Uncertainty Explicit Assessment of Off-the-Shelf Software: Selection of an Optimal Diverse Pair

Ilir Gashi, Peter Popov
Centre for Software Reliability, City University,
Northampton Square, London, EC1V 0HB
I.Gashi@city.ac.uk, ptp@csr.city.ac.uk

Abstract

Assessment of software COTS components is an essential part of component-based software development. Sub-optimal selection of components may lead to solutions with low quality. The assessment is based on incomplete knowledge about the COTS components themselves and other aspects, which may affect the choice such as the vendor's credentials, etc. We argue in favor of assessment methods in which uncertainty is explicitly represented ('uncertainty explicit' methods) using probability distributions. We have adapted a model (developed elsewhere [17]) for assessment of a pair of COTS components to take account of the fault (bug) logs that might be available for the COTS components being assessed. We also provide empirical data from a study we have conducted with off-the-shelf database servers, which illustrate the use of the method.

1. Introduction

Commercial-off-the-shelf (COTS) components often form an essential part in software development. Benefits of their use are wide ranging: from the incentive to cut-down on cost to reducing the development time and improving quality by using tried and tested components. An initial and essential part of component based software development is the assessment of available COTS components. There exist a plethora of available methods for COTS assessment [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. An often overlooked aspect in the existing assessment techniques is the inherent *uncertainty* in the values of the parameters being assessed. This is because the assessment is carried out with limited resources of time and budget. Therefore the true values of the assessed attributes will rarely be known with certainty.

For solutions with very stringent dependability requirements a single component may rarely be able to meet the required dependability target. It has been argued [15] that employing fault-tolerance in the form of software design diversity (i.e. using more than one component to perform the same function) is usually the best guarantee of achieving higher levels of dependability than what the available COTS components can offer. But, employing software diversity was seen in the past as an expensive method for increasing dependability due to the need of building more than one component. With off-the-shelf components this problem is overcome: there may be many different components that will have the required functionality therefore bespoke development may not be required[1]. Moreover many of these components are free and open-source, thus the cost of procurement may be non-existent. The problem of assessment though still exists. If we were interested in building a 1-out-of-2[2] system, simply choosing the two best components that exist in the market may not be enough. What is of interest is how well the pair works together. The optimal pair will be the one with the lowest probability of coincident failures of both components of the pair. The components that form the best pair may not necessarily be the ones which are the best individually. For further details on the subtleties of this problem the interested reader is referred to a recent survey [16].

In this paper we will provide details of an adaptation of the model in [17] which allows for an optimal selection of a pair of components to be used in a fault-

[1] Apart from *'glue code'* (usually referred to as *middleware)* which may be needed to ensure the components can be deployed for a given system in a coordinated manner as required by the particular system context.

[2] In this configuration the system performs correctly as long as 1 of the 2 components works correctly.

tolerant system. In this model the assessment results are subject to uncertainty and we discuss how this may impact the decisions about which pair of components we choose. The model also enables representing the dependencies that exists between uncertainties associated with the values of each COTS component in the pair.

The paper is structured as follows: section 2 contains a brief review of related work on COTS assessment; in section 3 we describe the model of assessment, in which model parameters are not known with certainty and argue in favor of using probability distributions as an adequate mechanism to capture this uncertainty; in section 4 we provide details of an empirical study with off-the-shelf database servers and illustrate how our approach can be used to select the optimal diverse pair; in section 5 we provide a discussion of the method and finally section 6 contains conclusions and provisions for further work.

2. Related work

There are a wide variety of COTS 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 candidates at a very early stage [9], [18]. It has even been suggested [18] 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 [18] suggests distinguishing between essential requirement 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 selection. The phases are: the search phase, the screening and evaluation phase and the analysis phase. In the first phase COTS products are identified. In the screening and evaluation phase the products 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 products for inclusion in the system.

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 products and their use within the wider system.

Other assessment methods include: CISD (COTS-based Integrated System Development) [4], STACE (Socio Technical Approach to COTS Evaluation) [10], CDSEM (Checklist Driven Software Evaluation Methodology) [3], CRE-COTS-Based Requirements Engineering Method [6], CEP (Comparative Evaluation Process Activities) [7], CBA Process Decision Framework [8], A Proactive Evaluation Technique [19], CAP-COTS Acquisition Process method [5], Storyboard Process [11], Combined Selection of COTS Components [12], PECA Process [13] or COTS-DSS [14].

