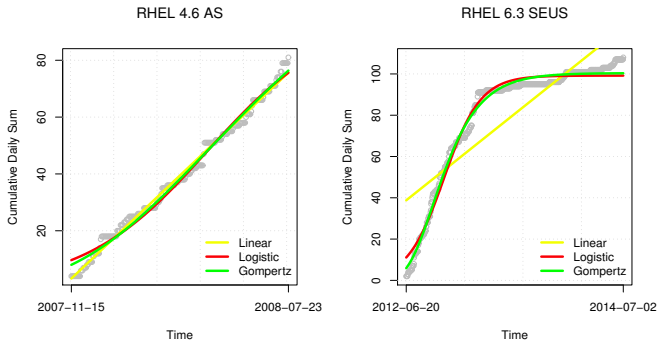


Figure 5: Security Patches for Two Red Hat Products



The general lack of fit for the linear curve becomes evident also when the individual estimated models are evaluated graphically. Consider one of the two products for which the linear function provides a decent fit according to the F-tests and AIC comparisons. This is shown on the left-hand side plot in Fig. 5. While the linear function fares better during the start of the time series, the sigmoidal functions do not yield entirely bad fits either. Rather, the wording about an entirely bad fit should be reserved for the many other linear fits such as the one that appears on the right-hand side plot. Clearly, the linear function is a bad choice to describe the growth of security patches in the RHEL 6.3 SEUS, which is the latest product that has reached the EOL stage in the primary dataset. When evaluated visually, the two sigmoidal curves yield particularly appealing fits also for the rest of the minor Red Hat releases. It seems that Hypothesis H_3 holds.

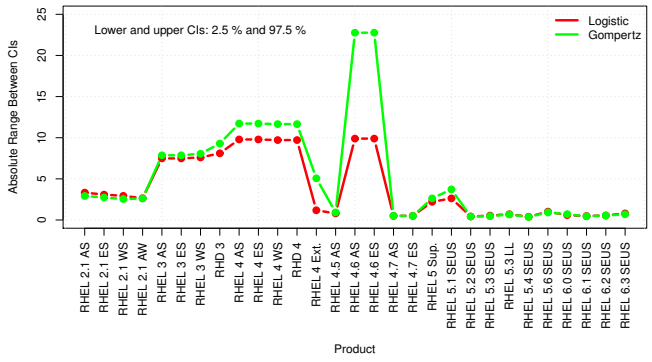
The estimates for the asymptotes are mostly sensible. Also the standard errors of the estimates are reasonably small for all but two Gompertz fits in the security patch sample. This is illustrated in Fig. 6 that shows the absolute range between the lower (2.5 %) and the upper (97.5 %) confidence intervals (CIs) for the estimated $\widehat{\alpha}_1, \ldots, \widehat{\alpha}_{29}$. That is, given the 95 % level confidence that

$$\text{Probability}\{a_i \leq \widehat{\alpha}_i \leq b_i\} = 0.95, \qquad (9)$$

the shown interval refers more specifically to the absolute

range between a_i and b_i. The wider the absolute range, the more imprecise the estimate. The standard errors from the Gompertz models are over twice as much as the standard errors from the logistic models for the two visible outliers (RHEL 4.6 AS and ES). This is also reflected in the shown confidence interval ranges. These are, moreover, the same products for which the linear model seems to give a reasonable approximation in the security patch sample. In general, tighter intervals are present for the minor releases, which is in accordance with the graphical evaluations. These effects should not be exaggerated.

Figure 6: CI Range for the Asymptote (Red Hat, Security Patches)



In fact, neither the parameter estimates for the asymptote nor the corresponding standard errors differ significantly between the logistic and the Gompertz fits. Following the idea of Wang and Bushman (2006), results from the nonparametric Mann-Whitney tests are shown in Table 1. As can be concluded from it, it is rather $\widehat{\beta}$ and $\widehat{\zeta}$ that show statistically significant differences between the two sigmoidal model estimates.

It should be also recalled that the statistical significance of the parameters is difficult to deduce in nonlinear models; even when the residuals seem to be normal, the t-values may not be correctly approximated in small samples (Fox and Weisberg, 2010). Rather than continuing with the confidence intervals and statistical significances, another comparative test can be carried out to compare the estimates between security patches and bug fixes. Table 2, thus, reports Kendall's rank correlation coefficients between the estimates using the columns of Appendix Table A.3 as the data source that is correlated against the equivalent columns from the bug fix sample; the idea is credited to Vaas et al. (2012). The AIC values are highly correlated between all fitted curves. Also the estimates for $\widehat{\alpha}$ and $\widehat{\zeta}$ are clearly correlated with each other. The magnitudes are difficult to interpret, but these correlations between the estimates for the asymptote and the rate parameter, nonetheless, indicate that the obtained fits might be roughly comparable in the two samples.

Although it remains debatable whether simple correlations provide a sufficient assertion for the specified Hypothesis H_2, it does seem that particularly the Gompertz function could be used to describe both the growth of released security patches and bug fixes. The few noted inaccuracies must be acknowledged, however. On the other hand, as these do not systematically discriminate major and minor releases, there is no reason to reject Hypothesis H_3. The specific wording in H_2 and H_3, however, make both hypotheses conditional on H_1. In other words, it may be possible that the fits are equally bad in both samples – or in all products.

4.2. Assessments

Statistical model assessments can be generally done by either considering the NLS optimization procedure or by examining the residual series. In addition, different resampling techniques can be used to assess the parameter estimates and the inference based on normal distribution (Fox and Weisberg, 2010). For the purposes of this experiment, a brief look to the residual series is sufficient in order to demonstrate that the estimated models are not free of potential statistical problems.

First of all, an assumption of normal distribution is not realistic for the majority of the examined residual series. For instance, a null hypothesis of normal distribution is rejected ($p < 0.05$) for all but five residual series (out of 174) according to the Shapiro-Wilk test. This is summarized in Table 3 that shows the W-statistics from the test. The results support the earlier observations from the technology diffusion field within with normal approximation has been noted to be problematic (Cunningham and Kwakkel, 2014). In other words, asymptotic inference based on the normal distribution is likely to be misleading. A similar conclusion can be expected to result from different resampling computations such as bootstrapping.

