International Journal of Reliability, Quality and Safety Engineering

Dept. of Math. University of Oslo
Statistical Research Report No 5
ISSN 0806–3842 July 2010

ON COMPONENT DEPENDENCIES IN COMPOUND SOFTWARE

MONICA KRISTIANSEN *
Østfold University College
1757 Halden, Norway
monica.kristiansen@hiof.no

RUNE WINTHER
Consultant at Risikokonsult
Oslo Area, Norway
rune.winther@wintherfamily.net

BENT NATVIG
Department of Mathematics
University of Oslo, Norway
bent@math.uio.no

Received (2 July 2010)

Predicting the reliability of software systems based on a component approach is inherently difficult, in particular due to failure dependencies between the software components. Since it is practically difficult to include all component dependencies in a system's reliability calculation, a more viable approach would be to include only those dependencies that have a significant impact on the assessed system reliability. This paper starts out by defining two new concepts: data-serial and data-parallel components. These concepts are illustrated on a simple compound software, and it is shown how dependencies between data-serial and data-parallel components, as well as combinations of these, can be expressed using conditional probabilities. Secondly, this paper illustrates how the components' marginal reliabilities put direct restrictions on the components' conditional probabilities. It is also shown that the degrees of freedom are much fewer than first anticipated when it comes to conditional probabilities. At last, this paper investigates three test cases, each representing a well-known software structure, to identify possible rules for selecting the most important component dependencies. To do this, three different techniques are applied: 1) direct calculation, 2) Birnbaum's measure and 3) Principal Component Analysis (PCA). The results from the analyses clearly show that including partial dependency information may give substantial improvements in the reliability predictions, compared to assuming independence between all software components.

Keywords: Compound software; component dependencies; Birnbaum's measure; Principal Component Analysis (PCA); system reliability; probability of failure on demand (pfd).

*Corresponding author.

1. Introduction

The problem of assessing reliability of software has been a research topic for more than 30 years, and several successful methods for predicting the reliability of an individual software component based on testing have been presented [23,25]. There are, however, still no really successful methods for predicting the reliability of compound software (software systems consisting of multiple software components) based on reliability data on the system's individual software components[9,29,32].

1.1. *Motivation*

For hardware components, even in critical systems, it is accepted to base the reliability assessment on failure statistics, i.e. to measure the failure probability of the individual components and compute the system reliability on the basis of this. This is for example applied for safety instrumented systems in petroleum [11].

The characteristics of software, however, make it difficult to carry out such a reliability assessment. Software is not subject to ageing, and any failure that occurs during operation is due to faults that are inherent in the software from the beginning. Any randomness in software failure is due to randomness in the input data. It is also a fact that environments, such as hardware, operating system and user needs change over time, and that the software reliability may change over time due to these activities [3].

Furthermore, having a system consisting of several software components, explicitly requires an assessment of the software components' failure dependencies [22]. So in addition to the fact that assessing the reliability of software is inherently difficult due to the complexity of software, and that software is sensitive to changes in its usage, failure dependencies between software components is a substantial problem.

Although several approaches to construct component-based software reliability models have been proposed [10,15,20], most of these approaches tend to ignore the failure dependencies that usually exist between software components, in spite of the fact that previous research shows that this is often unrealistic [5,14,21].

In principle, a single software component's failure probability can be assessed through statistical testing. However, since critical software components usually need to have low failure probabilities [22], the number of tests required to obtain adequate confidence in these failure probabilities often becomes practically very difficult to execute. An even more difficult situation arises when the probability for simultaneous failure of several software components need to be assessed, since these probabilities are likely to be significantly smaller than single failure probabilities.
Based on the fact that:

- software components rarely fail independently, and that
- using statistical testing alone to assess the probability for software components failing simultaneously is practically impossible in most situations

the main focus has been to develop a component-based approach for assessing the reliability of compound software, which is practicable in real situations, and where failure dependencies between the software components are explicitly addressed [16,17,18,19].

This paper starts out by defining two new concepts: data-serial and data-parallel components [a]. These concepts are illustrated on a simple compound software, and it is shown how dependencies between data-serial and data-parallel components, as well as combinations of these, can be expressed using conditional probabilities. Secondly, this paper illustrates how the components' marginal reliabilities put direct restrictions on the components' conditional probabilities. It is also shown that the degrees of freedom are much fewer than first anticipated when it comes to conditional probabilities. If the components' marginal reliabilities and four of the components' conditional probabilities are known in a simple three components system, the remaining 44 conditional probabilities can be expressed using general rules of probability theory. At last, this paper investigates three test cases, each representing a well-known software structure, to identify possible rules for selecting the most important component dependencies [b]. To do this, three different techniques are applied: 1) direct calculation, 2) Birnbaum's measure and 3) Principal Component Analysis (PCA).

The results from the analyses clearly show that including partial dependency information may give substantial improvements in the reliability predictions, compared to assuming independence between all software components. However, this is only as long as the most important component dependencies are included in the reliability calculations. It is also apparent that dependencies between data-parallel components are far more important than dependencies between data-serial components. Further the analyses indicate that including only dependencies between data-parallel components may give predictions close to the system's true failure probability, as long as the dependency between the most unreliable components is included. Including only dependencies between data-serial components may however result in predictions even worse than by assuming independence between all software components.

1.2. *Notation*

In this paper, capital letters are used to denote random variables and lower case letters are used for their realizations.

To indicate the state of the i th component, a binary value x_i is assigned to component i [1].

$$x_i = \begin{cases} 0 \text{ if component } i \text{ is in the failed state} \\ 1 \text{ if component } i \text{ is in the functioning state} \end{cases} \tag{1}$$

Similarly, the binary variable ϕ denotes the state of the system.

[a] See Definitions 3 and 4 in Section 1.3.
[b] See Definition 1 in Section 1.3.

$$\phi = \begin{cases} 0 \text{ if the system is in the failed state} \\ 1 \text{ if the system is in the functioning state} \end{cases} \tag{2}$$

It is assumed that the state of the system is uniquely determined by the states of the components, i.e. $\phi = \phi(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and n is the number of components in the system. ϕ is usually called the structure function of the system. A serial structure functions if and only if all the components in the system function. The structure function of a serial structure consisting of n components is given in Equation 3.

$$\phi(\mathbf{x}) = x_1 \cdot x_2 \cdots x_n = \prod_{i=1}^{n} x_i \tag{3}$$

A parallel structure functions if and only if at least one of the components in the system functions. The structure function of a parallel structure consisting of n components is given in Equation 4.

$$\phi(\mathbf{x}) = 1 - \prod_{i=1}^{n}(1 - x_i) \tag{4}$$

The reliability of component i are given as follows:

$$p_i = P(X_i = 1) \tag{5}$$

In addition, a simplified notation is used to describe conditional reliabilities. An example is given in Equation 6.

$$p_{3|1\bar{2}} = P(x_3 = 1 | x_1 = 1, x_2 = 0) \tag{6}$$

The main task of this paper is to find the system reliability $h(\mathbf{p})$, where $\mathbf{p}$ includes both the component reliabilities as well as their conditional reliabilities.

1.3. *Definitions*

Definition 1. The most important component dependencies are those dependencies that influence the system reliability the most, i.e. those dependencies that cannot be ignored without resulting in major changes in the predicted reliability of the system.

Definition 2. A dependency combination (DC) is a subset of the actual component dependencies in a compound software.

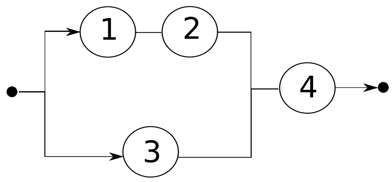

Fig. 1. An illustrative example.

Definition 3. Two components i and j are said to be data-serial components if either: 1) i receives data, directly or indirectly through other components, from j, or 2) j receives data, directly or indirectly through other components, from i.

$$i \stackrel{d}{\rightarrow} j \quad or \quad j \stackrel{d}{\rightarrow} i \tag{7}$$

Definition 4. Two components i and j are said to be data-parallel components if neither i or j receives data, directly or indirectly through other components, from the other.

$$i \stackrel{d}{\nrightarrow} j \quad and \quad j \stackrel{d}{\nrightarrow} i \tag{8}$$

To explain the concepts of data-serial and data-parallel components, the compound software given in Figure 1 is used as an illustrative example. The system consists of four components, and in Table 1 different pairs of data-serial and data-parallel components are listed. In addition, possible conditional reliabilities which can be used to express the dependency between these components are given.

Table 1. Different pairs of data-serial and data-parallel components.

data-serial component pairs	**stochastic dependence**
C1 and C2	$p_{2\|1}$ or $p_{1\|2}$
C1 and C4	$p_{4\|1}$ or $p_{1\|4}$
C2 and C4	$p_{4\|2}$ or $p_{2\|4}$
C3 and C4	$p_{4\|3}$ or $p_{3\|4}$
data-parallel component pairs	**stochastic dependence**
C1 and C3	$p_{3\|1}$ or $p_{1\|3}$
C2 and C3	$p_{3\|2}$ or $p_{2\|3}$

To express dependencies for sets of data-serial and data-parallel components, different conditional reliabilities can be used. For example, to express the dependency between the data-serial components 1 and 4 and the data-serial components 2 and 4, the conditional reliability $p_{4|12}$ can be used. In the same way, to express the dependency between the data-parallel components 1 and 3 and between the data-serial components 1 and 4, the conditional reliability $p_{1|34}$ can be used.

1.4. *Assumptions*

In this study, a software component is considered to be an entity that has a predefined and specified boundary and which is atomic, in the sense that it can't or won't be divided into sub-components. It is made no special assumptions whether the component is available in binary format or as source code. The context is essentially an Off-The-Shelf (OTS) situation, where custom developed and previously developed software (PDS) components are combined to achieve a larger piece of software.

In this paper, only on-demand types of situations are considered, i.e. situations where the system is given an input and execution is considered to be finished when a corresponding output has been produced.

The following assumptions are made:

- All structural relations between the components are known.
- The individual component reliabilities are known.
- The components, as well as the system, only have two possible states, a functioning state and a failure state.
- It is assumed positive correlation between the software components.
- The system has a monotone structure [27].

1.5. *The structure of this paper*

In Section 2, some of the work that has been done with regard to understanding the nature of failure dependency between software components is reviewed. Section 3 illustrates how the software components' marginal reliabilities put direct restrictions on the components' conditional reliabilities and failure probabilities. It is also shown that the degrees of freedom are much fewer than first anticipated when it comes to conditional probabilities. Section 4 describes the methods and analysis techniques used to identify possible rules for selecting the most important component dependencies. Section 5 presents the selected test cases, and Section 6 presents the results from the analyses. Section 7 summarizes the results and tries to come up with possible rules for selecting the most important component dependencies. Section 8 concludes and presents ideas for further work.

2. Earlier Work Related to the Problem of Component Dependency

The dominating case for discussions on software component dependency is multi-version designs, typically the N-version approach where output is decided by a voter using the results from N components as input. The idea behind N-version programming is that by forcing various aspects of the development process to be different, i.e. development team, methods, tools, programming languages etc. the likelihood of having the same fault in several components would become negligible.

The hypothesis that independently developed components would fail independently has been investigated from various perspectives. A direct test of this hypothesis was done in [14] where a total of 27 components were developed by different people. Although the results can be debated, this experiment indicated that assuming independence should be done with caution. The experiment showed that the number of tests for which several components failed was much higher than anticipated under the assumption of independence. While there are many different mechanisms that might cause even independently developed components to fail on the same inputs, it doesn't seem implausible that the simple fact that programmers are likely to approach a problem in much the same way would cause them to make the same mistakes, and thus cause dependency between the components' failure behavior.