3. Assessment of Diverse COTS Solutions: Bayesian Approach

3.1 Uncertainty in the assessment

Any assessment is conducted with limited resources and under various assumptions, which may not hold true in real operation. Therefore the outcome of the assessment is subject to uncertainty. For example, deciding to rate a COTS component exactly 7 out of 10 according to a chosen scale may be difficult to justify. The assessor may be certain that the values of the attribute outside the range {6,7} are unreasonable but be *indifferent* between the possible values inside this interval. Software reliability is a typical example of an attribute which is never known with certainty. Probability of failure on a randomly chosen demand (*pfd*) is unknown, but the assessor may be prepared to state, with confidence 99%, that it is less than, say 10^{-3}. The assessor may be even more specific of their doubts about the COTS *pfd* and state that the most likely range of the *pfd* is between 10^{-4} and 10^{-3}.

There are various methods for representing uncertainty [20]. Bayesian approach to probabilistic modeling is one of the best-known ones and used with some success in reliability assessment [21]. It allows one to combine, in a mathematically sound way, the prior belief (which is ‘subjective’ and possibly inaccurate) about the values of a parameter with the (‘objective’) evidence from seeing the modeled artifact (in this case a COTS component) 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. If the prior belief is represented as a probability *distribution* rather than a single value, then after seeing the observations we get a posterior distribution (quantification of uncertainty) which takes into account both the prior knowledge and the empirical evidence.

3.2 Model for Assessment of 1 COTS Component with one Attribute

In this section we illustrate how the Bayesian approach to assessment is normally applied to assessing a single attribute of a single COTS component. 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 (Fig.1), the Bayesian assessment proceeds as follows.

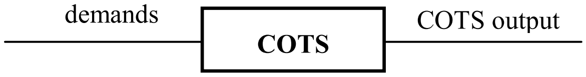

Fig. 1 - Black-box model of a COTS component

Let us denote the system *pfd* as *p*, with prior distribution $f_p(\bullet)$, which characterizes the assessor's knowledge about the system *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), and *r* failures are observed. The posterior distribution, $f_p(x \mid r,n)$, of *p* after the observations will be:

$$f_p(x \mid r,n) \propto L(n,r \mid x) f_p(x) \qquad (1)$$

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:

$$L(n,r \mid x) = \binom{n}{r} x^r (1-x)^{n-r} \qquad (2)$$

For any prior and any observation (*r*, *n*), including (*r*=0), the posterior can be calculated. Thus it can be applied to all the COTS components included in the assessment. Now, the selection can be based on the posterior distribution derived for the COTS components using different criteria:

- for a given reliability target the COTS component chosen will be the one which has the highest probability of having a *pfd* lower than the given target;
- for a predefined 'mission' of say, 1000 demands, the COTS component chosen will be the one which is most likely to survive the mission without a failure.

3.3 Model for Assessment of a Fault-Tolerant System Consisting of 2 COTS Components

The Bayesian assessment can also be applied to choosing a pair of components. In what follows we will describe how the assessment can be performed for a system made up of two components. The mathematical details can be found in [17] and Appendix A.

Let us assume that the attribute of interest is again the *pfd* of the system: that is of simultaneous failure of both components. Now assume that the system is subjected to a series of independently selected demands. On each demand the response received from each of the COTS components is characterized as correct/incorrect. Since we have two COTS components clearly 4 combinations exist, which can be observed on a demand, as shown in Table 1.

The four probabilities given in the last column of Table 1 sum to unity (i.e. they sum to 1). So if the last three probabilities are 0.2, 0.4 and 0.3 respectively then the first one p_{10} = 1- (0.2 + 0.4 + 0.3) = 0.1. Thus, the joint distribution of any three of these probabilities, will give an exhaustive description of the COTS pair behavior. In statistical terms, the model of the COTS component pair has three degrees of freedom. Since we have a three variate distribution we need to define three prior distributions (not a single one as in the previous section): the prior distributions for the *pfd* of each of the components, and then the conditional prior distribution for the *pfd* of both components simultaneously. The details of this joint distribution are given in [17, section 2] and Appendix A. From this distribution we can then derive the marginal distribution of common failures which will be used to choose the best pair of components in a 1-out-of-2 setup.