Second, the basic time series assumption of independence fails for many of the fits already when judged visually. As can be seen from Fig. 7, a few residuals series from the 2×29 Gompertz fits, for instance, exhibit severe fluctuations. This is particularly evident for the RHEL 3 and 4 product lines for which the cumulative amount of bug fixes tends to take the shape of a step function (see Fig. 2). If $\widehat{\rho}_1$ denotes the estimated (auto)correlation between $\widehat{\varepsilon}_t$ and $\widehat{\varepsilon}_{t-1}$, it is easy to verify formally that in all residual series $\widehat{\rho}_1 \neq 0$, thus establishing a first-order dependence structure. This can be done for instance with the conventional Durbin-Watson test (here see, for instance, Maddala, 2001). The test has been used also previously to assess growth curve fits (López et al., 2004). The d-statistic of interest is computed from

$$d = \sum_{t=2}^{T} \left(\widehat{\varepsilon}_t - \widehat{\epsilon}_{t-1}\right)^2 \Bigg/ \sum_{t=1}^{T} \widehat{\varepsilon}_t^2. \qquad (10)$$

Although the critical values are rather difficult to compute particularly in the nonlinear context, the basic interpretation of (10) is unambiguous: d is zero whenever $\widehat{\rho}_1 = 1$, and if $\widehat{\rho}_1 = -1$, then $d = 4$. When there is no correlation, the value should be close to $d = 2$. The results

Table 1: Logistic and Gompertz Models Compared (Red Hat, Mann-Whitney U)

		$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC	F
Parameter Estimates	Security Patches	381.0	832.00	603.00	475.00	508.50
	Bug Fixes	384.0	766.00	562.00	446.00	459.00
Standard Errors	Security Patches	393.0	655.00	562.00	475.00	508.50
	Bug Fixes	372.0	679.00	525.00	446.00	459.00

The shown values are U-statistics from the Mann-Whitney test (see the *wilcox.test*(·) function in R); colored entries imply $p < 0.05$. The parameter estimates, AICs, and (exact) F-statistics between the corresponding columns 3 – 7 and 8 – 12 in Appendix Table A.3 provide the input sources for the results on the first row. The remaining three rows are constructed analogously. The three parameter-columns on the last two rows show the results for the (two-sided) null hypotheses that the parameter standard errors are distributed equally between the logistic and Gompertz fits.

Table 2: Correlations Between Parameters in the Two Samples (Red Hat, Kendall τ)

		Linear	Logistic				Gompertz			
		AIC	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC
Linear	AIC	0.78				0.75				0.76
Logistic	$\widehat{\alpha}$		0.61	0.45	-0.62		0.63	0.36	-0.59	
	$\widehat{\beta}$		-0.28	0.15	0.17		-0.30	0.16	0.15	
	$\widehat{\zeta}$		-0.47	-0.39	0.65		-0.49	-0.32	0.68	
	AIC	0.83				0.79				0.80
Gompertz	$\widehat{\alpha}$		0.59	0.47	-0.60		0.61	0.38	-0.61	
	$\widehat{\beta}$		-0.28	0.11	0.16		-0.30	0.14	0.16	
	$\widehat{\zeta}$		-0.40	-0.35	0.68		-0.42	-0.28	0.71	
	AIC	0.83				0.79				0.80

The shown values are Kendall's τ-coefficients between the parameter estimates from the two samples; colored entries refer to values larger or equal to 0.5 in absolute value.

Table 3: Normality Assessment (Red Hat, Shapiro-Wilk W)

	Security Patches			Bug Fixes		
	Linear	Logistic	Gompertz	Linear	Logistic	Gompertz
RHEL 4.6 AS	0.979	0.978	0.990	0.990	0.982	0.978
RHEL 4.6 ES	0.979	0.978	0.990	0.990	0.982	0.978
RHEL 5.1 SEUS	0.984	0.971	0.989	0.968	0.960	0.963

The shown values are W-statistics from *shapiro.test*(·) in R. Non-significant ($p \geq 0.05$) values are colored, and only those rows are shown for which H_0 remained in force at least once.

Table 4: Autocorrelation Assessment (Red Hat, Durbin-Watson d)

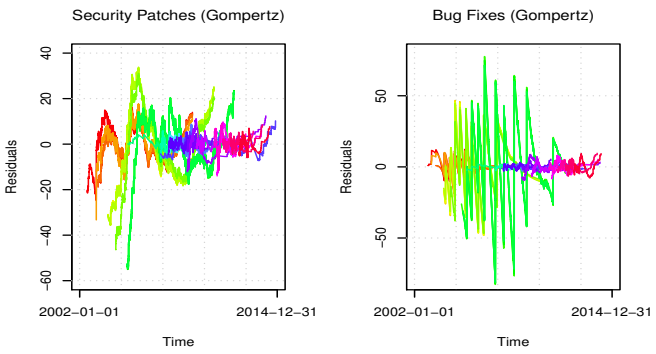
	Security Patches			Bug Fixes		
	Linear	Logistic	Gompertz	Linear	Logistic	Gompertz
Minimum	0.001	0.001	0.001	0.001	0.003	0.004
Maximum	0.149	0.166	0.250	0.093	0.133	0.134
Median	0.001	0.007	0.011	0.003	0.025	0.034

The summary refers to the d-values from the Durbin-Watson tests for the $2 \times 3 \times 29$ residuals.

in Table 4 establish that this is clearly not the case; the computed d-values are close to zero for all residual series. Given that a tabulated critical value for (one-sided) testing of $H_0 : \widehat{\rho}_1 = 0$ against $\widehat{\rho}_1 > 0$ is somewhere near 1.75 at 5 % level (Maddala, 2001), the presence of severe (positive) autocorrelation is more than evident already because the maximum computed d is as low as 0.25.

Third, some of the residual series in Fig. 7 do not have a constant variance, even though the mean may be relatively constant. This has been noted to be a problem in some growth curve estimates (López et al., 2004; Meade and Islam, 2006). In the statistical time series literature the non-constant variance characteristics are often tested and modeled in terms of the so-called autoregressive con-

Figure 7: Residuals from the Gompertz Fits (Red Hat)



ditional heteroskedasticity (Maddala, 2001; Tsay, 2004). The basic idea behind this complex term is that the conditional variance of the residual process follows some autoregressive pattern such as $\text{Var}(\varepsilon_t) = c_0 + c_1\varepsilon_{t-1}^2 + c_2\varepsilon_{t-2}^2$. These time series heteroskedasticity patterns may well be the source of the observed non-constant residual variances.

The three noted residual characteristics are not fatal in the sense that the consistency of the nonlinear estimates $\{\widehat{\alpha}, \widehat{\beta}, \widehat{\zeta}\}$ would be necessarily threatened. These violations rather reinforce the earlier remarks about the asymptotic assumptions; the standard errors for the parameter set are presumably misleading; the raw t-values are likely deceiving; neither the computed F-tests in Appendix Table A.3 nor the confidence interval ranges in Fig. 6 are entirely accurate; and so forth. Another question is whether these violations are enough to violate the wording about a statistically satisfactory fit that was used in Hypothesis H_1. In essence, this question restates the fundamental difference between growth curve modeling and statistical time series models. This question in turn is best evaluated against the practical purpose of empirical modeling. Since the primary purpose in this paper is evaluation of theory, the observed deficiencies can be argued to be minor obstacles; a sigmoidal growth curve is not a bad choice for a descriptive theoretical characterization. If the theory is further used to predict the future, however, these statistical problems imply a generally more uncertain future.

4.3. Forecasts

The time series literature is usually keen to maintain that forecasting can be considered only after the given entertained model is optimally calibrated and free of statistical problems (Tsay, 2004). As was demonstrated, these requirements are difficult to satisfy in the present empirical experiment. Rather than as an attempt to predict the future, the brief forthcoming forecast experiment is better understood as a simple assessment over what the curves indicate as the asymptotes are approached, and a constraint is placed over the actual fitting period.