A more theoretical approach on the same issue was presented in Eckhardt and Lee [5] and elaborated on a few years later in Littlewood and Miller [21]. Although Eckhardt and Lee present several interesting results, our primary interest is related to the considerations regarding whether independent development processes produce software components that fail independently. Note that a more comprehensive discussion is provided in [22].

The key variable in the Eckhardt and Lee model is the difficulty function $\theta(x)$, defined to be the probability that a component version chosen at random will fail on a particular input demand, x. The more difficult an input x is, the greater we would believe the chance that an unknown program will fail.

The main result in the Eckhardt and Lee model is that independently developed components do not imply independent components. The key point is that as long as some inputs are more difficult to process than others, even independently developed components will fail dependently. In fact, the more the difficulty varies between the inputs, the greater is the dependence in failure behavior between the components. Only in the special situation where all inputs are equally difficult, i.e. the difficulty function $\theta(x)$ is constant for all $x \in \Omega$, independently developed components will fail independently.

The Littlewood and Miller model [21] is a generalization of the Eckhardt and Lee model in which the different component versions are developed using diverse methodologies. In this context, the different development methodologies might represent different development environments, different types of programmers, different languages, different testing regimes etc.

The main result in the Littlewood and Miller model is that the use of diverse methodologies decreases the probability of simultaneous failure of several component versions. In fact, they show that it is theoretically possible to obtain component versions which exhibit better than independent failure behavior. So while it is natural to try to justify an assumption of independence, it is worthwhile noting that having independent components is not necessarily the optimal situation with regard to maximizing reliability.

Other relevant work on how to include component failure dependencies are summarized below.

Gokhale and Trivedi [8] look into problems associated with assuming independence in path-based approaches. The problem they address is that assuming independence of successively executing components is likely to produce pessimistic results, especially considering that the same component may be executed several times in a single path due to loop structures. The knowledge that a component did not fail on the previous loop iteration is likely to be a good indication that it will not fail on the next iteration either. This is an interesting observation and it indicates that thinking in terms of reliability block diagrams when it comes to software components is not straightforward. As a possible way to overcome the problem of a pessimistic estimate, the authors propose to treat multiple executions as a single execution. Their solution relies on 1) time-dependent notation of reliability and 2) time-dependent failure intensities of the individual components.

Zavala and Huhns [33] present an initial empirical study on the correlation of code complexity measures and coincident failures in multi-version systems (when two or more program versions are identically incorrect). Their study is based on 28 Java implementations and clearly shows a correlation between software metrics and coincident failures. At the current state the results cannot be generalized, however the authors have shown that the use of software complexity metrics as indicators of proneness to coincident failures in multi-version systems is worth exploring further.

In Popic et al. [28], the authors extend their previous work on Bayesian reliability prediction of component based systems by introducing the error propagation probability into the model. Like most other component-based reliability models, their old model assumed that system components will fail independently. The authors define the error propagation probability as the probability that an erroneous state generated in one component propagates to other components instead of being successfully detected and masked at its source. To describe error propagation, the model of Nassar et al. [26] is applied. Based on a case study, the authors conclude that error propagation may have significant impact on the system reliability prediction and argue that future architecture-based models should not ignore it.

Fricks and Trivedi [7] study the effect of failure dependencies in reliability models developed using stochastic Petri nets (SPN) and continuous-time Markov chains. Based on a set of examples, the authors conclude that failure dependencies highly influence the reliability models and that failure dependencies therefore never should be ignored. Of special interest is the authors classification of different types of failure dependencies that can arise in reliability modeling. The authors then illustrate how several of these failure dependencies can be incorporated into stochastic Petri net models.

Vieira and Richardson [31] argue that component dependencies should be treated as a first class problem in component-based systems (CBSs). They discuss issues related to component-based system dependencies and present a conceptual model for describing and analyzing dependencies in a CBS. To describe component dependencies, the authors use denotational semantics of partial-order multi-sets(pomsets).

In Huang et al. [12], the authors combine analytical models with simulation techniques for software reliability measurement. The authors present two failure-rate simulation techniques, which both take the functional dependency and error correlation among the components in a software system into account. In the first technique, the authors use a dependency coefficient to include dependencies between the components. This coefficient is based on test data from each component in the system. In the second technique, the transition probabilities between the components in the system are used. The authors do however not suggest any approaches to find these probabilities. The main contribution of their work is demonstrating an architecture-oriented simulation framework to analyze reliability measures for software systems with dependent components.

Reliability block diagrams (RBDs), fault trees (FTs) and reliability graphs (RGs) are all limited in their modelling capability, due to the assumption of stochastic independence among the system's units. Dynamic reliability block diagrams (DRBDs), presented in [4], extend RBDs with elements specific for representing dynamic behaviors. Examples of dynamic-dependent behaviors that can be handled in a DRBD include dependent, cascade, on-demand and/or common cause failures, as well as interferences between the system's units such as load sharing and inter/sequence-dependency. The DRBDs are based on the concept of dependency. The authors consider a dependency as the simplest dynamic relationship between two system units. A dependency is a unidirectional, one-way, dynamic relationship, which represents and quantifies the influence of one unit on another unit. More complex dynamic behaviors are than expressed as compositions of these simple dependencies. In [4], the authors investigate the reliability in two case studies and show that dynamic aspects and behaviors, usually not analyzable by other methodologies, can be handled in DRBDs.

Although previous work on software component dependencies is valuable, it was in [32] concluded that the scope of this work is too narrow. In [32], the authors take a deeper look at the nature of software component dependencies and try to increase the reader's understanding of the mechanisms that cause dependencies between software components. In the paper, the authors differ between degree of dependence between software components, which can be expressed through conditional or simultaneous failure probabilities, and the mechanisms that either cause or exclude events to occur together. These mechanisms are divided into two distinct categories:

- Development-cultural aspects (DC-aspects): Includes factors that cause different people, tools, methods, etc. to make the same mistakes, e.g. identical programming language, compiler, etc.
- Structural aspects (S-aspects): Includes factors that allow a failure in one component to affect the execution of another component, e.g. through shared resources, structural relation, etc.

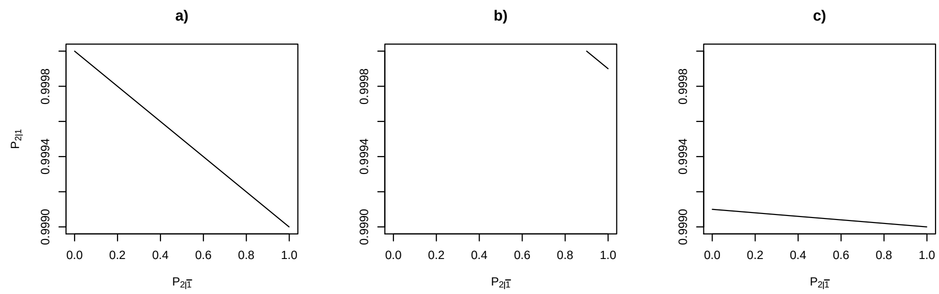

Fig. 2. Possible values for the conditional reliabilities in a two components system when a) $p_1 = 0.999$ and $p_2 = 0.999$, b) $p_1 = 0.999$ and $p_2 = 0.9999$ and c) $p_1 = 0.9999$ and $p_2 = 0.999$.

The main conclusions in [32] are that inter-dependencies between software components are more complicated than any existing methods consider.

3. Prior Information from the Software Components' Marginal Reliabilities

In the following, it will be shown how single components' marginal reliabilities, as well as the assumption of positive correlation, put directly restrictions on the components' conditional reliabilities. These restrictions may be used as direct input into a Bayesian belief net for establishing prior probability distributions for the probabilities that sets of software components will fail simultaneously. It may also be used as guidance for the experts as to which conditional reliabilities it is easiest to make any decisions about.

3.1. *Two components system*

Consider a general system consisting of only two software components. Assume further that the two components' marginal reliabilities p_1 and p_2 are known. In addition, positive correlation between component 1 and 2 is assumed ($p_{2|1} \geq p_2$). This means that information that component 1 is functioning cannot reduce the reliability of component 2. This is a reasonable assumption when the components are in series with each other. However, when the components are in parallel, this may not always be a natural assumption. If the components have been developed by different development teams, using different development methods and languages, it might in fact be natural to assume negative correlation. This means that if one component fails, this increases the reliability of the other component and visa versa. However, the consequences of assuming independence between all software components in a compound software are far more severe than by assuming positive correlation.

In a simple two components system, there are eight possible conditional probabilities between component 1 and 2 ($p_{2|1}$, $p_{2|\bar{1}}$, $p_{1|2}$, $p_{1|\bar{2}}$ etc.). If one of these con-

Table 2. Restrictions on the conditional reliabilities $p_{2|1}$ and $p_{2|\bar{1}}$ in a simple two components system for different combinations of the marginal reliabilities p_1 and p_2.

Marginal reliabilities		Conditional reliabilities
C1	C2	
$p_1 = 0.9$	$p_2 = 0.9999$	$p_{2\|1} \in [0.9999, 1]$ $p_{2\|\bar{1}} \in [0.999, 0.9999]$
$p_1 = 0.99$	$p_2 = 0.9999$	$p_{2\|1} \in [0.9999, 1]$ $p_{2\|\bar{1}} \in [0.99, 0.9999]$
$p_1 = 0.999$	$p_2 = 0.9999$	$p_{2\|1} \in [0.9999, 1]$ $p_{2\|\bar{1}} \in [0.9, 0.9999]$
$p_1 = 0.9$	$p_2 = 0.999$	$p_{2\|1} \in [0.999, 1]$ $p_{2\|\bar{1}} \in [0.99, 0.999]$
$p_1 = 0.99$	$p_2 = 0.999$	$p_{2\|1} \in [0.999, 1]$ $p_{2\|\bar{1}} \in [0.9, 0.999]$
$p_1 = 0.999$	$p_2 = 0.999$	$p_{2\|1} \in [0.999, 1]$ $p_{2\|\bar{1}} \in [0, 0.999]$
$p_1 = 0.9999$	$p_2 = 0.999$	$p_{2\|1} \in [0.999, 0.9990999]$ $p_{2\|\bar{1}} \in [0, 0.999]$
$p_1 = 0.99999$	$p_2 = 0.999$	$p_{2\|1} \in [0.999, 0.99900999]$ $p_{2\|\bar{1}} \in [0, 0.999]$
$p_1 = 0.999$	$p_2 = 0.99$	$p_{2\|1} \in [0.99, 0.99099099]$ $p_{2\|\bar{1}} \in [0, 0.99]$
$p_1 = 0.9999$	$p_2 = 0.99$	$p_{2\|1} \in [0.99, 0.990099]$ $p_{2\|\bar{1}} \in [0, 0.99]$
$p_1 = 0.99999$	$p_2 = 0.99$	$p_{2\|1} \in [0.99, 0.9900099]$ $p_{2\|\bar{1}} \in [0, 0.99]$

ditional probabilities is known, the others can easily be expressed by using general rules in probability theory. See proof in Appendix A.