3.4 Utilizing Multiple Sources of Data in the Assessment

In some areas of software engineering, especially in testing, the usefulness of *partitioning the demand space* has been recognized [22], [23], [24]. The

Table 1 - The outcomes and their frequency and probabilities for each demand

Event	*COTS A Correct*	*COTS B Correct*	*Observations in n demands*	*Probability*
A	*No*	*Yes*	r_1	p_{10}
B	*Yes*	*No*	r_2	p_{01}
Γ	*No*	*No*	r_3	p_{11}
Δ	*Yes*	*Yes*	r_4	p_{00}

demand space partitions typically represent different *types* of demands, which may have different likelihoods of occurring in 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 attribute 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 $\{S_1, S_2, \ldots S_M\}$, then for each of these the assessment will be performed as described above, e.g. with two COTS components the description provided in section 3.3 will apply. As a result M conditional distributions will be associated with each pair of COTS components from which the conditional uncertainty of interest will be expressed, that characterizes the behavior of the particular pair of COTS components in the specific partition. Finally, in order to compare the competing pairs of COTS components the unconditional distribution of the probability of joint failure should be derived for the particular profile defined *over the set of partitions*, which represents the targeted operational environment. In [25] we describe an approach of combining the assessment in partitions under the assumption of independence of uncertainties across the sub-domains. Mathematical details can be found in [25, section 3.3] and Appendix B.

4. Empirical Results from a Study with Off-The-Shelf Databases

We have reported previously results of a study on dependability of off-the-shelf database servers [26]. In this paper we will use the data collected in that study to demonstrate how the model explained in section 3.3 can be utilized to perform the selection of the best pair of 2 servers. We note that the ideal selection of the best pair is to perform statistical testing using the COTS products. This, however, is problematic in practice due to the lack of suitable middleware[3] for diverse database replication. Database replication is non-trivial as it requires synchronizing the operation of the copies while serving concurrent clients. Additionally the software vendor of the middleware may like to make a 'strategic' choice of an SQL server pair for use in the foreseeable future. The application(s), which may be developed by the users of the middleware in the future, will be clearly unknown at the time of making the selection, therefore performing statistical testing (which is crucially dependent on knowing the operational profile in the targeted environment) will be impossible.

Given these difficulties we can use alternative options. We will describe in this paper one such option: using stressful environments which increase the likelihood of failures occurring. After all the fault-tolerant solution with a pair of servers is intended to cope with the difficult situations (demands) where the individual channels are deficient. The set of bugs of a particular COTS product (in our case SQL server) defines one such stressful environment for a server. We have collected known bug reports for four SQL servers, namely PostgreSQL 7.0, Interbase 6.0, Oracle 8.0.5 and Microsoft SQL server 7 [26] (for the sake of brevity we will use the abbreviations PG, IB, OR and MS respectively throughout the rest of the text when referring to these servers). The union of the bugs reported for all the compared COTS products will form a demand space, in which there will be a partition stressing each of the products. The logs of the known bugs are treated as a sample (without replacement[4]) 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. The servers are then compared using the methodology described in sections 3.3 and 3.4.

4.1 Prior Distributions

The prior distributions used in this study are explained next. The joint prior distribution was constructed under the assumptions that the respective *pfd*s of a server A and a server B are independently distributed; in the general case of the failures being non-independent events, the conditional distributions of the probability of coincident failure are specified for every pair of values of the *pfd*s of servers A and B.

The distributions were assumed to be identical for

[3] Some rudimentary solutions such as C-JDBC [27] only allow for the use of a minimal subset of SQL with diverse SQL servers.

[4] 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 3.3 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).

each of the four servers across both their ‘own’ and ‘foreign’ partitions respectively. This assumption was made because we did not have reasons to believe otherwise. We discuss other options of deriving more accurate priors in the Conclusions section. A summary of the distributions used is given in Table 2.

For ‘own’ partitions the prior distributions of *pfd*s of both A and B were defined as uniform in the range [L, 1], where L < 1 accounts for the chance that some of the reported bugs might be Heisenbugs[5] [28], 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 [29]. These authors studied the fault reports for three off-the-shelf products: 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 products we interpreted these findings by setting L = 1-(2*0.14), that is twice the highest value of Heisenbugs reported, i.e. allowing for even higher proportion of the Heisenbugs than recorded in [29]. The prior, thus, 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 for both *pfd*s of A or B 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 theoretical work such as [30], [31] 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 them. On the other hand, empirical studies such as [32], [33], 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 servers are also plausible. In summary, we are indifferent between the values of the probability that a server will fail from a ‘foreign’ bug.

