Experience in Software Reliability: From Data Collection to Quantitative Evaluation

Karama Kanoun, Mohamed Kaâniche, Jean-Claude Laprie

Paper submitted to the International Symposium on Software Reliability Engineering
ISSRE'93, Denver, Colorado, November 3-6, 1993.

EXPERIENCE IN SOFTWARE RELIABILITY: FROM DATA COLLECTION TO QUANTITATIVE EVALUATION*

Karama KANOUN, Mohamed KAANICHE and Jean-Claude LAPRIE
LAAS-CNRS, 7, avenue du Colonel Roche, 31077 Toulouse (France)
e-mail: kanoun@lasubmitte@laas.fr
Phone: 33/ 61 33 62 00 — Fax: 33/ 61 33 64 11

ABSTRACT

This paper describes the different steps in software failure data processing in order to monitor the software development and to quantify the operational reliability. Processing consists in (a) filtering the raw data in order to keep only those corresponding to software failures without duplicate, (b) partitioning of data into sub-sets according for instance to failure severity or fault location, (c) performing descriptive analyses, (d) analyzing the trend and (e) when possible and needed, applying reliability growth models. These steps are part of an overall method experienced at LAAS on several real-life software systems. The goals of the paper are twofold: first, present the method and, second, show its benefits through its application to data collected on a switching system.

Key words: Software reliability, reliability growth, trend test, failure data processing

Presenting author: Karama KANOUN

* The work reported in this paper was partially supported by the ESPRIT Basic Research Action on Predictably Dependable Computing Systems (Action no. 6362).

1- INTRODUCTION

This paper addresses software reliability analysis and evaluation from failure data collected on the software. It reflects our experience in the field based on processing failure data collected on several real-life software systems [Kanoun 1987], [Kaâniche 1990], [Kanoun 1991a], [Kanoun 1991b]. Even though each reliability evaluation can be considered as a particular case [Troy 1986] due to the diversity of: (a) the nature of the software and the corresponding failure data, (b) the development and validation methods adopted, (c) the failure data collection organization, (d) the aims of the analysis, etc., our experience allowed us to adopt an overall method for software reliability analysis from development to operational life. This method, relying on qualitative as well as quantitative analyses, is intended to better satisfy the supplier and customer's real needs in the field of software reliability.

The supplier is interested in (a) monitoring the different development activities, (b) respecting the delivery delay, (c) predicting when a specified level of reliability would be reached (or, at least, checking how well the software satisfies the customer's requirements). On the other hand, the customer is interested in getting a reliable software, delivered at scheduled time and cost effective. The method we have elaborated helps reaching these aims.

As discussed in several publications, data collection raises several problems in itself, such as the goals and the method of data collection, the type of data to be collected, etc., (see e.g., [Basili 1984], [Comer 1989]). Even though we agree that it is very important to plan data collection from the beginning and to use well defined methods, this topic will not be dealt with in this paper due to space limitations. We will concentrate on the processing of failure data issued from development and/or operational life. We will address the supplier as well as the customer points of view. Emphasis will be put on real-time processing of the data in a controlled environment in order to allow efficient and real-time feedback. We will introduce the successive operations that have to be performed (in the framework of the overall method) on the collected failure data in order to (a) get valid failure data, (b) manage the evolution of the development and (c) evaluate the reliability of the software. Some of the benefits brought by this method are illustrated through its application to data collected on a real software system.

The paper is composed of seven sections. Section 2 gives a general overview of the overall method we developed and recalls related work allowing to situate the present work versus the already existing material including our previous publications on the subject. Section 3 details the qualitative analyses that can be performed on the collected data whereas Sections 4 and 5 deal with the statistical processing of software data, i.e., trend analysis and model application. Section 6 is concerned with processing of real software failure data in order to highlight the advantages of the proposed method and the improvement of reliability evaluation results. Finally, Section 7 concludes.

The terminology we shall be using is that defined in [Laprie 1992a].

2- GENERAL OVERVIEW

The aims of this section are to (a) state the objectives of a reliability study, (b) give a quick overview of the overall method (details of the various steps are given in Sections 3 to 5) and (c) briefly describe the related work.

2-1- Objectives of a reliability study

The objectives of a reliability study are numerous and directly related to the considered point of view (the supplier or the customer) and the life-cycle phase concerned. They are generally expressed in terms of measures.

When the software is under development, interesting measures are:

- **(M1)** the evolution of the trend (reliability growth/ decrease) in response to the test and debugging effort in order to monitor these activities,
- **(M2)** the expected number of failures for the following periods of time so as to plan the test effort and thus the testing time and the numerical importance of the test team.

When the software is in operation, two types of measures are useful:

- **(M3)** from the customer's point of view: the mean time to failure or failure rate (either the instantaneous failure rate or the expected residual failure rate in operational life) so as to evaluate the reliability of the whole system (hardware and software),
- **(M4)** from the supplier's point of view: the expected number of failures among all installations (or the number of corrections to be introduced in the software) in order to estimate the maintenance effort still needed.

2-2- Overview of the overall reliability analysis and evaluation method

Figure 1 summarizes the various operations that can be performed on the collected data set and the results that may be expected from these operations.

The collected data may include foreign data (see Section 3-1). Filtering these data items is thus needed in order to keep only data related to the software. Depending on the objectives of the study, reliability analysis may be performed using this data set or sub-sets of it obtained through data partition.

Several partitions can be accomplished, according to (a) failure severity, (b) fault location, (c) the life-cycle phase, (d) type of installation (when the software is installed on several systems with different operational profiles for instance), etc. These partitions allow more detailed analyses leading to more elaborate results. Two types of analysis may be performed on the whole data set or/and on the derived sub-sets: either statistical processing of the data for trend analysis and reliability growth model application or/and descriptive analysis. Statistical processing concerns the evolution with respect to time of reliability measures such as the time to failure, the number of failures over time, whereas descriptive analyses relate to other statistics such as (a) number of faults per kilo-lines of source code, (b)

number of faults per release, or (c) combined analyses such as fault location and failure severity.

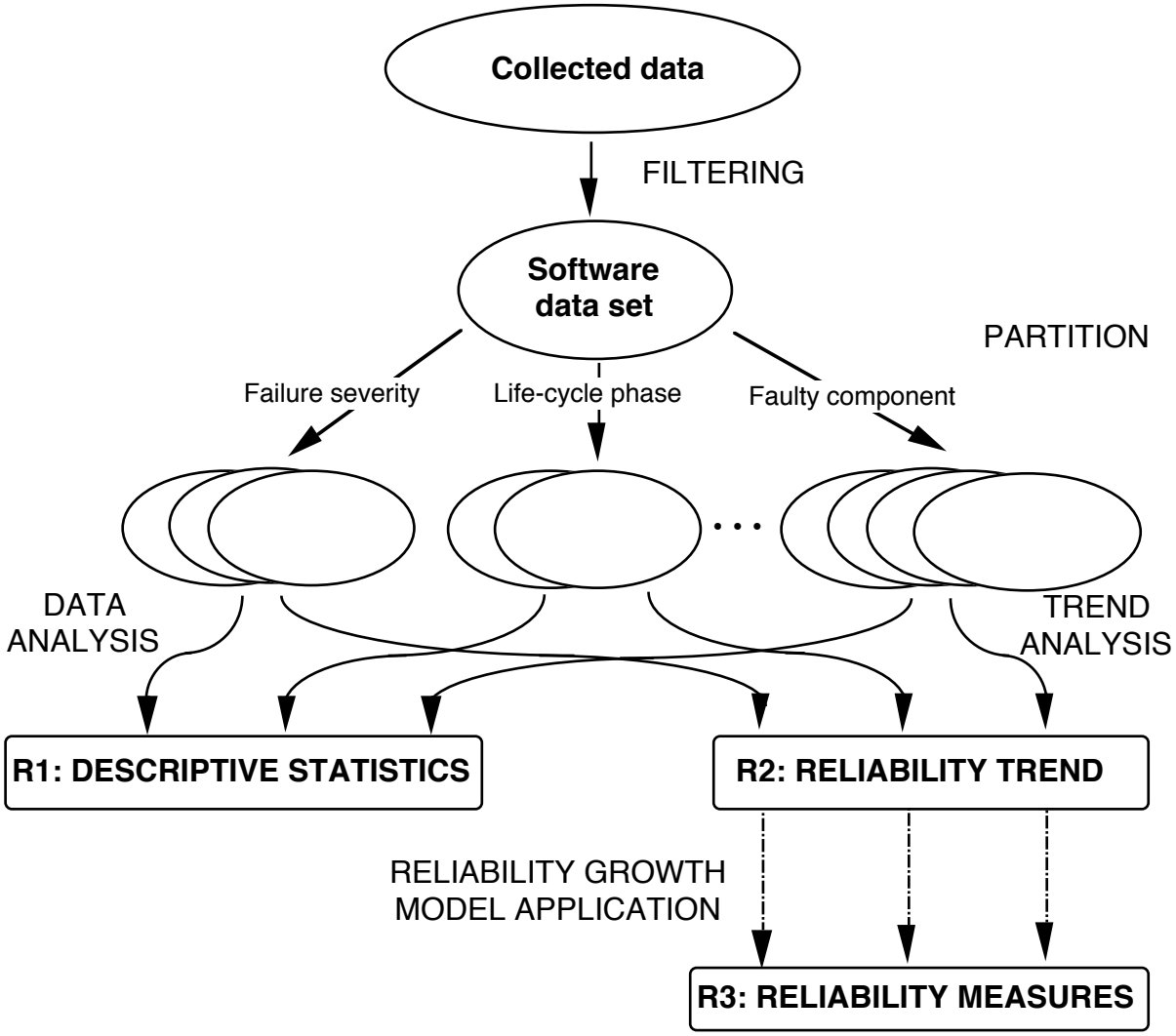


Figure 1: Different steps in reliability analysis and evaluation.

