Uncertainty Explicit Assessment of Off-The-Shelf Software: A Bayesian Approach

Ilir Gashi, Peter Popov, Vladimir Stankovic

Centre for Software Reliability,
City University,
Northampton Square
London EC1V 0HB
United Kingdom
Tel: 020 7040 0273, 020 7040 8963, 020 7040 0273,
Fax: 020 7040 8585
http://www.csr.city.ac.uk
{I.Gashi, V.Stankovic}@city.ac.uk,
{ptp}@csr.city.ac.uk

Abstract. Assessment of software COTS components is an essential part of component-based software development. Poorly chosen components may lead to solutions of low quality and that are difficult to maintain. The assessment may be based on incomplete knowledge about the COTS component itself and other aspects (e.g. vendor's credentials, etc), which may affect the decision of selecting COTS component(s). We argue in favor of assessment methods in which uncertainty is explicitly represented ('uncertainty explicit' methods) using probability distributions. We provide details of a Bayesian model, which can be used to capture the uncertainties in the simultaneous assessment of two attributes, thus, also capturing the dependencies that might exist between them. We also provide empirical data from the use of this method for the assessment of off-the-shelf database servers which illustrate the advantages of 'uncertainty explicit' methods over conventional methods of COTS component assessment which assume that at the end of the assessment the values of the attributes become known with certainty.

Keywords: ***COTS component assessment, reliability and performance assessment, Bayesian inference***

1. Introduction

The use of commercial-off-the-shelf (COTS) components in software development is ubiquitous. There are many benefits to using COTS components stemming from the incentive to cut down on cost and development time and to improve quality by using tried and tested components. An essential part of component-based software development is the assessment of available COTS components. Various assessment methods have been proposed [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. The results of these assessment techniques crucially depend on assuming that the values of the assessed attributes will be known with *certainty* at the end of the assessment. However, since the assessment is carried out with limited resources of time and budget the outcome is subject to *uncertainty*.

We propose an assessment method, in which the assessment results are subject to explicitly stated uncertainty and discuss how this may impact the selection of COTS software. The method also enables representing the dependencies that exist between the uncertainties associated with the values of the COTS component attributes, which affect the decision about which of the available COTS components to choose. It also encourages assessing the dependent attributes *simultaneously*, thus identifying the dependencies that may exist between the values of the attributes which may affect the choice of component(s). We provide empirical results from a study with off-the-shelf database servers, which demonstrate how the assessment method can be used in practice.

The paper is structured as follows: section 2 contains a brief review of related work on COTS component assessment and attribute definitions; section 3 contains an overview of the problems that need to be addressed during COTS component assessment; in section 4 we describe models of assessment, in which model parameters (values of the attributes to be assessed) are not known with certainty and argue in favor of using probability distributions as an adequate mechanism to capture this uncertainty; in section 5 we give details of an empirical study with off-the-shelf database servers and also some contrived numerical examples, which illustrate the advantages of handling uncertainty and dependence between the values of the attributes; section 6 contains a discussion of the scalability and applicability of the method proposed; and finally in section 7 we present conclusions and possible further work.

2. Related Work

2.1 COTS Assessment Methods

There is a wide variety of COTS component assessment approaches available. All of them start with an initial statement of requirements, which defines what is being sought. It has been proposed that the requirements initially should not be too stringent, since this would discard potentially appropriate COTS component candidates at a very early stage [9], [15]. It has even been suggested [15] that if the requirements are not flexible then the COTS-based development may not be appropriate and bespoke development could be more cost-effective. So initially [15] suggests distinguishing between essential requirements and those that are negotiable. The selection criteria are then based on the essential requirements.

Off-the-shelf-option (OTSO) [2] is a multi-phase approach to COTS component selection. The phases are: the search phase, the screening and evaluation phase and the analysis phase. In the first phase COTS components are identified. In the screening and evaluation phase the components are further filtered using a set of evaluation criteria (established from a number of sources, including the requirements specification, the high level design specification etc.). In the analysis phase results of the evaluation are analyzed, which lead to the final selection of COTS components for inclusion in the system. Other similar multiphase process approaches for COTS component evaluation that have been proposed include CEP (Comparative Evaluation Process Activities) [7], CBA Process Decision Framework [8] which in addition to defining a process for COTS component assessment also defines two other processes: COTS integration (“gluing”) and COTS configuration (“tailoring”); CAP-COTS Acquisition Process method [5] and PECA Process [13].

Procurement-oriented requirements engineering (PORE) [1] is a process in which requirements are defined in parallel with COTS component evaluation and selection. [1] propose using prototypes to develop knowledge

concerning COTS components and their use within the wider system. Other methods that are centred on the requirements to assist with the COTS component selection process are CRE-COTS-Based Requirements Engineering Method [6], Storyboard Process [11], Combined Selection of COTS Components [12] and COTS-DSS [14].

CISD (COTS-based Integrated System Development) [4] and CDSEM (Checklist Driven Software Evaluation Methodology) [3] are both checklist-based evaluation methodologies. STACE (Socio Technical Approach to COTS Evaluation) [10] is a socio-technical approach to evaluation which builds on work of [1] and [2] and emphasizes the organizational issues related to COTS selection.

2.2 Attribute Definition Methods

Extensive work has been also reported on definition of COTS component assessment attributes. A comprehensive list is given in [16]. They group the attributes in two categories depending on how they can be measured: Attributes Measurable at Runtime (which contain Accuracy, Security, Recoverability, Time Behavior and Resource Behavior) and Attributes Measurable during Component Life-Cycle (Suitability, Interoperability, Maturity, Learnability, Understandability, Operability, Changeability, Testability and Replaceability). These attributes are further divided into more fine-grained attributes, which are measurable using their proposed metrics of: presence, time, level and ratio. This work [16] follows the spirit of the guidelines for attribute definitions outlined by the international standardizing organizations ISO [17], and IEEE/ANSI [18] in a broader context, not specific to COTS component attributes. COCOTS framework by Abts et al. [19], and Torchiano and Jaccheri [20] also provides COTS attribute definitions.

3. Problems with COTS Component Assessment

3.1 Motivation

Any assessment is conducted with limited resources and under various assumptions, which may not hold true in real operation. As a result the outcome of the assessment is subject to uncertainty – the assessor may never be 100% sure that what they concluded during the assessment (both about the values of the attributes as well as the choice of a COTS component) will be confirmed when the COTS component is used in operation. This is clearly true for some parameters, which can be estimated *objectively*, e.g. failure rate, performance, etc. For failure rate, for instance, even after a very thorough testing one can only identify a range of rates which are more likely than others. For instance, Littlewood and Wright have shown [21] that starting with indifference between the values of the failure rate (i.e. uniform distribution of the failure rate in the range [0, 1]) and seeing a protection system process correctly 4603 demands translates into 99% confidence that this system's probability of failure on demand (*pfd*) is no worse than 10^{-3}. The same equally applies to attributes assessed subjectively, e.g. using the Likert scale [22], widely used in the COTS component assessment. It may be difficult for an assessor to justify that a COTS component must be ranked at exactly, say 7 out of 10, according to a chosen scale but he/she may be certain that the 'true' value of the attribute is in the range [6, 7].

The value of expressing the assessment results in the form (value, confidence) has been recognized in some other technical areas which dealt with assessment. The best performing software reliability-growth models (RGM) which predict the failure rate from the observed failures in the past, for instance, are those in which the model parameters are treated as *random variables* [23]. In these models the 'true' values of the attributes being assessed are never assumed known with certainty. Instead the attribute is characterized by a probability distribution, from which the true value of the attributes will come (i.e. are seen as drawn at random). For each reliability target, then, the assessor can tell the probability that the true reliability is lower than the target. Such models *systematically outperformed* the alternative simplistic methods in which the parameters were assumed to be known with certainty [24]. If the 'uncertainty explicit' models have been best with one specific method of assessment – software reliability – it seems natural to try similar 'uncertainty explicit' methods for other assessments, e.g. evaluation of COTS software and selecting the best out of a set of comparable alternatives. This is the focus of this paper.

There are various methods for representing uncertainty [25]. Bayesian approach to probabilistic modelling is one of the best-known ones and used with some success in reliability assessment [24], [21]. It allows one to combine, in a mathematically sound way, the prior belief (which may be 'subjective') about the values of a parameter or a set of parameters to be assessed with the ('objective') evidence from seeing the modelled artefact in operation. Combining the prior belief and the evidence from the observations in a mathematically correct way leads to a posterior belief about the values of the assessed attribute(s).

How does 'uncertainty explicit' assessment differ from the conventional deterministic assessment? With deterministic assessment point estimates of the attributes are used. A common approach of comparing the alternatives is then to use a *weighted sum* of the estimates for each of the alternatives. When uncertainty is accounted for, this approach is still possible – we can use various characteristics of the posterior distributions (mean, median, etc.) of the attributes as estimates and then calculate the weighted sum for each of the COTS components included in the assessment before deciding which is the best one. When uncertainty is explicitly used in the assessment, however, more refined ways of comparison are possible: from the posterior one can express *the uncertainty in the value of the comparison criterion*, e.g. the weighted sum of the attributes. Since the value of the weighted sum is now uncertain we have a range of options. We may give preference to the COTS component for which the mean (median) value of the weighted sum is the best (as we would have done with point estimates of the attributes). With uncertainty stated explicitly a range of new options exists, which is illustrated in Fig. 1.