Based on the law of total probability, the linear relationship between $p_{2|1}$ and $p_{2|\bar{1}}$ is given in Equation 9.

$$p_{2|1} = \frac{p_2}{p_1} - \frac{(1-p_1)}{p_1} p_{2|\bar{1}} \tag{9}$$

Equation 9 is used as basis for investigating the relation between the marginal reliabilities p_1 and p_2 and the conditional reliabilities $p_{2|1}$ and $p_{2|\bar{1}}$. In Table 2, different sets of marginal reliabilities and their restrictions on the components' conditional reliabilities are given. Restrictions on the conditional reliabilities $p_{2|1}$ and $p_{2|\bar{1}}$ for three different sets of marginal reliabilities p_1 and p_2 are also illustrated graphically in Figure 2.

The results in Table 2 and Figure 2 clearly shows that the marginal reliabilities p_1 and p_2 put direct restrictions on the conditional reliabilities $p_{2|1}$ and $p_{2|\bar{1}}$. In fact, in some cases the conditional reliabilities are restricted into small intervals. The restrictions depend heavily on the values of the marginal reliabilities.

3.2. *Three components system*

Let's move a step forward and look at a simple system consisting of three components. As for the two components system, it is assumed that the components' marginal reliabilities p_1, p_2 and p_3 are known. In addition, positive correlations are assumed.

In a simple three components system there are 48 possible conditional probabilities between components 1, 2 and 3 ($p_{3|1}$, $p_{3|\bar{1}}$, $p_{3|2}$, $p_{3|\bar{2}}$, $p_{3|12}$ etc.), including the eight possible conditional probabilities between components 1 and 2. If four of these conditional probabilities are known, the others can easily be expressed by using general rules of probability theory (see proof in Appendix A). In a three components system, one therefore for instance needs to know one conditional probability between components 1 and 2 and three conditional probabilities between components 1, 2 and 3 to find all the remaining conditional probabilities. One possible set of conditional probabilities may for example be: $p_{2|1}, p_{3|1}$, $p_{3|2}$ and $p_{3|12}$. However, this is only one possible selection of conditional probabilities that can be chosen. Another set may for example be: $p_{2|\bar{1}}, p_{3|\bar{1}}$, $p_{3|\bar{2}}$ and $p_{3|\bar{1}\bar{2}}$. Which set to choose should be considered thoroughly, since some conditional probabilities may be easier for an expert to determine than others.

The linear relationships between $p_{3|1}$ and $p_{3|\bar{1}}$ and between $p_{3|2}$ and $p_{3|\bar{2}}$ are parallel to the linear relationship between components 1 and 2 in Equation 9. The relations between the conditional reliabilities $p_{3|12}$, $p_{3|1\bar{2}}$, $p_{3|\bar{1}2}$ and $p_{3|\bar{1}\bar{2}}$ are shown in Appendix A to be:.

$$p_{3|1\bar{2}} = \frac{p_{3|1} - p_{3|12}p_{2|1}}{1 - p_{2|1}} \tag{10}$$

$$p_{3|\bar{1}2} = \frac{p_{3|2}p_2 - p_{3|12}p_{2|1}p_1}{p_2 - p_{2|1}p_1} \tag{11}$$

$$p_{3|\bar{1}\bar{2}} = \frac{p_3 - p_{3|1}p_1 - p_{3|2}p_2 + p_{3|12}p_{2|1}p_1}{1 - p_2 + (p_{2|1} - 1)p_1} \tag{12}$$

Equation 9 and the corresponding ones for $p_{3|1}$ and $p_{3|2}$, and Equations 10 - 12 are used as basis for investigating the relation between the marginal reliabilities p_1, p_2 and p_3 and the conditional reliabilities $p_{2|1}$, $p_{3|1}$, $p_{3|2}$ and $p_{3|12}$. In Tables 3 - 5,

Table 3. Restrictions on the conditional reliabilities $p_{2|1}$, $p_{3|1}$, $p_{3|2}$ and $p_{3|12}$ in a simple three components system when $p_1 = 0.9999$, $p_2 = 0.999$ and $p_3 = 0.99$.

Example 1	
First assumption:	
$p_1 = 0.9999$	
$p_2 = 0.999$	
$p_3 = 0.99$	
Results in:	
$p_{2\|1} \in [0.999, 0.9990999]$	$p_{2\|\bar{1}} \in [0, 0.999]$
$p_{3\|1} \in [0.99, 0.990099]$	$p_{3\|\bar{1}} \in [0, 0.99]$
$p_{3\|2} \in [0.99, 0.99099099]$	$p_{3\|\bar{2}} \in [0, 0.99]$
$p_{3\|12} \in [0.99, 0.99099999]$	$p_{3\|\bar{1}\bar{2}} \in [0, 0.99]$
Second assumption:	
$p_{2\|1} = 0.99905$	
$p_{3\|1} = 0.990085$	
Results in:	
$p_{3\|2} \in [0.990043, 0.990964]$	$p_{3\|\bar{2}} \in [0.026468, 0.947503]$
$p_{3\|12} \in [0.990085, 0.990999]$	$p_{3\|\bar{1}\bar{2}} \in [0, 0.140085]$
Third assumption:	
$p_{3\|2} = 0.9903$	
Results in:	
$p_{3\|12} \in [0.990336, 0.990342]$	$p_{3\|\bar{1}\bar{2}} \in [0, 0.140085]$

three different sets of marginal reliabilities and their restrictions on the components' conditional reliabilities are given. These tables should be read as follows:

- In the first assumption, it is assumed that the components' marginal reliabilities are known. Knowing these reliabilities put direct restrictions on all the remaining conditional reliabilities in the system. In some cases they limit the conditional reliabilities into small intervals.
- In the second assumption, it is assumed that the conditional reliabilities $p_{2|1}$ and $p_{3|1}$ are known, in addition to the marginal reliabilities. This put more strict restrictions on the remaining conditional reliabilities $p_{3|2}$ and $p_{3|12}$.
- In the third assumption, the conditional reliability $p_{3|2}$ is also assumed to be known and it can easily be seen that the more information that is available, the more strict are the restrictions on the remaining reliabilities.

4. Methods and Analysis

In this section, the techniques used to identify possible rules for selecting the most important component dependencies are described in detail. The techniques are applied on three test cases, each representing a well-known software structure. For detailed descriptions of the test cases and the sets of marginal and conditional reliabilities used see Section 5.

Table 4. Restrictions on the conditional reliabilities $p_{2|1}$, $p_{3|1}$, $p_{3|2}$ and $p_{3|12}$ in a simple three components system when $p_1 = 0.99$, $p_2 = 0.999$ and $p_3 = 0.9999$.

Example 2	
First assumptions:	
$p_1 = 0.99$	
$p_2 = 0.999$	
$p_3 = 0.9999$	
Results in:	
$p_{2\|1} \in [0.999, 1]$	$p_{2\|\bar{1}} \in [0.9, 0.999]$
$p_{3\|1} \in [0.9999, 1]$	$p_{3\|\bar{1}} \in [0.99, 0.9999]$
$p_{3\|2} \in [0.9999, 1]$	$p_{3\|\bar{2}} \in [0.9, 0.9999]$
$p_{3\|12} \in [0.9999, 1]$	$p_{3\|\bar{1}\bar{2}} \in [0, 0.9999]$
Second assumptions:	
$p_{2\|1} = 0.9999$	
$p_{3\|1} = 0.99999$	
Results in:	
$p_{3\|2} \in [0.9999081, 1]$	$p_{3\|\bar{2}} \in [0.9, 0.9918081]$
$p_{3\|12} \in [0.99999, 1]$	$p_{3\|\bar{1}\bar{2}} \in [0.9, 0.99099]$
Third assumptions:	
$p_{3\|2} = 0.99995$	
Results in:	
$p_{3\|12} \in [0.99999, 0.999995]$	$p_{3\|\bar{1}\bar{2}} \in [0.94446, 0.94995]$

4.1. *Direct calculation*

In the "direct calculation", the effects of including only a subset of the actual component dependencies when assessing the failure probability of compound software are examined. In this analysis, all marginal and conditional reliabilities are assumed to be known. This makes it possible to assess the system's "true" failure probability when all dependencies are taken into account. The system's "true" failure probability can then be compared to the failure probability predictions one gets when various component dependencies are ignored.

4.2. *Birnbaum's reliability importance measure*

Birnbaum's measure [2] for the reliability importance of component i, I_i^B, is defined by:

$$I_i^B = \frac{\delta h}{\delta p_i} \tag{13}$$

Hence, Birnbaum's measure is found by partial differentiation of the system reliability with respect to p_i. This approach is well known from classical sensitivity analysis and assumes independence between the components. If I_i^B is large, a small change in the reliability of component i will give a relatively large change in system reliability.

Table 5. Restrictions on the conditional reliabilities $p_{2|1}$, $p_{3|1}$, $p_{3|2}$ and $p_{3|12}$ in a simple three components system when $p_1 = 0.99$, $p_2 = 0.9999$ and $p_3 = 0.999$.

Example 3	
First assumptions:	
$p_1 = 0.99$	
$p_2 = 0.9999$	
$p_3 = 0.999$	
Results in:	
$p_{2\|1} \in [0.9999, 1]$	$p_{2\|\bar{1}} \in [0.99, 0.9999]$
$p_{3\|1} \in [0.999, 1]$	$p_{3\|\bar{1}} \in [0.9, 0.999]$
$p_{3\|2} \in [0.999, 0.9990999]$	$p_{3\|\bar{2}} \in [0, 0.999]$
$p_{3\|12} \in [0.999, 1]$	$p_{3\|\bar{1}\bar{2}} \in [0, 0.999]$
Second assumptions:	
$p_{2\|1} = 0.99999$	
$p_{3\|1} = 0.9999$	
Results in:	
$p_{3\|2} \in [0.9990802, 0.9990999]$	$p_{3\|\bar{2}} \in [0, 0.918808]$
$p_{3\|12} \in [0.9999, 0.99990999]$	$p_{3\|\bar{1}\bar{2}} \in [0, 0.9099]$
Third assumptions:	
$p_{3\|2} = 0.999085$	
Results in:	
$p_{3\|12} \in [0.9999, 0.9999085]$	$p_{3\|\bar{1}\bar{2}} \in [0.0556, 0.149085]$

Pivotal decomposition gives that:

$$\begin{aligned} h(\mathbf{p}) &= p_i h(1_i, \mathbf{p}) + (1 - p_i) h(0_i, \mathbf{p}) \\ &= p_i (h(1_i, \mathbf{p} - h(0_i, \mathbf{p})) + h(0_i, \mathbf{p}) \end{aligned} \tag{14}$$

Birnbaum's measure can therefore be written as:

$$I_i^B = \frac{\delta h}{\delta p_i} = h(1_i, \mathbf{p}) - h(0_i, \mathbf{p}) \tag{15}$$

Since $h(\cdot_i, \mathbf{p}) = E[\phi(\cdot_i, \mathbf{X})]$, the Birnbaum's measure can be written as:

$$\begin{aligned} I_i^B &= E[\phi(1_i, \mathbf{X})] - E[\phi(0_i, \mathbf{X})] \\ &= E[\phi(1_i, \mathbf{X}) - \phi(0_i, \mathbf{X})] \end{aligned} \tag{16}$$

When $\phi(\mathbf{X})$ is monotone, it can only take the values 0 and 1. I_i^B can therefore be given by:

$$\begin{aligned} I_i^B &= P(\phi(1_i, \mathbf{X}) - \phi(0_i, \mathbf{X}) = 1) \\ &= P(\phi(1_i, \mathbf{X}) = 1) - P(\phi(0_i, \mathbf{X}) = 1) \end{aligned} \tag{17}$$

Birnbaum's measure is therefore the probability that the system is in such a state that component i is critical for the system. If the components are dependent, which often is the case for software systems, the probability in Equation 17 can be used as the definition of the Birnbaum's measure.