All conditional prior distributions for coincident failures of the two servers for given values of the components’ *pfd*s were defined in the range [0, min (value of *pfd* of A, value of *pfd* of B)] (since it 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 a more accurate prior, thus ‘indifference’.

For the comparison we use a distribution defined on the partitions which does not favor any of the servers, i.e. we assumed that probability of each partition is 0.25 in the study with 4 servers[6].

Table 2 - The Prior distributions (identical for all four servers)

Partition	Range	Distribution
pfd of server A *or* B on ‘Own’ partitions	0.72 – 1	Uniform
pfd of server A *or* B on ‘Foreign’ partitions	0 – 1	Uniform
Conditional Distribution of ‘Coincident failures’ in both A *and* B on either partition	0 – min (value of *pfd* of A, value of *pfd* of B)	Uniform

4.2 Observations

The observations using the known bugs of four off-the-shelf servers are given in Table 3 [26]. Since we included 4 servers in our study and we are interested in diverse pairs of servers, then we have a total of 6 different server pairs. 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 reason 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

[5] Gray defines two types of bugs [28]: “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 usage pattern, other software and internal state)

[6] We could use 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, an ii) we would have used the bugs twice – once in the inference procedure and another time in defining the profile, which is theoretically unsound.

commercial servers (especially OR). Therefore, the sizes of the samples from the partitions on each server are different[7]. 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, thus, literally cannot exist in a server that lacks the extension. We called these "dialect-specific" 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.

Table 3 - The observations for the 6 diverse server pairs on the bug reports of the different partitions. In the partition column the subscript indicates for which server these bugs have been reported. *N* is the total number of bugs run and *r₁*, *r₂* and *r₃* are as defined in Table 1.

Server Pair	Partition	N	r_1	r_2	r_3	Server Pair	Partition	N	r_1	r_2	r_3
PG & IB	S_{PG}	24	21	0	0	IB & OR	S_{PG}	18	0	0	0
	S_{IB}	28	0	23	1		S_{IB}	31	25	0	0
	S_{OR}	3	0	0	0		S_{OR}	4	0	3	0
	S_{MS}	9	0	0	0		S_{MS}	10	1	0	0
PG & OR	S_{PG}	30	27	0	0	IB & MS	S_{PG}	21	0	1	0
	S_{IB}	24	1	0	0		S_{IB}	35	27	0	2
	S_{OR}	4	0	2	1		S_{OR}	4	0	0	0
	S_{MS}	7	0	0	0		S_{MS}	12	0	6	1
PG & MS	S_{PG}	33	28	0	2	OR & MS	S_{PG}	27	0	2	0
	S_{IB}	25	1	2	0		S_{IB}	30	0	2	0
	S_{OR}	3	0	0	0		S_{OR}	4	3	0	0
	S_{MS}	18	1	7	5		S_{MS}	12	0	7	0

4.3 Posteriors

Table 4 shows the percentiles of the priors and posteriors of the probability of a failure of a pair of components assuming a 1-out-of-2 setup. The values in the cells represent the confidence that the probability of a coincident failure of both components of a pair on the same randomly chosen demand is no greater than the respective confidence level, e.g. for PG & IB the value of 0.02 at the 50th percentile can be interpreted as "we are 50% confident that the probability of a coincident failure of both PG & IB on a randomly chosen demand is no greater than 0.02".

Table 4 - The percentiles of the probability of system failure for each server pair.

Server Pair	50th percentile		99th percentile	
	Prior	Posterior	Prior	Posterior
PG & IB	0.3	0.02	0.61	0.12
PG & OR		0.07		0.19
PG & MS		0.09		0.20
IB & OR		0.02		0.14
IB & MS		0.04		0.14
OR & MS		0.02		0.14

We can see that universally the best pair across the percentiles is the open-source server pair PG & IB. There are some interesting remarks to note from the results on Table 4, which highlight the value of handling the uncertainty explicitly using probability distributions, rather than using point estimates of attribute values and the value of exploiting the dependence in the failure behavior of the servers:

- It may seem surprising that the best server pair is PG & IB given that results in Table 3 show that one coincident failure (i.e. r_3) was observed for this pair and none for the commercial server pair OR & MS. But, in Table 3 we also saw that there is a much larger number of single channel failures (i.e. r_1 and r_2) observed for the open-source server pair than for the commercial server pair which increases our confidence of a strong *negative correlation* in the failure behavior of the open-source pair, i.e. we see extensive evidence that diversity does work: when one of the servers fails the other works correctly. No such evidence is available for the commercial servers.
- We cannot make a selection purely on the 50th percentile of the posterior distribution of the system *pfd* since 3 of the server pairs give identical results. Most of the conventional assessment techniques, which rely on median values of the assessment attributes would have also been unable to provide a clear choice. However we can make a selection from the 99th percentile of the same setup.

