WRAPPING THE FUTURE

Tom Anderson, Brian Randell and Alexander Romanovsky
School of Computing Science, University of Newcastle upon Tyne, UK
{tom.anderson, brian.randell, alexander.romanovsky}@newcastle.ac.uk

Abstract: Enclosing a component within a software "wrapper" is a well-established way of adapting components for use in new environments. This paper presents an overview of an experimental evaluation of the use of a wrapper to protect against faults arising during the (simulated) operation of a practical and critical system; the specific context is a protective wrapper for an off-the-shelf software component at the heart of the control system of a steam raising boiler. Encouraged by the positive outcomes of this experimentation we seek to position protective wrappers as a basis for structuring the provision of fault tolerance in component-based open systems and networks. The paper addresses some key issues and developments relating wrappers to the provision of dependability in future computing systems.

Key words: dependability; off-the-shelf components; fault tolerance; protective wrapping.

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 with 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 such a 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 (such as safety-critical systems). But it is in the nature of these systems that they really **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 (well, almost always). 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 (for example, 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, or may not, be valid, but such evidence cannot be relied upon if it is merely anecdotal or does not 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 paper is a simple application of diversity to provide an architectural solution. The OTS component will be enclosed in a bespoke *protective wrapper* [Voa98, Arl02], 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 deleterious effects from the rest of the system. This is thus a special case of the more general use of software wrapping, a technology that has a long history of use as a means of adapting existing components for use in new environments.

1. THE DOTS PROJECT

DOTS – "Diversity with OTS components" – is a joint project at CSR (Centre for Software Reliability) in Newcastle and City Universities, funded by the UK EPSRC. Work at Newcastle is exploring architectural approaches to diversity in the presence of OTS items, while colleagues at City concentrate on assessment of the benefits that can be expected.

Our architectural exploration has concentrated on *protective wrapper technology,* a phrase that 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:

- is protective wrapper technology feasible in practical systems?
- can protective wrappers detect and respond successfully to erroneous situations in practical systems?

In order to draw conclusions for practical systems we sought realism in our investigation. Experimentation with a real-world critical system would be fraught with peril, so we made use of a software model of a real-time software system. To maximise realism we adapted a model taken directly from industry: a Honeywell-supplied industrial grade simulation of a steam boiler and its associated control system. Written in Simulink [Mat], 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 (i.e. Proportional, Integral and Derivative) controller together with a range of smart sensors, actuators, and configuration controls (collectively referred to as the ROS (rest of system)), as illustrated in Figure 1.

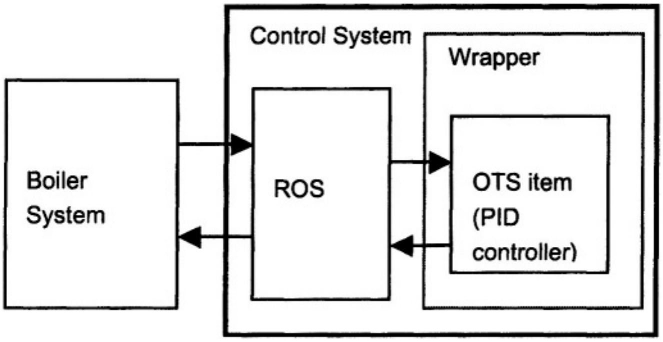

Figure 1. Boiler System and Control System

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 between the PID controller and the rest of the control system. As a purely protective wrapper there was no intention to adjust or upgrade the PID controller's behaviour at the interface; the aim was simply to provide fault-tolerant elements that can detect and recover from errors. In designing the wrapper we only had access to limited information about the operation of the boiler and the control system; furthermore we deliberately ignored any details of the inner working of the PID controller, treating it as a black box. Ideally we would have wished for access to 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) [Pop01] which stipulate what may be considered as acceptable behaviour at the interface between the PID controller and the rest of the control system.

2. 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 symptomatic "cues" at the interface to the PID controller. The wrapper was programmed to check inputs to the PID controller cyclically 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). Thus we were attempting to deal with both the PID's monitoring and its control activities.

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

- unavailability of signal (inputs or outputs),
- signal violating specified constraints (usually out of range errors),
- 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 mass 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 that concerned either steam pressure or the quantity of water in the drum were designated as needing an immediate and effectual response. (A boiler operated via a PID controller is one of the most widely deployed systems, installed in many industrial facilities and residential houses, for the safe and reliable generation of steam and/or hot water. Nevertheless, critical incidents in these systems lead to deaths and injuries every year [Nat03].)

3. 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 attempts to implement application-specific *forward* recovery, which does not discard the current state. Exception handling provides a general framework for such 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 PID input errors (except drum level and steam pressure) invoke H2
- excessive signal oscillation errors concerning drum level and steam pressure invoke H2
- all other errors concerning drum level and steam pressure invoke H3.

[Adopting this (or any other) recovery strategy for an actual boiler plant would, of course, require safety analysis and justification.]