Measure (M1) stated above is reached by means of trend analyses (results R2 of Figure 1) which constitute a major tool for monitoring the development process, they allow appreciating the efficiency of testing activities. Measures (M2), (M3) and (M4) are reached through application of one or several reliability growth models (results R3 of Figure 1).

Results R1 of Figure 1 are not directly related to reliability objectives as stated above, their aim is mainly to improve knowledge about the software and the corresponding failure data. Results R2 relative to trend analyses may constitute the totality of the results of a reliability study if the objectives are to monitor development progress. They are also of great help for reliability growth models to give better estimations (results R3) since these models can be applied to data displaying trend in accordance with their assumptions rather than blindly.

Operations concerning data filtering, partition and analyses are specific to (a) the system under study, (b) the way data have been collected, (c) the purpose and the structure of

the software, (d) the objectives of the analysis, etc. These operations are thus application dependent, hence, we will merely outline the aims of these steps in Section 3. Trend analyses and model application are more systematic (i.e., less application dependent); therefore, a section is devoted to each of them.

2-3- Related work

Many papers consider the different operations introduced above. Few, however, give an overall approach and examples of practical application. Examples of descriptive analyses are to be found in e.g., [Iyer 1982], [Basili 1984] or [Troy 1985]. Trend tests are studied at a theoretical level in [Cox 66], [Ascher 1984], [Gaudoin 1990]. More general results concerning trend tests and their use within software reliability analyses are to be found among our previous publications, see for instance [Kanoun 1991a]. The definition and application of reliability growth models are discussed in many publications, see e.g., [Musa 1987]. Application examples on concrete cases are far rarer, because results are often confidential; we have however interesting examples of application of reliability growth models in [Currit 1986], [Musa 1987], [Lyu 1992] for example.

In this paper, we synthesize our experience concerning processing of software failure data. Applications to real software systems following the main steps of the overall method can be found in [Kanoun 1987], [Kaâniche 1990], [Kanoun 1991a] for instance. The presentation of these main steps is given in [Kanoun 1988] where the role of trend analysis during validation was not emphasized, this latter has been published in [Kanoun 1991b]. The features of a tool, SoRel, supporting this method, are described in [Kanoun 1993b]. This paper is intended to give a more general view about the method including our last investigations in the field. It is worth noting that most of the results concerning the application considered in Section 6 have not been published yet.

3- DATA FILTERING, PARTITION AND ANALYSIS

All these successive operations may not be needed for all reliability studies. For instance, if the collected data is entered in a database and checked as it is entered, filtering is not needed. Data partition and analysis are performed or not depending on the level of detail looked for.

3-1- Data filtering

Usually the collected data include reports corresponding to actual software failures or to errors detected during software inspection as well as extraneous data such as: false trouble reports (e.g., inconsistent data or hardware failures) or duplicated data (i.e., multiple reporting of the same failure). It is worth noting that multiple manifestations of the same fault (usually called rediscoveries) have to be kept since they correspond to different failures occurring on the same installation or not (since they affect the observed reliability). Filtering of the raw data is thus needed in order to keep only those corresponding to (a) genuine software faults, i.e., faults activated during software execution (software failure data) and/or (b) faults detected during design or code inspection.

The data filtering activity is time-consuming and cumbersome. It necessitates precise knowledge about the system and the software as well as interview of the persons involved in software testing and in data collection. When the trouble reports are entered in a database, a part of the filtering activity can be automated as in the CREDO system for instance [Cannon 1991, Chap. 7]. From our experience [Kaâniche 1990] as well those reported in [Basili 1984], [Levendel 1990], the ratio between filtered data and raw data ranges around 50%. This step is essential since the reliability analysis is carried out on the filtered data; it has to be fulfilled carefully.

Filtering leads to retain two categories of reports: failure reports (FR) issued from software execution (either from development or operational life) and trouble reports (TR) issued from design and/or code inspection[1]. FRs (resp. TRs) allow creation of files containing data in the form of times *to failures* or *number of failures per unit of time* (resp. *number of troubles reported per unit of time*). The choice between one form or the other may be guided by the objective of the reliability study (development follow up, maintenance planning or reliability evaluation) and the life cycle phase concerned by the study. The use of data in the form of "number of failures per unit of time" reduces the impact of very local fluctuations on software reliability evaluation. The unit of time is function of the type of system usage as well as the number of failures occurring during the considered units of time and it may be different for different phases. The subsequent data processing will be carried out on the retained data sets or sub-sets derived from data partition.

3-2- Data partition

Data partition into sub-sets is needed whenever a more complete analysis is required. The most common partitions concern failure severity and software faulty components (i.e., fault location). Other partitions may also be carried out according for instance to failure occurrence conditions or to the type of hardware installation when the software is in operation with several copies installed on different systems.

Applying reliability growth models to the most severe failures allows for example evaluation of the software failure rate corresponding to the most critical behavior. This failure rate is generally more significant than the software overall failure rate which may also incorporate the failure rate relative to failures that do not have major impact on system behavior [Kanoun 1987]. Concerning partition according to the fault location, evaluation of the failure rate of the various components allows (a) weighting the influence of each one on the whole failure rate and (b) identification of the most (un)reliable ones as done in Section 6.

3-3- Descriptive analyses

Descriptive analyses consist in making syntheses of the observed phenomena in the form of control charts, tables or pies in order to identify the most significant ones. It may

---

[1] Usually only FRs are dealt with. However, due to the importance of design and code inspection in the software debugging process, we recommend to process TRs recorded during this phase in the same manner as FRs, in order to manage the inspection phase as well [Kanoun 1993a].

consist of simple analyses such as for example: failure distribution among software components (new, modified, re-used), the fault typology, or combined analysis such as the relationship between (a) failure occurrence conditions and criticality or (b) fault location and failure criticality or (c) number of faults in the components and the component size, etc. These statistics are very useful, and are commonly used by the companies and organizations [Grady 1987], [Ross 1989], [Valette 1988]: accumulation and analysis of information about several projects, products and releases allows them to (a) have better insights into their products and (b) establish links between their products and the process of developing them.

4- TREND ANALYSIS

Reliability growth can be analyzed through the use of trend tests: these tests give a better insight into the evolution of the reliability. Only the most used and significant trend tests are presented in this section, more complete reviews may be found in [Ascher 1984] or [Gaudoin 1990]. The presentation of trend tests is followed by a discussion on how they can be used in real situations. Some typical results that can be drawn from trend analysis are discussed in the last sub-section.

4-1- Reliability growth characterization

Regarding the time to failure, a natural definition of reliability growth is that the successive times to failure tend to become larger. Considering the number of failures over time, reliability growth is characterized by the fact that the increase in the expected number of failures tends to become lower; which means that the curve representing the cumulative number of failures is concave, or, equivalently, that the failure intensity is non-increasing. An alternative definition allowing for local fluctuations is that the expected number of failures in any initial interval is no lower than the expected number of failures in any interval of the same length occurring later: this property is known as the *subadditive property* (see e.g. [Hollander 1974]).

A graphical interpretation of the subadditive property has been derived in [Kanoun 1993a] as follows. Let H(x) denote the cumulative number of failures, C_t the portion of the curve between 0 and t and L_t the line joining the two ending points of C_t, i.e., the chord from the origin to point (t, H(t)) of C_t (Figure 2). Let $\mathcal{A}[C_t]$ denote the area delimited by C_t and the coordinate axes and $\mathcal{A}[L_t]$ denote the area delimited by L_t and the coordinate axes. The subadditive property can be stated as: H(x) is subadditive over [0,T] if $\mathcal{A}[C_t] - \mathcal{A}[L_t] \geq 0$ for all
$t \in [0,T]$. Figure 2 displays an interesting case: C_t is subadditive over [0,T], denoting a global reliability growth on [0,T] despite local fluctuations indicated by the changes in the concavity.

