Wrapping it up

by T. Anderson, M. Feng, S. Riddle, A. Romanovsky

Many siren voices, and some harsh economic facts, argue in favour of *off-the shelf* (OTS) components as a way to reduce the costs of software system development. Compared to bespoke design and development, the OTS option offers a number of potential benefits, including: immediate availability, proven in use, low price due to amortisation. The increasing scale and complexity of modern software systems is a powerful driver for modularity in design, which clearly chimes with a structured and therefore component (or sub-system) based approach.

The need for economy is often most keenly felt in expensive systems, and this can certainly be the case for systems that have critical requirements, including, of course, safety-critical systems. But it is in the nature of these systems that they **must** deliver on their requirements; their operational behaviour must exhibit dependability; they must do what they are supposed to do, and must not do what is prohibited (almost always, anyhow). With a completely bespoke development, designers can strive very hard to achieve a dependable system, and regulators can obtain access to extensive information on the development process (as well as the delivered product) in order to evaluate a documented justification that the system will meet its critical requirements (eg a safety case). Utilisation of an OTS software component is likely to inhibit this evaluation, since – in the extreme case - the component may have to be viewed as a black box, with no information on its inner workings or its development; the proven-in-use evidence for the component's suitability may be valid (or not) but will frequently be anecdotal and is unlikely to relate to an identical use environment.

So, consider the situation where use of an OTS software component is feasible and there is a strong financial reason for doing so, but the component's behaviour needs to be trusted, and we have insufficient evidence to justify that trust. How can we proceed? The approach to be considered in this article is a simple application of diversity to provide an architectural solution. The OTS component will be enclosed in a bespoke *wrapper* [1], a purpose designed additional component that intercepts all inputs to, and outputs from, the OTS component in order to monitor its behaviour. The aim is for the wrapper to deal with any problems arising from inadequate behaviour of the OTS component, ideally masking any such effects from the rest of the system.

The DOTS project

The authors are the Newcastle members of DOTS – "Diversity with OTS components", which is a joint project at CSR in Newcastle and City Universities, funded by EPSRC. Work at Newcastle is exploring architectural approaches to diversity in the presence of OTS items, while colleagues at City are focussing on assessment of the benefits that can be expected.

Our architectural exploration has concentrated on *wrapper technology*, a phrase which gives an appealing technical ring to this simplistic approach of enveloping a possibly suspect component. However, despite its apparent simplicity there are a range of issues to consider and questions to be asked. We believe that our work gives some encouragement that positive answers may be given to the following questions:

(i) is wrapper technology feasible in practical systems?

(ii) can wrappers detect and respond successfully to erroneous behaviour?

In order to draw conclusions for practical systems we seek realism in our investigation. Experimentation with a real world critical system would be fraught with peril, so we have made use of a software model of a real-time software system. Specifically, we have adapted an industrial grade simulation of a steam boiler and its associated control system. Written in Simulink, the model represents a real steam raising system in which a coal-fired boiler responds to demands for steam under the operational authority of an automated control system; the control system consists of a PID (which stands for proportional, integral and derivative) controller together with a range of smart sensors, actuators, and configuration controls.

We chose to treat the PID controller as an OTS item, and then developed a (simulated) protective wrapper that can monitor and, when appropriate, modify all input/output signals to the PID controller from the rest of the control system. In designing the wrapper we only had access to limited information about the operation of the boiler and the control system; furthermore we ignored any details of the inner working of the PID controller, treating it as a black box. Ideally we would have liked a full external specification of the PID controller, but the lack of this made our position even more realistic.

In creating our own, approximate, specification for the PID controller we built up a set of *Acceptable Behaviour Constraints* (ABCs) which stipulate what may be considered as acceptable behaviour at the interface between the PID controller and the rest of the control system.

Error detection

Given the limitations of the scenario we were exploring our strategy for error detection was necessarily based on a systematic application of generic criteria. Erroneous situations can arise anywhere in the system, but the wrapper can only check for "cues" at the interface to the PID controller. The wrapper was programmed to cyclically check inputs to the PID controller against constraints established as ABCs (missing, invalid, unacceptable, marginal or suspect values from the sensors or configuration variables). Similarly, outputs from the PID controller were also checked against ABCs (missing, invalid or unacceptable values intended for the actuators).