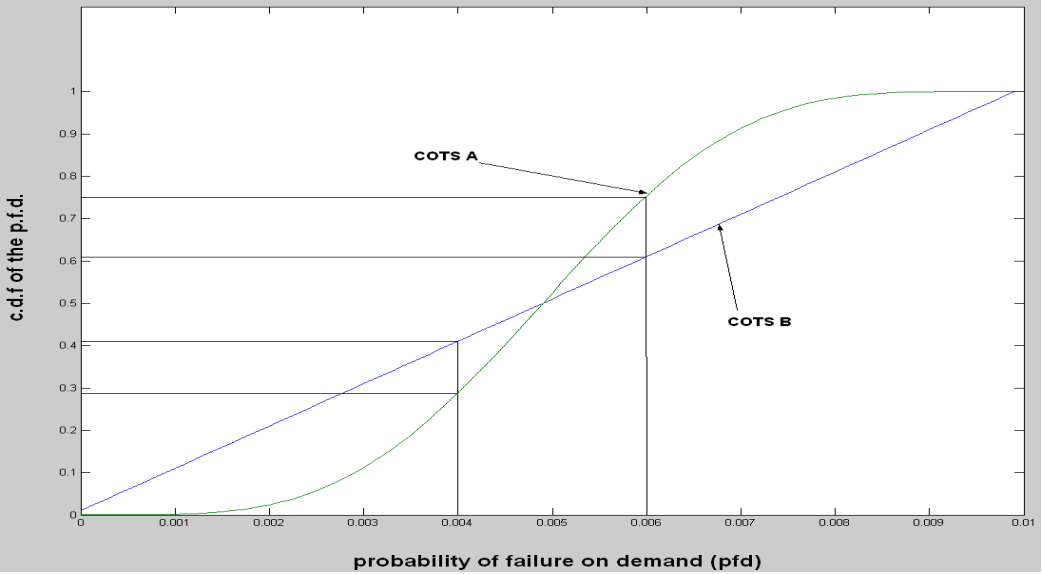

Fig. 1 - The *pfd* for two different COTS components

In this figure we illustrate the value of handling uncertainty explicitly even when dealing with a single assessment attribute, COTS component reliability. Let us assume that Fig. 1 illustrates the cumulative distribution function (c.d.f.) graph for two COTS components with the same average *pfd*. If we wanted to choose the COTS component that has the highest probability of having a *pfd* of no worse than $6*10^{-3}$ (i.e. the value of the x-axis of 0.006) then we would choose COTS component A, whereas the COTS component with the highest probability of having a *pfd* of no worse than 4.10^{-3} is COTS component B. We can also see clearly that the distribution of the *pfd* of COTS component B is much more spread than that of COTS component A (in fact the distribution of COTS component B is uniformly spread across all the values from 0 to 10^{-2}). Therefore there is a much higher uncertainty associated with the values of COTS component B than that of COTS component A. Stating uncertainty explicitly offers the assessor a wider range of options in selecting the most appropriate COTS component.

3.2 Dependence among Attributes

COTS component assessment requires dealing with multiple attributes of the COTS components being compared. The selection of a particular COTS component, thus, is a multi-criteria decision which taken under uncertain values of the attributes naturally leads to the question about the dependence between the uncertainties associated with the individual attributes. Ignoring the possible dependence between the uncertainties in attributes' values represents a particular form of belief: that assessing attribute X one can learn *nothing* about another attribute, Y. For example, performance of a COTS component will hardly tell anything about the quality of its documentation and vice versa. It is quite obvious, however, that not all COTS component attributes are like that. In many cases while assessing an attribute X the assessor may infer something about the values of another set of attributes. For instance, if we devise a prototype in order to assess the *functionality* of a COTS component we will also learn something about the *performance* (how quickly this COTS component responds to requests) and how *reliable* the COTS component is. A more subtle, but very useful concept, as we will see later, is that the uncertainties associated with the assessed attributes may be *dependent*. Informally, assume that we want to

assess the reliability and performance of a COTS component. We may assume that the uncertainties associated with these two attributes are independent, in the statistical sense. Under this assumption learning something about reliability will tell us nothing about performance and vice versa. Now suppose that we have run a very long testing campaign and have repeatedly observed that whenever the response was late it was also incorrect and no other incorrect response has been observed. With such evidence of a strong positive correlation between the failures (incorrect responses) and the responses being late, we may accept that any change of our belief about the rate of failure should also be translated into a change in our belief about the rate of late responses. The assessment models surveyed invariably assume that the values of the attributes are independent and do not allow for dependencies between their uncertain values to be captured adequately.

In summary, with the assessment method that we propose in this paper we aim to handle both the *uncertainties* in the values of the attributes and the *dependence* that exists between the values of the different attributes. The existing assessment methods we surveyed do not deal explicitly with either of these two uncertainties.

4. Assessment of COTS Components: a Bayesian Approach

In this section we briefly summarize how the Bayesian approach to assessment is normally applied to assessment of a single attribute. Assume that the attribute of interest is the component's probability of failure on demand (*pfd*). If the system is treated as a black box, i.e. we can only distinguish between COTS component's failures or successes, the Bayesian assessment proceeds as follows. Let us denote the system *pfd* as *p*, with prior distribution (probability density function, *pdf*) $f_p(\bullet)$, which characterises the assessor's knowledge about the COTS component *pfd prior* to observing the COTS component in operation. Assume further that the COTS component is subjected to *n* demands, independently drawn from a 'realistic' operational environment (profile)[1], and *r* failures are observed. The posterior distribution, $f_p(x \mid r,n)$, of *p* after the observations will be:

$$f_p(x \mid r,n) \propto L(n,r \mid x) f_p(x), \tag{1}$$

where $L(n,r \mid x)$ is the *likelihood* of observing *r* failures in *n* demands *if* the *pfd* were exactly *x*, which in this case of independent demands is given by the *binomial* distribution, $L(n,r \mid x) = \binom{n}{r} x^r (1-x)^{n-r}$. For any prior and any observation (*r*, *n*) the posterior can be calculated[2] for all the COTS components included in the assessment. Even if no failure is observed (i.e. $r = 0$), the posterior can be calculated. Other measures of interest can also be derived from this posterior, e.g. the probability that the COTS component will survive the next 5000

[1] An operational profile [26] can be defined as a statistically accurate characterization of how the component will be used in its 'true' environment.

[2] The posterior can be calculated either by using a conjugate prior distribution [27], in which case the posterior distribution is guaranteed to be in the same family as that of the prior for a given likelihood function (e.g. Beta distribution prior, with Binomial Likelihood function, gives us a Beta distributed posterior) or it can be calculated through numerical methods and approximations. In our case, since the conjugate family has limitations [28] we have used numerical methods to calculate the posterior.

randomly chosen demands. This probability can be calculated for each of the COTS components included in the assessment as follows:

$$\int_0^\infty (1-p)^{5000} f_p(p \mid r,n) dp \,.$$

Then the best COTS component will be the one, for which the integral above gets a maximum value.

4.1. A Model for Assessment of 2 Non-Independent Attributes

Typically, COTS component assessment is a multi-criteria decision with dozens of attributes usually assessed and taken into account (as detailed in [2], [1], [5], [8]). The Bayesian assessment can be applied to multiple attributes, too. For simplicity we first demonstrate the approach with two attributes and then discuss (in section 5) the implications of scaling it up to many attributes. A similar model to the one we describe here has been used in the past in assessing reliability of various systems built with components [28], [29].

Let us assume that two non-functional attributes must be assessed, such as the COTS component's *pfd* and performance, the latter assessed in the form of whether the response is received on time or not, i.e. the probability of a *late response* on demand, *pld*. Using a binary score – on time vs. late – is an adequate approach when the COTS component is planned for integration in a larger system. In these circumstances using an absolute scale, e.g. how long it takes a COTS component to respond to a demand, may be unnecessary: it will be sufficient to know whether the response is received with an *acceptable delay* as dictated by the wider system. In terms of comparison of several COTS components using the binary scale (on time/late) seems also adequate. Any COTS component, which responds with an acceptable delay, is sufficiently good from the point of view of the system's integrator.

Here we define a model to help with the comparison of COTS components assessed by subjecting them to a *series of independently selected demands*. Both, the COTS component's *pfd* and *pld*, are used in the comparison and different comparison criteria are discussed.

On each demand the response received from the COTS components is evaluated from two different viewpoints: correct/incorrect and on time/late. Clearly 4 combinations exist, which can be observed on a randomly chosen demand, as shown in Table 1.

Table 1 – The outcomes, their frequencies and probabilities for a random demand.

Event	Correct Response (Reliability)	Response On-Time (Performance)	Number of observations in *n* demands	Probability
α	No	Yes	r_1	p_{10}
β	Yes	No	r_2	p_{01}
χ	No	No	r_3	p_{11}
δ	Yes	Yes	r_4	p_{00}

The four probabilities given in the last column sum to 1. So if the first three probabilities are 0.2, 0.4 and 0.3, respectively, then the last one p_{00} = 1 - (0.2 + 0.4 + 0.3) = 0.1. This constraint remains even if we treat the probabilities in Table 1 as random variables: their sum will always be 1. Thus, the joint distribution of any three

of these parameters, e.g. $f_{p_{01},p_{10},p_{11}}(\bullet,\bullet,\bullet)$, gives an exhaustive description of the COTS component's behaviour. In statistical terms, the model of the COTS component with two binary attributes has three degrees of freedom.

The marginal probabilities of getting an incorrect response on a random demand, let's denote it p_I, and of getting the response late, p_L, respectively, can be expressed as:

$p_I = p_{10} + p_{11}$ and $p_L = p_{01} + p_{11}$.

p_{11} represents the probability of receiving late an incorrect response. Hence, the notation $p_{IL} \equiv p_{11}$, $p_I = p_{10} + p_{11}$ and $p_L = p_{01} + p_{11}$ will capture better the intuitive meaning of the event it is assigned to. Instead of using $f_{p_{10},p_{01},p_{11}}(\bullet,\bullet,\bullet)$ another distribution, which can be derived from it through a *change of variables* [30], can be used. In this section we use $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet)$ which can be factorised as:

$$f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet) = f_{p_I,p_L}(\bullet,\bullet) f_{p_{IL}}(\bullet \mid p_I, p_L) \tag{2}$$

For the prior joint distribution $f_{p_I,p_L}(\bullet,\bullet)$ above, we assume throughout this paper that the p_I and p_L are independently distributed[3]. We capture the possible dependencies between the two failures processes (characterized by p_I and p_L, respectively) by $f_{p_{IL}|p_I,p_L}(\bullet)$. Hence the full joint prior distribution is given by:

$$f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet) = f_{p_I}(\bullet) f_{p_L}(\bullet) f_{p_{IL}}(\bullet \mid p_I, p_L) \tag{3}$$

For a given observation (r_1, r_2, and r_3 in N demands) the posterior joint distribution can be calculated as:

$$f_{p_I,p_L,p_{IL}}(x,y,z \mid N, r_1, r_2, r_3) = \frac{f_{p_I,p_L,p_{IL}}(x,y,z) L(N, r_1, r_2, r_3 \mid p_I, p_L, p_{IL})}{\iiint_{p_I,p_L,p_{IL}} f_{p_I,p_L,p_{IL}}(x,y,z) L(N, r_1, r_2, r_3 \mid p_I, p_L, p_{IL}) dx dy dz} \tag{4}$$

where

$$L(N, r_1, r_2, r_3 \mid p_I, p_L, p_{IL}) = \frac{N!}{r_1! r_2! r_3! (N - r_1 - r_2 - r_3)!} (p_I - p_{IL})^{r_1} (p_L - p_{IL})^{r_2} p_{IL}^{r_3} (1 + p_{IL} - p_I - p_L)^{N - r_1 - r_2 - r_3} \tag{5}$$

is the multinomial likelihood of the observation (r_1, r_2, r_3 , N).