In our experimental situation, having implemented a protective wrapper with a detection and recovery capability we need to observe how well the system responds. Initial test exercises gave very positive indications, and we have just completed a first phase of setting up a range of fault injection scenarios, running these, and recording the outcomes. Our fault injection scenarios involve signal communication faults (bias, random noise, stuck-at previous, stuck-at random) and faults that impinge directly on the algorithms of the PID controller (transient zeros, control parameter overwrites). A preliminary examination of the experimental data generated indicates that the wrapper has been very effective in reducing serious failures of the boiler system.

4. PROTECTIVE WRAPPING – WHAT NEXT?

Thus far we merely claim to have built a reasonably realistic, albeit rather simplistic, demonstrator of a protective wrapper in action to enhance the dependability of an industrial OTS component, with encouraging early results. The exercise of working with the demonstrator has helped us to address a number of specific concerns, which we consider in more detail elsewhere [And03a, And03b, And03c]. However, and much more significantly, we now discern a salient role for wrapper technology as a means for structuring, designing and building future ICT systems. We believe that protective wrapping has considerable promise as a uniform approach for incorporating fault tolerance into new and existing complex systems, particularly when these are organised and created as integrative networks of interacting components.

Openness is often considered to be one of the defining characteristics of the ICT systems that are expected to be widely deployed in future – systems that will support mobile access and that are pervasive of society; systems that deliver ambient "intelligence" in terms of services, information, processing and communication; networks of heterogeneous systems/components which interact with (and depend upon) other networks; sub-networks that combine and decompose dynamically. Openness allows on-line composition, reconfiguration, evolution and upgrading, performed on the basis of a dynamic analysis of the available information representing possible changes. This flexibility is made possible by features that can be used dynamically to select or devise optimal configurations, and then to realise these configurations by locating, deploying and integrating the appropriate components. Openness is usually understood in the widest possible sense, in that it should allow systems to deal with changes in: requirements, location (mobility), quality of service (QoS) characteristics, the environment, component behaviour, users' expectations, users' behaviour, etc. Clearly this must include dealing with changes due to accidental or malicious faults.

There is a significant challenge in identifying and developing fault tolerance solutions that fit the specific characteristics of open systems; protective wrapping has considerable potential to be one of the fundamental fault tolerance techniques needed, primarily because it sits well with the open network based approach but also because it provides such a simple but general starting point. Of course, there are numerous issues that will need to be examined further before that potential can be converted into a fully effective approach. Among these issues we can draw attention to:

- wrapper deficiencies – the role of a protective wrapper is to improve system dependability by providing an error detection and recovery capability, but there is always a risk in including defensive mechanisms for fault tolerance that by adding an additional software component new opportunities for erroneous computation may arise; the best general guidance is to keep wrappers as simple as possible
- formal development of wrappers – the need to minimise the risk of a protective wrapper introducing additional fault modes is not only a driver for simplicity in wrapper design, but also for adherence to stringent and rigorous development practice; we see a basis for progress here based on contracts derived from constraint-based specification of component interfaces (e.g. ABCs), and by exploiting compositional semantics
- timing issues – there is considerable scope for considering how best a protective wrapper should stipulate and monitor deadlines, and react to

delays; strategies and protocols from distributed systems work will need to be adapted for open networks

- scoping issues – the most simplistic image of a protective wrapper gives it full access to all communications across the wrapped component's interface and no access to any variables elsewhere in the system (either internal to the wrapped component or in its environment), but any realistic implementation is likely to deviate from this artifice; it will rarely be feasible or necessary to control each and every component interaction, access to internal values (though often undesirable and/or inhibited by lack of knowledge) may be possible and of benefit in special circumstances, and supplementary information about external conditions could be an invaluable guide to the operation of the wrapper (both for error detection and response)
- wrapper interactions – a set of protective wrappers in a network of wrapped components may need to be able to communicate and interact in order to best achieve their several and collective dependability objectives; imposing constraint and mediation on inter-wrapper communication is likely to involve approaches based on interactive consistency solutions

The above list of bulleted topics could be extended almost indefinitely, for instance with issues from specific dependability domains (safety and security, for example) and more general systems engineering concerns (modelling and requirements, for example) but instead we move on, to relate the technology of protective wrapping to anticipated developments in the dependability of systems more generally.

A recently-commissioned Foresight document [Jon04] identified a number of core directions for future research in dependability, including: dependability-explicit systems, cost-effective formal methods, architecture theory, and adaptivity. In the remainder of this paper we outline how advances in these four areas could impact on protective wrapper technology, and vice-versa.