In the experimental study, the idea is to use Birnbaum' measure to check if the importance of the software components changes when various component dependencies are ignored. If this is the case, it may indicate that some component dependencies are more important than others.

In Section 6, the results from using Birnbaum's measure are presented as one or more of the following measures:

- Original Birnbaum's measures.
- Standardized Birnbaum's measures.
- Squared difference between the true Birnbaum's measures and the measures one gets when various component dependencies are ignored.
- Squared difference between the true standardized Birnbaum's measures and the standardized measures one gets when various component dependencies are ignored.

4.3. *Principal Component Analysis (PCA)*

A principal component analysis is concerned with explaining the covariance structure or the correlation structure of a set of variables through a few linear combinations of these variables [13]. These linear combinations are called the principal components (PC).

The objective of a principal component analysis is usually data reduction. Although p variables are required to reproduce the total system's variability, often much of this variability can be explained by a small number of k uncorrelated principal components ($k \leq p$). If this is the case, the k principal components can replace the p variables, and the data set can be reduced.

Let's assume that the system's predicted failure probabilities under different dependency combinations[c] represent the variables in a PCA. For example; variable 1 can be the system's failure probability when all dependencies are included, variable 2 can be the system's failure probability when all components are independent and so on. All these variables are than calculated for n unique observation vectors. These observation vectors represent different variations in the values for each of the test cases' conditional reliabilities and are identified using a "factorial design" [24].

One of the main results from a PCA analysis is a graphical representation of the data. These graphs should be studied in detail. Score plots express graphically the variation in data and loading plots express the original variables contribution to describe this variation. To get a better understanding of the variation in data, score plots and loading plots should be examined simultaneously. Especially, points that

[c]See Definition 2 in Section 1.3.

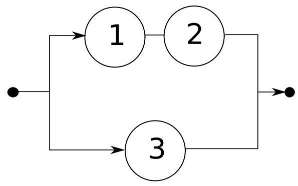

Fig. 3. Minimal path set representation of test case 1.

fall close together in the loading plots are of special interest. This indicates that the variables are highly correlated and therefore explain the same type of variation in data.

A good starting point would therefore be to try to identify the variables that load equally to the variable where all component dependencies are included. In this way the most important component dependencies may be identified.

5. Test Cases

To identify possible rules for selecting the most important component dependencies, this paper investigates three test cases, each representing a well-known software structure. In all test cases, the components are assumed to execute sequentially according to their numbers.

5.1. *Test case 1*

Test case 1 is a typical recovery block structure and consists of two independently developed, functionally identical software components that receive the same input data (see Figure 3). The first component is a super component consisting of sub components 1 and 2. Both the super component and component 3 receive the same input data, but they are not run in parallel like in N-version programming. First, the super component is run and its output is checked using an acceptance test. An acceptance test is a program specific fault detecting mechanism, which checks the results from a program execution. If the super component passes the acceptance test, its outcome is regarded as successful and the recovery block can be exited. If the test fails or if any errors are detected by other means during execution, an exception is raised and backward recovery is invoked. This restores the state of the system to what it was at entry, and component 3 is executed. Then the acceptance test is applied again. If both the super component and component 3 fail the acceptance test, the system fails.

Figure 3 only illustrates the redundant and diverse software components in the system. This is done to simplify the analysis. It should, however, be emphasized that the system is not complete without an additional component giving the redundant components inputs and an acceptance test validating the operation of the software components.

Table 6. The selected marginal and conditional reliabilities for test combinations 1.1 and 1.2.

Test combination 1.1	Test combination 1.2
$p_1 = 0.999$	$p_1 = 0.9999$
$p_2 = 0.999$	$p_2 = 0.999$
$p_3 = 0.9999$	$p_3 = 0.99$
$p_{2\|1} = 0.9999$	$p_{2\|1} = 0.99905$
$p_{3\|1} = 0.99999$	$p_{3\|1} = 0.990085$
$p_{3\|2} = 0.999985$	$p_{3\|2} = 0.9903$
$p_{3\|12} = 0.999992$	$p_{3\|12} = 0.99034$

The system in Figure 3 is evaluated in two different ways, representing test combination 1.1 and test combination 1.2. In test combination 1.1, it is assumed that component 3 is the "high-assurance" component, whereas the super component constitutes the "high-performance" component. In test combination 1.2, it is assumed that the super component is the "high-assurance" component, whereas component 3 is the "high-performance" component. In both combinations, it is assumed that the "high-assurance" component is more reliable than the "high-performance" component.

Based on the system's minimal path sets, the system reliability of test case 1 is given in Equation 18.

$$P(\phi(\mathbf{x}) = 1) = p_{2|1}p_1 + p_3 - p_{3|12}p_{2|1}p_1 \tag{18}$$

Since the main point of this paper is to investigate and evaluate the effect of including only partial dependency information when assessing a system's reliability, all the essential marginal and conditional reliabilities must be defined. Based on the assumptions made for test case 1 and the restrictions from the marginal reliabilities (see Section 3), a valid set of marginal and conditional reliabilities for test combination 1.1 and test combination 1.2 are given in Table 6.

The system's failure probability was assessed for the following dependency combinations[d]:

1. Including all software component dependencies.
2. Assuming independence between all software components.
3. Including only the dependency between data-serial components 1 and 2.
4. Including the dependencies between data-parallel components 1 and 3, and between data-parallel components 2 and 3.
5. Including only the dependency between data-parallel components 1 and 3.
6. Including only the dependency between data-parallel components 2 and 3.
7. Including the dependencies between data-parallel components 1 and 3, and between data-serial components 1 and 2.

[d]See Definition 2 in Section 1.3.

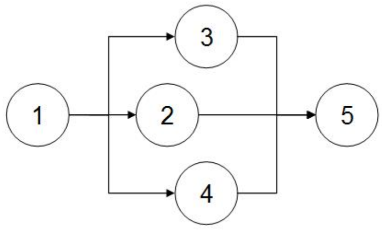

Fig. 4. System drawing of test case 2 and test case 3.

8. Including the dependencies between data-parallel components 2 and 3, and between data-serial components 1 and 2.

5.2. *Test case 2*

The second test case represents a more complex fault tolerant system capable of switching between two redundant components in case of failure. This type of structure is referred to as a simplex architecture [30], and are for instance used on software controllers in Boeing 777. The system consists of five components and includes both data-serial and data-parallel components (see Figure 4).

The test system is basically a redundant system with a hot standby and forward recovery. This means that the system switches to a "high-assurance" controller (component 4) if the normal "high-performance" controller (component 3) causes the system to enter states outside a predetermined boundary.

In this system, the sensor manager (component 1) receives data from the sensors that are monitoring the equipment under control (EUC). This information is collected by the manager and sent to the monitor (component 2) and the two controllers (components 3 and 4). Based on the information sent from the sensor manager, the monitor selects which controller to be used. The switch (component 5) will receive input from the monitor as to which controller to take its input from. Notice that both controllers continuously receive data and send output. It is only up to the monitor to decide which of the controllers that actually will be allowed to control the system. Data from the selected controller will be sent to the actuators which in turn control the EUC.

For simplicity, two assumptions are made. First of all, it is assumed that the switch does not fail. Secondly, it is assumed that the controllers are independent of the monitor. The system will function as long as the sensor manager functions in combination with either both controllers or with at least one controller and the monitor.

A minimal path set representation of the simplified system is illustrated in Figure 5. Based on the system's minimal path sets and the assumptions that are made, the system's reliability is given in Equation 19.

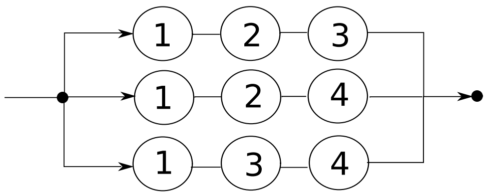

Fig. 5. Minimal path set representation of test case 2.

$$P(\phi(\mathbf{x}) = 1) = p_{3|1}p_{2|1}p_1 + p_{4|1}p_{2|1}p_1 + p_{4|13}p_{3|1}p_1 - 2p_{4|13}p_{3|1}p_{2|1}p_1 \quad (19)$$

Based on the assumptions made for test case 2 and the restrictions from the marginal reliabilities (see Section 3), a valid set of marginal and conditional reliabilities for test case 2 is given in Table 7.

The system's failure probability was assessed for the following dependency combinations:

1. Including all software component dependencies.
2. Assuming independence between all software components.
3. Including only the dependency between data-parallel components 3 and 4.
4. Including only the dependency between data-serial components 1 and 2.
5. Including the dependencies between data-serial components 1 and 2, and between data-parallel components 3 and 4.
6. Including the dependencies between data-serial components 1 and 3, and between data-parallel components 3 and 4.
7. Including the dependencies between data-serial components 1 and 4, and between data-parallel components 3 and 4.
8. Including the dependencies between data-serial components 1 and 3, between data-serial components 1 and 4, and between data-parallel components 3 and 4.
9. Including only the dependency between data-serial components 1 and 3.
10. Including only the dependencies between data-serial components 1 and 4.
11. Including the dependencies between data-serial components 1 and 3, and between data-serial components 1 and 4.
12. Including the dependencies between data-serial components 1 and 2, between data-serial components 1 and 3 and between data-serial components 1 and 4.
13. Including the dependencies between data-serial components 1 and 2 and between data-serial components 1 and 3.
14. Including the dependencies between data-serial components 1 and 2 and between data-serial components 1 and 4.

Note that we somewhat imprecisely use the characterizations data-serial and data-

Table 7. The selected marginal and conditional reliabilities for test cases 2 and 3.

Test case 2 and 3
$p_1 = 0.99999$
$p_2 = 0.999$
$p_3 = 0.99$
$p_4 = 0.9999$
$p_{2\|1} = 0.999005$
$p_{3\|1} = 0.990005$
$p_{4\|1} = 0.999905$
$p_{4\|3} = 0.999995$
$p_{4\|13} = 0.9999965$

parallel also in the simplified system. The same is done in test case 3.

5.3. *Test case 3*

Test case 3 is equal to test case 2, except that a failure of component 1 does not necessarily cause system failure. This is counterintuitive since component 1 is in series with the rest of the system, i.e. all other components are downstream of this component. To see that failure in component 1 doesn't necessarily cause the system to fail, what is meant by failure in component 1 must be defined.

It must be remembered that the context is a system consisting of multiple software components. For each of these components it is assumed that reliability data are available. This means that the reliability assessment of these components must have been done with reference to a given specification. It will in many cases, however, be uncertain whether this specification is completely in accordance with the requirements of the system the component is put into. Thus, what constitutes a failure, according to the component's specification, is not necessarily a failure in the context of the system. Limited accuracy of outputs is one example of "failures" that might not constitute a failure in a given context. As can be seen from the reliabilities in Table 7, failures in component 1 are considered to be serious. E.g., the reliability of component 3 is 0.990005 when component 1 is OK and 0.490005 when component 1 fails.

By assuming that a failure of component 1 does not necessary cause system failure, the assumption in Section 1.4 on binary component states is violated. If the system is robust to a failure in component 1, the component has two possible failure modes instead of one: 1) component 1 fails and leads to system failure and 2) component 1 fails but does not lead to system failure.