4.2. Combination of Uncertainties in the Values of Attributes

For comparison of the COTS components we will define the following criterion:

Probability of an *inadequate response*, P_{Ser}, by the COTS component: of getting either an incorrect or late response. Clearly, $P_{Ser} = P_I + P_L - P_{IL}$. Its posterior distribution, $f_{p_{Ser}}(\bullet \mid N, r_1, r_2, r_3)$, can be derived from

[3] The plausibility of making this independence assumption is explained in Appendix A.

the joint posterior, $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet \mid N, r_1, r_2, r_3)$, by first transforming it, to for example $f_{p_I,p_L,p_{Ser}}(\bullet,\bullet,\bullet \mid N, r_1, r_2, r_3)$, and then integrating out the nuisance parameters P_I and P_L.

An often used selection method [31] in the literature is the weighted sum of the values of the attributes. The weighted sum of the two attributes in our study can be calculated as follows: $P_S = kP_I + (1-k)P_L$, in which the constant k is defined by the assessor in the range 0-1. High values of k correspond to cases when incorrect results are highly undesirable while late results may be tolerable. On the other hand, low values of k correspond to cases when incorrect results may be tolerated by the system while late responses may have serious consequences. In order to derive the marginal distribution of P_S first the joint distribution $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet \mid N, r_1, r_2, r_3)$ is transformed to $f_{p_I,p_L,p_S}(\bullet,\bullet,\bullet \mid N, r_1, r_2, r_3)$ and then the nuisance parameters P_I and P_L are integrated out, as above for P_{Ser}. However we will not be using this method of selection since the new variable P_S does not have an obvious intuitive meaning. The difficulty is compounded in our case since the uncertainty is stated explicitly. It is impossible to say what a confidence of say 99% associated with a particular value of P_S tells us about the COTS component being assessed.

4.3. Partitioning the Demand Space

In some areas of software engineering, especially in testing, the usefulness of *partitioning the demand space* has been recognised [32], [33], [26]. The demand space partitions typically represent *different types of demands*, which may have different likelihoods of occurring in a realistic environment. Realistic testing, thus, would require generating mixes of demands, which take into account the likelihood of the types of demands.

In our context, operating in a partitioned demand space may imply that the uncertainty associated with the attributes of interest may differ among the partitions, e.g. as a result of different number of observations being made for the different partitions.

If the demand space is partitioned into M partitions $\{S_1, S_2, \ldots S_M\}$, with a probabilistic measure $\{ P(S_1),..., P(S_M)\}$ [4], then for each of these the assessment will be performed as described above, e.g. with two attributes the description provided in section 4.1 will apply. As a result M *conditional distributions* will be associated with each COTS component, e.g. using reliability and performance these can be denoted as $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet \mid S_i)$, from which the conditional distribution $f_{p_{Ser}}(\bullet \mid S_i)$ will be expressed. This distribution characterises the probability of failure (incorrect or late response), $P_{Ser} \mid S_i$, of the particular COTS component in the specific partition. Finally, in order to compare the competing COTS components the *unconditional distribution* $f_{p_{Ser}}(\bullet)$ should be derived for the particular profile defined over the set of partitions, which represents the targeted operational environment.

[4] The meaning of these random variables is that a demand chosen at random with probability $P(S_i)$ will be drawn from S_i.

Let us assume the profile of the targeted environment is *known with certainty*[5]. The marginal probability of failure of a COTS component, according to the formula of full probability is:

$$P_{Ser} = \sum_{i=1}^{M} P_{Ser} \mid S_i \times P(S_i) \tag{6}$$

The distribution of this random variable, P_{Ser}, depends on the joint distribution, $f_{(P_{Ser}|S_1),...,(P_{Ser}|S_M)}(\bullet,...,\bullet)$, i.e. of the conditional probabilities of failure in sub-domains. In some setups it may be plausible to assume that the conditional probabilities of failure (in the partitions that is) are *independently distributed*, i.e.:

$$f_{(P_{Ser}|S_1),...,(P_{Ser}|S_M)}(\bullet,...,\bullet) = \prod_{i=1}^{M} f_{P_{Ser}|S_1}(\bullet)... f_{P_{Ser}|S_M}(\bullet). \tag{7}$$

Such an assumption represents the assessor's belief that learning something about the probability of failure, $P_{Ser} \mid S_i$, of a particular COTS component in partition *i* will not change their belief about the probability of failure, $P_{Ser} \mid S_j$, of the same COTS component in another partition. The assumption is consistent with applying inferences to the individual partitions, i.e. conditional on the demands coming from a particular partition.

Under (7) the unconditional probability of COTS component failure (6) can be expressed as a *convolution* of the distributions of the random variables $P_w(i) = P_{Ser} \mid S_i \times P(S_i)$, i.e.:

$$P_{Ser}^{w} = \otimes P_w(i) \tag{8}$$

The selection of the best COTS component, out of the available alternatives, will then be based on the marginal distributions, $f_{p_{Ser}^{w}}(\bullet)$, associated with the available COTS components.

5. Numerical Examples: a Study with Off-The-Shelf Database Servers

We have reported recently results of studies on dependability and performance of database servers [34], [35], [36], [37]. The focus of these earlier studies was in measuring the amount of "diversity", in both correctness and response time, which exists between different servers, i.e. certain servers might give an incorrect and/or late response in one input but the others might not. The motivation behind this work was to get preliminary measurements on the improvements in reliability and performance that can be had from using more than one component in parallel in a multi-channel diverse configuration.

In this paper we will use the data collected in those studies to demonstrate our approach to COTS component selection. SQL servers are a very complex category of off-the-shelf components, with many reported faults in

[5] This assumption is needed for the comparison only. We do not require here that we know the 'real' operational environment, in which the system together with the chosen COTS component will be deployed. Taking into account the uncertainty about the profile is possible at the expense of making the calculations more complicated, which is beyond the scope of this paper.

each release. In total six off-the shelf SQL servers from four different vendors were used. Four of the servers are open-source, namely PostgreSQL 7.0, PostgreSQL 7.2, Interbase 6.0 and Firebird 1.0[6]. The other two servers are commercial closed development servers, anonymised here due to the restrictive 'End User License Agreements'. We will refer to these components as CS1 (Commercial Server 1) and CS2 as they are from different vendors.

An ideal selection of an SQL server based on the results of statistical testing of the COTS components may be problematic in practice. We will highlight two circumstances in which these difficulties can occur:

- Assume that we are interested in choosing between several SQL servers, based on their reliability and performance. The ideal situation for choosing the most appropriate SQL server based on measurements *after* deploying the COTS components is clearly *unrealistic* since we would like to select the best server *before* the application is developed.
- Assume that the system integrator (e.g. a software house) would like to make a *strategic* choice of a SQL server for use in the foreseeable future. In this scenario the application(s), which may be developed in the future may be even unknown at the time of making the selection.

Given these difficulties we can use alternative options:

- Use well-known benchmark applications. In the context of SQL servers this might be the TPC-C benchmark for on-line transaction processing [38]. In this case, the performance of the components can be measured directly on the target platform, but there might be problems observing failures. This is because it would be reasonable to expect that an SQL server would correctly process the statements defined in the TPC-C benchmark application. Thus, in this case the selection of the SQL server would be significantly influenced by the performance attribute. Even if failures are observed, such a measurement of the reliability of the COTS components may be very expensive; the likely candidates to choose from will be reliable components. Thus the amount of testing to observe a few failures may be prohibitively high [39]. We illustrate the assessment method with data collected from experiments with an implementation of the TPC-C client benchmark. For the TPC-C experiments we used all six aforementioned SQL servers.
- Use *stressful environments* (in terms of the reliability attribute) for comparing the components, i.e. environments which increase the likelihood of failures occurring, even if we do not know how likely these are to occur in operation. The set of bugs of a particular COTS component (in our case SQL server) defines one such stressful environment for a server. The union of the bugs reported for all the compared COTS components will form a demand space, in which there will be a partition stressing each of the components. We have collected known bug reports for four of the SQL servers in our studies, namely PostgreSQL 7.0, Interbase 6.0, CS1 and CS2 and used them as a sample from a 'stressful' environment, in which to compare the COTS components.

Detailed results for each of these studies are given in the next two sub-sections. We did not use *partitioning of the demand space* approach in the study with the TPC-C benchmark application (even though the TPC-C transactions types could form basis for such partitioning). This is because we did not have any reason to expect that the servers will perform differently (in terms of timeliness and correctness) for each transaction type. We however did use partitioning of the demand space in the study with the bug reports of the servers, since we had compelling reasons to expect that the servers will perform differently (this will be explained in section 5.2).

[6] Firebird is the open-source descendant of Interbase. The later releases of Interbase are issued as closed-development by Borland.

