Issues in Predicting the Reliability of Composed Components

Judith Stafford
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213
+1-412-268-5051
jas@sei.cmu.edu

John D. McGregor
Clemson University
Department of Computer Science
Clemson, SC 29634
+1-864-656-5859
johnmc@cs.clemson.edu

ABSTRACT

Availability is one of the most frequently specified quality attributes for computerized systems and the computation of availability requires knowledge about the reliability of the system. Although much research has been devoted to software system reliability, much work remains to be done in identifying ways to predict reliability of assemblies of components. We are designing an experiment for use as a foundation for creating a reliability prediction-enabled component technology (PECT), which is to be used to produce systems that are predictably reliable by construction; in the course of that work we have recognized the need to evolve combinatorial reliability models for use in computing reliability of assemblies based on the reliabilities of constituent components. In this paper, we describe and discuss aspects of current models that need to be adapted and how they affect the design of our experiment.

1. INTRODUCTION

In this paper we discuss the prediction of reliability of component assemblies where the components are reusable assets. Rather than present a solution to the prediction problem, we present a set of issues related to currently accepted combinatorial reliability models that must be resolved in order to predict reliability of component-based software systems and we describe the skeleton of an experiment that is in its developmental stage.

Reliability is an important quality attribute. System availability, the likelihood that the system will be capable of performing a function when needed, is one of the most common quality requirements specified by commissioners of computerized systems and availability is based on system reliability. Reliability is generally defined as the probability for failure-free execution for a specified interval of time or other natural unit. A survey of reliability models by Farr may be found in Lyu's handbook [1] and Goseva-Popstojanova and Trivedi have compiled an overview of architecture-based reliability models [3].

Submitted to the 5th ICSE Workshop on Component-Based Software Engineering, Orlando, Florida, May 2002.

System[1] reliability has its origins in the realm of hardware. This heritage, and the fact that availability is a system requirement that involves both software and hardware, has strongly influence research in software reliability. System reliability was, and still is, calculated in much the same way as hardware reliability. It is widely recognized that there are fundamental differences in the causes of failure in hardware and software. For instance hardware degrades because it is used and wears but using an executable piece of software does not cause it to wear out. It might become more likely to fail over time, but that failure is not due to wear and tear on the executable.

While hardware is expected to be failure free when first used, for software, failure is generally not time dependent, but rather is dependent on the probability that a failure-causing input will be used. Software will always fail when used in the same way and all copies of a program will fail in exactly the same way; failure depends on whether a path is executed that includes the execution of a faulty piece of code or if communication among program units fails. On the other hand, that same program can be rerun in a different context and perform perfectly well with no modification to the code.

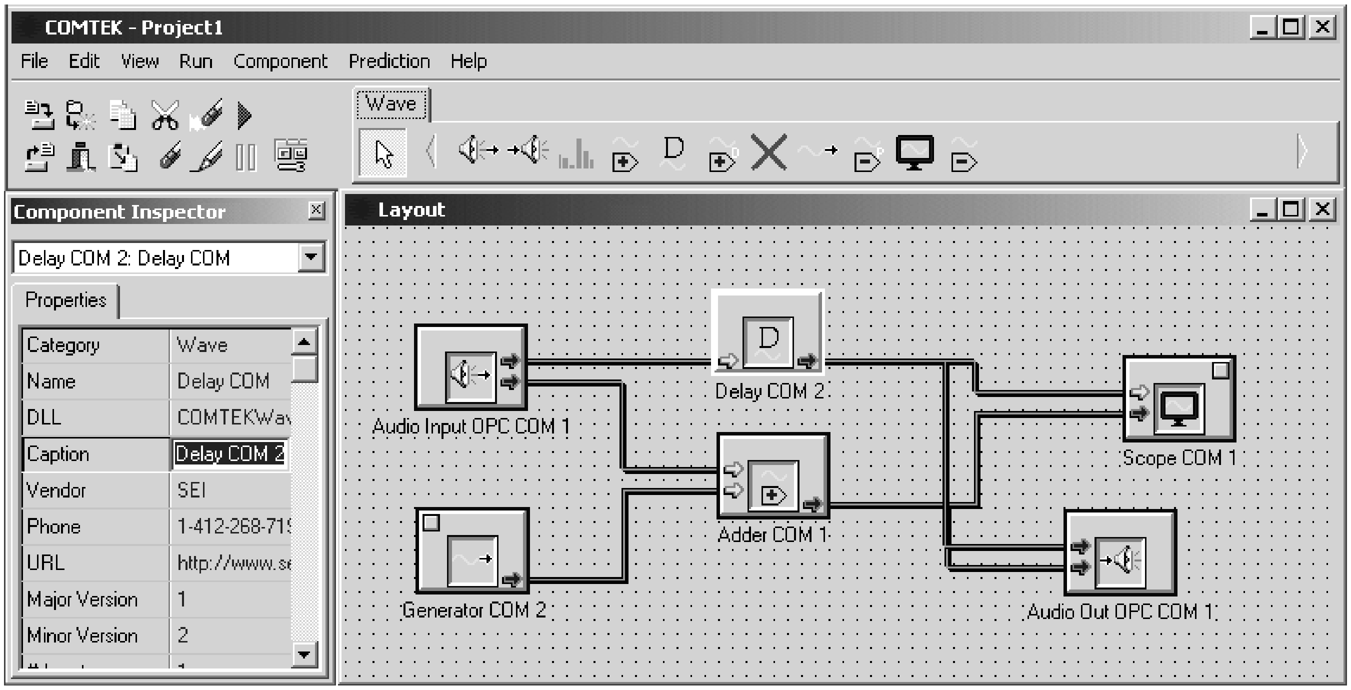
These issues are magnified when using third party components and require evolution or adaptation of traditional combinatorial reliability models, which are based on the assumption that the following three items have been determined:

– what constitutes a failure
– a common interval of time or other natural unit
– an operational profile

Issues beyond those associated with system reliability, such as these that are related to component reliability, arise because information computed by the component developer will be used in unknown contexts for unknowable reasons.

In the rest of this paper we describe our work towards developing a reliability prediction-enabled component technology. We describe and discuss issues that have broader implications for

[1] When we use the term *system* we are referring to combinations of both hardware and software components.



Category	Wave
Name	Delay COM
DLL	COMTEKWav
Caption	Delay COM 2
Vendor	SEI
Phone	1-412-268-71
URL	http://www.se
Major Version	1
Minor Version	2

compositional analysis and that should inspire discussion at the workshop.

2. Developing a Reliability Prediction-Enabled Component Technology

We are developing a reliability prediction-enabled component technology. This work is the second step in the development of a general model for creating prediction-enabled component technologies (PECTs) that is being carried out as part of the PACC[2] project at the Software Engineering Institute (SEI). The first step involved developing a latency prediction-enabled component technology, ComTek-λ, which is based on the ComTek component technology [5]. ComTek is a component technology that provides a design environment, a component model, and the runtime environment in which to execute assemblies of components. The ComTek development environment is shown in Figure 1. The menu bar at the top includes a button labeled "Prediction". At present, one is able to click on the button and select latency prediction, at which time the user will be provided a prediction of average latency for the component assembly on the palette. The result of the current experiment will support a similar prediction of reliability.

> Given a set of ComTek wave application components, predict the reliability of an assembly of these components deployed in the ComTek runtime environment.

[2] http://www.sei.cmu.edu/pacc

2.1 Defining Assembly Reliability

The reliability model chosen for this work is based on the widely used definition of software reliability that is given by Musa [6]:

> The probability of execution without failure for some specified interval of time or other natural units.

Musa defines two basic configurations of computations for which reliabilities can be calculated. The AND configuration represents a series of computations, all of which must be successful for the assembly to be successful. The formula is given by $\prod_{i=0}^{N} R_i$ where N is the number of components in the path.

The OR configuration represents subassemblies, one of which must be successful in order for the assembly to be successful. The formula is given by $R = 1 - \prod_{i=1}^{N} (1 - R_i)$ where N is the number of components in the path.

These two operators are sufficient for describing many classes of assemblies including those that will be used in our initial experiment. In fact, it will be sufficient to use the AND formula because the cyclic, pipe&filter, non-pre-emptive nature of the scheduler used in ComTek requires successful execution for all tasks for successful execution of the assembly.

Issues arise when attempting to use simple combinatorial reliability models if one considers issues related to the degradation for software over time. These models assume independence of failure and the ability to predict the rate of failure but this is not necessarily the case for assemblies of software components. Although software itself does not wear out, that is it does not degrade when it sits unused, there are environmental and assembly related conditions that can change over time and affect the ability of a given component to do its job. The term “software aging” has recently been coined by Garg et al. [2] to describe the effect of these types of software faults. Software aging is caused by situations such as memory leakage, incorrect or insufficient exception handling, and interdependent timing of events. Software aging is prevalent in component-based systems because many of the causes result from unanticipated component interactions. Research in predicting the effect of software aging on assembly reliability is a wide-open research topic.