We have also used the model described in section 3.1 to calculate the posteriors of single servers (using the same prior definitions as for the pairs, the observations for each individual server and utilizing the

[7] It may seem desirable to have a similar amount of data for the different servers, but in reality there are different reporting practices for each server. Such differences simply translate into different amounts of empirical evidence available for the servers, with which our method can cope easily.

partitions theory described in section 3.4). The posteriors for each server are shown in Table 5. We can see that even the worst pair from Table 4 on all percentiles performs better than the best single server in Table 5. This is hardly surprising given the fact that coincident failures are very rare despite the choice of a stressful demand profile (known bug reports). We can also see that the differences in the *pfd* values of a single server vs. a diverse pair of servers are quite significant.

The worst performing server pair has a *pfd* of no worse than 0.20 with confidence 99% whereas the best performing single server has a *pfd* of no worse than 0.32 with the same confidence level. These results indicate that the use of a diverse server pair would bring significant dependability gains: the best single server may fail up to once in 3 demands while the worst pair – up to once in 5 demands.

Table 5 - The percentiles of the probability of failure on demand for each single server.

Posteriors	PG	IB	OR	MS
50th percentile	0.41	0.30	0.26	0.30
99th percentile	0.54	0.43	0.32	0.42

5. Discussion

The Bayesian model explained in sections 3.3 and 3.4 can be used for selection of an optimal pair of COTS components, as was illustrated in section 4, when the attribute of interest is the probability of failure on demand. It is a common practice that COTS components are assessed in terms of more than 2 attributes, usually many more. The obvious question, therefore, is whether the proposed 'uncertainty explicit' assessment 'scales up' to:

- more than one attribute
- fault-tolerant configurations in which more than two COTS components are used (for example, three COTS components to enable majority voting on the results)

In both of these cases, the question is how the method applies if we have to define multivariate distributions. Even though mathematically possible, Bayesian inference with multivariate distributions is difficult. The difficulty has two aspects:

- specifying a multivariate prior distribution becomes problematic 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.

For scenarios where the COTS components need to be assessed in terms of more than one attribute, 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 (i.e. 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); then, due to the independence assumption, the final distribution is the product of the distributions of the individual groups. More details on this approach can be found in [25, section 5.1].

The limitations we outlined in this section 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 have shown in [25] that even when the assessment of single COTS components is done using just 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.

The "divide and conquer" approach to 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. [34], 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 can proceed [35].

6. Conclusions

Software diversity is a well known and well studied subject in the literature [36]. It is recognized that often the only way of obtaining dependability assurances is to employ software diversity [15]. With the plethora of off-the-shelf components available fault tolerance through software diversity becomes a much more achievable and affordable solution especially since many of the components are open-source and free. The important questions for a given project is how much dependability gains there will actually be from employing diversity, or at least given a set of diverse software alternatives which is the best for a given application.

We applied methods of Bayesian assessment developed elsewhere [17], [25]. We illustrated how our model can be used with the collected evidence to perform the assessment and choose the best server pair. We then compared the results of the posteriors of server pairs with those of single servers and we saw that even the worst server pair still performs much better than the best single server. This indicates that significant dependability gains may be obtained from using diverse off-the-shelf database servers. It is also interesting to note that in our assessment the best single server is a commercial server, namely Oracle, whereas the best pair of components is the pair PostgreSQL & Interbase both of which are free and open-source components.

The prior definition in Bayesian assessment is crucial. In our study we have assumed that prior distributions for each component are the same. This was due to the unavailability of other known evidence that we could use to define more accurate priors. However this problem can be remedied by utilizing evidence from *earlier versions* of the servers and then doing multiple steps of inference, i.e. if we want to perform the assessment with later versions of the servers in our study we can use the posteriors from this step 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.

Future work that is desirable would be to enable effective assessment with a higher number of COTS components in a diverse setup (more than two components may be desirable in a diverse setup to enable majority voting on the results from the components).

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.