5.1 Study with TPC-C Benchmark Application

We first describe the results obtained using the TPC-C benchmark application as a basis of selecting the best SQL server. In the empirical study we used our own implementation of TPC-C. The benchmark defines five transaction types (New-Order, Payment, Order-Status, Delivery and Stock-Level) and sets the probability of execution for each, i.e. the particular transaction mix (profile) is defined. The specified performance measure is the number of *New-Order* transactions completed per minute. However, our measurements were more detailed - we recorded the transaction response times instead. The benchmark specifies an upper bound on the 90^{th} percentile values for each transaction type. It requires that the average response time of each transaction type is less than or equal to the respective 90^{th} percentile value. The values are as follows:

- New-Order - 5 seconds
- Payment - 5 seconds
- Delivery – 80 seconds
- Order-Status – 5 seconds
- Stock-Level – 20 seconds

The test harness consisted of two machines:

- A server machine, on which one of the six database servers was run.
- A client machine, which executed a JAVA implementation of the TPC-C standard.

Each experiment comprised the *same sequence* of 1000 transactions. We ran two types of experiments:

- *single client* - a TPC-C compliant client modifies the database by executing the specified transaction mix.
- *multiple clients* - a TPC-C compliant client modifies the database and additional 10 clients concurrently execute the read-only transactions (Order-Status and Stock-Level).

Multiple clients experiment enabled us to increase the load on the servers and measure the effect of the increased load on their performance.

A timeout value, specific to each transaction type, was used to distinguish between late and timely responses. We defined two sets of timeouts[7]:

- The 90^{th} percentile values specified by TPC-C (*TPC-C timeout*),
- One fifth of the 90^{th} percentile values (*short timeout*).

We defined four scenarios, varying the number of clients and timeout values respectively:

- Scenario 1 - single client / TPC-C timeouts
- Scenario 2 - single client / short timeouts
- Scenario 3 - multiple clients / TPC-C timeouts
- Scenario 4 - multiple clients / short timeouts

The SQL servers were compared for each of the scenarios.

[7] The choice of these was made after a personal communication of one of the authors with a TPC-C affiliate and auditor who confirmed that the values were conservative for a wide range of on-line transaction processing applications.

5.1.1 Prior Distributions

The prior, $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet)$, was constructed under the assumption that P_I and P_L are independently distributed random variables, i.e. $f_{p_I,p_L}(\bullet,\bullet) = f_{p_I}(\bullet)f_{p_L}(\bullet)$. We made this assumption since we did not have any objective evidence to believe otherwise. In case there are reasons (objective or subjective) then the assumption of independence may be dropped. In this case the particular form of $f_{p_I,p_L}(\bullet,\bullet)$ should be defined explicitly. Additionally the conditional distributions $f_{p_{IL}|P_I,P_L}(\bullet \mid P_I, P_L)$ were defined for every pair of values of P_I and P_L, in the range [0, min(P_I, P_L)] since the probability of incorrect and late responses cannot be greater than the probability of *either* of the two individually. In passing we note that the choice of the prior is not critical with the benchmark application since an arbitrarily large number of demands can be generated, i.e. 'the data will speak for itself'.

We anticipated observing mainly late responses while the incorrect result failures were expected to be very rare. We have assumed 'ignorance prior' (Uniform distribution) for performance in the range $P_L \in [0,1]$. For incorrect result failures we have also assumed ignorance but using an upper bound of 10^{-2}, likely to be very conservative in the context of TPC-C, i.e. we used the range $P_I \in [0,10^2]$. We assumed ignorance priors for both P_I and P_L since we did not have any preference regarding their values. In this study we used the same distribution for all the servers since for the scenarios tested we did not have any reason to prefer one server over the others. There might, however, be cases – some discussed later in section 6.4 - whereby the assessor may have different prior beliefs about the competing COTS components.

A summary of the distributions used and the range in which they are defined is given in Table 2.

Table 2 - The Prior distributions (identical for all six servers and all four scenarios).

Prior Distribution	Range	Distribution Type
Reliability $f_{p_I}(\bullet)$	0 – 0.01	Uniform
Performance $f_{p_L}(\bullet)$	0 – 1	Uniform
Conditional distribution: $f_{p_{IL}\mid p_I,p_L}(\bullet \mid P_I, P_L)$	0 – min(P_I,P_L)	Uniform

5.1.2 Observations

The observations from the TPC-C experiments are given in Table 3. The number of demands for all servers is 1000. Five out of six servers exhibit late result failures only. Incorrect result failures are observed only for CS2. In addition, whenever a result was incorrect on CS2 it was late, too. The incorrect results observed were due to the specific concurrency control mechanism used by CS2 [34]. The locks on resources, e.g. database rows, were not released properly when the lock holding transactions were completed. To resolve the problem we had to install timeout watchdogs and abort transactions when the timeout expired. Each aborted transaction was repeated as many times as necessary to eventually commit successfully. We decided to use *transaction repetition count* as the criterion of an incorrect response on CS2. In particular, we defined a threshold of 5 as the value, beyond which the transaction would be considered to have failed.

We used transaction timeout values and *transaction repetition count* to classify each demand on each server in the categories r_1 to r_4 (defined in section 4.1).

Table 3 - The observations of the six database servers for the four scenarios. The number of demands (n) is 1000 for each server. We did not observe any incorrect-only failures, i.e. r_1=0 for all servers.

COTS	Scenario 1			Scenario 2			Scenario 3			Scenario 4		
	r_1	r_2	r_3	r_1	r_2	r_3	r_1	r_2	r_3	r_1	r_2	r_3
PG 7.0	0	1	0	0	30	0	0	0	0	0	644	0
PG 7.2	0	6	0	0	33	0	0	3	0	0	489	0
IB 6.0	0	0	0	0	24	0	0	1	0	0	434	0
FB 1.0	0	0	0	0	1	0	0	0	0	0	439	0
CS1	0	0	0	0	33	0	0	19	0	0	303	0
CS2	0	0	0	0	4	0	0	0	1	0	329	1

5.1.3 Posteriors

The percentiles derived from the posterior distribution for the 4 scenarios are given in Table 4[8]. One can see that the ordering between the servers changes as the number of clients and/or the timeout values vary (to improve the readability of the table we have explicitly shown the ranking order of the servers in each scenario).

Under Scenario 1 (the least demanding scenario) four servers (IB 6.0, FB 1.0, CS1 and CS2) produce identical results since they completed without any failure (i.e. on time and correctly) the 1000 transactions. We are indifferent in the choice among them. The two versions of PostgreSQL exhibit late responses and they are ranked lowest.

When we decrease the timeout value (Scenario 2) the ranking changes: now there are late responses with all the servers. The two worst servers are CS1 and again PostgreSQL 7.2. Interestingly, Firebird 1.0, an open-source server, is ranked the best.

In Scenario 3 the percentile values are close again as in the first scenario, though the earlier version of PostgreSQL, PG 7.0, is ranked the best, alongside Firebird 1.0 while CS1 is the worst performing server. Firebird 1.0 is consistently among the best servers in the first 3 scenarios. An interesting observation is the 50th percentile value of the posteriors CS2 and IB 6.0. Even though the total number of failures for these two servers were the same (1 each, see Table 3), the nature of the failure was different: the result from CS2 was both incorrect and late whereas from IB 6.0 it was only late. Exploring this dependence we can still see a difference in the 50th percentile values of these two servers (even though the difference is marginal and on the chosen accuracy of expressing the percentile values is not observed in the 99th percentile). We will further scrutinize the interplay between the failures of the individual components and the correlation between their failures with contrived examples in section 5.4.

The ranking changes again in the most demanding scenario (Scenario 4). The best server is now CS1.

[8] As we assumed the same priors for all the servers the differences of the posteriors will be solely due to the observations.

Table 4 – Percentiles (abbreviated P-tile) for the distribution of the system quality $P_{Ser} = P_I + P_L - P_{IL}$ classified per scenario. To improve the readability we have also provided the ranking order for each of the servers based on the percentiles values. The prior distribution is the same for all servers across all scenarios.

P-tile	COTS	Prior	Scenario 1		Scenario 2		Scenario 3		Scenario 4	
			Posterior	Rank	Posterior	Rank	Posterior	Rank	Posterior	Rank
0.5	PG 7.0	0.502	0.0021	5	0.0310	4	0.0012	1	0.6436	6
	PG 7.2		0.0071	6	0.0340	5	0.0041	5	0.4888	5
	IB 6.0		0.0012	1	0.0250	3	0.0021	4	0.4340	3
	FB 1.0		0.0012	1	0.0021	1	0.0012	1	0.4392	4
	CS1		0.0012	1	0.0340	5	0.0200	6	0.3032	1
	CS2		0.0012	1	0.0051	2	0.0020	3	0.3300	2
0.99	PG 7.0	0.992	0.0076	5	0.0456	4	0.0060	1	0.6780	6
	PG 7.2		0.0152	6	0.0492	5	0.0108	5	0.5256	5
	IB 6.0		0.0060	1	0.0384	3	0.0076	3	0.4704	3
	FB 1.0		0.0060	1	0.0076	1	0.0060	1	0.4756	4
	CS1		0.0060	1	0.0492	5	0.0324	6	0.3376	1
	CS2		0.0060	1	0.0124	2	0.0076	3	0.3652	2

5.2 Study with the Known Bugs of the Servers

Now we compare the servers using the methodology described in section 4.3. We have collected known bug reports for four SQL servers. We will use the union of the bugs reported for each of these SQL servers. Each of these bug reports will constitute a 'demand' to the server. These demands form a partition of the demand space for each server[9]. In contrast to the TPC-C study where partitioning of the demand space was not used, in the study with the bug reports we apply inferences to the partitions. The reason for doing so was the very different prior beliefs about the behaviour of servers in the different partitions as will be discussed in section 5.2.1. The logs of the *known bugs* are treated as a sample (without replacement[10]) from the corresponding partition (representing the server, for which the bug has been reported). We label the partitions $S_{Server\ name}$. Partition S_X is called an 'own' partition for server X and a 'foreign' partition for any other server Y≠X.

[9] We have observed no bug reported for two or more servers, thus the logs of the known bugs indeed formed partitions of the union of the bugs. Even if we had observed bugs reported from more than one server we could construct a partition of the intersection of the bugs reported for several servers by putting them in their own partition. Thus, a server may have more than one own partition in the demand space and the description provided here will apply.

