

Diversity for off-the-Shelf Components

Peter Popov*, Lorenzo Strigini*, Alexander Romanovsky**

*Centre for Software Reliability, City University, London

**Department of Computing Science, University of Newcastle upon Tyne
{ptp, strigini}@csr.city.ac.uk, Alexander.Romanovsky@newcastle.ac.uk

"Commercial-off-the-shelf" (COTS) or, generally, "off-the-shelf" (OTS) software items are increasingly used in building systems, instead of only relying on bespoke software items¹. This trend is driven by a wish to reduce costs, and by some hope that greater re-use of software may lead to higher quality (via more feedback from use). Thus, for instance, the U.S. Dept of Defence policy is now to encourage the use of COTS items. This trend extends to critical systems with high dependability requirements, like a computer-based railway signalling systems by Alcatel (Austria) [1].

A serious problem with OTS items (software, but also complex digital hardware) is that they often lack the guarantee of good development practice, and the extensive documentation of it, which are traditionally the basis for accepting/certifying software for critical applications. Even for commercial applications with modest dependability requirements, using OTS items requires some trust that they will not become a "weak link", making the final product intolerably unreliable. If the OTS items have already seen much operational use, this experience could be used to forecast their dependability in a new context; but this experience is seldom documented with sufficient accuracy and detail to allow confident predictions. Much of the on-going discussion about OTS items addresses this issue. A notorious case of a U.S. warship being disabled by a crash of Windows NT in 1997 [2] is often quoted to illustrate the problems with dependence on COTS items.

The debate is open about how COTS items could be certified to have sufficient reliability but what is clear is that a solution will not be widely available any time soon.

¹ Non-commercial OTS items are widespread in specific applications, some of them with high aggregated economic value (e.g., Apache Web servers). Their situation differs in some respects from that with commercial OTS items, but the differences are of degree within a spectrum of possibilities, not of kind. E.g., "open-source" software removes the problem of inaccessible source code, but not those of poor requirement traceability and undocumented coverage of the V&V processes.

Fault tolerance ("design diversity") [3] is an attractive way of improving reliability, despite problems in evaluating the actual gain. An integrator or customer, faced with an OTS item which nominally satisfies the functional requirements for a certain component in a new system, but whose reliability is insufficient (or not proven to be sufficient) for the purpose, cannot easily improve the OTS item's reliability (e.g. by further debugging) or the evidence thereof (e.g. by further inspection and testing): the design documentation is inaccessible or in any case unfamiliar to the customer's staff, and paying for an extensive validation effort would deny the economic advantage of using an OTS item (i.e., the sharing of fixed costs among many customers). Fault tolerance, instead, may be applied without requiring access to the internals of the OTS item:

1. one can procure one or more additional OTS items with similar functionality, and configure them as a 1-out-of-N or majority-voted system. In some applications, building 1-out-of-N systems is trivial: e.g., adding alternates to complete communication or alarm systems, with the human users "adjudicating" among the alternative results available), or to servers operating in distributed systems with good "fail-stop" properties. Diversity in the form of multiple, functionally equivalent, diverse components ("multiple version software") is often seen as intolerably expensive, unless required by extreme dependability requirements. But OTS items make it much *cheaper*: using multiple OTS versions is often cheaper than developing one, high-quality item. So, the availability of OTS components may make the multiple-version approach both desirable and *affordable*.
2. alternatively, the system can be protected against its OTS components by additional components that either monitor them for deviations from their specified behaviour, or for violations of a known "safety envelope" of behaviours that do not endanger the rest of the system. Such components, which in the practice of dependable design are variously called watchdogs,

safety monitors, acceptance tests, audit programs, etc., are in many cases comparatively simple and cheap to develop even if they must be developed ad hoc for a specific project.

So, fault-tolerance techniques are good candidates for improving the dependability of OTS items, and especially for providing *modest increases* in dependability at *modest cost* (although we do not exclude their use - at higher cost in either development or assurance activities - for satisfying more stringent requirements in the more critical applications).