Bibliography

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 p. 55-69.
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 p. 361-367.
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 p. 285-294.
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*. 2002, http://www.stsc.hill.af.mil/crosstalk/2002/04/phillips.html.
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 p. 8-17.
9. Dean, J., *An Evaluation Method for COTS Software Products*. 2000, http://www.stc-online.org/cd-rom/cdrom2000/webpages/johndean/paper.pdf.
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 p. 132-141.
12. Burgués, X., et al. *Combined Selection of COTS Components*. in *ICCBSS 2002*. 2002. Florida, USA: Springer-Verlag p. 54-64.
13. Comella-Dorda, S., et al. *A Process for COTS Software Product Evaluation*. in *ICCBSS 2002*. 2002. Florida, USA: Springer-Verlag p. 86-92.
14. Ruhe, G. *Intelligent Support for Selection of COTS Products*. in *Web, Web-Services, and Database Systems*. 2003: Springer p. 34-45.

15. Littlewood, B. and L. Strigini, *Validation of Ultra-High Dependability for Software-based Systems.* Communications of the ACM, 1993. 36(11): p. 69-80.
16. Littlewood, B., Popov, P. and Strigini, L., *Modelling software design diversity - a review.* ACM Computing Surveys, 2001. 33(2): p. 177 - 208.
17. 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 p. 294-308.
18. Lewis, P., et al., *Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software Systems*. 2000, Software Engineering Resource Center.
19. Dean, J. and M. Vidger. *COTS Software Evaluation Techniques*. in *Proceedings of The NATO Information Systems Technology. Symposium on Commercial Off-the-shelf Products in Defence Applications*. 2000. Brussels, Belgium.
20. Wright, D. and K.-Y. Cai, *Representing Uncertainty for Safety Critical Systems, PDCS2 Tech. Rep. 135. Center for Software Reliability, City University, London,*. 1994.
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. Jeng, B. and E.J. Weyuker, *Analyzing partition testing strategies.* IEEE Transactions on Software Engineering, 1991. 17(7): p. 703-711.
23. Hamlet, D. and R. Taylor, *Partition testing does not inspire confidence.* IEEE Transactions on Software Engineering, 1990. 16(12): p. 1402-1411.
24. Musa, J.D., *Operational Profiles in Software-Reliability Engineering.* IEEE Software, 1993. March: p. 14-32.
25. Gashi, I., Popov, P., Stankovic, V., *Uncertainty Concious Assessment of Off-The-Shelf Software*. 2006, http://www.csr.city.ac.uk/people/ilir.gashi/COTS/.
26. 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 p. 389-398.
27. ObjectWeb, *C-JDBC*. 2006 http://c-jdbc.objectweb.org/.
28. Gray, J. *Why do computers stop and what can be done about it?* in *6th International Conference on Reliability and Distributed Databases*. 1987.
29. 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 p. 97-106.
30. 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.
31. 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.
32. 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.
33. 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.
34. 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 p. 54-66.
35. Ncube, C. and N. Maiden, *Acquiring COTS Software Selection Requirements,* IEEE Software, 1998, 15(2), p. 46-56.
36. Anderson, T. and P.A. Lee, *Fault Tolerance: Principles and Practice (Dependable Computing and Fault Tolerant Systems, Vol 3)*. 2nd Revised ed. 1990: Springer Verlag.

Appendix A – Component-Pair Assessment

Assume that the attribute of interest is the probability of failure on demand (*pfd*). Now assume that the system is subjected to a series of independently selected demands. On each demand the response received from the COTS components is characterized as correct/incorrect. But since we have two COTS components clearly 4 combinations exist, which can be observed on a randomly chosen demand, as shown in Table 1 of section 3.

The four probabilities given in the last column of Table 1 sum to unity (i.e. they sum to 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 probabilities, e.g. $f_{p_{01},p_{10},p_{11}}(\bullet,\bullet,\bullet)$, gives an exhaustive description of the behavior of a COTS components pair. In statistical terms, the model has three degrees of freedom.