[10] Strictly, there might be a difference between sampling with and without replacement. Our model is based on sampling without replacement while the inference procedure described in section 4.1 implies sampling with replacement. This is a simplification, which in many cases is acceptable (e.g. sampling from a large population of units, none of which dominates the sampling process, which seems a plausible assumption in our case of SQL servers being very complex products and likely to contain many unknown bugs).

5.2.1 Prior Distributions

The prior distributions $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet \mid S_i)$ used in this study are explained next. The prior distribution, $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet \mid S_i)$, was constructed under similar assumptions to those of the TPC-C study: that P_I and P_L are independently distributed random variables; in the general case of incorrect and late responses being non-independent events, the conditional distributions, $f_{p_{IL}|P_I,P_L}(\bullet \mid S_i, P_I, P_L)$, are specified for every pair of values of P_I and P_L.

The distributions were assumed to be identical for each of the four servers in both their 'own' and 'foreign' partitions. Again, this assumption was made because we did not have objective evidence to believe otherwise. We discuss other options in section 6.4. A summary of the distributions used and their respective parameters including the range of each distribution are given in Table 5, and we will discuss these choices in the rest of this sub-section.

Table 5 – The Prior distributions (identical for all four severs).

	Proportion	Range	Distribution
Reliability	$f_{p_I}(\bullet \mid S_{own})$	0.72 – 1	Uniform
	$f_{p_I}(\bullet \mid S_{foreign})$	0 – 1	Uniform
Performance	$f_{p_L}(\bullet \mid S_{own})$	0 – 1	Uniform
	$f_{p_L}(\bullet \mid S_{foreign})$	0 – 1	Uniform
Conditional distribution: $f_{p_{IL}\mid p_I,p_L}(\bullet \mid S, P_I, P_L)$		0 – min(P_I,P_L)	Uniform

Prior distributions for Incorrect Results $f_{p_I}(\bullet \mid S_i)$

For 'own' partitions the prior distribution was defined as Uniform in the range [L, 1], where L < 1 accounts for the chance that some of the reported bugs might be Heisenbugs[11], i.e. we expect most of the bugs that have been reported for a particular server to cause failures when they are run on that server (hence the probability of observing an incorrect results failure is very close to 1) but, due to Heisenbugs, not always so. As a source for L we used the study by Chandra and Chen [41]. These authors studied the fault reports for three off-the-shelf components: MySQL database server, GNOME desktop environment and the Apache web-server and reported that 5%, 7% and 14%, respectively, of the reported bugs were Heisenbugs. Given the variation between the components we cautiously interpreted these findings by setting L = 1-(2*0.14), that is twice the highest value of Heisenbugs reported, thus the prior is expected to be within the range [0.72, 1]. Notice that here the prior distribution for incorrect results is being defined at a range close to 1 (i.e. high unreliability). This is because of the unusual profile of the demands: since we are using known bug reports as demands we expect most of the bugs to cause failures when we run them on the server for which they were reported.

For 'foreign' partitions, however, the prior distributions were defined as uniform in the range [0, 1]. This is due to the absence of any comparative study to guide our expectation about the likely value. In passing we note that

[11] Gray defines two types of bugs [40]: "Bohrbugs" for bugs that appear to be deterministic (they manifest themselves each time the bug script is executed); and "Heisenbugs" for those that are difficult to reproduce as they only cause failures under special conditions (e.g., created by the internal state affected by the other applications etc.)

theoretical work such as [42], [43] suggest that diverse software versions will tend to fail coincidentally on 'difficult' demands. Since all the bugs are 'difficult' – they are known to be problematic at least for one of the servers – we may consider them genuinely difficult, hence assume as plausible that the other servers too, are likely to fail. On the other hand, empirical studies such as [44], [45], have shown that significant gains can be had via design diversity – hence low chances that a particular server will fail on bugs reported for other server are also plausible. In summary, we are indifferent between the values of the probability that a server will fail from a 'foreign' bug.

Prior Distributions for Performance $f_{p_L}(\bullet \mid S_i)$

We have not found a public domain source, which would allow us to define a prior distribution for performance failures (in the context we have defined). This is also because the number of late results that would be observed would be conditional on how the timeout threshold is set. The only remaining source is to look at the data (either our own or of various vendors) from the experiments using the TPC-C [38] benchmark. However it is not clear how reasonable a prior based on these results would be due to the differences in the profile that will exist between the TPC-C client application and the bug scripts. Therefore we have decided to define the prior distribution for all proportions as uniformly distributed in the range 0 to 1, i.e. be 'indifferent' between the possible chances of the servers exceeding the set timeout.

Prior Distributions for Incorrect and Late Results $f_{p_{IL}|P_I,P_L}(\bullet \mid S_i, P_I, P_L)$

All conditional prior distributions of the probability of a result being at the same time incorrect and late were defined in the range [0, $min(P_I,P_L)$] (since the probability of incorrect and late responses cannot be greater than the probability of *either* of the two individually). This is again due to the rather unique profile, under which we apply the inference and the lack of comparable studies that would enable us to define different priors than assuming 'indifference'.

Priors for Probabilities of a Bug Being Selected From the Partitions

For the comparison of the servers we use a distribution defined on the set of partitions, which does not favour any of the servers, i.e. we assumed that probability of each partition is 0.25 in the study with 4 servers[12].

5.2.3 Observations

The observations using the known bugs of four off-the-shelf servers are given in Table 6. We can see that the number of bugs collected for each server was different, which indicated that the empirical evidence differs between the partitions. The reasons for this was merely differences in the reporting practices operated by the vendors of the servers, e.g. unavailability in the public domain of fully reproducible bug scripts for the commercial servers (especially CS1). Therefore, the sizes of the samples from the partitions on each server are different. Additionally, these servers are not functionally identical: they offer different degrees of compliance with the SQL standard(s) and even proprietary extension to SQL. Bugs affecting one of these extensions, therefore, cannot exist in a server that lacks the extension. In other words, such bug scripts will provide empirical evidence for the server they were reported for but not for the other servers. We called these "dialect-specific"

[12] We could have used the number of known bugs for each of the partition to construct a profile consistent with the observations. This is not acceptable for two reasons: i) it will favour poor bug reporting practices, and ii) we would have used the bugs twice – once in the inference procedure and another time for the profile, which is theoretically unsound.

bugs. Due to this, not all the bugs reported for a server can be run on the other servers. Therefore the number of 'foreign' bug reports varies between the servers. The interested reader will find an extensive discussion of the study with the bugs in [37].

Table 6 – The observations for the 4 off-the-shelf servers on the bug reports of the different partitions. In the *partition* column we have stated for which server these bugs have been reported.

COTS	Partition	Number of demands *n*	r_1	r_2	r_3
PG 7.0	$S_{PG7.0}$	57	41	0	11
	$S_{IB6.0}$	28	1	0	0
	S_{CS1}	4	1	0	0
	S_{CS2}	18	6	0	0
IB 6.0	$S_{PG7.0}$	24	0	0	0
	$S_{IB6.0}$	55	37	3	7
	S_{CS1}	4	0	0	0
	S_{CS2}	12	1	0	0
CS1	$S_{PG7.0}$	30	0	0	0
	$S_{IB6.0}$	31	0	0	0
	S_{CS1}	18	10	1	3
	S_{CS2}	12	0	0	0
CS2	$S_{PG7.0}$	33	2	0	0
	$S_{IB6.0}$	35	2	0	0
	S_{CS1}	4	0	0	0
	S_{CS2}	51	28	6	5

5.2.4 The Posterior Results

The 50th and 99th percentiles of the marginal distribution, $f_{p^w_{Ser}}(\bullet)$ [13], associated with each server is shown in Table 7. Since the prior distributions are identical for each of the components, then the ordering of the components in the posteriors will be determined by the observations. The best server, across all the percentiles is CS1. This is not surprising since CS1 did not fail for any of the foreign bugs. The second best server is CS2, although IB 6.0 is very close, both at the 50% and the 99% level of confidence. This is somewhat surprising at first given that this server failed more on the foreign bugs than IB6.0. However, the total number of foreign bugs that could be run on CS2 (72) is much higher than IB6.0 (40). Additionally the number of Heisenbugs for CS2 is also much higher (23.5%) than IB6.0 (14.5%), which leads to the CS2 being better in the posteriors.

Table 7 - The table shows the percentiles of the system quality $f_{p^w_{Ser}}(\bullet)$ for each server.

Percentiles	0.5				0.99			
COTS	PG7.0	IB6.0	CS1	CS2	PG7.0	IB6.0	CS1	CS2
Priors	0.77	0.77	0.77	0.77	0.94	0.94	0.94	0.94
Posterior	0.42	0.32	0.24	0.3	0.55	0.45	0.32	0.42

[13] We omitted the detailed results related to the individual partitions due to a lack of space.

5.3. Discussion of the Results for the Two Setups

We have seen that under the more 'stressful' profiles (i.e. Scenario 4 in the TPC-C study and the Bugs study) the best COTS component is CS1. The fact that we have come to the same conclusion using rather different testing methods and different profiles would give us an extra assurance that CS1 is indeed the best component for applications with more stringent reliability and performance requirements which operate at greater transaction load and level of concurrency. However if the concurrency is low, then even with more rigid performance requirements (Scenario 2) Firebird 1.0 server, which is open-source and freely available, comes out as the best server.

The two studies are also in agreement with respect to the worst server – these are the PostgreSQL components.

We could also use the outcome of the studies as a validation of the proposed method. CS1, which came out best, is widely accepted by the database community to be the best SQL server and has by far the largest share in the market of SQL servers. This gives some confidence that both the data that we used is sufficiently informative to allow for meaningful and accurate discrimination between the competing components and the method itself is trustworthy to provide rigorous ground for accurate COTS component selection.