This class of faults is outside the scope of our initial experiment therefore we will not attempt to address this issue when adapting the AND formula for use in our experiment. We will however attempt to monitor the behavior of the assembly in such a way as to capture data that might provide input to the development of such a theory.

2.2 Defining Component Reliability

When developing a PECT it is assumed that the components are black-boxes, independently deployable, and reusable components. Applying the Musa reliability combinatorics require the analyst define what failure means and determine a common interval for natural units or time that is most useful for those needing information about the system. In addition, it is necessary to have knowledge about expected system usage. Understanding how to determine each of these must be re-examined for use in the context of component composition.

Consider the issue of defining a common interval of time. When a software assembly is executing, not every constituent component is executing all the time. In fact, a given use of an assembly might not activate a particular component at all. Using the total execution time for a program as the execution time for its parts for the purpose of calculating each component’s contribution to overall reliability is not accurate and, in fact, may overstate the reliability of a component. For instance, the error-handling component in a highly reliable assembly will only execute for a very small fraction of the total execution time of the assembly and therefore the strength or weakness of its reliability matters little to the overall reliability of the system.

We are considering applying other reliability models that use “natural units” other than time. We recognize that the close tie between reliability and availability raise issues with this approach and therefore are exploring the potential for using conversion mappings from non-time based reliability measures to time. Candidates include using a path-based approach to reliability that uses individual program paths of execution, average number of invocations for a specific service, and predicted average latency.

2.2.1 Operational Profiles

Regardless of the definition, reliability is an operational attribute that is measured by execution. To empirically validate a prediction theory requires execution that is constrained by an operational profile[3]. If a component could be exhaustively tested, which in most cases is not possible, and if one could know exactly how the component was to be used, then one could accurately calculate the probability of using a given failure-causing input within a specified amount of time.

However, even in such a perfect world there would be problems for using operational profiles to assist with calculating system reliability in component-based development. Operational profiles are used to guide testing efforts and to assign reliability values to certain aspects of components. Because software components are third party composable, determining an operational profile is at least very complex and may be intractable. Creating operational profiles assumes knowledge of the context in which the component is to be used. When creating a components that are to be used in assemblies, all that can be known is that they provides certain services. The nature or identity of the users of the services is unknowable. Hamlet et al. discuss this problem in their ICSE 2001 paper [1]. They suggest supplying profile mappings in component data sheets that can be used in combination to predict overall system reliability before actual assembly. We have concerns about the scalability of this approach, but even in the event that this does turn out to be the case, their work provides good insights into problems associated with this aspect of compositional reliability prediction.

Another issue related to operational profiles is the fact that a component is typically sufficiently large that different users will use the services of the component in very different ways and, in fact, usually do not use a majority of the component’s services. This invalidates the profile and leaves the question of reliability unresolved. Given that operational profiles are crucial to computing reliability, understanding the impact that component-based development has on identification and use of operational profiles is perhaps the most pressing issue for developing compositional approaches to reliability analysis.

2.2.2 Time

Both context dependencies and software aging affect evaluation of components to support combinatorial reliability models. These issues are sketched below.

Context Dependencies and Time Intervals

Definition of time interval is assembly dependent. Recall that Musa's definition of reliability is based on some natural unit or time. Certification and testing of reliability has been assumed to happen within the context of component use. The reported reliability will be with respect to some specific interval of natural or time unit but this may have no relationship to that which is required in the deployment environment. This number must be adapted in order to reflect the components contribution to each assembly in which it is deployed.

As a simple example, Suppose there are three components A, B, and C to be used in an assembly. A executes for 10% of the time,

[3] An operational profile is the frequency distribution that describes the relative frequency with which each operation of the system is selected for use.

B executes for 60% of the time, and C executes for 30% of the time. For every 100 minutes of assembly execution A has executed 10 minutes, B has executed for 60 minutes and C has executed for 30 minutes. It is necessary to extrapolate the reliability of each component to reflect its contribution to overall reliability of the assembly, otherwise the impact of using a component of questionable reliability may overly influence the calculation. In general it is necessary to adjust reliability values over periods of time other than that supplied with the component. This will be the approach taken in the initial experiment. We will assume a linear relationship between probability of failure and time. In general this is not true but may prove sufficient. Depending on the results of our experiment we will explore more precise extrapolation functions.

Software Aging

The causes of software aging can be traced to specific types of assumptions that components make about the environment into which they will be deployed as well as coding errors. We will be simulating coding errors in our experiment and attributed the components with appropriate property value. As mentioned above, we will not be addressing issues of software aging in this experiment and will not explore issues of evaluating components for their contribution to aging at this time.