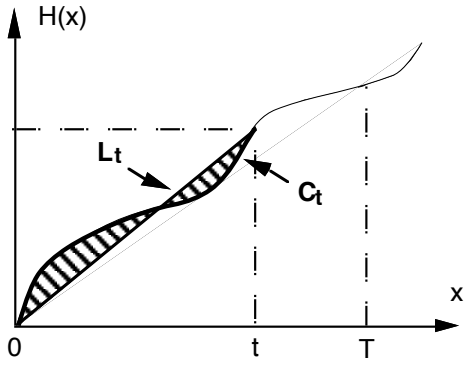


Figure 2: Graphical interpretation of the subadditive property

From what precedes subadditivity can be seen as a generalization of the concavity: a function which is concave is subadditive but the converse is not true.

4-2- Trend tests

There are a number of trend tests (either graphical or analytical) which can be used to help determine whether the system undergoes reliability growth or reliability decrease. Three very simple graphical tests can be used: the plots giving the evolution of the observed (a) times to failure, or (b) cumulative number of failures or (c) the number of failures per unit of time (i.e., the failure intensity). Among the existing analytical tests our choice goes to the Laplace test [Cox 1966] for several reasons stated beneath.

Detailed presentation, analysis and comparison of several analytical tests (Laplace, MIL-HDBK 187, Gnedenko, Spearman and Kendall tests) are carried out in [Gaudoin 1990] where it is shown that from a practical point of view, all these tests give similar results in their ability to detect reliability trend variation. From the optimality point of view, it is shown in this reference that the Laplace test is superior and it is recommended to use it when the Non Homogeneous Poisson Process (NHPP) assumption is used (even though its significance level is not exact and it is not possible to estimate its power). These results confirm our experience on processing real failure data: we have observed the equivalence of the results from these various tests and the superiority of the Laplace test.

The Laplace test consists in calculating the Laplace factor, u(x), for the observation period [0,x]. The expression of this factor is given in Annex A. The practical use in the context of reliability growth is:

- negative values indicate a decreasing failure intensity (i.e., reliability growth),
- positive values suggest an increasing failure intensity (i.e., reliability decrease),
- null values indicate stable reliability (reliability is neither increasing nor decreasing).

These practical values are deduced from the significance levels associated with the statistics; for example, for a significance level of 5%, values oscillating between -2 and +2 indicate stable reliability.

In addition, there is a link between the Laplace test and the subadditive property [Kanoun 1993a]: the sign of the Laplace factor when calculated from the number of failures per unit of time, indicates in fact whether the cumulative number of failures is subadditive or not (see Annex A). As a direct consequence, the Laplace factor may be used in order to identify local fluctuations which are typical of experimental data; this is made clear in the next sub-section.

4-3- Local and global trends

Let us consider the plot of the Laplace factor as indicated on Figure 3 where u(k) is evaluated step by step (at each unit of time or after each failure), ignoring for the time being the comments on the figure concerning local and global trends. If we change the origin of the considered interval, and let the observation interval start at T_{L1}, u(k) becomes negative over the whole (remaining) interval. It is noteworthy that the change in the origin does not result in a simple translation: removal of failure data from 0 to T_{L1} underlines the local variations and amplifies as thus the Laplace factor variation, however the points of trend change are preserved. This topic will be illustrated in Section 6 on real data.

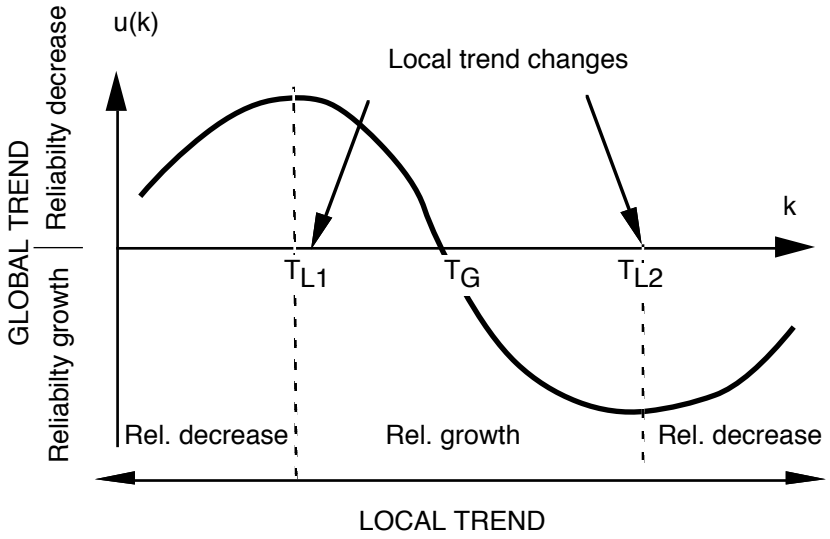


Figure 3: Laplace factor and local fluctuation

From a pragmatic point of view, i.e., using the Laplace factor as a trend indicator, the above remarks enable *local* and *global* trends to be defined as:

- negative (resp. positive) values of the Laplace factor indicate global reliability growth (resp. global reliability decrease) over an interval,
- decreasing (resp. increasing) values of the Laplace factor over a sub-interval indicate local reliability growth (resp. local reliability decrease) over this sub-interval.

Figure 4 summarizes some typical situations of trend evolution and shows the link between the cumulative number of failures, the failure intensity and the Laplace factor evolution. In real situations, we use the Laplace test to analyze the trend considering the sign

of its factor as well as the evolution of this factor with time as it enables both local and global trends to be identified "at a glance".

4-4- Typical results that can be drawn from trend analyses

Trend tests enable to draw attention to problems that might otherwise not be noticed until too late, and to solve these problems as early as possible. They cannot replace the interpretation of someone who knows the software that the data are related to, as well as the development process and the user environment. In the following, three typical situations are outlined.

Reliability decrease at the beginning of a new activity such as (a) start of a new life cycle phase, (b) changes in the test sets within the same phase, (c) addition of new users or (d) activation of the system according to a different profile of use, etc., is generally expected and is considered as a normal situation (see e.g., [Kenney 1992]). Reliability decrease may also result from regression faults. Trend tests allow this kind of behavior to be noticed. If the period of decrease seems long, one has to pay attention and, in some situations, if it keeps decreasing this can point out some problems within the software: the analysis of the reasons of this decrease as well as the nature of the activated faults is of prime importance in such situations. Such analysis may help in the decision to re-examine the corresponding piece of software.

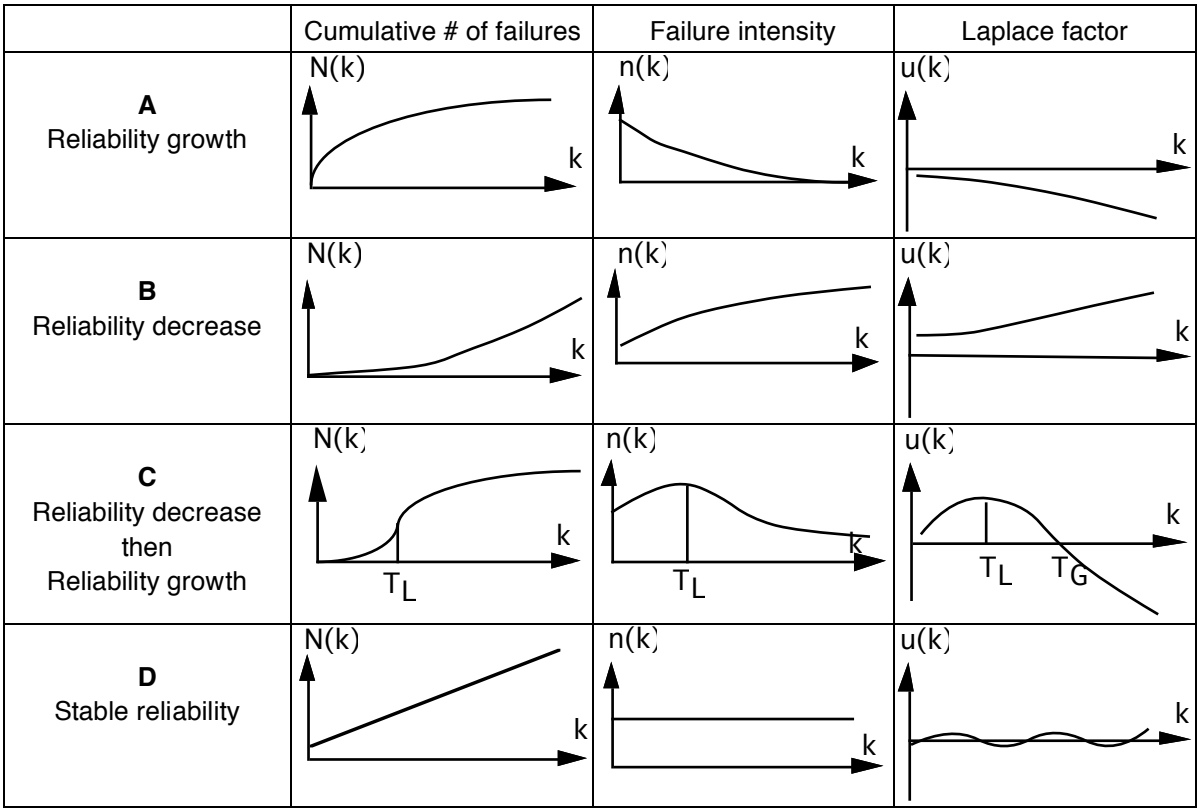


Figure 4: Cumulative number of failures, failure intensity, Laplace factor and trend evolution

Reliability growth after reliability decrease is usually welcomed since it indicates that, after first faults removal, the corresponding activity reveals less and less faults. When calendar time is used, mainly in operational life, sudden reliability growth may result from a period of time during which the system is less used or is not used at all; it may also result from the fact that some failures are not recorded. When such situation is noticed, one has to be very careful and, more important, an examination of the reasons of this sudden increase is essential.