There are a couple of important decisions that must be made beforehand. First, a decision is required over the used statistic to assess the forecast accuracy. The multitude of available measures can be grouped into magnitude measures, distributional measures, and directional measures (Tsay, 2004). While measures from the last group, which assess whether the future direction goes up or down, have been sometimes used in the curve fitting context (López et al., 2004), the magnitude measures have been more common (Suominen and Seppänen, 2014; Young, 1993). The usual suspects in this group are the mean square error, the mean absolute deviation, and the mean absolute percentage error (MAPE), which all have their own limitations (Hendry and Clements, 2000; Roumani et al., 2015). As the last one is arguably the most intuitive, it was adopted for the tentative forecast evaluation.

This measure can be written as

$$\text{MAPE} = 100 \times \frac{1}{T_F} \sum_{i=s}^{T} \left| \frac{\widehat{y}_i}{y_i} - 1 \right|, \qquad (11)$$

where $T_F \geq 1$ denotes the length of the 1-through-T_F-steps-ahead forecast window and $s = T - T_F + 1$, while y_i and $\widehat{y}_i$ refer to the actual and forecast values in the window, respectively. The actual curve fitting is done via R's *predict.nls*$(\cdot)$, given the initial training interval $t = 1, \ldots, T - T_F$. Otherwise the same computational routine is used (see Section 3.3). These computational details exemplify such terms as projections (Höök et al., 2011) or trend extrapolations (Suominen and Seppänen, 2014) that are sometimes used in the context to signify the difference to more conventional statistical time series forecasting. Finally, the actual definition in (11) is analogous to the one used by Tsay (2004). A small value is desirable.

Second, the choice over the statistical measure is arguably only a computational detail when compared to the choice over T_F. The technology diffusion field (within which trend extrapolations are commonly considered) usually operates with short annual time series, which imply only a small T_F by construction. For instance, Young (1993) uses $T_F \in \{1, 3\}$. Since both the security patch and the bug fix time series are based on daily observations, slightly larger forecast windows can be considered for the present experiment. Given the varying length of the univariate time series (see the columns T_S and T_B in Appendix Table A.1), the length of the forecast window is defined as

$$T_F = \lfloor T \times 0.03 \rfloor, \quad T \in \{T_S, T_B\}, \qquad (12)$$

where the choice to use approximately 3 % of T is arbitrary. The definition in (12) implies that the computed MAPEs are again only comparable across the three different curves fitted for a given product in either sample.

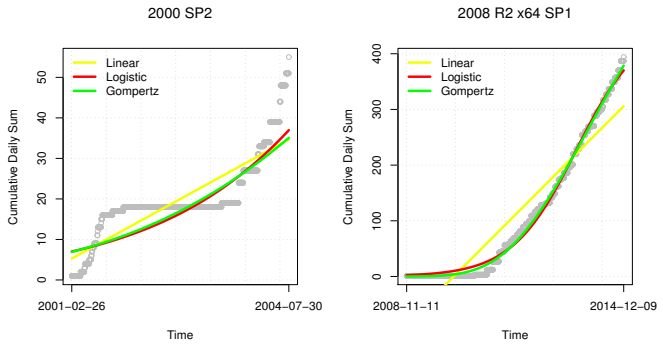
According to the results shown in Appendix (Table A.4), the Gompertz model is again the winner in terms of the smallest MAPEs. Although there is expectedly a little more variation than with the earlier AIC and F-test comparisons, the Gompertz model still loses only five times in

the security patch sample. Three of these cases (RHEL 4. Ext., RHEL 4.6 AS, and RHEL 4.6 ES) are same with respect to the AIC comparisons (see Appendix Table A.3), and two of these refer to the outliers that were discussed in Section 4.1. While the logistic model seems to work slightly better in the bug fix sample, the Gompertz model is the winner also therein. In conclusion, the forecast experiment does not bring much new information to the earlier comparisons; that is, the mean absolute percentage errors generally agree with the earlier empirical results.

4.4. Generalizability

The same threefold estimation strategy was used to evaluate the growth of security patches in the secondary dataset of Microsoft products. A summary of the results is shown in Appendix (Table A.5). The results contain no surprises: the Gompertz function delivers the best fits, although a linear trend is again appropriate for two outlying products. While the datasets are not directly comparable, it is worth remarking that the rate parameter estimates are rather similar to those obtained for the Red Hat products. This suggests that the inflection points occur generally earlier than the logistic growth function would entail.

Figure 8: Security Patches for Two Microsoft Products



The same conclusion is reached by comparing the trend extrapolations and MAPEs. Also the basic statistical assessments are comparable to the primary dataset; non-normality, autocorrelation, and non-constant variances obstruct the estimates. Despite of these problems, subjective graphical evaluation allows to characterize the fits as appealing. However, none of the three growth functions can adequately describe the growth in one outlying product, as can be seen from the left-hand side plot in Fig. 8. This is clearly an abnormal product; like the example on the right-hand side plot, all remaining sigmoidal curves display rather good-looking solutions. In short: sigmoidal growth characterizes also the trends in the secondary dataset.

Despite of the presented new evidence, it remains impossible to strictly infer whether the results generalize to the whole historical population of operating system software products. If the discussed theoretical restrictions in Section 2.3 are extended with a constraint that limits the scope to the server market, the empirical results can be expected to generalize at least towards this market segment. This is the segment within which Microsoft and Red Hat cover a large market share – and within which the operating system products are roughly comparable in terms of such characteristics as functionality, size, and life cycles.

5. Conclusion

Answering to the urge for more replications (Boylan et al., 2015; Evanschitzky and Armstrong, 2010), the paper evaluated, replicated, and extended the empirical results that were initially obtained for the sigmoidal growth model of security vulnerabilities (Alhazmi et al., 2005, 2007; Alhazmi and Malaiya, 2008). Given the different functions for replication (Gómez et al., 2014), neither sampling errors nor researcher biases (Shepperd et al., 2014) were observed. Although generalizability remains to be a general problem, the replication revealed no particular problems with respect to the presumed operating system population. The hypotheses that were used to verify the model lead to some theoretical problems, however. Besides summarizing the empirical results, the remainder of this paper briefly discusses the potential for practical applications, limitations, and future directions.

5.1. Results

The empirical experiment tested the growth model with nearly seventy operating system products, using security advisories as a proxy for the cumulation of known vulnerabilities. The analysis was based on the suggested criteria for good practices in growth curve modeling: (1) the basic *model validity* was satisfied since the upper bound was explicitly known in the primary dataset; (2) brief *statistical validity* assessments were made; (3) and tentative *forecast ability* was demonstrated (Meade and Islam, 2006). Depending on the viewpoint, the latter two aspects are tantamount to the decision to either accept or challenge the three hypotheses that were specified in Section 2.3. In essence, the answer depends on the viewpoint one is willing to adopt for growth curve modeling in general.

In terms of the binary distinction between statistical and mechanical growth modeling (López et al., 2004), the reported statistical problems – such as autocorrelation, outliers, and non-normality – indicate that the growth cannot be taken to mechanically follow a clear-cut sigmoidal growth function. If one adopts a more descriptive standpoint, as advocated for instance by Höök et al. (2011), a sigmoidal growth curve is not an ill-chosen pattern to empirically characterize the cumulative amount of security patches that have affected most of the analyzed Red Hat and Microsoft operating system products. As the more descriptive understanding is also endorsed in this paper, all three hypotheses can be accepted based on the results (R) that can be summarized as follows.