Birnbaum's measure assumes binary component states and can therefore not be calculated for components having multiple failure modes. One possible way to overcome the problem of multiple failure modes in component 1, is to treat component 1 as an environmental factor and not as a regular component in the system.

Another way is to redefine what is meant by a failure of component 1, and say that component 1 only fails if it leads to system failure as well. In test case 3, component 1 is treated as an environmental factor and Birnbaum measures are only calculated for components 2, 3 and 4.

The system in test case 3 will function as long as either both controllers function, or if at least one controller and the monitor function. Based on the simplified system's minimal path sets, the assumptions that are made and the law of total probability, the system reliability is given in Equation 20.

$$\begin{aligned} P(\phi(\mathbf{x}) = 1) = &(p_{3|1}p_{2|1} + p_{4|1}p_{2|1} + p_{4|13}p_{3|1} \\ &- 2p_{4|13}p_{3|1}p_{2|1})p_1 \\ &+ (p_{3|\bar{1}}p_{2|\bar{1}} + p_{4|\bar{1}}p_{2|\bar{1}} + p_{4|\bar{1}3}p_{3|\bar{1}} \\ &- 2p_{4|\bar{1}3}p_{3|\bar{1}}p_{2|\bar{1}})q_1 \end{aligned} \tag{20}$$

The system's failure probability was assessed for same dependency combinations as in test case 2.

6. Results

For each test case described in Section 5, the following procedure was applied:

1. Direct calculation was performed using a selected set of marginal and conditional reliabilities.
2. Birnbaum's measures were studied assuming the same marginal and conditional reliabilities.
3. PCA was performed by varying the values of the test case's conditional reliabilities.

The results from the analyses are summarized below.

6.1. *Test case 1.1*

6.1.1. *Direct calculation*

Using the marginal and conditional reliabilities in Table 6, the system's failure probability in test case 1.1 was calculated assuming the eight dependency combinations listed in Section 5.1. The results are summarized in the line plot in Figure 6 and clearly show that the system's failure probability divides into four different groups depending on the dependency combination used. The groups are summarized below.

- Group 1 consists of dependency combinations 1 and 4. Both these dependency combinations result in the system's exact failure probability (0.000092). This indicates that dependency combination 4, which includes the dependencies between data-parallel components 1 and 3 and between data-parallel components 2 and 3, can replace the true dependency combination in test case 1.1 without significantly underestimating the system's failure probability.

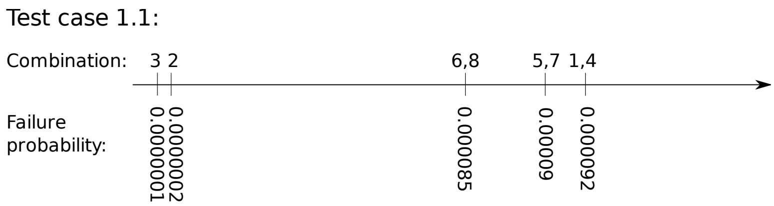

Fig. 6. Results from direct calculation in test case 1.1.

- Group 2 consists of dependency combinations 5 and 7, which both include the dependency between data-parallel components 1 and 3. Using one of these dependency combinations results in a minor underestimation of the system's failure probability (0.00009).
- Group 3 consists of dependency combinations 6 and 8, which both include the dependency between data-parallel components 2 and 3. Using one of these dependency combinations results in a minor to average underestimation of the system's failure probability (0.000085).
- Group 4 consists of dependency combinations 2 and 3. Common for these two dependency combinations is that none of them include any dependencies between data-parallel components. Dependency combination 2 assumes independence between all software components whereas dependency combination 3 only includes the dependency between data-serial components 1 and 2. Using one of these dependency combinations results in a major underestimation of the system's failure probability (0.0000001).

6.1.2. *Birnbaum's measure*

Based on the original Birnbaum measures in Table 8, it can easily be seen that dependency combination 4 is the dependency combination that alters the Birnbaum measures the least. This is especially apparent for the Birnbaum measures of components 1 and 2. While dependency combination 4 has the same Birnbaum measures for components 1 and 2 as the correct dependency combination, the remaining dependency combinations significantly overestimate these measures. Dependency

Table 8. Original Birnbaum measures and standardized squared difference for components 1, 2 and 3 in test case 1.1.

DC	I_1^B	I_2^B	I_3^B	st. sqrd. diff.
1	8.0×10^{-6}	8.0×10^{-6}	1.1×10^{-3}	0
2	1.0×10^{-4}	1.0×10^{-4}	2.0×10^{-3}	8.8×10^{-3}
3	1.0×10^{-4}	1.0×10^{-4}	1.1×10^{-3}	2.9×10^{-2}
4	8.0×10^{-6}	8.0×10^{-6}	2.0×10^{-3}	6.1×10^{-5}
5	1.0×10^{-5}	1.0×10^{-5}	2.0×10^{-3}	2.9×10^{-5}
6	1.5×10^{-5}	1.5×10^{-5}	2.0×10^{-3}	2.9×10^{-7}
7	1.0×10^{-5}	1.0×10^{-5}	1.1×10^{-3}	1.9×10^{-5}
8	1.5×10^{-5}	1.5×10^{-5}	1.1×10^{-3}	2.2×10^{-4}

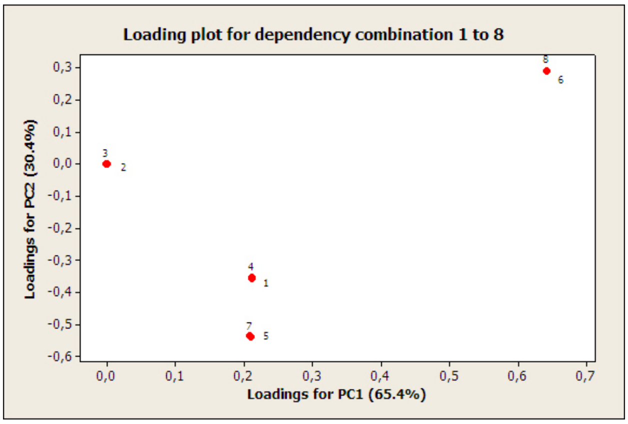

Fig. 7. Loading plot for test case 1.1.

combinations 2 and 3, which are the combinations that overestimates the Birnbaum measures the most, are also the dependency combinations that underestimates the system's failure probability the most.

6.1.3. *PCA*

Variables that fall close together in a PCA loading plot indicate that the variables are highly correlated and that they explain the same type of variation in data. The loading plot in Figure 7 shows that the different dependency combinations in test case 1.1 divide into four different groups based on their PCA loadings. The groups are summarized below.

- Group 1 consists of dependency combinations 1 and 4, since these dependency combinations fall close together in the loading plot. The results from the PCA analysis show that using dependency combination 4 results in the exact or a minor overestimation of the system's failure probability. Using all other dependency combinations will in almost all cases underestimate the system's failure probability, however to varies degrees.
- Group 2 consists of dependency combinations 5 and 7. The results from the PCA analysis show that using one of these dependency combinations mainly results in a minor underestimation of the system's failure probability. However, in some special cases these dependency combinations may result in a major underestimation of the system's failure probability.
- Group 3 consists of dependency combinations 6 and 8. The results from the PCA analysis show that using one of these dependency combinations mainly will result in a minor underestimation of the system's failure probability. However, in some special cases these dependency combinations may result in a major underestimation of the system's failure probability.
- Group 4 consists of dependency combinations 2 and 3, which constantly

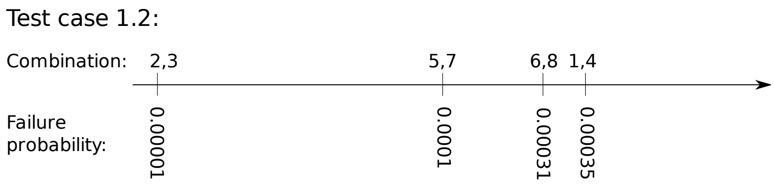

Fig. 8. Results from direct calculation in test case 1.2.

result in a major underestimation of the system's failure probability. Results from the PCA analysis show that these dependency combinations may underestimate the failure probability by a factor of 1000 compared to the system's true failure probability. In addition, the results show that by only including the dependency between data-serial components 1 and 2 may result in even worse results than by assuming independence between all components.

6.1.4. *Results test case 1.1*

The results from the analyses performed on test case 1.1 show that:

- Since the data-parallel components 1 and 3 and the data-parallel components 2 and 3 have equal reliabilities, both dependencies should be included in the reliability prediction. In fact, including only one of the dependencies may result in a major underestimation of the system's failure probability.
- Including only the dependency between the data-serial components 1 and 2 results in a major underestimation of the system's failure probability. In some cases, the results are even worse than by assuming independence between all components.

6.2. *Test case 1.2*

6.2.1. *Direct calculation*

Using the marginal and conditional reliabilities in Table 6, the system's failure probability in test case 1.2 was calculated assuming the same dependency combinations as in test case 1.1. The results are summarized in the line plot in Figure 8 and clearly show that the system's failure probability divides into four different groups depending on the dependency combination used. The groups are summarized below.

- Group 1 consists of dependency combinations 1 and 4. Both these dependency combinations result in the system's exact failure probability (0.00035). This indicates that dependency combination 4 can replace the true dependency combination in test case 1.2 without significantly underestimating the system's failure probability.
- Group 2 consists of dependency combinations 6 and 8. Using one of these dependency combinations results in a minor underestimation of the sys-

Table 9. Original Birnbaum measures and squared difference for components 1, 2 and 3 in test case 1.2.

DC	I_1^B	I_2^B	I_3^B	sqrd. diff.
1	0.0097	0.0097	0.001	0
2	0.01	0.01	0.0011	2.3×10^{-7}
3	0.01	0.01	0.001	2.3×10^{-7}
4	0.0097	0.0097	0.0011	2.5×10^{-9}
5	0.0099	0.0099	0.0011	1.3×10^{-7}
6	0.0097	0.0097	0.0011	5.7×10^{-9}
7	0.0099	0.0099	0.001	1.4×10^{-7}
8	0.0097	0.0097	0.001	3.2×10^{-9}

tem's failure probability (0.00031).

- Group 3 consists of dependency combinations 5 and 7. Using one of these dependency combinations results in an average underestimation of the system's failure probability (0.0001).
- Group 4 consists of dependency combinations 2 and 3. Using one of these dependency combinations results in a major underestimation of the system's failure probability (0.00001).

6.2.2. *Birnbaum's measure*

Based on the original Birnbaum measures and the squared differences in Table 9, it can easily be seen that dependency combinations 4, 6 and 8 are the dependency combinations that alter the Birnbaum measures the least. This is especially apparent for the Birnbaum measures of components 1 and 2. While dependency combinations 4, 6 and 8 have the same Birnbaum measures for components 1 and 2 as the correct dependency combination, the remaining dependency combinations overestimate these measures. Dependency combinations 2 and 3, which are the combinations that overestimates the Birnbaum measures the most, are also the dependency combinations that underestimates the system's failure probability the most. In addition, it can easily be seen that dependency combinations 2 and 3 have the highest squared difference between their Birnbaum measures and the Birnbaum measures calculated including all component dependencies.

6.2.3. *PCA*

The loading plot in Figure 9 shows that the different dependency combinations in test case 1.2 can be divided into four different groups based on their PCA loadings. The groups are summarized below.