To help inform the next stage of wrapper design we categorised these error cues as follows:

(a) Unavailability of signal (inputs or outputs),

(b) Signal violating specified constraints (usually out of range errors),

(c) Excessive signal oscillations (in amplitude or frequency)).

Additionally, we recognised that with respect to the safety of the system, some erroneous situations are much more acute than others. In a steam boiler, a key parameter is the "drum level"; this parameter measures the quantity of water contained in the boiler drum more accurately than the "water level" (which is also monitored, of course). Too little water and the boiler tubes are exposed to heat stress, too much and water could go over the header causing corrosion. The danger of excessive steam pressure is obvious and explosive. Thus detected errors involving any of drum and water level, drum and bus steam pressures were designated as needing an immediate and effectual response.

Error recovery

The purpose of error recovery is to transform a system state that contains errors to one that does not. *Backward* recovery returns the system to a previous state, prior to the incidence of error, and is unlikely to be available for OTS components. So our protective wrapper should attempt to implement application-specific *forward* recovery, which does not discard the current state. Exception handling provides a general framework for forward recovery.

We implemented three elementary recovery actions:

H1: reset signal to normal value and alert operator

H2: wait Δt, if error goes away no action taken; otherwise send alarm to operator and wait ΔT, no further action if error goes away; otherwise invoke H3 [delay times Δt and ΔT chosen by the wrapper designer]

H3: shut system down and send alarm to operator.

We then devised a rationale for a recovery strategy in which:

All errors for PID controller outputs invoke H1;

All errors from configuration controls invoke H1;

All errors for PID inputs (except drum level and steam pressure) invoke H2;

Category (a) and (b) errors on drum level and steam pressure invoke H3;

Category (c) errors on drum level and steam pressure invoke H2.

Adopting this (or any other) strategy for an actual boiler plant would require safety analysis and justification. In our experimental situation, having implemented a wrapper with a detection and recovery capability we now wish to see how well the system responds. Initial test exercises have given very positive results, and the next phase is to set up, run and record the outcomes for a range of scenarios. We will also explore, with our colleagues at City, the extent to which indicative statistical measures of performance may be obtained.

Conclusion and further issues

Thus far we merely claim to have built a reasonably realistic, albeit rather simplistic, demonstrator of a protective wrapper in action, with encouraging early results. The exercise of working with the demonstrator has helped us to address a number of generic concerns, which we consider in more detail elsewhere [2,3]. We intend to extend the evaluation of the demonstrator and use it to examine a range of further issues. These may include:

wrapper deficiencies – the role of the wrapper is to improve dependability, including safety, but there is always a risk that adding an additional software component could introduce new failure modes; the best general guidance is to keep wrappers as simple as possible

formal development of wrappers – another approach to minimising problems induced by a wrapper is to adhere to a fully rigorous development methodology; we see a basis for progress here based on contracts derived from the ABCs, and exploiting compositional semantics

timing issues – we have skirted around these, but there is considerable scope for considering deadlines and delays (or, much more basically, how long before a signal is deemed to be missing?)

scoping issues – to what extent should a wrapper have access to variables elsewhere in the system (almost always excluding internal variables within the OTS component)

modelling issues – how best to minimise concerns that decisions and design based on modelled systems will not reflect real-world implementations?

safety issues – standards, hazard analysis, safety cases …

Work has only just begun.

References

1. J Voas, Certifying OTS Software Components, IEEE Computer 31, 1998, pp 53-59.

2. T Anderson *et al*, Protective Wrapper Development, ICCBSS 2003, Ottawa, pp1-14.

3. T Anderson *et al*, Error Recovery for a Boiler System with OTS PID Controller, CS Tech. Rep 798, May 2003, Univ. of Newcastle presented at ECOOP 2003, Darmstadt

The authors are all at CSR, Univ. of Newcastle and can be contacted via csr@ncl.ac.uk