R_1 The Hypothesis H_1 presumed that a commonly used sigmoidal growth function can be used to statistically describe the cumulative amount of known post-release operating system security vulnerabilities that are observed with univariate per-release time series. Despite of apparent statistical problems, the results indicate that particularly the classical Gompertz growth function can be used to tentatively characterize the vulnerability (security advisory) growth trends of most, but not all, examined products. Since highly similar results were obtained with the logistic growth curve that was originally specified for the model, it can be expected that any common sigmoidal function can be used for descriptive purposes. This agrees with the general conclusion of López et al. (2004).

R_2 The second Hypothesis H_2 was specified as a conditional corollary to the first hypothesis: when a sigmoidal growth function characterizes the vulnerability trends, it should also characterize the cumulative amount of corrected software faults. Although exact correspondence was not observed, the empirical results indicate that sigmoidal functions describe also the observed long-run fault trends relatively well.

R_3 The final Hypothesis H_3 asserted that a supportive result for H_1 should not imply divergence between major and minor operating system releases. While there were a few visible outliers, sigmoidal growth patterns mostly apply to both release types.

In other words, *the paper supports the contested theoretical growth model*. As in the original empirical experiment of Alhazmi et al. (2007), there were, however, a few outlying products for which a linear trend was more appropriate. The results do not, therefore, fully generalize to the population of Red Hat and Microsoft operating systems. It should be also emphasized that some other nonlinear function may describe the growth better. Furthermore, the latter two Hypotheses H_2 and H_3 were specified in advance as theoretical beacons of potential validity problems.

5.2. Applications

The fundamental theoretical argument can be seen to be implicitly excessively pessimistic in terms of concrete software engineering work. If the growth of security vulnerabilities mechanically follows a sigmoidal pattern that is primarily dependent on the market characteristics, there is only a limited theoretical space for software and security engineering to impact the observed S-shaped growth trends. This limits also the room for practical applications.

The theory was originally postulated with historical data on the cumulative amount of security vulnerabilities that had affected operating system software products when these were in active use. This excludes the periods during which the products were actually developed. While security is an essential aspect of post-development software maintenance, the initial development phase is arguably more relevant in terms of different security engineering practices. The sigmoidal growth model cannot, however, explain the impact of such security engineering practices upon the post-release growth of security vulnerabilities. In other words, historical analysis with postmortem variables cannot explain what happened before the software product entered a given market (Hein and Saiedian, 2009). This is another limitation for practical applications.

Since the theoretical argument is closely related to the inflection points in sigmoid functions, it seems that a further practical limitation may be the requirement of fixed software life cycles. The model may be difficult to apply in case the software in question is continuously evolving with irregular release cycles.

Despite of these constraints, there is still potential also for practical use particularly from the management point of view. Since the patching of security vulnerabilities tends to follow a similar trend from a release to another, a projection from a previous release can be used in planning, staffing, and resource allocation in general. The same rationale applies to post-release reliability modeling (Jalote et al., 2008). The practical release engineering aspect is further reinforced by the empirical results according to which sigmoidal trends apply also to the minor product releases within major product lines. The observed sigmoidal trends can be portrayed also from the perspective of business continuity planning; it may be worthwhile to revisit the past software engineering of a product that failed to follow the supposedly relatively common sigmoidal trend. From a customer perspective, moreover, a sigmoidal trend might be used to evaluate the risks associated with the use of old operating systems, or to evaluate the right time to switch to new systems. Finally, quantitative models should not be undermined in the ongoing global efforts to improve security education and public awareness.

5.3. Limitations and Future Directions

The practical limitations cannot be separated from the vulnerability growth model. There are, in addition, at least four important research limitations, all of which also require further empirical evaluations. The first is related to the central theoretical argument behind the model.

First, the argument that sigmoidal vulnerability trends are driven by popularity is the factor that generally separates the evaluated model from reliability modeling. In this paper software reliability was further related to the vulnerability growth model with an assertion that also the post-release cumulative amount of fixed software faults tends to follow a sigmoidal growth pattern (H_2). As this was indeed observed to be the case in many Red Hat operating system products, it is not entirely clear whether the observed empirical quantities really reflect what these are meant to reflect theoretically; whether the sigmoidal growth is unique to software vulnerabilities.

If security bugs are comparable to normal bugs, the well-established reliability models should deliver the adequate

answers. Given the fundamental nature of the issue, further empirical research is required to actually evaluate the popularity argument. A concrete research approach should be relatively straightforward to formulate. For instance, both the logistic growth model and the Gompertz function contain well-defined inflection points that can be estimated from data. If suitable high-frequency data would be available for the adoption rates or installation bases, it would be easy to evaluate the theoretical validity of the imposed inflection points for the vulnerability trends.

Second, the model is not immune to the general reliability and validity concerns that surround empirical vulnerability data (Massacci and Nguyen, 2014). In particular, further empirical research is required to understand the potential limitations of security advisory data.

Third, there is room for further empirical work related to the statistical implications of the sigmoidal vulnerability growth trends. While different scaling solutions were considered in the original research (Alhazmi et al., 2007), there may be also covariates that improve the univariate nonlinear growth curve estimates. Perhaps more importantly, the existence of sigmoidal growth trends may affect the assumptions made in more general time series modeling that uses vulnerabilities as the dependent variable.

Last, the replication triggered another assertion (H_3): the specified sigmoidal growth curves described particularly well the trends of minor operating system releases that have extended the life cycle of the initial major releases. This leads to a potential theoretical contradiction: if both the major releases and the subsequent minor releases have their own distinct S-shaped curves, it is difficult, but not necessarily impossible, for the whole active life cycle to follow the intended sigmoidal pattern. The observation signifies the relevance to consider multiple subsequent products that are combined into a single time series. In this regard the extensions (Kim et al., 2007; Woo et al., 2011) to the original model can be seen as prolific for pursuing the vulnerability growth model further. Different combined approaches have been also actively developed in the field of technology diffusion (Meade and Islam, 2006; Norton and Bass, 1987), which may provide a source for further interdisciplinary theoretical insights.

References

Alhazmi, O., Malaiya, Y., Ray, I., 2005. Security Vulnerabilities in Software Systems: A Quantitative Perspective. In: Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 2005). Springer-Verlag, pp. 281–294.

Alhazmi, O. H., Malaiya, Y. K., 2008. Application of Vulnerability Discovery Models to Major Operating Systems. IEEE Transactions on Reliability 57 (1), 14–22.

Alhazmi, O. H., Malaiya, Y. K., Ray, I., 2007. Measuring, Analyzing and Predicting Security Vulnerabilities in Software Systems. Computers & Security 26 (3), 219–228.

Annadurai, G., Babu, S. R., Srinivasamoorthy, V. R., 2000. Development of Mathematical Models (Logistic, Gompertz and Richards Models) Describing the Growth Pattern of *Pseudomonas putida* (NICM 2174). Bioprocess Engineering 23 (6), 607–612.