- Group 1 consists of dependency combinations 1 and 4, since these dependency combinations fall close together in the loading plot. This indicates that dependency combination 4 can replace dependency combination 1 in test case 1.1 without any serious consequences. In fact, the results from the PCA analysis show that using dependency combination 4 results in the

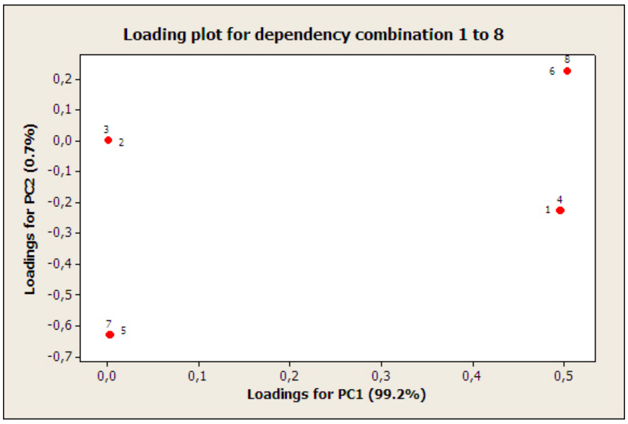

Fig. 9. Loading plot for test case 1.2.

exact or a minor overestimation of the system's failure probability. Using all other dependency combinations will in almost all cases underestimate the system's failure probability, however to varies degrees.

- Group 2 consists of dependency combinations 6 and 8. Since, principal component 1 explains 99.2% of the variation in data in test case 1.2, dependency combinations 6 and 8 also load closely to dependency combinations 1 and 4. The results from the PCA analysis show that using one of these dependency combinations mainly results in a minor underestimation of the system's failure probability. In fact, the results show that using dependency combinations 6 or 8 may underestimate the failure probability by a factor of 9 compared to the system's true failure probability.
- Group 3 consists of dependency combinations 5 and 7. The results from the PCA analysis show that using one of these dependency combinations may underestimate the system's failure probability by a factor of 78 compared to the system's true failure probability.
- Group 4 consists of the dependency combinations that constantly result in a major underestimation of the system's failure probability. Results from the PCA analysis show that these dependency combinations may underestimate the failure probability by a factor of 86 compared to the system's true failure probability. In addition, the results show that by only including the dependency between data-serial components 1 and 2 may result in even worse results than by assuming independence between all components.

6.2.4. *Results test case 1.2*

The results from the analyses performed on test case 1.2 show that:

- Including the dependency between the most unreliable data-parallel components 2 and 3 gives predictions close to the system's true failure probability.

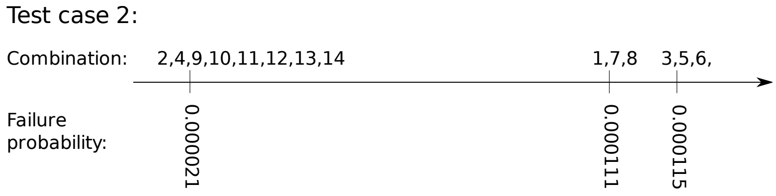

Fig. 10. Results from direct calculation in test case 2.

Ignoring this dependency may, however, result in a major underestimation of the system's failure probability.

- Including the additional dependency between data-parallel components 1 and 3 may improve the predictions even more.
- Including only the dependency between data-serial components 1 and 2 results in a major underestimation of the system's failure probability. In some cases, the results are even worse than by assuming independence between all components.

6.3. *Test case 2*

6.3.1. *Direct calculation*

Using the marginal and conditional reliabilities in Table 7, the system's failure probability in test case 2 was calculated assuming the fourteen dependency combinations listed in Section 5.2. The results are summarized in the line plot in Figure 10 and clearly show that the system's failure probability divides into three different groups depending on the dependency combination used. The groups are summarized below:

- Group 1 consists of dependency combinations 1, 7 and 8. All these dependency combinations result in the system's exact failure probability (0.000111). This indicates that dependency combinations 7 and 8, which both include the dependencies between data-parallel components 3 and 4 and between data-serial components 1 and 4, can replace the true dependency combination in the system, without significantly underestimating the system's failure probability.
- Group 2 consists of dependency combinations 3, 5 and 6, which all include the dependency between data-parallel components 3 and 4. Using one of these dependency combinations results in a minor overestimation of the system's failure probability (0.000115).).
- Group 3 consists of dependency combinations 2, 4, 9, 10, 11, 12, 13 and 14. Using one of these dependency combinations results in a major underestimation of the system's failure probability (0.000021). Common for these dependency combinations is that none of them include the dependency between data-parallel components 3 and 4. Dependency combination 2 assumes independence between all software components, whereas the other combinations only include dependencies between data-serial components.

Table 10. Standardized Birnbaum measures and squared difference for components 1, 2, 3 and 4 in test case 2.

DC	I_1^B	I_2^B	I_3^B	I_4^B	st. sqrd. diff.
1	0.9786	0.0097	0.001	0.0107	0
2	0.9783	0.0099	0.0011	0.0107	1.3×10^{-7}
3	0.9786	0.0097	0.001	0.0107	5.0×10^{-10}
4	0.9783	0.0099	0.0011	0.0107	1.2×10^{-7}
5	0.9786	0.0097	0.001	0.0107	1.0×10^{-10}
6	0.9786	0.0097	0.001	0.0107	1.0×10^{-10}
7	0.9786	0.0097	0.001	0.0107	5.0×10^{-10}
8	0.9786	0.0097	0.001	0.0107	1.0×10^{-10}
9	0.9783	0.0099	0.0011	0.0107	1.2×10^{-7}
10	0.9783	0.0099	0.0011	0.0107	1.2×10^{-7}
11	0.9783	0.0099	0.0011	0.0107	1.2×10^{-7}
12	0.9783	0.0099	0.0011	0.0107	1.1×10^{-7}
13	0.9783	0.0099	0.0011	0.0107	1.1×10^{-7}
14	0.9783	0.0099	0.0011	0.0107	1.2×10^{-7}

6.3.2. *Birnbaum's measure*

Based on the standardized Birnbaum measures and squared differences in Table 10, it can easily be seen that dependency combinations 3, 5, 6, 7 and 8 are the dependency combinations that alter the standardized Birnbaum measures the least. In addition, it can easily be seen that dependency combinations 2, 4, 9, 10, 11, 12, 13 and 14 have the highest squared difference between their standardized Birnbaum measures and the standardized Birnbaum measures calculated including all component dependencies.

6.3.3. *PCA*

The loading plot in Figure 11 shows that the different dependency combinations in test case 2 can be divided into three different groups based on their PCA loadings. The groups are summarized below.

- Group 1 consists of dependency combinations 1, 7 and 8, since these dependency combinations fall close together in the loading plot. This indicates that dependency combinations 7 and 8 can replace dependency combination 1 in test case 2 without any serious consequences. In fact, the results from the PCA analysis show that using dependency combination 7 or 8 results in the exact or a minor overestimation of the system's failure probability.
- Group 2 consists of dependency combinations 3, 5 and 6, which all fall close together in the loading plot. Since, principal component 1 explains 99.9% of the variation in data in test case 2, dependency combinations 3, 5 and 6 also load closely to dependency combinations 1, 7 and 8 .The results from the PCA analysis show that using one of these dependency combinations mainly results in a minor overestimation of the system's failure probability.
- Group 3 consists of the dependency combinations that constantly underestimate the system's failure probability, and includes dependency combi-

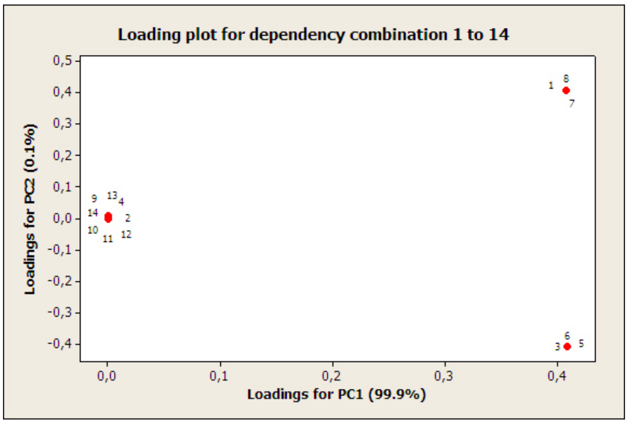

Fig. 11. Loading plot for test case 2.

nations 2, 4, 9, 10, 11, 12, 13 and 14. Results from the PCA analysis show that these dependency combinations may underestimate the failure probability by a factor of 5 compared to the system's true failure probability. In addition, the results show that by only including dependencies between data-serial components may result in even worse results than by assuming independence between all components.

6.3.4. *Results test case 2*

The results from the analyses performed on test case 2 show that:

- Including the dependency between data-parallel components 3 and 4 gives predictions close to the system's true failure probability. Ignoring this dependency will have major consequences on the system's failure probability.
- Including the additional dependency between the most reliable data-serial components 1 and 4 results in even better predictions.
- Including only dependencies between data-serial components results in a major underestimation of the system's failure probability. In some cases, the results are even worse than by assuming independence between all components.

6.4. ***Test case 3***

6.4.1. *Direct calculation*

Using the marginal and conditional reliabilities in Table 7, the system's failure probability in test case 3 was calculated assuming the same dependency combinations as in test case 2. The results are summarized in the line plot in Figure 12 and clearly show that the system's failure probability divides into two major groups depending on the dependency combination used. The groups are summarized below.

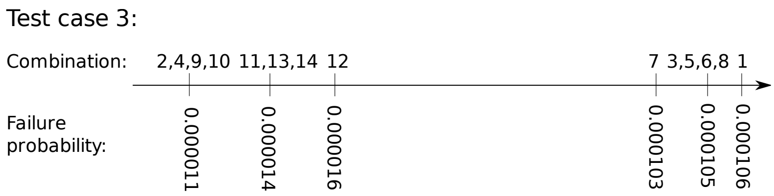

Fig. 12. Results from direct calculation in test case 3.

- Group 1 consists of dependency combinations 1, 3, 5, 6, 7, and 8. Using one of these dependency combinations only results in a minor underestimation of the system's failure probability (0.000103, 0.000105). This indicates that all these dependency combinations can replace the correct dependency combination in test case 3 without any major consequences on the system's failure probability.
- Group 2 consists of dependency combinations 2, 4, 9, 10, 11, 12, 13 and 14. Using one of these dependency combinations results in a major underestimation of the system's failure probability (0.000011).

6.4.2. *Birnbaum's measure*

Based on the standardized Birnbaum measures and squared differences in Table 11, it can easily be seen that dependency combinations 3, 5, 6, 7 and 8 are the dependency combinations that alter the standardized Birnbaum measures the least. In addition, it can easily be seen that dependency combinations 2, 4, 9, 10, 11, 12, 13 and 14 have the highest squared difference between their standardized Birnbaum measures and the standardized Birnbaum measures calculated including all component dependencies.

Table 11. Standardized Birnbaum measures and squared difference for components 2, 3 and 4 in test case 3.