Stable reliability with almost no failures indicates that the corresponding activity has reached a "saturation": application of the corresponding set of tests does not reveal new faults, or the corrective actions performed are of no perceptible effect on reliability. One has either to stop testing or to introduce new sets of tests or to proceed to the next phase. More generally, it is recommended to continue to apply a test set as long as it exhibits reliability growth and to stop its application when stable reliability is reached. As a consequence, in real situations, the fact that stable reliability is not reached may lead the validation team (as well as the manager) to take the decision to continue testing before software delivery since it will be more efficient to continue testing and remove faults.

5- MODEL APPLICATION

Several models have been developed during the last twenty years, and all the practitioners agree on the fact that no particular reliability growth model is superior in assessing software behavior for all software systems in any circumstances. More recently, methods for improving model performance have been developed. Among these methods, we have the linear combination model approach [Lyu 1992], the recalibration technique [Brocklehurst 1992] and our approach based on trend analysis, first used in [Kanoun 87], [Kanoun 1988] and [Kanoun 1991a] and presented in this paper.

Our objectives in this section are to (a) analyze the relevance of reliability growth models in software reliability assessment according to the life-cycle phase considered and (b) show how to use trend test results and reliability growth models in combination in order to improve the model performance. We will not present the fifty already existing reliability models which can be found in the literature neither address the problems of parameter estimation and validation criteria. These items may be found for example in [Musa 1987] or [Xie 1991].

However, in order to ease the comprehension of Section 6, a brief description of the models that are used is given in Annex B.

5-1- Relevance of reliability growth models

Application of reliability growth models to a particular real-world software enables to (a) evaluate software reliability during validation, in order to manage the delays and ensure that the software meets its reliability requirements, and (b) assess the software reliability in operation, in order to estimate the overall system dependability. Estimation of the number of failures that will occur the next period of time is useful in planning maintenance effort;

during development, due to the frequent changes in the environment of use, one can only expect very short term estimations.

The relevance of the measures obtained from reliability growth model application varies with the considered life-cycle phase, for instance:

- application of reliability growth models during early stages of the validation is not convincing: when the observed times to failure are of the order of magnitude of minutes or hours, the predictions for the mean time to failure performed from such data can hardly exceed minutes or hours… which is so distant of any expected reasonable reliability and is thus not very helpful; in this case, software validation should be guided by trend analyses as shown in the previous section,
- when the program under validation becomes more reliable, the times to failure may be large and the application of reliability growth models is more convincing particularly when the software is activated under operational profile,
- when the software is in operation, on multiple installations, the results are usually of high relevance since we have true statistical failure data (see e.g. [Adams 1984], [Kanoun 1987] for examples of such studies).

5-2- Reliability growth models and trend analysis

Applying blindly reliability growth models may lead to non realistic results when the trend displayed by the data differs from the one assumed by the model. However, if they are applied to data displaying trend in accordance with their assumptions, results may be improved considerably as shown in our previous publications. This difficulty results from the fact that the already existing reliability growth models only allow two types of behavior to be modeled: (a) decreasing failure intensity (i.e. reliability growth) or (b) increasing failure intensity prior to undergoing decreasing failure intensity. Thanks to trend analyses, failure data can be partitioned according to their trend and reliability growth models can be selected as follows:

- in case of reliability growth (situation A of Figure 4), most of the existing reliability growth models can be applied,
- when the failure data exhibit reliability decrease followed by reliability growth (situation C of Figure 4), an S-Shaped model [Ohba 1984] can be applied,
- when stable reliability is noticed (situation D of Figure 4), a constant failure intensity model can be applied (HPP model): reliability growth models are in fact not needed.

The idea of partitioning data into subsets according to trend has already been considered in [Tohma 1989] but in an empirical way: the Laplace test tells where to do it.

5-3- How to use reliability growth models according to trend in real time?

Models can be applied either in a retrodictive way (the aim is to reproduce the observed behavior based on the observed failure data) or in a predictive way (the aim is then to predict future behavior based on the observed failure data). In a predictive situation,

statements have to be made regarding the future reliability of a software and we can only make use of the information available at that time. A trend test carried out on the available data helps in choosing the reliability growth model(s) to be applied and the subset of data to which this (or these) model(s) will be applied.

The models are applied as long as the environmental conditions remain significantly unchanged (no major changes in the testing strategy or in the specifications, no new system installation with different operational profile…). In fact even in these situations, reliability decrease may be noticed. Initially, one can consider that it is due to a local random fluctuation and that reliability will increase some time in the near future: predictions are still made without partitioning data. If reliability keeps decreasing, one has to find out why and new predictions may be made by partitioning data into subsets according to the new trend displayed by the data.

If a significant change in the development or operational conditions takes place, great care is needed since local reliability trend changes may result, leading to erroneous predictions:

- if there is insufficient evidence that a different phase in the program's reliability evolution has been reached, application of reliability growth models can be continued,
- if there is an obvious reliability decrease, reliability growth models application has to be stopped until a new reliability growth period is reached again; the observed failure data have to be partitioned according to the new trend.

6- CASE STUDY

TROPICO R-4096 is an Electronic Switching System (ESS) allowing connection of 4096 subscribers[2]. The data regarding software faults detected and removed are recorded in appropriate failure reports (FR). Collected data have been filtered as they were entered in the database: the 211 FRs we received from TELEBRAS-CPqD, the Brazilian telecommunication company, correspond to genuine software faults that have been removed. These 211 FRs have been recorded over 32 units of time including the end of validation (8 units of time, 77 FRs) and the beginning of operation (24 units of time, 134 FRs) during which 42 ESSs were installed progressively.

Due to space limitation, we cannot illustrate all the different aspects presented in the previous sections. We will limit ourselves to the following: we will first analyze the trend for the whole data set and then we concentrate on operational reliability. Since several functions are fulfilled by the software, we will consider the associated components in order to evaluate their reliability as well. Maintenance planning will then be addressed.

---

[2] The reliability of its predecessor, the TROPICO R-1500, has been studied in [Kanoun 1991a]

6-1-Software decomposition

The software is decomposed into four components corresponding to the four main functions that are: (a) **telephony**, including all modules providing switching, (b) **defense**, including all on-line testing and reconfiguration mechanisms, (c) **interface**, including all interfaces with local devices (memories, terminals, alarms, etc.), and (d) **management**, including programs allowing communication with external devices.

For this purpose, the failure data is partitioned into four sub-sets according to these components. Figure 5 gives the size of the various components as well as the number of corrections carried out on each of them during operational life. It is worth noting that the sum of the number of corrections in operation (146) is higher than the number of corrections performed on the whole software in operation (134). This is due to the fact that 8 failures led to modify more than one component: these components are thus not totally independent, however dependent faults amount only to 6%.

	Size (kilo-bytes)	# corrections
Telephony	75	41
Defense	101	47
Interface	112	40
Management	44	18
Total	332	146

Figure 5: Size and number of corrections in operation for the various components

6-2- Trend analysis

Figure 6 plots the Laplace factor for the whole data set when considering all the installed ESSs. Between time units 4 and 6 reliability decrease took place as a result of changes in the nature of tests within the validation phase: new parts of programs have been activated. The software has been put into operation before observing a noticeable reliability growth. It can be seen that reliability growth took place only during operational life. Evaluating the Laplace factor for FRs relative only to operation leads to Figure 7b. Figure 7b shows reliability fluctuation from time unit 14 up to 24 which was not evident when considering the whole data set. This confirms the fact that removal of a part of the data corresponding to reliability decrease (i.e., positive Laplace factor) leads to (a) negative values of Laplace factor and (b) amplifies the variation in the trend (see § 4-3). This fluctuation is mainly due to the installation of 40 ESSs during this period. Note that on Figure 7a, the considered unit of time has been divided by 4 in order to obtain more precise information.