The possible cost-effectiveness benefits of employing

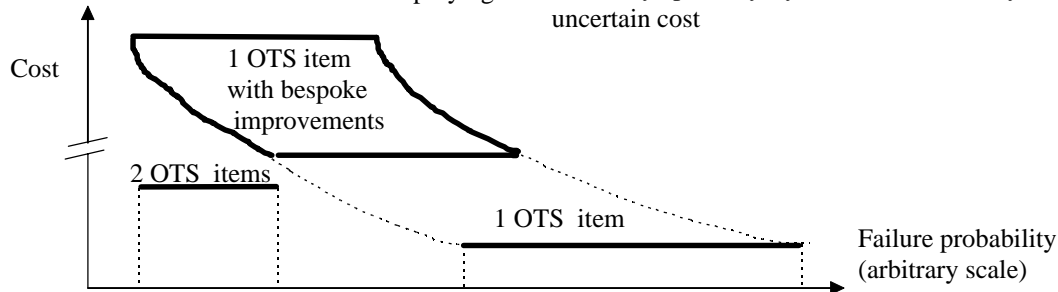


Fig. 1 A sketch of the possible cost-effectiveness of a 1-out-of-2 COTS-based solution compared with a single-version OTS based solution with additional efforts to improve reliability.

The communities affected by the "COTS trend" recognise to some extent the potential of diversity techniques, without actually using this term. A typical approach is to give some protection function, as in case 2 above, to "wrappers". However, not recognising that these are special cases of software fault-tolerance may prevent the re-use of the extensive knowledge existing on the topic. Even the simple design need of matching the protection functions to the failure modes of the (OTS or bespoke) components is not usually addressed systematically.

Another example of employing diversity, similar to case 1 above, is reported by Allaire Corporation [5]. The authors advocate running several Java virtual machines (JVM) on a server for executing Java servlets, as a remedy against the 'dialects' of Java, implemented by the competing vendors. Servlets 'speaking' Microsoft's dialect of Java (with Microsoft extensions) will be directed for execution to Microsoft JVM, while those 'speaking' Symantec dialect will be directed to Symantec JVM, etc. This idea can easily be extended so as not to require the complete a priori knowledge of which is the 'native' JVM for the executed servlet, instead making the VM a fault-tolerant architecture which automatically masks failures to execute a servlet by any proprietary VM.

In summary, the proposed use of fault-tolerance with OTS components seems a promising compromise between the cost, which is substantially lower than for bespoke systems, and the gain in reliability, which may be significant. If the allocated budget allows for using

diverse OTS items (solution 1 above) are illustrated in Fig. 1.

The cost of a diverse OTS solution is roughly twice the cost of a single OTS solution. The reliability is not known with certainty, which is represented by the wide bars in the graph. We have shown elsewhere [4] that under plausible assumptions the uncertainty will be lower in the case of a two-version system, than in the case of a single version: hence the lower bar associated with a single OTS solution is wider than the bar associated with the two-OTS solution. Improving the single OTS item may improve reliability (possibly by much) but certainly at a high and uncertain cost

multiple OTS items or combination of OTS items and bespoke monitor components, and the required level of reliability/confidence is not too much greater than what each of the individual COTS can provide, fault-tolerance is very likely to deliver the required reliability. The known experience with bespoke fault-tolerant software is in favour of such conclusion: a modest improvement in reliability (2-10 times lower probability of failure per demand, for instance) is a reasonable expectation.

References

- [1] H. Kantz and A. Veider, "Design of a Vital Platform for Railway Signalling Applications", in Proc. 10th European Workshop on Dependable Computing (EWDC-10), Vienna, Austria, 1999, pp. 37-41.
- [2] G. Slabodkin, "Software glitches leave Navy Smart Ship dead in the water", Government Computer News (GCN), 1998:
<http://www.gcn.com/archives/gcn/1998/july13/cov2.htm>,
- [3] M. R. Lyu (Ed.), "Software Fault Tolerance", Wiley, 1995.
- [4] P. Popov, L. Strigini and M. Pizza, "The efficacy of diverse redundancy against design error: some practical considerations", in Proc. INucE 3rd Int. Conf. on Control and Instrumentation in Nuclear Installations, Edinburgh, U.K., 1998,
Also http://www.csr.city.ac.uk/csr_city/projects/diversity/
- [5] C. Allaire, "Allaire Run: Edition comparison", Corporation Web site,
www.allaire.com/products/jrun/MoreInformation/ChoosingTheEdition.cfm.