DC.	I_2^B	I_3^B	I_4^B	st. sqrd. diff.
1	0.4527	0.0458	0.5014	0
2	0.4553	0.0496	0.4951	6.1×10^{-5}
3	0.4526	0.0459	0.5015	2.7×10^{-8}
4	0.4553	0.0496	0.4951	6.1×10^{-5}
5	0.4526	0.0459	0.5015	2.7×10^{-8}
6	0.4526	0.0459	0.5015	2.7×10^{-8}
7	0.4527	0.046	0.5014	2.2×10^{-8}
8	0.4526	0.046	0.5016	4.4×10^{-8}
9	0.4553	0.0496	0.4951	6.1×10^{-5}
10	0.4553	0.0496	0.4951	6.1×10^{-5}
11	0.4552	0.0496	0.4952	5.9×10^{-5}
12	0.4554	0.0494	0.4952	5.7×10^{-5}
13	0.4554	0.0496	0.4950	6.3×10^{-5}
14	0.4554	0.0494	0.4952	5.8×10^{-5}

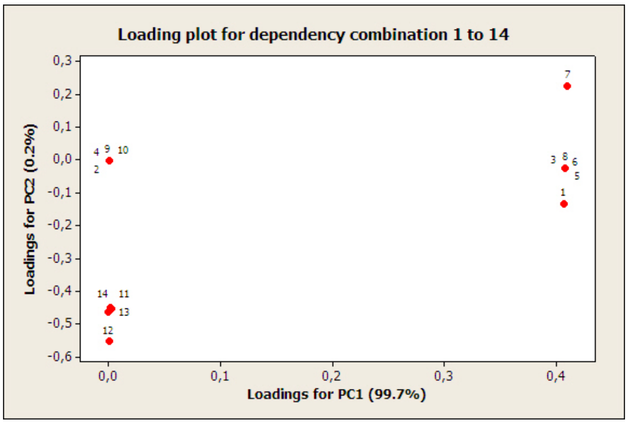

Fig. 13. Loading plot for test case 3.

6.4.3. *PCA*

The loading plot in Figure 13 shows that the different dependency combinations in test case 3 can be divided into four different groups based on their PCA loadings. The groups are summarized below:

- Group 1 consists of dependency combinations 1, 3, 5, 6 and 8, since these dependency combinations fall close together in the loading plot. This indicates that these dependency combinations can replace dependency combination 1 in test case 3 without any serious consequences. The results from the PCA analysis show that using one of the dependency combinations in group 1 may result in the exact or a minor underestimation of the system's failure probability.
- Group 2 consists of dependency combination 7. Since, principal component 1 explains 99.7% of the variation in data in test case 3, dependency combination 7 also load closely to the dependency combinations in group 1. The results from the PCA analysis show that using dependency combination 7 may result in the exact or a minor underestimation of the system's failure probability.
- Group 3 consists of dependency combinations 11, 12, 13 and 14. The results from the PCA analysis show that using one of the dependency combinations in group 2 may underestimate the system's failure probability by a factor of 9.
- Group 4 consists of the dependency combinations 2, 4, 9 and 10. Since, principal component 1 explains 99.7% of the variation in data in test case 3, dependency combinations 2, 4, 9 and 10 also load closely to the dependency combinations in group 3. The results from the PCA analysis show that using one of the dependency combinations in group 4 may underestimate the system's failure probability by a factor of 10.

6.4.4. *Results test case 3*

The results from the analyses performed on test case 3 show that:

- Including the dependency between data-parallel components 3 and 4 gives predictions close to the system's true failure probability. Ignoring this dependency will have major consequences on the system's failure probability.
- Including only dependencies between data-serial components results in a major underestimation of the system's failure probability. In some cases, the results are even worse than by assuming independence between all components.

7. Summary of the Results and Discussion

The results from the analyses performed in Section 6 show that the three techniques "direct calculation", Birnbaum's measure and PCA in most cases identify the same dependency combinations as the "best" dependency combinations. The results can be summarized as follows:

- Including only partial dependency information may give a substantial improvement in the reliability predictions, compared to assuming independence between all software components. However, this is only as long as the most important component dependencies are included.
- It is also apparent that dependencies between data-parallel components are far more important than dependencies between data-serial components.

For a system consisting of both data-parallel and data-serial components, the results indicate that:

- Including only dependencies between data-serial components may result in a major underestimation of the system's failure probability. In some cases, the results are even worse than by assuming independence between all components.
- Including only dependencies between data-parallel components may give predictions close to the system's true failure probability, as long as the dependency between the most unreliable components is included.
- Including additional dependencies between data-parallel components may improve the predictions further.
- Including additional dependencies between data-serial components may also give better predictions, as long as the dependency between the most reliable components is included.

One of the key results in [6] is the following theorem:

Theorem 1.
Let $X_1 \dots X_n$ be associated random variables such that $0 \leq X_i \leq 1$ for $i = 1 \dots n$. Then

$$E\prod_{i=1}^{n} X_i \geq \prod_{i=1}^{n} EX_i \tag{21}$$

$$E\coprod_{i=1}^{n} X_i \leq \coprod_{i=1}^{n} EX_i \tag{22}$$

By using this theorem on components having binary states, the theorem says that falsely assuming independence between components in a series structure will overestimate the system's failure probability. The theorem also says that falsely assuming independence between components in a parallel structure will underestimate the system's failure probability. The author of [27] therefore concludes that for an arbitrary component structure, the consequence of assuming independence will be impossible to predict.

The results in Section 6 do, however, indicate that it may in fact be possible to say something about the consequences of assuming independence between some components in an arbitrary system structure. For a system where there are dependencies between both data-serial and data-parallel components, it is quite clear that the effect of falsely assuming independence between data-serial components is greatly diminished as long as the dependencies between data-parallel components are included. In the opposite case, when wrongly assuming independence between data-parallel components and including the dependencies between data-serial components, the system's failure probability may however be underestimated even more than by assuming independence between all software components in the system.

8. Conclusions and Further Work

In this paper, it is shown that the difficult task of including component dependencies in the reliability calculations can be simplified in three ways:

1. The components' marginal reliabilities put direct restrictions on the components' conditional reliabilities in a compound software.
2. The degrees of freedom are much fewer than first anticipated when it comes to conditional probabilities. If the components' marginal reliabilities and four of the components' conditional probabilities are known in a simple three components system, the remaining 44 conditional probabilities can be expressed using general rules of probability theory. This is shown mathematically in Appendix A.
3. Including only partial dependency information may give substantial improvements in the reliability predictions, compared to assuming independence between all software components. However, this is only as long as the most important component dependencies are included.

It should be emphasized that the rules for selecting the most important component dependencies are based on case studies, where the individual component reliabilities are assumed to be known. It is also assumed that all components in the test cases, as well as the system, only have two possible states. In addition, the research is restricted to on-demand types of situations.

It should also be emphasized that the objective of this research is to include dependency aspects in the reliability calculations of critical systems, and not to handle component dependencies in systems consisting of a huge amount of components.

To follow up on these results, a more analytical approach should be considered. In addition, an evaluation of the proposed rules by studying other well-known software structures is essential.

Acknowledgements

This work has been performed as part of a PhD-study in collaboration between Østfold University College, Institute for Energy Technology and the University of Oslo.

Appendix A. Theorems and Proofs

Theorem 1. *Consider a general system consisting of two components. Assume further that the components' marginal reliabilities* p_1 *and* p_2 *are known. In such a system there are eight possible conditional probabilities between components 1 and 2:* $p_{2|1}$, $p_{\bar{2}|1}$, $p_{2|\bar{1}}$, $p_{\bar{2}|\bar{1}}$, $p_{1|2}$, $p_{\bar{1}|2}$, $p_{1|\bar{2}}$ *and* $p_{\bar{1}|\bar{2}}$. *If one of these conditional probabilities is known, the remaining seven can be found using general rules in probability theory.*

Proof. This proof uses Bayes theorem, the rule of complementation and the following rule of total probability:

$$p_2 = p_{2|1}p_1 + p_{2|\bar{1}}p_{\bar{1}} \tag{A.1}$$

Assume that the conditional probability $p_{2|1}$ is known. As shown in Equations A.2-A.8, the seven remaining conditional probabilities can be expressed as functions of p_1, p_2 and $p_{2|1}$.

$$p_{\bar{2}|1} = 1 - p_{2|1} \tag{A.2}$$

$$p_{2|\bar{1}} = \frac{p_2 - p_{2|1}p_1}{1 - p_1} \tag{A.3}$$

$$p_{\bar{2}|\bar{1}} = 1 - \frac{p_2 - p_{2|1}p_1}{1 - p_1} \tag{A.4}$$

$$p_{1|2} = \frac{p_{2|1}p_1}{p_2} \tag{A.5}$$

$$p_{\bar{1}|2} = 1 - \frac{p_{2|1}p_1}{p_2} \tag{A.6}$$

$$p_{1|\bar{2}} = \frac{(1 - p_{2|1})p_1}{1 - p_2} \tag{A.7}$$

$$p_{\bar{1}|\bar{2}} = 1 - \frac{(1 - p_{2|1})p_1}{1 - p_2} \tag{A.8}$$

□

Theorem 2. *Consider a general system consisting of three components. Assume further that the components' marginal reliabilities* p_1, p_2 *and* p_3 *are known. In such a system there are 48 possible conditional probabilities between components 1, 2 and 3:* $p_{2|1}$, $p_{\bar{2}|1}$, $p_{2|\bar{1}}$, $p_{\bar{2}|\bar{1}}$, $p_{1|2}$, $p_{\bar{1}|2}$, $p_{1|\bar{2}}$, $p_{\bar{1}|\bar{2}}$, $p_{3|1}$, $p_{\bar{3}|1}$, $p_{3|\bar{1}}$, $p_{\bar{3}|\bar{1}}$, $p_{1|3}$, $p_{\bar{1}|3}$, $p_{1|\bar{3}}$, $p_{\bar{1}|\bar{3}}$, $p_{3|2}$, $p_{\bar{3}|2}$, $p_{3|\bar{2}}$, $p_{\bar{3}|\bar{2}}$, $p_{2|3}$, $p_{\bar{2}|3}$, $p_{2|\bar{3}}$, $p_{\bar{2}|\bar{3}}$, $p_{3|12}$, $p_{\bar{3}|12}$, $p_{3|\bar{1}2}$, $p_{\bar{3}|\bar{1}2}$, $p_{3|1\bar{2}}$, $p_{\bar{3}|1\bar{2}}$, $p_{3|\bar{1}\bar{2}}$, $p_{\bar{3}|\bar{1}\bar{2}}$, $p_{2|13}$, $p_{\bar{2}|13}$, $p_{2|\bar{1}3}$, $p_{\bar{2}|\bar{1}3}$, $p_{2|1\bar{3}}$, $p_{\bar{2}|1\bar{3}}$, $p_{2|\bar{1}\bar{3}}$, $p_{\bar{2}|\bar{1}\bar{3}}$, $p_{1|23}$, $p_{\bar{1}|23}$, $p_{1|\bar{2}3}$, $p_{\bar{1}|\bar{2}3}$, $p_{1|2\bar{3}}$, $p_{\bar{1}|2\bar{3}}$, $p_{1|\bar{2}\bar{3}}$, $p_{\bar{1}|\bar{2}\bar{3}}$. *If four of these conditional probabilities are known, the remaining 44 can be found using general rules in probability theory.*

Proof. This proof uses Bayes theorem, the rule of complementation and the following rules of total probability:

$$\begin{aligned} p_3 = {} & p_{3|12}p_{2|1}p_1 + p_{3|1\bar{2}}p_{\bar{2}|1}p_1 \\ & + p_{3|\bar{1}2}p_{2|\bar{1}}p_{\bar{1}} + p_{3|\bar{1}\bar{2}}p_{\bar{2}|\bar{1}}p_{\bar{1}} \end{aligned} \tag{A.9}$$

$$p_{3|1} = p_{3|12}p_{2|1} + p_{3|1\bar{2}}p_{\bar{2}|1} \tag{A.10}$$

$$p_{3|2} = p_{3|12}p_{1|2} + p_{3|\bar{1}2}p_{\bar{1}|2} \tag{A.11}$$

Assume that the conditional probabilities $p_{2|1}$, $p_{3|1}$, $p_{3|2}$ and $p_{3|12}$ are known. As shown in Theorem 1, $p_{\bar{2}|1}$, $p_{2|\bar{1}}$, $p_{\bar{2}|\bar{1}}$, $p_{1|2}$, $p_{\bar{1}|2}$, $p_{1|\bar{2}}$, $p_{\bar{1}|\bar{2}}$ can be expressed as functions of p_1, p_2 and $p_{2|1}$. In the same way $p_{\bar{3}|1}$, $p_{3|\bar{1}}$, $p_{\bar{3}|\bar{1}}$, $p_{1|3}$, $p_{\bar{1}|3}$, $p_{1|\bar{3}}$, $p_{\bar{1}|\bar{3}}$ can be expressed as functions of p_1, p_3 and $p_{3|1}$, and $p_{\bar{3}|2}$, $p_{3|\bar{2}}$, $p_{\bar{3}|\bar{2}}$, $p_{2|3}$, $p_{\bar{2}|3}$, $p_{2|\bar{3}}$, $p_{\bar{2}|\bar{3}}$ can be expressed as functions of p_2, p_3 and $p_{3|2}$.