Arora, A., Forman, C., Nandkumar, A., Telang, R., 2010. Competition and Patching of Security Vulnerabilities: An Empirical Analysis. Information Economics and Policy 22 (2), 164–177.

Bass, F. M., 1969. A New Product Growth for Model Consumer Durables. Management Science 15 (2), 215–227.

Berger, R. D., 1981. Comparison of the Gompertz and Logistic Equations to Describe Plant Disease Progress. Phytopathology 71 (7), 716–719.

Birch, C. P. D., 1999. A New Generalized Sigmoid Growth Equation Compared with the Richards Growth Equation. Annals of Botany 83, 713–723.

Boylan, J. E., Goodwin, P., Mohammadipour, M., Syntetos, A. A., 2015. Reproducibility in Forecasting Research. International Journal of Forecasting 31 (1), 79–90.

Cunningham, S. W., Kwakkel, J. H., 2014. Tipping Points in Science: A Catastrophe Model of Scientific Change. Journal of Engineering and Technology Management 32, 185–205.

Evanschitzky, H., Armstrong, J. S., 2010. Replications of Forecasting Research. International Journal of Forecasting 26 (1), 4–8.

Fisher, J. C., Pry, R. H., 1971. A Simple Substitution Model of Technological Change. Technological Forecasting and Social Change 3, 75–88.

Fox, J., Weisberg, S., 2010. Nonlinear Regression and Nonlinear Least Squares in R, an Appendix to *An R Companion to Applied Regression*. Available online on October, 2014: http://socserv.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Nonlinear-Regression.pdf.

Gao, L., Porter, A. L., Wang, J., Fang, S., Zhang, X., Ma, T., Wang, W., Huang, L., 2013. Technology Life Cycle Analysis Method Based on Patent Documents. Technological Forecasting and Social Change 80 (3), 398–407.

Gómez, O. S., Juristo, N., Vegas, S., 2014. Understanding Replication of Experiments in Software Engineering: A Classification. Information and Software Technology 56 (8), 1033–1048.

Gompertz, B., 1825. On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies. Philosophical Transactions of the Royal Society of London 115, 513–583.

Hein, D., Saiedian, H., 2009. Secure Software Engineering: Learning from the Past to Address Future Challenges. Information Security Journal: A Global Perspective 18 (1), 8–25.

Hendry, D. F., Clements, M. P., 2000. Economic Forecasting in the Face of Structural Breaks. In: Holly, S., Weale, M. (Eds.), Econometric Modelling: Techniques and Applications. Cambridge University Press, Cambridge, pp. 3–37.

Höök, M., Li, J., Oba, N., Snowden, S., 2011. Descriptive and Predictive Growth Curves in Energy System Analysis. Natural Resources Research 20 (2), 103–116.

Jalote, P., Murphy, B., Sharma, V. S., 2008. Post-Release Reliability Growth in Software Products. ACM Transactions on Software Engineering and Methodology 17 (4), 17:1–17:20.

Jones, J. R., 2007. Estimating Software Vulnerabilities. Security & Privacy 5 (4), 28–32.

Kahm, M., Hasenbrink, G., Lichtenberg-Fraté, H., Ludwig, J., Kschischo, M., 2010. grofit: Fitting Biological Growth Curves in R. Journal of Statistical Software 33 (7), 1–21.

Kim, J., Malaiya, Y. K., Ray, I., 2007. Vulnerability Discovery in Multi-Version Software Systems. In: Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE 2007). IEEE, pp. 141–148.

Lehman, M. M., Belady, L. A., 1985. Programs, Cities, Students – Limits to Growth. Reprinted from a 1974 publication. In: Lehman, M. M., Belady, L. A. (Eds.), Program Evolution: Processes of Software Change. Academic Press, London, pp. 134–163.

Lehman, M. M., Fernández-Ramil, J. C., 2006. Software Evolution. In: Madhavji, N. M., Fernández-Ramil, J. C., Perry, D. E. (Eds.), Software Evolution: Theory and Practice. John Wiley & Sons, Chichester, pp. 7–39.

López, S., Prieto, M., Dijkstra, J., Dhanoa, M. S., France, J., 2004. Statistical Evaluation of Mathematical Models for Microbial Growth. International Journal of Food Microbiology 96 (3),

289–300.
Maddala, G. S., 2001. Introduction to Econometrics, 3rd Edition. John Wiley & Sons, Chichester.
Marinakis, Y. D., 2012. Forecasting Technology Diffusion with the Richards Model. Technological Forecasting and Social Change 79 (1), 172–179.
Massacci, F., Nguyen, V. H., 2014. An Empirical Methodology to Evaluate Vulnerability Discovery Models. IEEE Transactions on Software Engineering 40 (12), 1147–1162.
Meade, N., Islam, T., 2006. Modeling and Forecasting the Diffusion of Innovation – A 25-Year Review. International Journal of Forecasting 22 (3), 519–545.
Microsoft, Inc., 2015. Microsoft Security Bulletin Data, available online on April, 2015: http://www.microsoft.com/en-us/download/details.aspx?id=36982.
Miranda, L. C. M., Lima, C. A. S., 2013. Technology Substitution and Innovation Adoption: The Cases of Imaging and Mobile Communication Markets. Technological Forecasting and Social Change 80 (6), 1179–1193.
Norton, J. A., Bass, F. M., 1987. A Diffusion Theory Model of Adoption and Substitution for Successive Generations of High-Technology Product. Management Science 33 (9), 1069–1086.
Ohishi, K., Okamura, H., Dohi, T., 2005. Gompertz Software Reliability Model and Its Application. In: Proceedings of the 29th Annual International Software and Applications Conference (COMPSAC 2005). IEEE, pp. 405–410.
Pearl, R., Reed, L. J., 1920. On the Rate of Growth of the Population of the United States since 1790 and Its Mathematical Representation. Proceedings of the National Academy of Sciences 6 (6), 275–288.
Rajlich, V. T., Bennett, K. H., 2000. A Staged Model for the Software Life Cycle. Computer 33 (7), 66–71.
Red Hat, Inc., 2014. Security Alerts, Bug Fixes, and Enhancements for Unsupported Products, available online on November, 2014: https://access.redhat.com/security/updates/eol/.
Richards, F. J., 1959. A Flexible Growth Function for Empirical Use. Journal of Experimental Botany 10 (2), 290–301.
Roumani, Y., Nwankpa, J. K., Roumani, Y. F., 2015. Time Series Modeling of Vulnerabilities. Computers & Security 51, 32–40.
Shepperd, M., Bowes, D., Hall, T., 2014. Researcher Bias: The Use of Machine Learning in Software Defect Prediction. IEEE Transactions on Software Engineering 40 (6), 603–616.
Suominen, A., Seppänen, M., 2014. Bibliometric Data and Actual Development in Technology Life Cycles: Flaws in Assumptions. Foresight 16 (1), 37–53.
Suominen, A., Tuominen, A., 2010. Analyzing the Direct Methanol Fuel Cell Technology in Portable Applications. The Journal of Business Chemistry 7 (3), 117–130.
Tsay, R. S., 2004. Nonlinear Models and Forecasting. In: Clements, M. P., Hendry, D. F. (Eds.), A Companion to Economic Forecasting. Blackwell Publishing, Oxford, pp. 453–484.
Vaas, L. A. I., Sikorski, J., Michael, V., Göker, M., Klenk, H., 2012. Visualization and Curve-Parameter Estimation Strategies for Efficient Exploration of Phenotype Microarray Kinetics. PLoS ONE 7 (4), e34846.
Wang, G. P., Bushman, F. D., 2006. A Statistical Method for Comparing Viral Growth Curves. Journal of Virological Methods 135 (1), 118–123.
Wang, J., Wu, Z., Shu, Y., Zhang, Z., 2014. An Imperfect Debugging Model Considering Log-Logistic Distribution Fault Content Function. The Journal of Systems and Software 100, 167–181.
Winsor, C. P., 1932. The Gompertz Curve as a Growth Curve. Proceedings of the National Academy of Sciences 18 (1), 1–8.
Woo, S.-W., Joh, H., Alhazmi, O. H., Malaiya, Y. K., 2011. Modeling Vulnerability Discovery Process in Apache and IIS HTTP Servers. Computers & Security 30 (1), 50–62.
Wright, J. L., McQueen, M., Wellman, L., 2013. Analyses of Two End-User Software Vulnerability Exposure Metrics (Extended Version). Information Security Technical Report 17 (4), 173–184.
Yamada, S., Ohba, M., Osaki, S., 1983. S-Shaped Reliability Growth Modeling for Software Error Detection. IEEE Transactions on Reliability R-32 (5), 475–484.
Young, P., 1993. Technological Growth Curves: A Competition of Forecasting Models. Technological Forecasting and Social Change 44 (4), 375–389.
Zwietering, M. H., Jongenburger, I., Rombouts, F. M., van 'T Riet, K., 1990. Modeling of the Bacterial Growth Curve. Applied and Environmental Microbiology 56 (6), 1875–1881.