Taking into account the number of installations during operation and considering an average system (i.e., dividing the observed failure intensity by the number of installations in service during the corresponding unit of time) leads to Figure 8. It is worth noting that the curve of Figure 8 is smooth compared to that of Figure 7b. Reliability fluctuation between 14 and 17 results from the introduction of 6 ESSs. Notice that reliability growth tends to be regular from time unit 17. This behavior is also observed for the four components as shown in Figure 9 where the Laplace factor is displayed for the various components (considering an

average system also). Figure 9 shows that all the components exercised reliability fluctuation between 14 and 16 and that Management was more sensible to this fluctuation.

6-3- Reliability growth model application

We will first adopt the customer viewpoint by addressing measure M3 of § 2-1, which leads to estimate the residual failure rate of the software in operation. This will be achieved through the application of the Hyperexponential model to data corresponding to an average system. Then, we will adopt the supplier viewpoint by addressing measure M4 of § 2-1, which leads to estimate the number of failures expected to occur during the next period of time on all the installed systems in order to plan maintenance.

6-3-1- Software and component residual failure rates

The results of the trend analyses evidence regularity over the last 15 units of time (Figures 7 and 8). These results guide the model application. The residual failure rate of the software is evaluated from the Hyperexponential model which will be applied to the whole software and to the components using failure data observed during the last 15 units of time.

Results of application of the Hyperexponential model to each component separately are displayed in Figure 10 which gives the observed and the estimated failure intensities. Notice that the failure intensity evolution is quite different: reliability was improved faster for Telephony and Management, which also seem to be the most unreliable at the beginning.

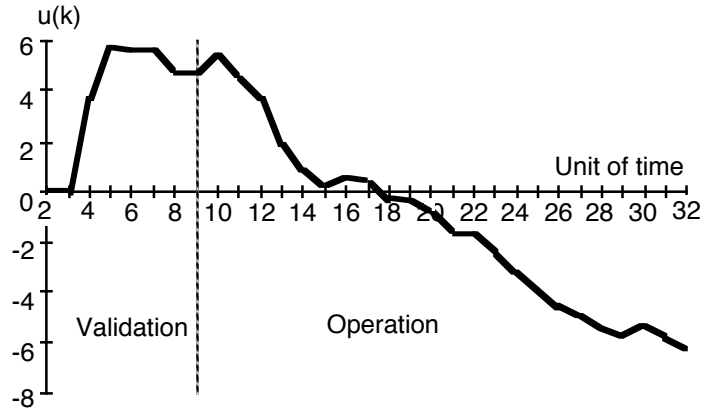


Figure 6: Laplace factor for all the installed systems and for the whole data set

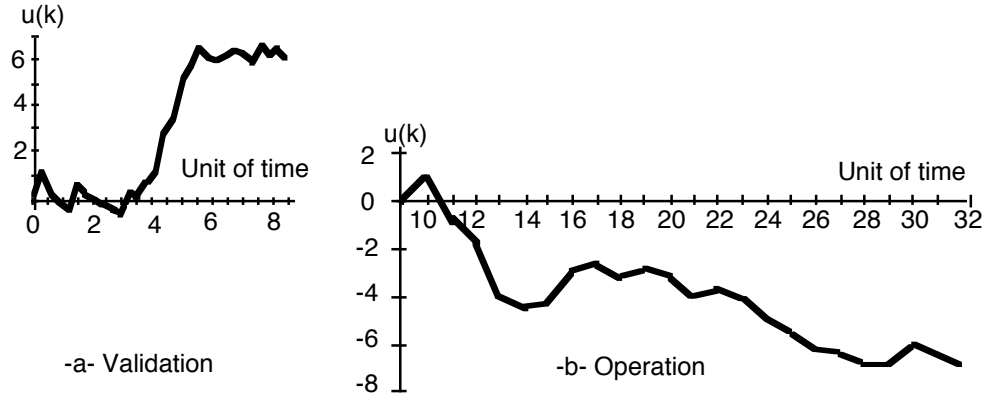


Figure 7: Laplace factors for all the installed systems for validation and operational phases

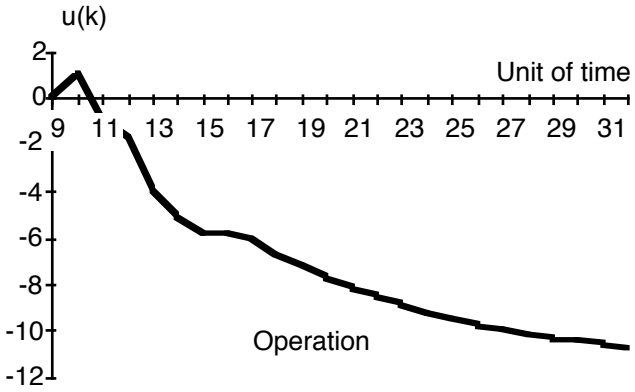


Figure 8: Laplace factors for an average system

Defense and Interface had lower failure intensity at the beginning and since they exercised reduced improvement they become less reliable at the end. Interface seems to have been debugged later leading to a greater failure intensity at the end of the considered period. These results are confirmed by the values of the residual failure rates issued from model application and given in Figure 11: Interface has the highest failure rate whereas Telephony has the lowest one.

The Interface component deserves the following comment: reliability decrease at time units 19 and 30 showed in Figure 10 is not identified by the Laplace factor since they are masked by the global reliability growth.