2.2.3 Defining Failure

There are many ways in which a system can fail and many ways in which a component can contribute to a given type of system failure. Thus, it might be that a system developer is interested in predicting the likelihood that the system will produce an incorrect value but one or more of the components that are being assembled define reliability only in terms of failure to continue operating. If suppliers are to provide reliability information, it would be best for them to have knowledge of the types of failure that are of interest to component users so that the appropriate test cases can be used to inform the reliability prediction. At the very least, the component reliability value must be accompanied by the definition of failure for which it applies. Exploring other solutions to this dilemma is another interesting and open research problem.

2.3 Communicating Reliability Information

The assumptions of a reliability theory impose a heavy burden on the component supplier for anticipating component usage and reliability concerns of expected users. In addition to the basic concerns, time-based reliability theories might benefit from knowledge of the rate at which failures were incurred during testing so reliability can be extrapolated based on testing data and expected component execution time within a given assembly.

Reliability values are typically reported as a single value, a percentage, which indicates the percentage of time in which the software is available to perform the requested operation. As noted above, this value is valid in the context of the operational profile that was used to structure the reliability test suite. To fully understand the reliability number, the user must understand that context.

We are considering a number of formats for reporting information on a component's data sheet. Three possible formats are:

– A single component reliability – It is assumed that the user will use the component in a pre-determined manner including invoking specific services with specific frequency. Only the single value would be communicated along with a description of the operational profile.

– A set of component reliabilities – It is assumed that the user will use the component in one of several modes. Each mode represents a separate operational profile. For example, a component may be embedded in a larger component and accessed by its API or it might be used as part of the GUI for an application and the component is accessed through its user interface. The set of values would be communicated along with a mode diagram that illustrates the major operational modes of the component. An operational profile description for each mode would also be provided.

– Path information – It is assumed that the user will use detailed path information to form their own reliability computation based on which services they will use. Knowledge of provides-requires dependencies can supports computation of more precise inter-component dependencies when a subset of the provided services are being utilized [8]. A brief description of how to compute reliabilities for a service or a mode from the path results would also be included.

3. EXPERIMENTING WITH COMTEK

Our experimental method follows the procedure used to develop the latency prediction enabled variant of ComTek [5], in which the predicted estimates are compared to empirical measurements to ascertain the adequacy of the predictions. At the time of the workshop we will be able to report those results. In this section we sketch out the plan for the experiment.

3.1 The ComTek Component Technology

ComTek has the following high-level characteristics that are relevant to our development of a reliability PECT:

– It enforces a typed pipe-and-filter architectural style.

– A fixed round-robin schedule is calculated from component input/output dependencies.

– The execution of an assembly is sequential, single-threaded, and non-preemptive.

The development of any PECT is expected to involve some degree of co-refinement between the component technology and the analytic theory. That is, bringing the assumptions of the analytic theory and the constraints imposed by the component technology in line might require adaptation to either or both of these elements of the PECT. In the case of our initial experiment the characteristics of ComTek support the assumptions of the AND reliability theory described in Subsection 2.1. The sequential pipe&filter nature of ComTek assemblies requires that all components succeed for the assembly to succeed. In addition, reasoning about input profiles is simplified by the fact that and all required inputs must be available before a component executes and we are adapting our components to fail based on the value of a randomly assigned value at each invocation of the component, thus neither the methods called, nor the data supplied, will affect the rate of failure of the components.

The latency PECT can be used to predict the average contribution each component will make to the execution of a given assembly and can be used to extrapolate assembly specific reliability values for individual components.

3.2 Seeding ComTek Components

As mentioned in the previous section, we are using a seeding approach to introduce unreliability into already existing ComTek components. Each component will be assigned a reliability value and will "fail" conditionally at the appropriate rate based on a randomly generated integer.

3.3 Creating Representative Assemblies

We will be using the Wave family of ComTek components for this experiment, primarily because this is the set of components that is latency prediction-enabled. The ComTek development environment is shown in Figure 1. The Layout window shows members of the wave family of components on the palette connected. One might be fooled into thinking that this is a specification of an assembly but, in fact, the components were lifted from the menu of component implementations that is located just above the palette, and then connected to form an assembly. At that point it is possible to play an actual CD and listen to the music as well as to view it on the scope that is available if one clicks on the little box on the upper right hand corner of the SCOPE component. Allowable connections are constrained by a naming scheme and connectors are pipes through which streams of audio data are passed.

