

Bayesian approaches to software testing

Simon C Shaw & Michael Goldstein
Department of Mathematical Sciences
University of Durham, UK

October 11, 2001

Topic areas: 3, 8, 14

Testing of large software systems, prior to release, is an enormously expensive, time consuming and critical component of the software development process. Exhaustive testing and remedying of all faults prior to release is rarely feasible. Competition and consumer demand are driving forces behind a rapidly changing market for software. Products have to be released more frequently and produced in shorter time scales to prevent the evolution of the users' needs making them obsolete. Test managers have to attempt to minimise the time and resources allocated to testing whilst ensuring that the subset of all possible test cases chosen maximises the confidence in the product of the supplier and consumer.

Rees *et al.* (2001) provides an overview of the problems faced by software testers and test managers, highlighting five main themes. (i) The quality and completeness of the information available on the system under test. (ii) The ability to use all system information and the testers' experience and domain knowledge in the best way. (iii) The ability to optimise testing against different criteria, e.g. finding faults fastest, maximising reliability or business value, and minimising business risk. (iv) The ability to use system knowledge and test results to predict the source of faults and the effects of any corrections. (v) The ability to judge what test results indicate about the quality of the system.

As software testing may be considered as the process of reducing uncertainty about software reliability by careful choice of test-suite, it is ideally suited to a Bayesian formulation. Recent research at Durham, see Wooff *et al.* (2001), Rees *et al.* (2001) and our project homepage (<http://maths.dur.ac.uk/stats/softtest/home.html>), has seen the development of such Bayesian methods to support testers working from a black-box testing perspective. The approach is based on the creation of a novel Bayesian graphical model for software testing as follows:

1. Creation of a form of pseudo-code which identifies (by plausible conjecture) the operations that must be performed on the inputs to achieve the various output effects. This establishes a partition over the input space and a representation of the sequence of processes to be undergone to achieve each of the functions of the software.
2. This representation is used to develop the qualitative form of the graphical model. We partition the input space into exchangeable subsets, for which each test input is considered to be equally informative for the function that is being tested. The nodes on the graph correspond to observable 'test' nodes, divided according to the relevant partitions of inputs for each aspect of functionality that is being tested, and unobservable link nodes which express 'sources' of the various errors that we might observe in the tests of the software, selected so that tests are conditionally independent given error sources.

3. We then quantify this structure, expressing probabilities over the root nodes for sources of error and conditional probabilities over the arcs which join the nodes to express beliefs about the likely propagation of errors through to the test outcomes.

4. Because the model is held in full probabilistic form, as each test is carried out the model may be coherently updated, and all uncertainties contributing to the reliability of the software recalculated. This model may therefore be used to determine how much testing is required to meet any management specified requirement on likely software quality prior to release, to derive efficient test suites to ensure high probability of meeting such overall reliability targets, and to sequence optimally the order of the tests to minimise the expected amount of retesting that is required.

The Bayesian graphical model for software testing thus offers an integrated approach to test suite construction, analysis and risk management. The approach has been tested with industrial collaborators and considered to be very promising.

Although the Bayesian approach offers a powerful method for combining expert judgements with experimental results, in large complex problems we may be unable to specify carefully a meaningful full prior distribution over all the variables or if we can give such a full prior specification the analysis may be too difficult to carry out. In particular, in the approach outlined above, there are certain aspects of the structure, based around the conditional error levels within related test-types, for which we currently do not make joint probabilistic specifications for precisely these reasons. An alternative approach which is intended to cope with such complexity lies in Bayes linear methodology, which, through taking expectation as primitive, minimally requires specification and analysis of variance structures and can therefore be used to introduce a much richer dependency structure into our models. Coolen *et al.* (2001) describes the use of Bayes linear methods related to partition testing for software. A very common assumption made when discussing partition testing is that test results for inputs in one subdomain do not provide any information about inputs in other subdomains. Knowledge about the relationship between different subdomains may be weak and a full specification may be a unrealistic representation of our actual beliefs. In a Bayes linear approach, a dependency may be introduced by considering test results in the same subdomain to be second-order exchangeable and co-exchangeable across subdomains. The additional specification requirement is a single covariance between each pair of subdomains and belief revision remains straightforward. The potential downside is that, by adopting a Bayes linear approach, we may lose probabilistic information.

As an example, the case study contained in Wooff *et al.* (2001) considers a piece of software whose function is to process an input number, such as a credit card number, in order to perform an action, such as to check whether the account corresponding to the credit card is in credit. A partition over the input space is formed identifying subdomains such as short numbers and long numbers. We must consider the proportion p_S and p_L of short and of long numbers respectively, which will give the correct response when tested. Given p_s , each test on a short number is a Bernoulli trial with a success probability of p_s : a Bayes linear approach would not fully capture this relation. However, a full Bayes approach would require a meaningful joint probability specification between p_S and p_L : such a specification is extremely difficult in practice when there are many such test domains.

Can we develop methods which gain the strengths of both approaches? We shall describe methods for Bayesian augmentation for Bayes linear models, introducing graphical models where beliefs relating connected nodes are either fully specified or partially specified. We term these Bayes linear/Bayes (BLB) models. In the software testing framework, the models concerned have fully specified test subdomains but the relationships between the error levels for the subdomains is modelled using a Bayes linear approach. The propagation of test results from a single subdomain is considered and revised expectations and variances are obtained

using a mixture of conditioning and Bayes linear belief adjustment. Propagation of test results from multiple subdomains is then developed. The approach exploits the conditional independence structure of graphical models and a local computation algorithm for global adjustment of the BLB model is provided. This algorithm builds upon that developed by Goldstein & Wilkinson (2000).

In summary, we shall describe:

- (i) a variety of Bayesian approaches to software testing;
- (ii) methods for using Bayesian graphical models to direct general system testing to high reliability;
- (iii) new types of Bayesian graphical models with fully and partially specified components.

References

- Coolen, F.P.A., Goldstein, M. and Munro, M. (2001). Generalized partition testing via Bayes linear methods. *Information and Software Technology* (to appear).
- Goldstein, M. and Wilkinson, D.J. (2000). Bayes linear analysis for graphical models: The geometric approach to local computation and interpretive graphics. *Statistics and Computing* **10**, 311-324.
- Rees, K., Coolen, F., Goldstein, M. and Wooff, D. (2001). Managing the uncertainties of software testing: a Bayesian approach. *Quality and Reliability Engineering International*, **17**, 191-203.
- Wooff, D.A., Goldstein, M. and Coolen, F.P.A. (2001). Bayesian graphical models for software testing. *IEEE Transactions on Software Engineering* (to appear).