The probabilities of getting an incorrect response on a random demand from COTS A, let's denote it p_A, or COTS B, p_B, respectively, can be expressed as:

$p_A = p_{10} + p_{11}$ and $p_B = p_{01} + p_{11}$.

p_{11} represents the probability of receiving an incorrect response from both the COTS components. Hence, a notation $p_{AB} \equiv p_{11}$ will capture better the intuitive meaning of the event it is assigned to. Instead of using $f_{p_{01},p_{10},p_{11}}(\bullet,\bullet,\bullet)$ another distribution, which can be derived from it through functional transformation, can be used. We use $f_{p_A,p_B,p_{AB}}(\bullet,\bullet,\bullet)$. We define the joint prior distribution as:

$$f_{p_A,p_B,p_{AB}}(\bullet,\bullet,\bullet) = f_{p_{AB}|p_A,p_B}(\bullet \mid p_A,p_B)\ f_{p_A,p_B}(\bullet,\bullet) \qquad (3)$$

under the assumption that p_A and p_B are independently distributed, i.e.

$$f_{p_A,p_B}(\bullet,\bullet) = f_{p_A}(\bullet)\ f_{p_B}(\bullet) \qquad (4)$$

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

$$f_{p_A,p_B,p_{AB}}(x,y,z \mid N,r_1,r_2,r_3) = \frac{f_{p_A,p_B,p_{AB}}(x,y,z)L(N,r_1,r_2,r_3 \mid p_A,p_B,p_{AB})}{\underset{p_A,p_B,p_{AB}}{\iiint} f_{p_A,p_B,p_{AB}}(x,y,z)L(N,r_1,r_2,r_3 \mid p_A,p_B,p_{AB})dxdydz} \qquad (5)$$

where

$$L(N,r_1,r_2,r_3 \mid p_A,p_B,p_{AB}) = \frac{N!}{r_1!r_2!r_3!(N-r_1-r_2-r_3)!}(p_A-p_{AB})^{r_1}(p_B-p_{AB})^{r_2}\,p_{AB}^{\,r_3}(1+p_{AB}-p_A-p_B)^{N-r_1-r_2-r_3} \qquad (6)$$

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

The marginal distribution $f_{p_{AB}}(\bullet)$, which is used for comparison of the COTS component pairs, can be derived from $f_{p_A,p_B,p_{AB}}(\bullet,\bullet,\bullet)$ by integrating out p_A and p_B, i.e.

$$f_{p_{AB}}(\bullet) = \underset{p_A p_B}{\iint} f_{p_A,p_B,p_{AB}}(\bullet,\bullet,\bullet)dp_A dp_B \qquad (7)$$

Appendix B – Partitions Theory

If the demand space is partitioned into M partitions $\{S_1, S_2, \ldots S_M\}$, then for each of these the assessment will be performed as described in section 3.3, e.g. with two COTS components the description provided in section 3.3 (with details given in Appendix A) will apply. As a result M conditional distributions will be associated with each pair of COTS components, e.g. using two components these can be denoted as $f_{p_A,p_B,p_{AB}}(\bullet,\bullet,\bullet \mid S_i)$, from which the conditional uncertainty $f_{p_{AB}}(\bullet \mid S_i)$ will be expressed. This distribution characterizes the probability of failure, $P_{AB} \mid S_i$, of both components in the specific partition. Finally, in order to compare the competing COTS pairs the unconditional distribution $f_{p_{AB}}(\bullet)$ should be derived for the particular profile defined over the set of partitions, which represents the targeted operational environment.

Let us denote the profile of the targeted environment as $\{ P(S_1),\ldots,P(S_M) \}$, and assume that these are known with certainty. The marginal probability of failure of a COTS component pair, according to the formula of full probability is:

$$P_{AB} = \sum_{i=1}^{M} P_{AB} \mid S_i \times P(S_i) \qquad (8)$$

The distribution of this random variable, P_{AB}, depends on the joint distribution, $f_{P_{AB}|S_1,\ldots,P_{AB}|S_M}(\bullet,\ldots,\bullet)$, 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.:

$$f_{P_{AB}|S_1,\ldots,P_{AB}|S_M}(\bullet,\ldots,\bullet) = \prod_{i=1}^{M} f_{P_{AB}|S_1}(\bullet)\ldots f_{P_{AB}|S_M}(\bullet) \quad (9)$$

Such an assumption represents the assessor's belief that learning something about the probability of failure, $P_{AB} \mid S_i$, of a particular COTS component pair in partition i will not change their belief about the probability of failure, $P_{AB} \mid S_j$, of the same COTS component pair 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 (9) the unconditional probability of COTS component pair failure (8) can be expressed as a convolution of the distributions of the random variables $P_w(i) = P_{AB} \mid S_i \times P(S_i)$, i.e.:

$$P_{AB}^{w} = \otimes P_w(i) \qquad (10)$$

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