Components can be replicated and combined as desired, within the bounds of the connection constraints imposed by the ComTek component model. Although it is not possible to define a representative set of assemblies, we will create 30 applications using the automated assembly generator that has been created by the PACC team, varying the topology and the number of components in the assembly in order to base our results on a variety of assemblies that we feel mimics that of a representative set.

3.4 Validating Reliability Prediction Theory

The experiment will include creating and executing all assemblies 100 times. The reliability of each assembly will be predicted using the AND formula, after extrapolating appropriate reliability values for the components based on each component's average contribution to the overall execution of the assembly. The reliability will be stated as the expectation for failure-free execution for a specified time period, which will vary by assembly. The assembly will be executed for that duration or until failure, whichever is shorter. For each execution the time of failure, or successful completion will be recorded.

When all data has been collected we will apply the empirical validation assessment that has been developed for this purpose within the PACC project [5].

3.5 Expected Outcomes

We reiterate that this initial experiment is not being carried out for the purposes of validating a reliability theory, but rather to explore the design of such an experiment and the development of a reliability prediction enabled component technology.

Through the exercise of designing and carrying out this experiment we expect to determine the

- information that must accompany a component when delivered from supplier to consumer
- way in which this information is adapted for use by the compositional reliability theory in a variety of contexts
- design of an experiment that can be used to validate other compositional reliability theories

Due to the design of this initial experiment we fully expect our reliability PECT to be rated highly with respect to the confidence a user could place in the reliability values computed by the PECT.

4. CLOSING THOUGHTS

There is much work remaining to be done in defining a compositional reliability theory. These issues involve understanding and defining the relationships between system reliability measures and component reliability measures, each of which depend on defining time and failure, as well as understanding usage. It is apparent that as the number of assumptions on which a compositional theory is based increase, the difficulty of employing it in component-based development also increases. Perhaps the development of a new business model for component-based software development could help simplify these problems.

Given that reliability values are typically derived from testing data, it seems reasonable to assume that the best source of reliability information is the component supplier and that functions be developed to specialize available information. However, given the issues discussed in this paper, it might be more reasonable to create a new business model that supports increased interaction between the component supplier and manufacturer that allows for negotiation related to the definition of component reliability. Perhaps component consumers would be allowed to "test drive" components before making the purchase at which time they could test a component based on the intended operational profile and determine reliability based on failures of concern to the purchaser.

5. ACKNOWLEDGMENTS

The authors thank colleagues: Scott Hissam, Mark Klein, Magnus Larsson, Gabe Moreno, Kurt Wallnau, and Bill Wood at the Software Engineering Institute and Il-Hyung Cho and Jessica Bamford at Clemson University who participated in insightful discussions. The work of Judith Stafford was supported by the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. The work of John D. McGregor was partially supported by the Software Engineering Institute and Clemson University.

6. REFERENCES

[1] W. Farr. “Software Reliability Modeling Survey.” In M.R. Lyu, (Ed.), Handbook of Software Reliability Engineering. McGraw-Hill, New York, 1996.

[2] S. Garg, A. van Moorsel, K. S. Trivedi, and K. Vaidyanathan. “A methodology for detection and estimation of software aging.” In Proceedings of the Ninth International Symposium on Software Reliability Engineering, pages 282--292, Paderborn, Germany, November 1998.

[3] K. Goseva-Popstojanova and K. S. Trivedi. “Architecture-based approach to reliability assessment of software systems.” *Performance Evaluation*, 45(2-3):179-204, 2001.

[4] D. Hamlet, D. Mason and D. Woit. “Theory of Software Reliability Based on Components.” Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada, May 2001.

[5] S.A. Hissam, G.A. Moreno, J.A. Stafford, K.C. Wallnau, "Packaging and Deploying Predictable Assembly," IFIP Working Conference on Component Deployment, Berlin, June 2002 (to appear).

[6] J. Musa. *Software Reliability Engineering*. McGraw-Hill, New York, N.Y., 1998.

[7] M. R. Lyu, Editor. Handbook of Software Reliability Engineering, McGraw-Hill, 1996.

[8] J.A. Stafford and A.L. Wolf, "Annotating Components to Support Component-Based Static Analyses of Software Systems." Proceedings of the Grace Hopper Celebration of Women in Computing 2000, Hyannis, Massachusetts, September 2000 (on CD-ROM). Also available as also available as University of Colorado Technical Report No. CU-CS-896-99.

[9] K.S.Trivedi, K. Vaidyanathan, K.Goseva-Popstojanova. "Modeling and Analysis of Software Aging and Rejuvenation", Proc. 33rd Annual Simulation Symposium, Washington D.C., Apr. 2000.