A. Appendix

Table A.1: The Primary Red Hat Dataset

Product	Abbreviation	N_S	N_B	T_S	T_B
RH Enterprise Linux AS (v. 2.1)	RHEL 2.1 AS	336	33	2531	2119
RH Enterprise Linux ES (v. 2.1)	RHEL 2.1 ES	293	26	2267	1924
RH Enterprise Linux WS (v. 2.1)	RHEL 2.1 WS	287	25	2267	1924
RH Linux Advanced Workstation 2.1 for IA-64	RHEL 2.1 AW	295	28	2310	2037
RH Enterprise Linux AS (v. 3)	RHEL 3 AS	383	53	2547	2353
RH Enterprise Linux ES (v. 3)	RHEL 3 ES	383	53	2547	2353
RH Enterprise Linux WS (v. 3)	RHEL 3 WS	379	50	2547	2353
RH Desktop (v. 3)	RHD 3	351	45	2360	2185
RH Enterprise Linux AS (v. 4)	RHEL 4 AS	477	158	2572	2502
RH Enterprise Linux ES (v. 4)	RHEL 4 ES	477	142	2572	2464
RH Enterprise Linux WS (v. 4)	RHEL 4 WS	474	137	2572	2464
RH Desktop (v. 4)	RHD 4	474	137	2572	2464
RH Enterprise Linux Extras (v. 4)	RHEL 4 Ext.	109	23	2542	2408
RH Enterprise Linux AS (v. 4.5.z)	RHEL 4.5 AS	54	17	479	525
RH Enterprise Linux AS (v. 4.6.z)	RHEL 4.6 AS	52	20	252	257
RH Enterprise Linux ES (v. 4.6.z)	RHEL 4.6 ES	52	20	252	257
RH Enterprise Linux AS (v. 4.7.z)	RHEL 4.7 AS	65	22	1127	833
RH Enterprise Linux ES (v. 4.7.z)	RHEL 4.7 ES	65	22	1127	833
RHEL Supplementary (v. 5 server)	RHEL 5 Sup.	156	28	2760	2024
RH Enterprise Linux Server EUS (v. 5.1.z)	RHEL 5.1 SEUS	39	19	197	196
RH Enterprise Linux Server EUS (v. 5.2.z)	RHEL 5.2 SEUS	50	35	658	597
RH Enterprise Linux Server EUS (v. 5.3.z)	RHEL 5.3 SEUS	65	47	715	700
RH Enterprise Linux Long Life (v. 5.3 server)	RHEL 5.3 LL	82	60	1889	1880
RH Enterprise Linux Server EUS (v. 5.4.z)	RHEL 5.4 SEUS	46	72	693	677
RH Enterprise Linux Server EUS (v. 5.6.z)	RHEL 5.6 SEUS	65	73	1065	1112
RH Enterprise Linux Server EUS (v. 6.0.z)	RHEL 6.0 SEUS	67	39	755	581
RH Enterprise Linux Server EUS (v. 6.1.z)	RHEL 6.1 SEUS	70	58	735	630
RH Enterprise Linux Server EUS (v. 6.2.z)	RHEL 6.2 SEUS	60	76	765	763
RH Enterprise Linux Server EUS (v. 6.3.z)	RHEL 6.3 SEUS	77	81	743	660

The first column reports the products under the names given by Red Hat, Inc. (2014), using an abbreviation RH to denote "Red Hat". The following column lists the used abbreviations in this paper. The symbols N_S and N_B refer to the number of individual security patches and bug fixes before cumulation, respectively. The subsequent symbols T_S and T_B refer to the number of days in the cumulated security patch and bug fix samples. More specifically: given the first and the last security advisory that were issued for the given product at days t_1 and t_{T_S}, respectively, the symbol T_S gives the number of days between $[t_1, t_{T_S}]$. The symbol T_B is defined analogously.