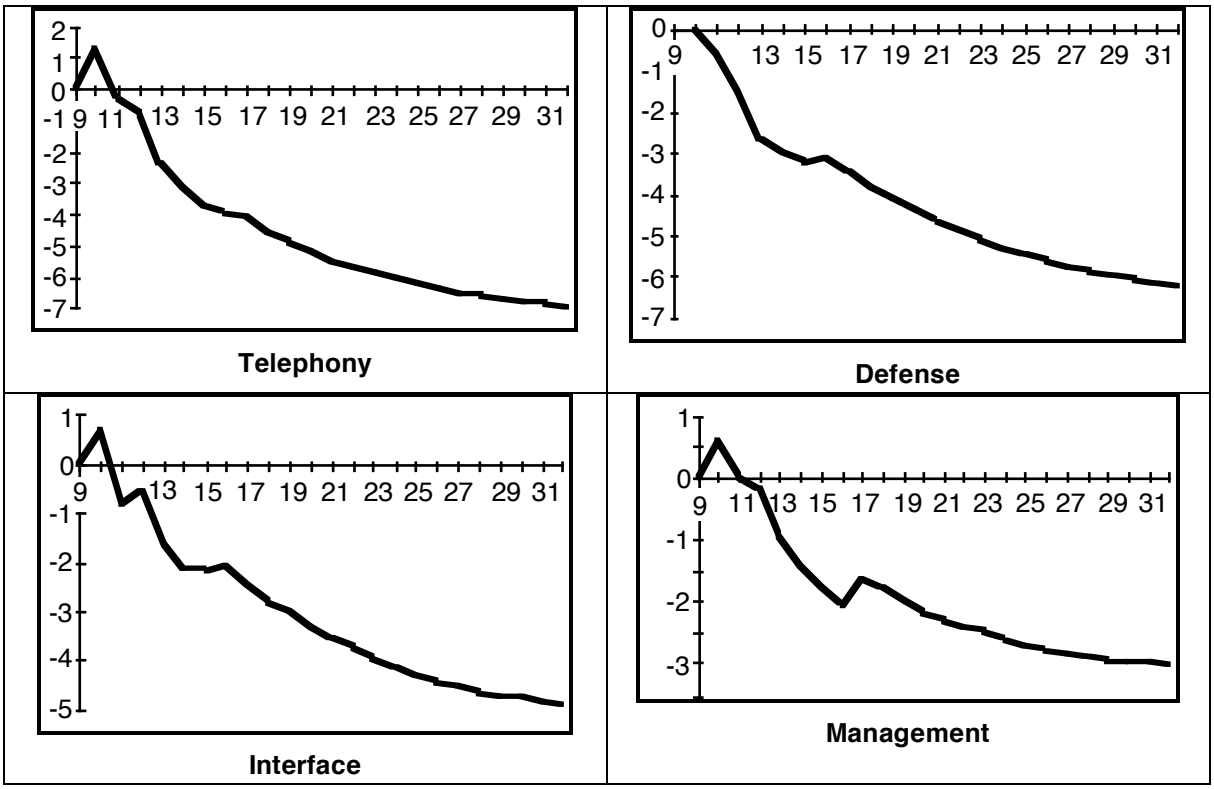


Figure 9: Laplace factor for software components

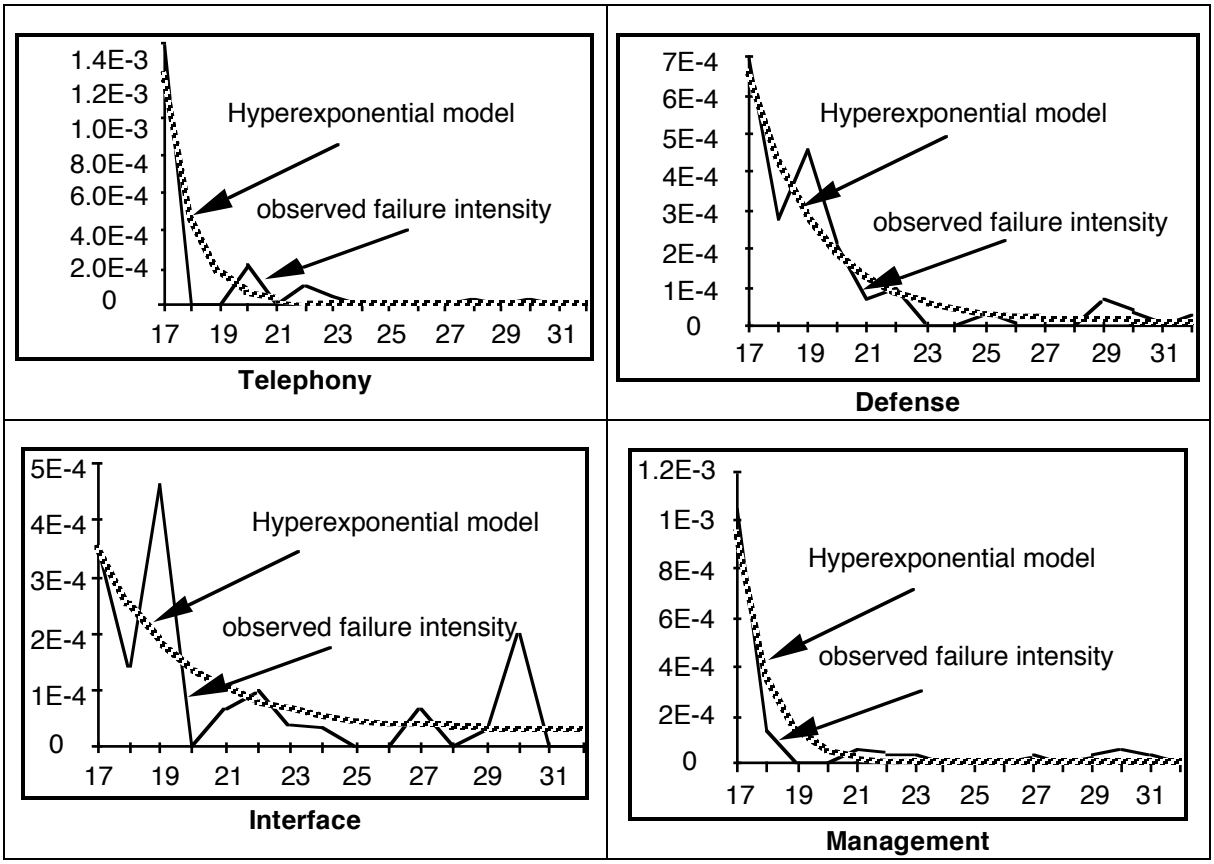


Figure 10: Failure intensities observed and estimated by the Hyperexponential model

Telephony	$2.85\ 10^{-6}$/h
Defense	$6.47\ 10^{-6}$/h
Interface	$2.76\ 10^{-5}$/h
Management	$4.79\ 10^{-6}$/h
Sum	$4.17\ 10^{-5}$/h

Figure 11: Residual failure rates evaluated by the Hyperexponential model

Figure 12 gives the results obtained from (a) direct application of the Hyperexponential model to the whole data set and (b) the sum of the failure intensities of Figure 10. This figure deserves the following remarks:

- evaluation of the whole failure intensity masks the real evolution of the failure intensities of the various components,
- direct application and summation lead to very close results, however the failure intensity obtained by summation is slightly higher, which is due to the dependency of the various components and to the precision of the results,
- the residual failure rate derived from direct application of the Hyperexponential model is $4.53\ 10^{-5}$/h which is to be compared to that obtained by summing over the residual failure rates of the four components ($4.17\ 10^{-5}$/h), these results are coherent.

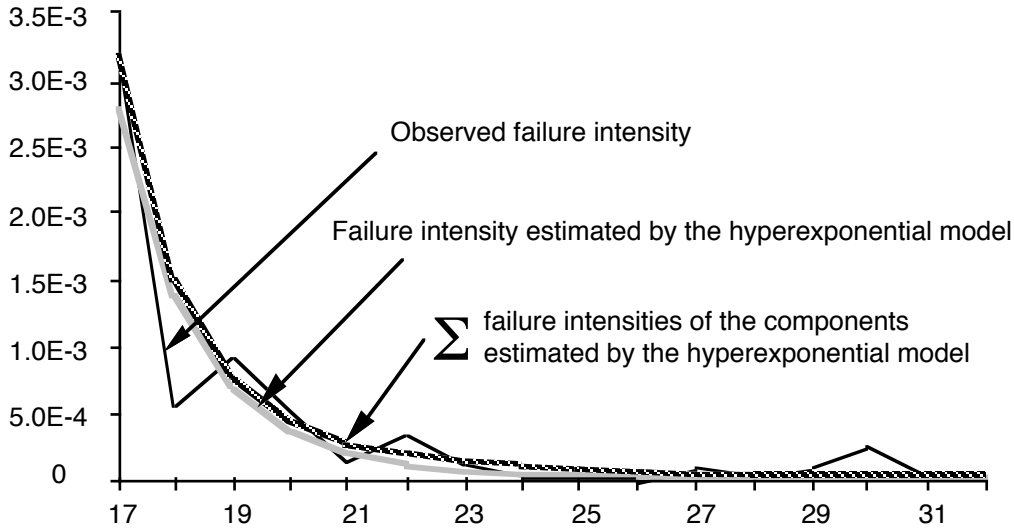


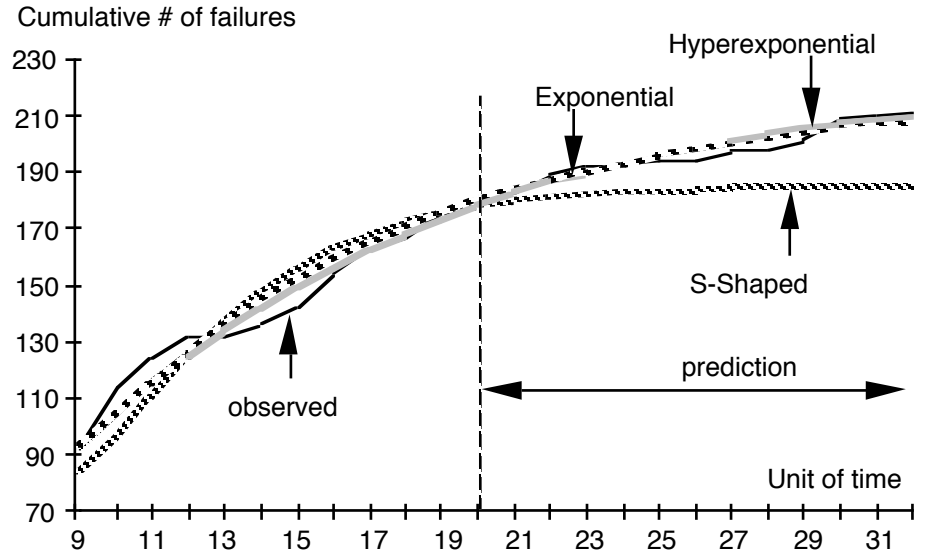
Figure 12: Failure intensities observed and estimated by the Hyperexponential model

This failure rate may be regarded as high when compared to other systems, however this has to be moderated by the fact that the severity of failures are not distinguished, it includes failures with minor consequences as well.

6-3-2- Maintenance planning

In the previous section, we adopted the customer point of view and we evaluated the failure intensity and the residual failure rate in operation for an average system. We are now addressing the supplier (more precisely the maintainer) point of view and we will use the reliability growth models in order to help maintenance decisions. For this purpose, we consider data collected from all the installed systems.

Figures 6 and 7 indicate global reliability growth over the operational life and suggest application of models with decreasing failure intensity: we will use the Hyperexponential and the Exponential models for data pertaining to operation. However, the results of an S-Shaped model will be presented in order to allow for comparison and to highlight the advantage of using trend analysis prior to model application. We will use failure data observed during units of time 9 to 19 in order to predict the number of failures that will occur during the rest of the observation period. Results are displayed in Figure 13: as expected, the Hyperexponential and the Exponential models give good results whereas the S-Shaped model is very optimistic. The Hyperexponential and the Exponential models predict respectively 37 and 33 failures for period 20-32 whereas the S-Shaped model predicts only 9 (these values are to be compared to 34 which is the observed number of failures during this period).