5.4. Further, Contrived Examples

In the empirical study with the SQL servers we could not fully illustrate the interplay between the dependence and the uncertainty in the values of the attributes due to the empirical results often being strikingly different for each server and also because the prior distributions that we started with were the same for each server. In this section we provide some further numerical examples, which illustrate the usefulness of handling uncertainty and dependence between the attribute values explicitly. We comment on the cases where the choice of the best COTS component would differ with conventional assessment methods which rely on point estimates of the attribute values and make assumptions of independence between the values of the attributes. We also discuss the effect of the priors on the selection, including different priors for each of the competing components. The choice of prior distributions and the observations serve illustrative purposes only. The prior, $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet)$, was constructed under the assumption that P_I and P_L are both Beta independently distributed random variables, $Beta(\bullet,a,b)$, defined in the interval [0, 0.01][14], i.e. $f_{p_I,p_L}(\bullet,\bullet) = f_{p_I}(\bullet) f_{p_L}(\bullet)$. The conditional distributions, $f_{p_{IL}|p_I,p_L}(\bullet \mid P_I, P_L)$, for every pair of values of P_I and P_L are also assumed to be Beta distributions, $Beta(\bullet,a,b)$. Clearly they are defined in the range [0, min(P_I, P_L)]. Note that we do not provide any justification for the choice of the prior distributions used here, and neither for the interval on which the distribution is defined; the particular choice of the type of the prior is dictated by some convenience offered by Beta distribution in the examples given below. The assessor can choose any prior distribution and interval that best represents his/her prior beliefs.

---

[14] In all numerical calculations we used the function BETADIST (x, alfa, beta, lowerbound, upperbound) implemented in many standard libraries, see for instance [46]. The last two parameters, lowerbound, upperbound $\in$ [0, 1], define the domain (i.e. the range of x) of the distribution.

5.4.1 Same Priors

In the first example we consider 3 different COTS components, referred to as C1, C2 and C3 respectively for which the prior information does not give any reasons to prefer one to another, i.e. we are indifferent between C1, C2 and C3. The prior distributions, therefore, for all three COTS components are identical. We assumed Beta distributions as described above, with parameters given as follows:

- Beta (2, 10) defined on [0, 0.01] for *pfd* associated with incorrect results $f_{p_I}(\bullet)$;
- Beta (2, 10) defined on [0, 0.01] for *pld* associated with late results $f_{p_L}(\bullet)$;
- Beta (3, 3) for the conditional distribution $f_{p_{IL}|p_I,p_L}(\bullet \mid P_I, P_L)$.

This completes the definition of the tri-variate distribution, $f_{p_I,p_L,p_{IL}}(\bullet,\bullet,\bullet)$.

The assumed observations for these three COTS components are given in Table 8.

For Observation 1 the total number of incorrect or late results are the same for C2 and C3: 5 each. But the failure correlations differ in the two components: for C2 these failures happen on 5 demands (i.e. each of these 5 demands gives both an incorrect and a late response), whereas for C3 they happen on 10 demands (the responses are either incorrect or late). For Observation 2 both the total number of failures and the failure correlation are different in the three COTS components.

Table 8 - Observations from testing the COTS components. All observations are from test campaigns of 5000 demands. The observations differ by the number of incorrect (r_1+ r_3) and late (r_2+ r_3) responses and the number of incorrect & late (r_3) responses.

Observation ID	Number of demands, *n*	COTS	r_1+r_3	r_2+r_3	r_3
Observation 1	5000	C1	0	0	0
		C2	5	5	5
		C3	5	5	0
Observation 2	5000	C1	20	10	10
		C2	13	13	10
		C3	10	10	0

Table 9 shows the results using the percentiles of the prior/posterior distributions of the probability of an inadequate response P_{Ser}. The posterior distribution for Observation 1 reveals that C1 is clearly the best component, since testing revealed no failures for this component. The interesting results are for C2 and C3. Even though the total number of failures observed for C2 and C3 is the same we can still distinguish between them since the types of failures observed in both cases differ. Positive correlation between the two types of failures is observed for C2 whereas the correlation observed between the types of failure for C3 is negative. As a result, the posterior distribution of C2 after testing with Observation 1 is better than that of C3 for all percentiles. Using conventional methods of assessment, where the *attributes are assessed independently*, this distinction would have not been possible since the marginal distributions for the two attributes are the same in both C2 and C3 leading to identical results for these two components. We commented on a similar observation for IB 6.0 and CS2 servers in section 5.1.3.

The posterior after the Observation 2 is also interesting. The total number of failures observed in C3 is the lowest (20 in total) in comparison with C2 (26) and C3 (30). However the correlation between the two types of failures is very different. In C3 there is a maximum negative correlation between the two types of failure (the observed failures are either incorrect or late responses but not both). For C2 we see 10 incorrect results which are also late. And for C1 we see that all late results are also incorrect. Thus, the observations indicate different degrees of

correlation between the two types of failure, which as a result, translates into quite different posteriors for the three COTS components. We would choose C2 as the best COTS component despite the total number of failures (26) observed during testing for this component being higher than the total for C3 (20). This example clearly indicates that the 'uncertainty explicit' assessment method proposed in this paper and conventional assessment methods[15] would have concluded differently regarding which of C2 and C3 should be chosen. The reason for this difference is the correlation between the two types of failure, which we take into account while the conventional methods, which are based on separate assessment of the attributes, would ignore.

Table 9 - The table shows the percentiles of the chosen parameters of system quality.

Percentiles	0.5			0.99		
COTS	C1	C2	C3	C1	C2	C3
System Quality $P_{Ser} = P_I + P_L - P_{IL}$						
Priors	0.0025	0.0025	0.0025	0.0061	0.0061	0.0061
Observation 1	0.0005	0.0011	0.002	0.0015	0.0024	0.0037
Observation 2	0.0033	0.0027	0.0036	0.0051	0.0044	0.0056

5.4.2 Different Priors, Same Observations

In the second example we will consider 2 different COTS components, COTS 1 and COTS 2 referred to as C1, and C2. The assumed testing results for C1 and C2 are identical. The prior distributions, however, for the two COTS components are now different. We will define Beta distributions again but with different parameters for each COTS component, as given in Table 10. The ranges on which the marginal distributions are defined remain the same as in section 5.4.1

Table 10 - The parameters *(a, b)* for the Beta prior distributions defined for each COTS components.

COTS	Reliability $f_{p_I}(\bullet)$	Performance $f_{p_L}(\bullet)$	Conditional distribution: $f_{p_{IL}\|p_I,p_L}(\bullet \mid P_I, P_L)$
C1	(5,5)	(5,5)	(3,3)
C2	(15,14)	(15,14)	(9,9)

A high value for parameter *a* of the Beta distribution means that the distribution is shifted to the right – in our context it represents a prior belief that the number of failures will be high, whereas the *b* parameter shifts the distribution to the left (i.e. a prior belief that the number of failures will be low). The higher the values are, the smaller the uncertainty. We can see for example that C1 and C2 are going to have very similar mean values (the mean of the Beta distribution being a / (a + b)) but the prior distribution for C2 is being expressed with much greater certainty. Therefore the prior distribution of C2 will be much less 'spread' from that of C1 as is illustrated in Fig. 2.

[15] The conventional methods not exploring the dependence in the values of the attributes would conclude that C3 is better than C2.

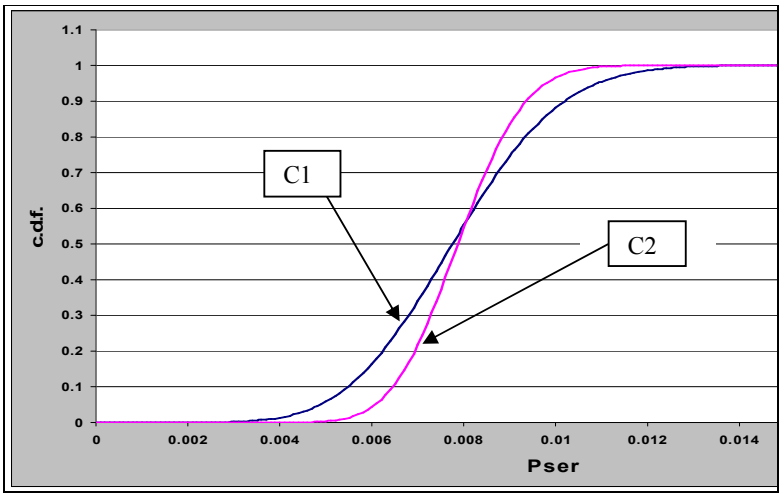

Fig. 2 - The prior distribution for the probability of an inadequate response (P_{Ser}) for C1 and C2. We can see that the prior distribution for C1 is more 'spread' than that of C2 which reflects the assessor's higher uncertainty in the prior beliefs for the values of C1.

We do not make any claims that the priors used in the examples should be used in practical assessment. They serve illustrative purposes only and yet, have been chosen from a reasonable range. For example, the mean of P_I for C1 is $5*10^{-3}$, which is a value from a typical range for many software components.

One set of observations were used for the calculations with the number of trials, $N = 5000$ as shown in Table 11.

Table 11 - Observation 3 from testing the COTS components.

Observation ID	Number of tests, N	COTS	r_1+r_3	r_2+r_3	r_3
Observation 3	5000	All	4	4	1

Table 12 shows the results using the percentiles of the prior/posterior distributions of the probability of an inadequate response P_{Ser}. The structure of the table is similar to that of Table 9.

Interesting points with reference to these posterior values are:

- At the 50th percentile, if the selection is based on the prior values then C1 is the best component. However at the 99th percentile[16] then the ordering changes: C2 is now the preferred choice over the two. For those assessors who prefer to minimize the risk of making 'wrong' decisions with high confidence (i.e. 90%+), C2 is the better choice. This type of distinction would have not been possible in the conventional methods of COTS component assessment, which use point values rather than distributions.
- The posterior values of C2 have shifted significantly in comparison with the priors but not as much as those of C1, even though the testing results for these two components are the same. This is due to the prior distributions: for C1 the prior distribution was highly spread, signifying that the uncertainty was high prior to testing; the opposite is true for C2. Therefore the posterior distribution of C1 is influenced by the testing results much more than that of C2.

Table 12 - The table shows the percentiles of the chosen parameters of system quality.

Percentiles	0.5		0.99	
COTS	C1	C2	C1	C2
System Quality $P_{Ser} = P_I + P_L - P_{IL}$				
Priors	0.0078	0.0079	0.0122	0.0106
Observation 3	0.0028	0.0046	0.0048	0.0065

[16] The same ordering was observed for all percentiles higher than 90th.