Table A.2: The Secondary Microsoft Dataset

Product	Abbreviation	N_S	T_S
Microsoft Windows 2000	2000	69	1385
Microsoft Windows 2000 SP1	2000 SP1	26	344
Microsoft Windows 2000 SP2	2000 SP2	28	1251
Microsoft Windows 2000 SP3	2000 SP3	22	812
Microsoft Windows 2000 SP4	2000 SP4	86	2639
Microsoft Windows XP	XP	40	1301
Microsoft Windows XP SP1	XP SP1	43	1729
Microsoft Windows XP SP2	XP SP2	73	2667
Microsoft Windows XP SP3	XP SP3	83	2726
Microsoft Windows XP Professional x64 Edition	XP Prof. x64	65	2212
Microsoft Windows XP Professional x64 Edition SP2	XP Prof. x64 SP2	113	4055
Windows Vista	Vista	39	1891
Windows Vista SP1	Vista SP1	43	1464
Windows Vista SP2	Vista SP2	73	2710
Windows Vista x64 Edition	Vista x64	38	1107
Windows Vista x64 Edition SP1	Vista x64 SP1	43	1464
Windows Vista x64 Edition SP2	Vista x64 SP2	73	2710
Microsoft Windows Server 2003	2003	48	1477
Microsoft Windows Server 2003 SP1	2003 SP1	49	1527
Microsoft Windows Server 2003 SP2	2003 SP2	102	3592
Microsoft Windows Server 2003 x64 Edition	2003 x64	64	2009
Microsoft Windows Server 2003 x64 Edition SP2	2003 x64 SP2	103	3592
Microsoft Windows Server 2003, Enterprise Edition for IA-64	2003 IA-64	28	1477
Microsoft Windows Server 2003 for IA-64 SP1	2003 IA-64 SP1	48	1527
Microsoft Windows Server 2003 for IA-64 SP2	2003 IA-64 SP2	100	3592
Windows Server 2008 for 32-bit Systems	2008 i386	43	1464
Windows Server 2008 for 32-bit Systems SP2	2008 i386 SP2	74	2710
Windows Server 2008 for 32-bit Systems (SC)	2008 i386 SC	31	1282
Windows Server 2008 for 32-bit Systems SP2 (SC)	2008 i386 SC SP2	57	2010
Windows Server 2008 for x64-based Systems	2008 x64	43	1464
Windows Server 2008 for x64-based Systems SP2	2008 x64 SP2	74	2710
Windows Server 2008 for x64-based Systems SP2 (SC)	2008 x64 SC SP2	29	1275
Windows Server 2008 for IA-64	2008 IA-64	42	1464
Windows Server 2008 for IA-64 SP2	2008 IA-64 SP2	72	2710
Windows Server 2008 R2 for x64-based Systems	2008 R2 x64	47	1611
Windows Server 2008 R2 for x64-based Systems SP1	2008 R2 x64 SP1	53	2220
Windows Server 2008 R2 for x64-based Systems (SC)	2008 R2 x64 SC	37	1275
Windows Server 2008 R2 for x64-based Systems SP1 (SC)	2008 R2 x64 SC SP1	47	1534
Windows Server 2008 R2 for IA-64	2008 R2 IA-64	45	1611
Windows Server 2008 R2 for IA-64 SP1	2008 R2 IA-64 SP1	51	2220

The information is presented analogously to Table A.1. The product names used by Microsoft, Inc. (2015) were shortened by using abbreviations for the terms "Service Pack" (SP), "Server Core installation" (SC), and "Itanium-Based Systems" (IA-64).

Table A.3: Parameter Estimates for Security Patches in the Primary Dataset (Red Hat)

	Linear	Logistic					Gompertz				
Product	AIC	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC	F	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC	F
RHEL 2.1 AS	24245	499.9	2.4	0.003	20666	✓	535.7	1.2	0.002	17963	✓
RHEL 2.1 ES	20795	411.8	2.2	0.003	17757	✓	440.7	1.1	0.002	15445	✓
RHEL 2.1 WS	20566	398.7	2.2	0.003	17485	✓	427.0	1.1	0.002	15058	✓
RHEL 2.1 AW	21317	436.6	2.0	0.003	17303	✓	468.1	1.0	0.002	15360	✓
RHEL 3 AS	23463	555.9	2.2	0.002	23318	✓	601.1	1.0	0.001	21458	✓
RHEL 3 ES	23463	555.9	2.2	0.002	23318	✓	601.1	1.0	0.001	21458	✓
RHEL 3 WS	23364	546.9	2.2	0.002	23341	✓	591.6	1.1	0.001	21530	✓
RHD 3	21058	521.9	1.9	0.002	21135		567.0	0.9	0.001	19754	✓
RHEL 4 AS	21494	873.9	2.0	0.002	22796		1006.3	0.9	0.001	20535	✓
RHEL 4 ES	21494	873.9	2.0	0.002	22796		1006.3	0.9	0.001	20535	✓
RHEL 4 WS	21402	869.5	2.0	0.002	22735		1002.2	0.9	0.001	20469	✓
RHD 4	21402	869.5	2.0	0.002	22735		1002.2	0.9	0.001	20469	✓
RHEL 4 Ext.	18191	144.4	3.8	0.002	9716	✓	209.6	1.7	0.001	10787	✓
RHEL 4.5 AS	3599	82.0	2.0	0.018	2295	✓	83.5	1.0	0.013	2275	✓
RHEL 4.6 AS	1033	96.2	2.2	0.014	1195		133.1	1.0	0.006	1113	
RHEL 4.6 ES	1033	96.2	2.2	0.014	1195		133.1	1.0	0.006	1113	
RHEL 4.7 AS	9208	86.0	2.8	0.014	5875	✓	87.1	1.4	0.009	5666	✓
RHEL 4.7 ES	9208	86.0	2.8	0.014	5875	✓	87.1	1.4	0.009	5666	✓
RHEL 5 Sup.	14662	183.2	2.7	0.002	17302		212.9	1.3	0.001	14652	✓
RHEL 5.1 SEUS	764	55.1	1.8	0.022	749	✓	62.0	0.8	0.013	666	✓
RHEL 5.2 SEUS	4882	65.1	2.3	0.018	2745	✓	65.9	1.2	0.012	2597	✓
RHEL 5.3 SEUS	5666	80.7	2.3	0.020	3466	✓	81.6	1.2	0.014	3164	✓
RHEL 5.3 LL	14708	90.9	1.6	0.011	12199	✓	91.6	0.7	0.008	11819	✓
RHEL 5.4 SEUS	5154	65.5	2.1	0.019	2856	✓	66.2	1.1	0.013	2648	✓
RHEL 5.6 SEUS	7708	81.4	1.1	0.007	6247	✓	82.1	0.5	0.006	5884	✓
RHEL 6.0 SEUS	6279	102.6	1.9	0.020	3923	✓	103.5	0.9	0.014	4104	✓
RHEL 6.1 SEUS	5959	88.5	2.2	0.021	3498	✓	89.3	1.1	0.015	3270	✓
RHEL 6.2 SEUS	5940	74.8	2.3	0.022	3882	✓	75.5	1.1	0.015	3756	✓
RHEL 6.3 SEUS	6081	99.2	2.1	0.016	4119	✓	100.4	1.1	0.011	3767	✓

The F-column refers to (8) as discussed in Section 3.3, that is, the symbol ✓ is used to denote $p < 0.05$ in the F-tests, while colored values are shown for the smallest AIC in a given row.

Table A.4: Forecast Assessment in the Primary Dataset (Red Hat, MAPE)