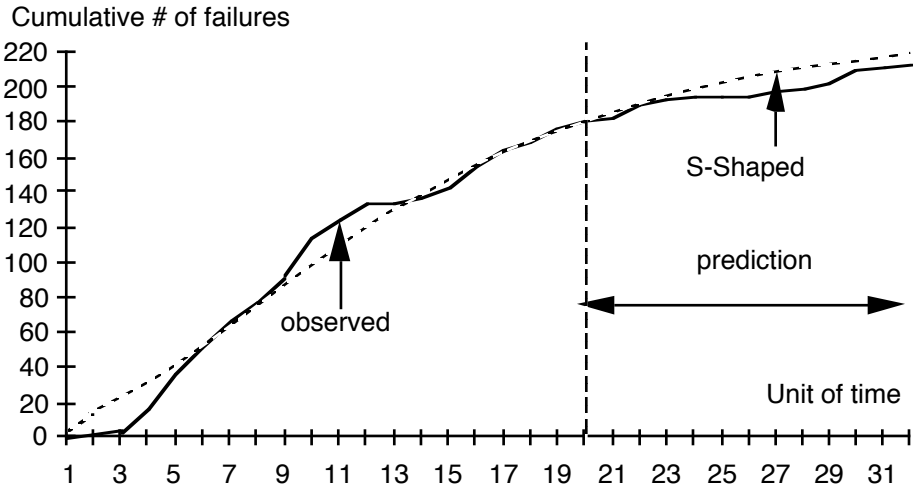


# failures	Exponential	Hyperexponential	S-Shaped	Observed
from 20 to 32	33	37	9	34

Figure 13: Cumulative number of failures observed and estimated by the models using operational data set

However, Figure 6 suggests application of the S-Shaped model from the beginning in order to include the trend change (around 5 to 9) which is more in kept with its assumptions. Results are displayed in Figure 14: predictions improvement is significant.

What precedes shows how much utilization of trend test results can improve the predictions; moreover, several models based on different assumptions applied on periods of time which may be different (one is a subset of the other) give equivalent results.



# failures	S-Shaped	Observed
from 20 to 32	40	34

Figure 14: Cumulative number of failures estimated by the S-Shaped model using the whole data set

7- CONCLUSION

We have presented a method enabling analysis and evaluation of software reliability. We have presented the various steps in analyzing data collection on the software and put emphasis on their aims and the results that can be expected from each step. Application to failure data collected on an ESS highlighted the benefits of such analyses. This method has already proved its worth to evaluate reliability of several real-life software systems, the characteristics of which are given in Figure 15. It has evolved all along our experience and we think that it is now mature enough to be integrated into the software quality control process allowing thus for quantified aspects.

System	Languages	Volume	Observation	Phases	# Systems	# FR and/or CR
E10-B	Assembler	100 k-bytes	3 years	Val. / Op.	1400	58 FR / 136 CR
TROPICO-R 1500	Assembler	300 k-bytes	27 months	Val. / Op.	15	461 FR/CR
TROPICO-R 4096	Assembler	350 k-bytes	32 months	Val. / Op.	42	211 FR/CR
Telecommunication Equipment	PLM-86	$5\ 10^5$ inst.	16 months	Val.	4	2150 FR

Work station	various	--	4 years	Op.	1	414 FR

FR: failure report CR: correction report Val.: validation Op.: operation

Figure 15: Characteristics of some real-life software systems whose reliability has been evaluated using the method presented in this paper

ANNEX A: LAPLACE FACTOR

The Laplace test [Cox 1966] consists of calculating the Laplace factor, u(T), for the observation period [0, T]. When expressed in terms of times to failure, the Laplace factor is:

$$u(T) = \textbf{Erreur !} \qquad \text{(A1)}$$

where: θ_j is the time to failure j counted from system restart after failure j-1

N[T] the number of failures in [0,T]

Following the method outlined in [Cox 1966], we have derived [Kanoun 1991a] the expression of the Laplace factor in terms of failures per unit of time:

$$u(k) = \textbf{Erreur !} \qquad \text{(A2)}$$

where n(i) is the number of failures during the i-th unit of time

Relation (A2) can also be written as a function of the cumulative number of failures N(k) = **Erreur !**; elementary manipulations lead to:

$$u(k) = - \textbf{Erreur !} \qquad \text{(A3)}$$

It is worth noting that the numerator of u(k) is nothing else than ($\mathcal{A}[C_k] - \mathcal{A}[L_k]$) (see § 4-1) which is the difference of areas between the curve plotting the cumulative number of failures and the chord joining the origin and the current cumulative number of failures. Therefore, testing for the sign of u(k) is indeed testing for the sign of ($\mathcal{A}[C_k] - \mathcal{A}[L_k]$). This shows that the Laplace indicator is directly related to the subadditive property.

ANNEX B: RELIABILITY GROWTH MODELS

The failure intensity of the **Hyperexponential** model [Laprie 84], [Laprie 91] is given by:

$h(t) =$ **Erreur !** with $0 \leq \omega \leq 1$, ω+**Erreur !**=1

h(t) is non-increasing with time. ζ_{inf} corresponds to the residual failure rate. This model admits as special cases:

- the stable reliability situation when (a) $\zeta_{sup} = \zeta_{inf}$, or (b) $\omega = 0$ or $\omega; \bar{} = 0$,
- a failure intensity tending asymptotically towards zero: $\zeta_{inf} = 0$.

The failure intensity of the **Exponential** model [Goel 1979] is given by:

$h(t) = a\ b\ e^{-bt}$ $a, b > 0$

h(t) is decreasing from h(0) = ab to zero.

The failure intensity of the **S-Shaped** model [Yamada 1983] is given by:

$h(t) = a\ b^2\ t\ e^{-bt}$ $a, b > 0$

h(t) is increasing and then decreasing denoting reliability decrease before reliability growth. The trend changes at point t = 1/b (which corresponds to the inflexion point of the cumulative number of failures).

Figure A1 gives the evolution of these failures intensities in time.

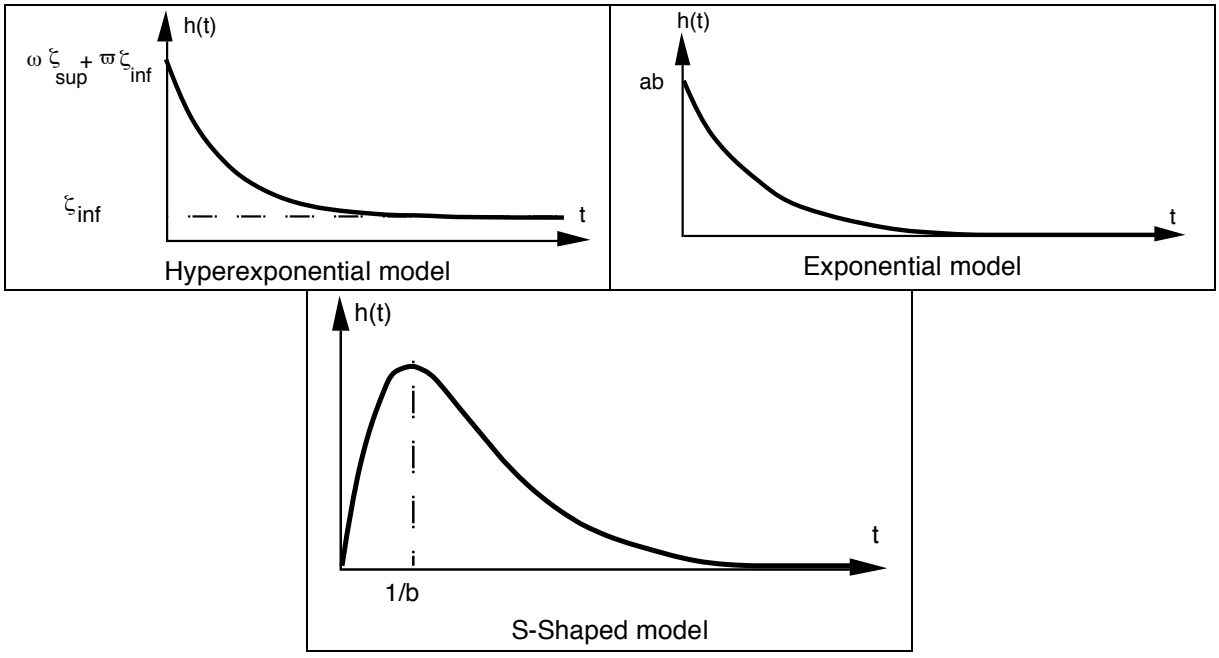


Figure A1: Typical failure intensity for the Hyperexponential, the Exponential and the S-Shaped models

Acknowledgement

The data of the TROPICO-R system have been provided by Marta Bastos Martini and Jorge Moreira de Souza, in the framework of a cooperation program between Telebras-CPqD and LAAS-CNRS.We wish also to thank Sylvain Metge for his help in processing failure data, within the framework of a student project.