6 Discussion of Applicability of the Proposed Assessment Method

6.1 Many Assessment Attributes

We have illustrated in the previous sections how the assessment can be done for the *reliability* and *performance* attributes, which are usually the most important attributes for software COTS components compliant with a *known specification (e.g.* SQL servers, J2EE Application servers, Business Process Execution Engines [47] etc.). For such software components (products) we consider repeating the measurements described in section 5 are feasible. We illustrated that, even for assessments in which only two attributes are considered, taking account of the dependence that exists between these attributes can lead to a different decision on which COTS component to choose compared with the methods that do not account for this dependence.

It is a common practice that COTS components are assessed in terms of more than 2 attributes, usually many more. Among attributes that are suitable for quantitative measurement are:

- "Recoverability" (which again can be characterised in terms of correctness of the recovery and the timeliness of the recovery) [48].
- "Usability" (which can also be characterised in terms of the correctness with which a user performs an action with a given system, and timeliness of their actions). This kind of objective characterisation of usability may be especially important in socio-technical applications, especially those which are safety-critical, such as Air Traffic Control, in which the controllers' accuracy and timeliness of their actions are important.

The obvious question, therefore, is whether the proposed 'uncertainty explicit' assessment 'scales up' to many attributes. Formally, the question is how the method applies if we have to compare COTS components, each of which is represented by a multivariate distribution $f_p(a_1, a_2, ..., a_n)$. The assessment will deliver posterior distributions $f_p(a_1, a_2, ..., a_n \mid assessment)$, which will be used in the comparison. A new variable should be defined as a function of the variates of the distribution $\{a_1, a_2, \ldots, a_n\}$, e.g. a weighted sum of all the attributes. The uncertainty associated with this aggregate variable is easily derived from the joint posterior $f_p(a_1, a_2, ..., a_n \mid assessment)$. Even though mathematically possible, Bayesian inference with multivariate distributions is difficult. The difficulty has two aspects:

- Specifying a multivariate prior distribution becomes very difficult because many non-intuitive dependencies between the attributes must be defined and *justified*.
- Manipulating a multivariate distribution is non-trivial even using the most advanced math/statistical tools. Calculating the posterior distribution is impracticable with more than 3 variates and without simplifying assumptions about the dependencies between them.

To partially overcome these difficulties a divide-and-conquer approach can be employed. First the attributes can be grouped into smaller groups so that there are dependencies within the groups, which the assessment can capture, but the groups are assumed independent. In other words, knowing the values of the attributes in one group does not change the assessor's knowledge (belief) about the values of the attributes included in the other group. Assume that our initial multivariate distribution can be represented as two independent groups of attributes:

- $f_{p_1, p_2, \ldots p_n}(a_1, a_2, ..., a_n) =$

$$f_{p_1,p_2,...p_s}(a_1,a_2,...,a_s)\times f_{p_{s+1},p_{s+2},...p_{s+n}}(a_{s+1},a_{s+2},...,a_n)$$

- the likelihood of observing the COTS component in operation can be expressed as:

$$L\left(observation \mid f_{p_1,p_2...p_n}(a_1,a_2,...,a_n)\right)= L\left(observation_1 \mid f_{p_1,p_2...p_s}(a_1,a_2,...,a_s)\right)\times L\left(observation_2 \mid f_{p_{s+1},p_{s+2},...p_{s+n}}(a_{s+1},a_{s+2},...,a_n)\right)$$

In this case, it trivially follows that:

$$f_{p_1,p_2,...p_n}(a_1,a_2,...,a_n \mid assessment)=$$

$$f_{p_1,p_2,...p_s}(a_1,a_2,...,a_s \mid assessment)\times f_{p_{s+1},p_{s+2},...p_{s+n}}(a_{s+1},a_{S+2},...,a_n \mid assessment)$$

We have so far defined three groups of attributes with which assessment can be performed (Reliability/Performance, Recoverability and Usability). Other groups of attributes may also exist (even though our survey of COTS attribute definition papers (e.g. [16], [20]) found very few attributes that lend themselves to objective assessment). We should note, however, that there are dependencies even amongst the groups identified (for example, we may observe that incorrect results and incorrect recovery actions are highly correlated). Assuming that these belong to independent groups will prevent us from learning about these dependences. Despite this potential deficiency, however, there is clearly an improvement over the existing methods, in which every single attribute is treated independently.

6.2 Decisions on How to Perform the Assessment

We outlined the problems with assessment of a large number of attributes due to the complex set of dependencies that may exist between the different attributes. The higher the number of attributes to be assessed and the higher the number of independence assumptions that are made the more difficult it becomes to place a high degree of confidence in the results obtained from the assessment. The limitations we have outlined in section 6.1, however, are *not specific* to our assessment method; in fact they are more serious for the conventional methods in which the individual attributes are assessed separately. We illustrated with numerical examples in section 5.4 that even when the assessment is done using two attributes, ignoring the dependence between the values of the attributes may lead to wrong decisions: a sub-optimal component may wrongfully be chosen as the best one. If this could be observed with only two attributes, then it is bound to be much more pronounced with more than two attributes, where many more dependencies may exist between the values of the attributes.

Doing the assessment with 'independent groups' of attributes also has its problems. It can only be applied if the assessor can justify that assuming a set of independent pairs is plausible. Despite this problem, however, using small independent groups is still an improvement compared with the extreme assumption used implicitly in the existing assessment methods surveyed, that all of the attributes are independent.

It is worth pointing out that many of the attributes, such as 'has the required functions', various forms of compliance, e.g. 'complies with certain standards', "*backward compatibility*", etc. [16], do not require any uncertainty attached to their values. Assessment with respect to such attributes normally leads to a reduction of the number of the COTS components (which satisfy all these 'binary' attributes), for which the more thorough assessment with respect to the remaining 'non-binary' attributes needs to be done [49].

6.3 The Types of COTS Components for Which the Assessment Method Can be Applied

The method of assessment proposed in this paper would be applicable to any family of COTS components. The setup described in section 4 and illustrated in section 5 is particularly relevant for COTS components with *stringent reliability and performance* requirements. In section 5 we provided empirical results using off-the-shelf database servers. There is a plethora of off-the-shelf database servers, both open source and commercial. Deciding which one to choose among the many choices available overwhelmingly depends on the reliability of servers and their performance.

Java Virtual Machines (JVMs), various Application Servers, Web Servers and Business Process Execution Engines [47] are also examples of COTS components where reliability and performance requirements are usually the deciding attributes for selection. Fault and failure reports, which can be used as observations, do exist for these products and so do performance benchmarks (e.g. *ab* benchmarking tool for web servers [50]). Therefore, similar measurements to what we did for database servers are also possible with these other families of COTS components. In many cases for these components one may not need to deal with more than 2 attributes, which makes our 2-attribute model proposed in section 4 immediately applicable without any further simplifications.

6.4 Other Ways of Eliciting the Prior Distributions

The prior definition in Bayesian assessment is crucial. In our studies we have assumed that prior distributions for each component are the same. This was due to the unavailability of other known 'objective' evidence that we could use to define more reasonable priors. Anecdotal evidence about the servers does exist, but is difficult to translate these subjective beliefs into priors in the form required by our method. By assuming that the prior distributions were the same for each server, the decision on which server is chosen is dictated by the observations only. As a result the decision of the types of distributions for the random variables in our study becomes less important.

However there are other ways of deriving more reasonable priors. We could, for example, utilize evidence from *earlier versions* of the servers and then do multiple steps of inference, i.e. if we want to perform the assessment with later versions of the servers in our study (e.g. with versions of PostgreSQL after release 7.2 or Firebird after release 1.0) we can use the posteriors derived here as priors for the later versions, collect the new evidence for the later versions and then use the model to derive the posteriors for each. This approach has also been reported elsewhere [21].

7. Conclusion

To handle the inherent uncertainty in the COTS component assessment we propose the use of "uncertainty explicit" methods. As a Bayesian approach to representing uncertainty has been successfully applied in other contexts of assessment we have defined a Bayesian model that can be used for assessment of COTS components with respect to two related attributes. This approach complements the conventional selection procedures with more powerful calculus, which can take into account the uncertainty explicitly.

We have conducted an empirical study with off-the-shelf database servers which also illustrated the use of the method. The contribution of this paper is in several aspects:

- We have demonstrated in the context of the COTS component assessment how to apply the Bayesian methods which have had some popularity in reliability assessment of both software and hardware.
- We have described the use of the model in selecting the best off-the-shelf database server from a sample of different servers, using two sources of data:
  - Experiments using an implementation of TPC-C benchmark for database severs.
  - Known faults reported for four different servers.
- We recommend that the 'uncertainty explicit' assessment methods be considered at least as a non-expensive warranty against badly sub-optimal decisions possible with the conventional COTS component selection methods (we provided contrived numerical examples which show examples of sub-optimal selections of COTS components if uncertainty or dependence in the values of the attributes are ignored).
- We have also demonstrated how our model can be extended and used with a partitioned demand space which allows utilization in the inference of all the evidence available from the different partitions.

An interesting observation from the study with SQL servers is that the results of the inference with the more stressful setups (Scenario 4 of the TPC-C study and the bugs study) both lead to CS1 being preferred as the best server and PG servers being the worst. This may give the assessor further assurance of preferring CS1 for an application with more stringent reliability, performance and concurrency requirements given that it performed best under two very different but 'stressful' profiles. Interestingly, CS1 is considered by the DB community as a leader among the SQL servers vendors, which may be seen as validation of the method's usefulness for making a correct choice among several similar COTS components despite the scarcity of the data that we could use.

There are several well-known difficulties of using Bayesian assessment - it does not scale up well due to:

- The difficulty with specifying a multivariate prior distribution when the number of attributes to be assessed increases, unless independence is assumed among the attributes. Defining the prior is crucial. It may be difficult for practitioners, not comfortable with non-trivial math, to express their individual beliefs as probability distributions.
- The difficulty with manipulating a multivariate distribution, which becomes impracticable with more than 3 variates if no simplifying assumptions are made.