Dependability-explicit system development [Kaâ00] is an emerging area of research which supports the explicit incorporation of dependability-related information into system development artefacts right through the development life-cycle, starting from the earliest phases of development, and continuing through to on-line support for maintaining, updating and exporting this dependability information within the operational system (with reference to the current state of the system and its environment). Examples of dependability information are fault descriptions, expected normal/abnormal behaviour, redundancy resources, mappings between errors and handlers, abnormal situations that components are capable of handling, failure frequency data, etc. Protective wrapping fits this approach extremely well, since such data will be relevant for informing decisions by the wrappers relating to error detection and recovery, potentially allowing the evolution of wrappers in response to changes in system and environment behaviour, and after network reconfiguration. Furthermore, wrappers are an obvious candidate mechanism for processing the dependability information, and publishing it across the network.

Research on *cost-effective formal methods* will contribute to overall system dependability by accumulating a set of advanced tools (operating within an open platform) for cutting-edge formal development methods focussed on fault tolerance, mobility and adaptivity. Formal models of future open systems will enable wrappers to be rigorously described, and the systems containing them to be formally analysed. A key research objective is the development of tools that can analyse the models of a system and a specific component, from that analysis determine requirements for a protective wrapper for the component, and then generate the wrapper model – with the long term objective that this analysis and generation can be performed fully automatically as an adaptive open system evolves.

Architecture theory research aims to provide methods for reasoning about dependability concerns at an architectural level much earlier in system development. Protective wrappers provide a major structuring approach that embodies fault tolerance capabilities (including confinement of error propagation, and exception handling), but they will need to have adequate architectural support. In particular, there is a need to introduce recursive architectural solutions that can integrate wrappers within the architectural styles that will be typical for future open systems. Wrappers will then serve as a cardinal structure for introducing and managing redundancy at the architectural level. The focus should be on preserving architectural representations throughout all development phases until runtime execution to enable dynamic changes of architecture to be made online to improve overall system dependability.

It seems clear that future systems will need to have *adaptivity,* so that they can respond to changing environments, altered patterns of use, modified requirements and more. They will need to be dynamically upgradeable and reconfigurable; they will need to have a capacity for adjustment and evolution. Despite this mutability, users will demand that the dependable delivery of service be sustained. Wrappers could provide the fundamental structure supporting component-level adaptation and evolution; protective wrappers could embody fault tolerant defences in support of dependable operation.

5. CONCLUSION – ONLY A BEGINNING!

Protective wrappers offer a simplicity of concept and a generality of applicability that is attractive and encouraging. But it must be acknowledged that this welcome simplicity defers many of the difficult issues to the next stage of research and development.

We close this paper with the observation that all computing systems are (eventually) embedded in groups of humans – that is, in society. Members of society will need future computing systems to be wrapped as a protective mechanism and, in turn, it may be appropriate (in effect) to wrap the users to protect the systems. Very basic protective wrappers are already essential to shield us from the excesses created by something as trivial as spam e-mails!

ACKNOWLEDGEMENT

This work was partially supported by the UK EPSRC DOTS project.

REFERENCES

[And03a] T. Anderson, M. Feng, S. Riddle, A. Romanovsky. Error Recovery for a Boiler System with OTS PID Controller, *CS Technical Report 798,* May 2003, School of Computing Science, Univ. of Newcastle (presented at ECOOP 2003, Darmstadt).

[And03b] T. Anderson, M. Feng, S. Riddle, A. Romanovsky. Protective Wrapper Development. *Proc. 2nd Int. Conf. on COTS-Based Software Systems,* Ottawa, Canada, February 2003, pp. 1-14.

[And03c] T. Anderson, M. Feng, S. Riddle, A. Romanovsky. Investigative Case Study: Protective Wrapping of OTS items in Simulated Environments. *CS Technical Report 821,* December 2003, School of Computing Science, Univ. of Newcastle (submitted to COMPSAC 2004).

[Arl02] J. Arlat, J.-C. Fabre, M. Rodríguez, F. Salles. Dependability of COTS Microkernel-Based Systems. *IEEE Trans. on Computers.* **51**, 2, 2002, pp. 138-163.

[Jon04]C. B. Jones, B. Randell. *Dependable Pervasive Systems.* Foresight report, DTI, UK. 2004.

[Kaâ00] M. Kaâniche, J.-C. Laprie, J.-P. Blanquart, A Dependability-Explicit Model for the Development of Computing Systems. *Proc. SAFECOMP 2000.* Rotterdam, The Netherlands, 2000, pp. 107-116.

[Mat] Mathworks, Using Simulink: reference guide, http://www.mathworks.com

[Nat03] The National Board of Boiler and Pressure Vessel Inspectors. *2002 National Board Incident Report,* http://www.nationalboard.org/incidents/02-IR.html, 2003.

[Pop01] P. Popov, S. Riddle, A. Romanovsky, L. Strigini. On Systematic Design of Protectors for Employing OTS Items. *Proc. 27th Euromicro Conf.,* Warsaw, Poland, September 2001, pp. 22-29.

[Voa98] J. Voas. Certifying OTS Software Components, *IEEE* Computer **31**, 1998, pp. 53-59.