REFERENCES

Adams 1984 E.N.Adams, "Optimizing preventive service of software products", IBM J. of Research and Development, vol. 28, no. 1, Jan. 1984, pp. 2-14.

Ascher 1984 H.Ascher, H.Feingold, *Repairable systems reliability: modeling, inference, misconceptions and their causes*, Lecture notes in statistics, Vol. 7, 1984.

Basili 1984 V.R.Basili, D.M.Weiss, "A methodology for collecting valid software engineering data", *IEEE Trans. on Software Engineering,* vol. 10, no. 6, Nov. 1984, pp.728-738.

Brocklehurt 1992 S.Brocklehurt, B.Littlewood, "New ways to get accurate reliability measures", *IEEE Software*, July 1992, pp. 34-42.

Cannon 1991 A.G. Cannon, A. Bendell, *Reliability data bank*, Elservier Applied Science, 1991.

Comer 1989 P.Comer, "Software data collection and the software data library", Proc. 6th *EUREDATA Conference*, Siena, Italy, March 1989, pp. 824-839.

Cox 1966 D.R.Cox, P.A.W.Lewis, *The statistical analysis of series of events*, London, Chapman & Hall, 1966.

Currit 1986 P.A. Currit, M. Dyer, H.D. Mills, "Certifying the reliability of software", *IEEE Trans.on Software Engineering*, vol. SE-12, no. 1, Jan. 1986, pp. 3-11.

Hollander 1974 M.Hollander, F.Proschan, "A test for superadditivity for the mean value function of a Non Homogeneous Poisson Process", *Stoch. Proc. and their applications*, vol. 2, 1974, pp. 195-209.

Gaudoin 1990 O.Gaudoin, "Statistical tools for software reliability evaluation", Phd thesis, Joseph Fourier Univ. Grenoble I, Dec. 1990, in French.

Goel 1979 A. L.Goel, K.Okumoto, "Time Dependent Error-detection Rate Model for Software and other Performance measures", *IEEE Trans. on Reliability*, vol R-28, no. 3, 1979, pp. 206-211.

Grady 1987 R.B.Grady, D.R.Caswell, *Software metrics: establishing a company-wide program*, Hewlett Packard Company, Prentice Hall, Inc., 1987.

Hollander 1974 M.Hollander, F.Proschan, "A test for superadditivity for the mean value function of a Non Homogeneous Poisson Process", *Stoch. Proc. and their applications*, vol. 2, 1974, pp. 195-209.

Iyer 1982 R.K.Iyer, S.E.Butner, E.J.McCluskey, "A statistical failure/load relationship: results of a multi-computer study", *IEEE Trans. on Computers*, vol. C-31, July 1982, pp. 697-706.

Kaâniche 1990 M.Kaâniche, K.Kanoun, S.Metge, "Failure analysis and validation monitoring of a telecommunication equipment software system", *Annales des Télécommunications*, Vol. 45, no. 11-12, 1990, pp. 657-670, *in French*.

Kanoun 1987 K.Kanoun, T.Sabourin, "Software dependability of a telephone switching system", Proc. 17th *IEEE Int. Symp. on Fault-Tolerant Comp. (FTCS-17),* Pittsburgh, Pennsylvania, July, 1987, pp. 236-241.

Kanoun 1988 K.Kanoun, J.C.Laprie, T.Sabourin, "A method for software reliability growth analysis and assessment", Proc. of *Software Engineering & its Applications*, Toulouse, France, Dec. 1988, pp. 859-878.

Kanoun 1991aK.Kanoun, M.Bastos Martini, J.Moreira De Souza, "A method for software reliability analysis and prediction — application to the TROPICO-R switching System", *IEEE Trans. on Software Engineering,* vol. 17, no. 4, April 1991, pp. 334-344.

Kanoun 1991b K.Kanoun, J.C.Laprie, "The role of trend analysis in software development and validation", Proc. of *Safecomp'91,* Trondheim, Norway, 30 Oct. - 1 Nov. 1991, pp. 169-174.

Kanoun 1993aK.Kanoun, J.C.Laprie, "Software reliability trend analyses: from theoretical to practical considerations", LAAS research report, n. 93.001, Jan., 1993.

Kanoun 1993b K.Kanoun, M.Kaâniche, J.C.Laprie, S.Metge, "SoRel: a tool for software reliability analysis and evaluation from statistical failure data", accepted for publication in Proc. 23th *IEEE Int. Symp. on Fault-Tolerant Comp. (FTCS-23),* Toulouse, France, June, 1993.

Kenney 1992 G.Q.Kenney, M.A.Vouk, "Measuring the Field Quality of Wide-Distributed Commercial Software", Proc. of the *Third International Symposium on Software Reliability Engineering*, ISSRE'92, Research Triangle Park, North Carolina, Oct. 1992, pp. 351-357.

Laprie 1984 J.C. Laprie, "Dependability modeling and evaluation of hardware-and-software systems", *Proc. 2nd GI/NTG/GMR Conf. on Fault Tolerant Computing,* Bonn, Germany, Sept. 1984, pp. 202-215.

Laprie 1991 J.C.Laprie, K.Kanoun, C.Béounes, M.Kaâniche, "The KAT (Knowledge-Action-Transformation) Approach to the Modeling and Evaluation of Reliability and Availability Growth", *IEEE Trans. on Software Engineering*, vol. 17, no. 4, April 1991, pp. 370-382

Laprie 1992a J.C.Laprie, *Dependability: basic concepts and terminology,* Dependable Computing and Fault-Tolerant Systems, Vol. 5, J.-C. Laprie Editor, Springer Verlag, Wien, New York, 1992.

Laprie 1992b J.C.Laprie, "For a Product-in-a-Process Approach to Software Reliability Evaluation", Proc. of the *Third International Symposium on Software Reliability Engineering*, ISSRE'92, Research Triangle Park, North Carolina, Oct. 1992, pp. 134-139.

Levendel 1990 Y.Levendel, "Reliability analysis of large software systems: defect data modeling", *IEEE Trans. on Software Engineering*, vol. 16, no. 2, Feb. 1990, pp. 141-152.

Littlewood 1979 B.Littlewood, "How To Measure Software Reliability and How Not To", *IEEE Trans. on Reliability*, vol. R-28, no. 2, June 1979, pp. 103-110.

Lyu 1992 M.R.Lyu, A.Nikora, "Applying reliability models more effectively", *IEEE Software*, July 1992, pp. 43-52.

Musa 1987 J.D.Musa, A.Ianino, K.Okumoto, *Software reliability: Measurement, Prediction, Application*, McGraw-Hill, Singapour, 1987.

Ohba 1984 M.Ohba, S.Yamada, "S-Shaped software reliability growth models", Proc. *4th International Conference on Reliability and Maintainability*, Perros Guirec, France, 1984, pp. 430-436.

Ross 1989 N.Ross, "The collection and use of data for monitoring software projects", Measurement for software control and assurance, Edited by B.A.Kitchenham and B.Littlewood, Elsevier Applied Science, London and New York, 1989, pp. 125-154.

Tohma 1989 Y.Tohma K.Tokunaga, S.Nagase, Y.Murata, "Structural Approach to the Estimation of the Number of Residual software faults Based on the Hyper-geometric Distribution", *IEEE Trans. on Software Eng.*, vol. 15, no 3, March 1989, pp. 345-355.

Troy 1985 R.Troy, C.Baluteau, "Assessment of software quality for the Airbus A-310 Automatic Pilot", Proc. 15th *IEEE Int. Symp. on Fault-Tolerant Comp. (FTCS-15),* Ann Arbor, Michigan, June, 1985, pp. 438-443.

Troy 1986 R.Troy, Y.Romain, "A statistical methodology for the study of the software failure process and its application to the ARGOS center", *IEEE Trans. on Software Engineering*, vol. SE-12, no. 9, Sep. 1986, pp. 968-978.

Valette 1988 V.Valette, "An environment for software reliability evaluation", Proc. of *Software Engineering & its Applications*, Toulouse, France, Dec. 1988, pp. 879-897.

Xie 1991 M.Xie, *Software reliability modeling*, Word Scientific, 1991.

Yamada 1983 S.Yamada, M.Ohba, S.Osaki, "S-Shaped Reliability Growth Modeling for Software error Detection", *IEEE Trans. on Reliability*, vol. R-32, no. 5, 1983, pp. 475-478.