The conditional probabilities $p_{\bar{3}|12}$, $p_{3|\bar{1}2}$, $p_{\bar{3}|\bar{1}2}$, $p_{3|1\bar{2}}$, $p_{\bar{3}|1\bar{2}}$, $p_{3|\bar{1}\bar{2}}$ and $p_{\bar{3}|\bar{1}\bar{2}}$ can further be expressed as functions of p_1, p_2, p_3, $p_{2|1}$, $p_{3|1}$, $p_{3|2}$ and $p_{3|12}$. This is shown in Equations A.12 - A.18. Especially, to express $p_{3|\bar{1}\bar{2}}$ in Equation A.17, Equations A.9, A.13, A.15, A.3 and A.4 are used as basis (the equations are listed in the sequence of their usage).

$$p_{\bar{3}|12} = 1 - p_{3|12} \tag{A.12}$$

$$p_{3|1\bar{2}} = \frac{p_{3|1} - p_{3|12}p_{2|1}}{1 - p_{2|1}} \tag{A.13}$$

$$p_{\bar{3}|1\bar{2}} = 1 - \frac{p_{3|1} - p_{3|12}p_{2|1}}{1 - p_{2|1}} \tag{A.14}$$

$$p_{3|\bar{1}2} = \frac{p_{3|2}p_2 - p_{3|12}p_{2|1}p_1}{p_2 - p_{2|1}p_1} \tag{A.15}$$

$$p_{\bar{3}|\bar{1}2} = 1 - \frac{p_{3|2}p_2 - p_{3|12}p_{2|1}p_1}{p_2 - p_{2|1}p_1} \tag{A.16}$$

$$p_{3|\bar{1}\bar{2}} = \frac{p_3 - p_{3|1}p_1 - p_{3|2}p_2 + p_{3|12}p_{2|1}p_1}{1 - p_2 + (p_{2|1} - 1)p_1} \tag{A.17}$$

$$p_{\bar{3}|\bar{1}\bar{2}} = 1 - \frac{p_3 - p_{3|1}p_1 - p_{3|2}p_2 + p_{3|12}p_{2|1}p_1}{1 - p_2 + (p_{2|1} - 1)p_1} \tag{A.18}$$

In the same way as shown above,

$$p_{2|13} = \frac{p_{3|12}p_{2|1}p_1}{p_{3|1}p_1} \tag{A.19}$$

gives $p_{\bar{2}|13}$, $p_{2|\bar{1}3}$, $p_{\bar{2}|\bar{1}3}$, $p_{2|1\bar{3}}$, $p_{\bar{2}|1\bar{3}}$, $p_{2|\bar{1}\bar{3}}$ and $p_{\bar{2}|\bar{1}\bar{3}}$.
Furthermore,

$$p_{1|23} = \frac{p_{3|12}p_{2|1}p_1}{p_{3|2}p_2} \tag{A.20}$$

is leading to $p_{\bar{1}|23}$, $p_{1|\bar{2}3}$, $p_{\bar{1}|\bar{2}3}$, $p_{1|2\bar{3}}$, $p_{\bar{1}|2\bar{3}}$, $p_{1|\bar{2}\bar{3}}$ and $p_{\bar{1}|\bar{2}\bar{3}}$. □

References

1. T. Aven, Reliability and Risk Analysis, Elsevier, London (1992)
2. Z. W. Birnbaum, On the importance of different components in a multicomponent system, *Multivariate analysis-II, ed. P. R. Krishnaiah*, Academic Press, New York, p. 581–592, (1969)
3. Gustav Dahll and Bjørn Axel Gran, The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems, *International Journal of General Systems*, **29** (2), p. 205–229, (2000)
4. S. Distefano and A. Puliafito, Reliability and availability analysis of dependent-dynamic systems with DRBDs, *Reliability Engineering and System Safety*, **94** (9), pp. 1381–1393, Elsevier, (2009)
5. D. E. Eckhardt and L. D. Lee, A theoretical basis for the analysis of redundant software subject to coincident errors, *NASA tech. Memo 86369*, (1985)

6. J. D. Esary, F. Proshan and D. W. Walkup, Association of Random Variables, with Applications, *The Annals of Mathematical Statistics*, **38** (5), pp. 1466–1474, (1967)
7. Fricks, R.M. and Trivedi, K.S., Modeling Failure Dependencies in Reliability Analysis Using Stochastic Petri Nets, *Proc. of 11th European Simulation Multiconf.*, Citeseer, (1997)
8. S. S. Gokhale and K. S. Trivedi, Dependency Characterization in Path-Based Approaches to Architecture-Based Software Reliability Prediction, *IEEE Workshop on Application-Specific Software Engineering and Technology*, IEEE Computer Society, pp. 86–90, (1998)
9. K. Goseva-Popstojanova and K. S. Trivedi, Architecture-based approach to reliability assessment of software systems, *Performance Evaluation*, Elsevier, **45** (2-3), pp. 179–204, (2001)
10. D. Hamlet, D. Mason, and D. Woit, Theory of Software Reliability Based on Components, *International Conference on Software Engineering*, **23**, pp. 361–370, (2001)
11. S. Hauge, M. A. Lundteigen, P. R. Hokstad and S. Haabrekke, Reliability Prediction Method for Safety Instrumented Systems - PDS Method Handbook, *Sintef report no. A13503*, **ISBN 978-82-1404850-4**, Sintef, (2010)
12. () R. Huang and M. R. Lyu and K. Kanoun, Simulation Techniques for Component-Based Software Reliability Modeling with Project Application, *Proceedings of International Symposium on Information Systems and Engineering (ISE'01)*, CSREA Press, pp. 283–289, (2001)
13. R. A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis, Prentice Hall, New York, **ISBN 0-13-834194-x**, (1998)
14. J. C. Knight and N. G. Leveson, An experimental evaluation of the assumption of independence in multiversion programming, *IEEE Transactions on Software Engineering*, **12** (1), pp. 96–109, (1986)
15. S. Krishnamurthy and A. Mathur, On the Estimation of Reliability of a Software System Using Reliabilities of its Components, *Proceedings of the 8th International Symposium on Software Reliability Engineering (ISSRE'97)*, IEEE Computer Society Press, pp. 146–155, (1997)
16. M. Kristiansen, Finding Upper Bounds for Software Failure Probabilities - Experiments and Results, *Proceedings of the 24th International Conference on Computer Safety, Reliability and Security (Safecomp'05)*, winther, Gran & Dahll, Balkema, pp. 179–193, (2005)
17. M. Kristiansen and R. Winther, Assessing Reliability of Compound Software, *Risk, Reliability and Social Safety (ESREL 2007)*, Aven & Vinnem, Taylor & Francis Group, pp. 1731–1738, (2007)
18. M. Kristiansen and R. Winther and John E. Simensen, Identifying the Most Important Component Dependencies in Compound Software, *Risk, Reliability and Safety (ESREL 2009)*, Bris, Soares & Martorell, Taylor & Francis Group, pp. 1333–1340, (2009)
19. M. Kristiansen and R. Winther and M. van der Meulen and M. Revilla, The Use of Metrics to Assess Software Component Dependencies, *Risk, Reliability and Safety (ESREL 2009)*, Bris, Soares & Martorell, Taylor & Francis Group, pp. 1359–1366, (2009)
20. Building a system failure rate estimator by identifying component failure rates, *Proceedings of the 10th International Symposium on Software Reliability Engineering (ISSRE'99)*, pp. 32–41, (1999)
21. B. Littlewood and D. R. Miller, Conceptual Modeling of Coincident Failures in Multiversion Software, *IEEE Transactions on Software Engineering*, **15** (12), pp. 1596–1614, (1989)

22. B. Littlewood, P. Popov and L. Strigini, Modelling software design diversity: a review, *ACM Computing Surveys*, **33** (2), pp. 177–208, (2001)
23. M. R. Lyu, editor. *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, **ISBN 0-07-039400-8**, (1995)
24. Douglas C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, **ISBN 0-471-15746-5**, (1997)
25. John D. Musa, Software Reliability Engineering, McGraw-Hill, **ISBN 0-07-913271-5**, (1998)
26. D. M. Nassar, W. A. Rabie, M. Shereshevsky , N. Gradetsky, H. H. Ammar, B. Yu, S. Bogazzi, and A. Mili, A., Estimating Error Propagation Probabilities in Software Architectures, *International Symposium on Software Metrics No10, Chicago IL, ETATS-UNIS*, pp. 384–393, Citeseer, (2004)
27. B.Natvig, Reliability analysis with technological applications (in Norwegian), Institute for Mathematics, University of Oslo, (1998)
28. Petar Popic, Dejan Desovski, Walid Abdelmoez and Bojan Cukic, Error Propagation in the Reliability Analysis of Component based Systems, *Proceedings of the 16th IEEE International Symposium on Software Reliability (ISSRE'05)*, IEEE Computer Society, pp. 53–62, (2005)
29. S. S.Gokhale, Architecture-based software reliability analysis: Overview and limitations, *IEEE Transactions on dependable and secure computing*, **4** (1), pp. 32–40, (2007)
30. L. Sha, J. B. Goodenough and B. Pollak, Simplex architecture: Meeting the challenges of using COTS in high-reliability systems, *Crosstalk*, Citeseer, pp. 7–10, (1998)
31. Vieira, M. and Richardson, D., The role of dependencies in component-based systems evolution, *Proceedings of the International Workshop on Principles of Software Evolution*, ACM, pp. 62–65, (2002)
32. R. Winther and M. Kristiansen, On the Modelling of Failure Dependencies Between Software Components, *Safety and Reliability for Managing Risk (ESREL'06)*, Guedes Soares & Zio, Taylor & Francis Group, pp.1443-1450, (2006)
33. L. Zavala and M.N. Huhns, Analysis of coincident failing ensembles in multi-version systems, *Presented at the 19th IEEE International Symposium on Software Reliability Engineering - Dependable Software Engineering Workshop (ISSRE'08)*, IEEE Computer Society, (2008)