	Security Patches				Bug Fixes			
	T_F	Linear	Logistic	Gompertz	T_F	Linear	Logistic	Gompertz
RHEL 2.1 AS	75	12.023	5.605	2.931	63	27.537	1.197	0.496
RHEL 2.1 ES	68	11.068	6.187	3.604	57	27.418	2.409	0.901
RHEL 2.1 WS	68	10.904	6.155	3.594	57	27.395	2.329	0.837
RHEL 2.1 AW	69	11.024	5.186	2.700	61	27.888	0.625	0.905
RHEL 3 AS	76	7.854	6.516	4.593	70	25.413	0.220	1.892
RHEL 3 ES	76	7.854	6.516	4.593	70	25.470	0.248	1.937
RHEL 3 WS	76	7.659	6.645	4.744	70	25.725	0.375	2.065
RHD 3	70	6.784	5.722	4.152	65	24.713	0.479	1.988
RHEL 4 AS	77	3.983	4.620	2.692	75	15.374	2.702	0.192
RHEL 4 ES	77	3.983	4.620	2.692	73	15.321	2.720	0.124
RHEL 4 WS	77	3.958	4.584	2.656	73	15.360	2.763	0.115
RHD 4	77	3.958	4.584	2.656	73	15.307	2.820	0.121
RHEL 4 Ext.	76	5.389	0.592	4.267	72	4.621	3.239	1.670
RHEL 4.5 AS	14	20.860	1.595	0.127	15	1.841	3.757	3.643
RHEL 4.6 AS	7	6.338	7.662	6.401	7	3.753	3.472	3.043
RHEL 4.6 ES	7	6.338	7.662	6.401	7	3.753	3.472	3.043
RHEL 4.7 AS	33	21.044	4.663	3.441	24	23.034	1.564	0.545
RHEL 4.7 ES	33	21.044	4.663	3.441	24	23.034	1.564	0.545
RHEL 5 Sup.	83	1.747	7.819	5.175	60	9.598	3.883	6.311
RHEL 5.1 SEUS	5	4.267	5.050	3.385	5	4.040	4.088	3.200
RHEL 5.2 SEUS	19	19.084	4.667	3.535	17	16.245	4.306	2.644
RHEL 5.3 SEUS	21	16.692	6.519	5.569	21	13.145	11.256	9.165
RHEL 5.3 LL	56	8.048	10.749	10.052	56	16.103	4.362	3.653
RHEL 5.4 SEUS	20	20.957	2.526	1.364	20	18.669	5.057	3.616
RHEL 5.6 SEUS	31	13.429	5.349	4.719	33	16.045	5.766	4.820
RHEL 6.0 SEUS	22	15.859	4.349	3.503	17	13.871	3.103	2.003
RHEL 6.1 SEUS	22	16.083	5.872	4.987	18	18.091	4.859	3.265
RHEL 6.2 SEUS	22	11.014	10.373	9.589	22	14.740	9.234	8.179
RHEL 6.3 SEUS	22	15.105	7.823	6.753	19	11.316	10.308	8.442

The symbol T_F refers to (12), while the remaining columns are MAPEs from (11) under the specified forecast window; colored entries are shown for the smallest row-wise MAPE in the two samples, respectively.

Table A.5: Parameter Estimates for Security Patches in the Secondary Dataset (Microsoft)

	Linear	Logistic					Gompertz				
Product	AIC	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC	F	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{\zeta}$	AIC	F
2000	7865	86.2	1.8	0.003	8408		94.6	0.8	0.002	8062	
2000 SP1	1451	28.5	2.2	0.017	1082	✓	30.5	1.1	0.011	1083	✓
2000 SP2	7772	331.4	3.8	0.001	7651	✓	699.6	1.5	0.000	7727	✓
2000 SP3	5063	91.9	3.8	0.007	3724	✓	143.5	1.7	0.003	3540	✓
2000 SP4	21191	441.3	3.2	0.002	20399	✓	594.0	1.4	0.001	18049	✓
XP	7298	67.3	3.9	0.004	4838	✓	101.3	1.7	0.002	4275	✓
XP SP1	12545	149.9	3.7	0.004	9719	✓	169.8	1.8	0.002	10443	✓
XP SP2	24690	393.9	4.1	0.002	20478	✓	583.1	1.8	0.001	18614	✓
XP SP3	25846	532.0	3.9	0.002	22055	✓	665.7	1.8	0.001	19106	✓
XP Prof. x64	17184	295.1	3.4	0.002	13875	✓	443.3	1.5	0.001	12066	✓
XP Prof. x64 SP2	44063	737.2	4.4	0.001	29782	✓	1320.6	1.9	0.001	31808	✓
Vista	16179	202.8	5.4	0.003	11385	✓	412.3	2.2	0.001	10851	✓
Vista SP1	12546	276.9	4.1	0.004	8985	✓	504.0	1.8	0.001	8426	✓
Vista SP2	27407	553.1	4.4	0.002	22250	✓	763.1	2.0	0.001	19822	✓
Vista x64	7504	233.8	2.9	0.003	6844	✓	529.4	1.4	0.001	6658	✓
Vista x64 SP1	12622	280.2	4.2	0.004	9116	✓	499.6	1.8	0.001	8526	✓
Vista x64 SP2	27545	553.6	4.4	0.002	22482	✓	744.8	2.0	0.001	19975	✓
2003	10337	191.1	3.4	0.003	8263	✓	331.0	1.5	0.001	7453	✓
2003 SP1	10210	229.7	3.1	0.003	9324	✓	306.4	1.4	0.002	8044	✓
2003 SP2	37160	661.2	4.3	0.002	29658	✓	852.8	2.0	0.001	24849	✓
2003 x64	15856	309.2	3.3	0.002	11160	✓	571.6	1.5	0.001	10214	✓
2003 x64 SP2	37207	661.5	4.3	0.002	29616	✓	841.1	2.0	0.001	24546	✓
2003 IA-64	11532	111.9	5.7	0.005	7858	✓	159.5	2.5	0.002	7332	✓
2003 IA-64 SP1	9181	194.0	2.9	0.003	9173	✓	255.3	1.3	0.001	8092	✓
2003 IA-64 SP2	35761	534.5	4.3	0.002	27129	✓	685.8	2.0	0.001	22013	✓
2008 i386	12361	276.6	4.0	0.004	9176	✓	474.1	1.7	0.001	8476	✓
2008 i386 SP2	27495	571.5	4.3	0.002	22179	✓	810.3	1.9	0.001	19771	✓
2008 i386 SC	6430	111.6	2.8	0.004	6096	✓	142.5	1.3	0.002	5702	✓
2008 i386 SC SP2	9611	185.5	2.4	0.002	12452		218.6	1.1	0.001	10818	
2008 x64	12585	289.9	4.1	0.004	9370	✓	499.5	1.8	0.001	8671	✓
2008 x64 SP2	27544	573.7	4.4	0.002	22469	✓	794.8	2.0	0.001	20020	✓
2008 x64 SC SP2	8319	78.5	4.7	0.006	5787	✓	92.8	2.3	0.003	5013	✓
2008 IA-64	11800	229.7	4.0	0.004	8749	✓	423.0	1.7	0.001	8120	✓
2008 IA-64 SP2	24926	344.2	4.4	0.003	20412	✓	422.6	2.1	0.001	17530	✓
2008 R2 x64	13797	245.6	4.5	0.004	10700	✓	312.4	2.1	0.002	9111	✓
2008 R2 x64 SP1	22811	455.6	5.3	0.003	16462	✓	730.0	2.2	0.001	14858	✓
2008 R2 x64 SC	6803	131.9	3.0	0.004	7652		160.1	1.4	0.002	6790	✓
2008 R2 x64 SC SP1	9863	215.4	3.1	0.003	9536	✓	280.9	1.4	0.002	8681	✓
2008 R2 IA-64	13017	201.1	4.4	0.004	10434	✓	250.3	2.0	0.002	9008	✓
2008 R2 IA-64 SP1	20539	262.6	5.2	0.003	15090	✓	333.8	2.4	0.002	13278	✓

The shown information is constructed analogously to Table A.3. The products are listed in Table A.2.