Future work that is desirable includes:

- Methods are needed which would allow for effective assessment with a *large number of related attributes*. Currently the 'uncertainty explicit' assessment only works with a limited number of related attributes (or with independent groups of attributes in which the number of attributes in the groups is small).
- Further development of the theoretical framework is needed for cases of groups of more than 2 dependent attributes. Conceptually, the multivariate inference is no different than the 1- and 2-variate inferences. Its practical use, however, is currently problematic.

Acknowledgement

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under the Interdisciplinary Research Collaboration in Dependability of Computer-Based Systems (DIRC) project. We

would also like to thank Dr Andrey Povyakalo, Professor Bev Littlewood and the anonymous reviewers for their comments on an earlier version of this paper.

References

1. Ncube, C. and N. Maiden. *PORE:Procurement Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm*. in *International Workshop on Component-Based Software Engineering*. 1999.
2. Kontio, J., et al. *A COTS Selection Method and Experiences of Its Use*. in *Twentieth Annual Software Engineering Workshop,NASA Goddard Space Flight Center*. 1995. Greenbelt, Maryland.
3. Jeanrenaud, J. and P. Romanazzi. *Software Product Evaluation: A Methodological Approach*. in *Software Quality Management II: Building Software into Quality*. 1994.
4. Tran, V. and D.-B. Liu. *A Risk Mitigating Model for the Development of Reliable and Maintainable Large-Scale Commercial-Off-The-Shelf Integrated Software Systems*. in *Proceedings of the 1997 Annual Reliability and Maintainability Symposium (RAMS)*. 1997.
5. Ochs, M., et al. *A Method for Efficient Measurement-based COTS Assessment and Selection -Method Description and Evaluation Results*. in *Proceedings of the 7th International Symposium on Software Metrics*. 2001. London, England: IEEE Computer Society.
6. Alves, C. and J. Castro. *CRE: A Systematic Method for COTS Components Selection*. in *XV Brazilian Symposium on Software Engineering (SBES)*. 2001. Rio de Janeiro, Brazil.
7. Phillips, B.C. and S.M. Polen, *Add Decision Analysis to Your COTS Selection Process*, in *CroosTalk - The Journal of Defence Software Engineering*. 2002.
8. Boehm, B., et al. *Composable Process Elements for Developing COTS-Based Applications*. in *Proceedings 2003 International Symposium on Empirical Software Engineering. ISESE'2003*. 2003: ACM-IEEE.
9. Dean, J., *An Evaluation Method for COTS Software Products*. 2000.
10. Kunda, D. and L. Brooks. *Applying Social-Technical Approach for COTS Selection*. in *Proceedings of the 4th UKAIS Conference*. 1999. University of York, England.
11. Gregor, S., J. Hutson, and C. Oresky. *Storyboard Process to Assist in Requirements Verification and Adaptation to Capabilities Inherent in COTS*. in *ICCBSS 2002*. 2002. Florida, USA: Springer-Verlag.
12. Burgués, X., et al. *Combined Selection of COTS Components*. in *ICCBSS 2002*. 2002. Florida, USA: Springer-Verlag.
13. Comella-Dorda, S., et al. *A Process for COTS Software Product Evaluation*. in *ICCBSS 2002*. 2002. Florida, USA: Springer-Verlag.
14. Ruhe, G. *Intelligent Support for Selection of COTS Products*. in *Web, Web-Services, and Database Systems*. 2003: Springer.
15. Lewis, P., et al. *Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software Systems*. 2000; Available from: http://www.cebase.org/www/researchActivities/COTS/LessonsLearned.pdf.
16. Bertoa, M.F. and A. Vallecillo. *Quality Attributes for COTS Components*. in *Proc. of the 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)*. 2002. Málaga, Spain.
17. ISO/IEC-9126-1:2001, *Information technology – Product Quality – Part1: Quality Model*. 2001, International Standard ISO/IEC 9126, International Standard Organization, June, 2001.
18. IEEE/ANSI, *Recommended Practice for Software Requirements Specifications, International Standard 830-1993*. 1993, IEEE.
19. Abts, C., B. Boehm, and E.B. Clark. *COCOTS: A software COTS-Based System (CBS) Cost Model*. in *Proceedings 12th Annual Conference on European Software Control and Metrics (ESCOM)*. 2001. London, UK.
20. Torchiano, M. and L. Jaccheri. *Assessment of Reusable COTS Attributes*. in *2nd Int. Conference on COTS Based Software Systems (ICCBSS 2003)*. 2003. Ottawa, Canada: Springer-Verlag.
21. Littlewood, B. and D. Wright, *Some conservative stopping rules for the operational testing of safety-critical software*. IEEE Transactions on Software Engineering, 1997. **23**(11): p. 673-683.
22. Likert, R., *A Technique for the Measurement of Attitudes*. 1932, New York: McGraw-Hill.
23. Brocklehurst, S., et al., *Recalibrating software reliability models*. IEEE Transactions on Software Engineering, 1990. **SE-16**(4): p. 458-470.

24. Lyu, M.R., ed. *Handbook of Software Reliability Engineering*. 1996, IEEE Computer Society Press and McGraw-Hill.
25. Wright, D. and K.-Y. Cai, *Representing Uncertainty for Safety Critical Systems, PDCS2 Tech. Rep. 135. Center for Software Reliability, City University, London,*. 1994, SHIP Project.
26. Musa, J.D., *Operational Profiles in Software-Reliability Engineering.* IEEE Software, 1993. **March**: p. 14-32.
27. Johnson, N.L. and S. Kotz, *Distributions in Statistics: Continuous Multivariate Distributions*. Wiley Series in Probability and Mathematical Statistics, ed. R.A. Bradley, Hunter, J. S., Kendall, D. G., Watson, G. S. Vol. 4. 1972: John Wiley and Sons, INc. 333.
28. Littlewood, B., P. Popov, and L. Strigini. *Assessment of the Reliability of Fault-Tolerant Software: a Bayesian Approach*. in *Proc. 19th International Conference on Computer Safety, Reliability and Security, SAFECOMP'2000*. 2000. Rotterdam, the Netherlands: Springer.
29. Popov, P. *Reliability Assessment of Legacy Safety-Critical Upgraded with Off-the-Shelf Components*. in *SAFECOMP-2002*. 2002. Catania, Italy: Springer.
30. Kaplan, W., *Advanced Calculus*. 3rd ed. 1984, Reading, MA: Addison-Wesley.
31. Port, D. and S. Chen. *Assessing COTS assessment:How much is enough?* in *ICCBSS'04, Inetrnational Conference on COTS Based Software Systems*. 2004. Redondo Beach, California: Springer-Verlag.
32. Jeng, B. and E.J. Weyuker, *Analyzing partition testing strategies.* IEEE Transactions on Software Engineering, 1991. **17**(7): p. 703-711.
33. Hamlet, D. and R. Taylor, *Partition testing does not inspire confidence.* IEEE Transactions on Software Engineering, 1990. **16**(12): p. 1402-1411.
34. Gashi, I., Popov, P., Strigini, L. *Fault diversity among off-the-shelf SQL database servers*. in *DSN'04 International Conference on Dependable Systems and Networks*. 2004. Florence, Italy: IEEE Computer Society Press.
35. Gashi, I., Popov, P., Stankovic, V., Strigini, L., *On Designing Dependable Services with Diverse Off-The-Shelf SQL Servers*, in *Architecting Dependable Systems II*, C.G.a.A.R. R. de Lemos, Editor. 2004, Springer-Verlag. p. 191-214.
36. Stankovic, V. and P. Popov. *Improving DBMS Performance through Diverse Redundancy*. in *25th IEEE Symp. on Reliable Distributed Systems (SRDS'06)*. 2006. Leeds, UK: IEEE Computer Society, .
37. Gashi, I., P. Popov, and L. Strigini, *Fault tolerance via diversity for off-the-shelf products: a study with SQL database servers.* IEEE Transactions on Dependable and Secure Computing, 2007. **4**(4): p. 280-294.
38. TPC, *TPC-C, An On-Line Transaction Processing Benchmark, v. 5.2*. 2004.
39. Adams, E.N., *Optimizing Preventive Service of Software Products.* IBM Journal of Research and Development, 1984. **28**(1): p. 2-14.
40. Gray, J. *Why do computers stop and what can be done about it?* in *6th International Conference on Reliability and Distributed Databases*. 1987.
41. Chandra, S. and P.M. Chen. *Whither Generic Recovery from Application Faults? A Fault Study using Open-Source Software*. in *DSN 2000, International Conference on Dependable Systems and Networks*. 2000. NY, USA: IEEE Computer Society Press.
42. Littlewood, B. and D.R. Miller, *Conceptual Modelling of Coincident Failures in Multi-Version Software.* IEEE Transactions on Software Engineering, 1989. **SE-15**(12): p. 1596-1614.
43. Eckhardt, D.E. and L.D. Lee, *A theoretical basis for the analysis of multiversion software subject to coincident errors.* IEEE Transactions on Software Engineering, 1985. **SE-11**(12): p. 1511-1517.
44. Knight, J.C. and N.G. Leveson, *An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming.* IEEE Transactions on Software Engineering, 1986. **SE-12**(1): p. 96-109.
45. Eckhardt, D.E., et al., *An experimental evaluation of software redundancy as a strategy for improving reliability.* IEEE Transactions on Software Engineering, 1991. **17**(7): p. 692-702.
46. GNOME. *BetaDist*. 2008; Available from: [http://www.gnome.org/projects/gnumeric/doc/gnumeric-BETADIST.shtml].
47. Andrews, T., F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, and I. Trickovic., *Business Process Execution Language for Web Services version 1.1*. 2003, BEA, IBM, Microsoft, SAP, Siebel, Systems.
48. Patterson, D., et al. *Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies*. 2002; Available from: http://roc.cs.berkeley.edu/papers/ROC_TR02-1175.pdf.
49. Ncube, C. and N. Maiden, *Acquiring COTS Software Selection Requirements.* IEEE Software, 1998. **15**(2): p. 46-56.
50. Apache-Software-Foundation. *ab - Apache HTTP server benchmarking tool*. 2008; Available from: http://httpd.apache.org/docs/2.0/programs/ab.html.