Error Recovery for a Boiler System with OTS PID Controller

Tom Anderson, Mei Feng, Steve Riddle, Alexander Romanovsky
School of Computing Science, University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, UK
{tom.anderson, mei.feng, steve.riddle, alexander.romanovsky}@ncl.ac.uk

Abstract.

We have previously presented initial results of a case study which illustrated an approach to engineering protective wrappers as a means of detecting errors or unwanted behaviour in systems employing an OTS (Off-The-Shelf) item. The case study used a Simulink model of a steam boiler system together with an OTS PID (Proportional, Integral and Derivative) controller. The protective wrappers are developed for the model of the system in such a way that they allow detection and tolerance of typical errors caused by unavailability of signals, violations of range limitations, and oscillations. In this paper we extend the case study to demonstrate how forward error recovery based on exception handling can be systematically incorporated at the level of the protective wrappers.

1. Introduction

Although integration of Off-The-Shelf (OTS) components into systems with high dependability requirements (including those that are safety-critical) is becoming a viable option for system developers, care must be taken to avoid a deterioration in overall system dependability. OTS components can often be of a lower quality than bespoke components, may not have been specifically intended for the environment in which they are to be deployed, and may be poorly documented. These factors can all contribute to a higher risk of failure for complex systems employing OTS components.

It must be accepted that OTS components will be utilised in such systems, and that their use will be a source of failure in spite of all efforts to improve the quality of OTS components and of the system in which they are to be integrated. The solution we advocate is to tailor specialised fault tolerance techniques to support the integration of OTS components into complex systems.

1.1. Protective Wrappers

In previous work [1,7] we illustrated an approach to the development of protective wrappers: a bespoke software module which intercepts all information going to and from an OTS item. This approach is developed further in this paper using the same case study.

Fault tolerance techniques have three main phases: error detection, error diagnosis and error recovery [5]. The first phase identifies an erroneous state; error diagnosis is then used to examine and assess the damaged area, to enable the system to move to an error-free state by means of error recovery. We have previously concentrated on detection and diagnosis, providing only limited recovery actions.

Component wrapping is an established technique used to intercept data and control flow between a component and its environment [6]. A protective wrapper may detect errors or suspicious activities, and initiate appropriate recovery when possible, and must be rigorously specified, developed and executed as a means of protecting OTS items against faults in the Rest Of the System (ROS), and the ROS against faults in OTS items. Sources of information for wrapper development include specification of the OTS item behaviour, known "erroneous" behaviour of the OTS item, and specification of the correct behaviour of the ROS with respect to the OTS item.

1.2. Case Study

The case study used in this paper concerns the development of a protective wrapper for an Off-The-Shelf PID (Proportional, Integral and Derivative) controller. This case study is intended to illustrate how the approach could be applied in practice, employing software models of the PID controller and the steam boiler system rather than conducting a potentially risky experiment in a real-world environment. Use of such software models is an active area of research and development carried out by many leading control product companies (including Honeywell [8]), and we

used a third-party model of a steam boiler in this case study. We believe that this decision adds credibility to our results. The model simulates a real controller and steam boiler system, enabling us to investigate the effect of wrapping with a representative model. In the course of our work we extended the model by incorporating protective wrappers.

1.3. Roadmap

The remainder of this paper is organised as follows. In the following section we describe the simulation environment, the controller and the boiler models we are using, and our approach to monitoring the model variables. Section 3 discusses the requirements for a protective wrapper, and the next three sections discuss design and implementation of the wrapper to detect, diagnose and select an appropriate recovery action for errors caused by unavailability of signals, violations of range limits, and signal oscillations. Section 7 concludes the paper by discussing the generic error recovery strategy and the possible impact of wrappers on the overall execution of the integrated system.

2. Simulation

2.1. Simulink

Simulink (Mathworks) [10] is one of the built-in tools in MATLAB, providing a platform for modelling, simulating, and analysing dynamical systems. It supports linear and nonlinear systems modelled in continuous time and sampled time, as well as hybrids of both. Systems can also be multi-rate, i.e., have different parts that are sampled or updated at different rates. Simulink contains a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors to allow modelling of very sophisticated systems. Models can also be developed through self-defined blocks by means of the *S-functions* feature of Simulink or by invoking MATLAB functions. After a model has been defined, it can be simulated and, using scopes and other display blocks, simulation results can be displayed while the simulation is running.

Simulink provides a practical and safe platform for simulating the boiler system and its PID control system, for detecting operational errors when boiler and control system interact, and for developing and implementing a protective wrapper dealing with such errors.

2.2. The Structure of the Model

The abstract structure of the system we modelled is shown in Figure 1. The overall system has two principal components: the boiler system and the control system. In turn, the control system comprises a PID controller (the OTS item), and the ROS which is simply the remainder of the control system.

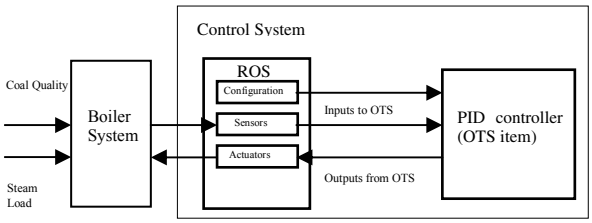

Figure 1. Boiler System and Control System (including the PID Controller)

The ROS consists of :

- the boiler sensors. These are "smart" sensors which monitor variables providing input to the PID controller: Drum Level, Steam Flow, Steam Pressure, Gas Concentrations and Coal Feeder Rate;
- actuators. These control a heating burner which can be ON/OFF, and adjust inlet/outlet valves in response to outputs from the PID controller: Feed Water Flow, Coal Feeder Rate and Air Flow;
- configuration settings. These are the "set-points" for the system: oxygen and bus pressure, which must be set up in advance by the operators.

Smart sensors and actuators interact with the PID controller through a standard protocol. Simulink output blocks can be introduced into the model in such a way that the variables of the MATLAB working space can be controlled as necessary. Working with the Simulink model we were able to perform repeatable experiments by manipulating any of the changeable variables and the connections between system components so as to produce and analyse a range of possible errors that would be reasonably typical for the simulated system.

2.3. The Simulink Model

The Simulink model (shown in Figure 2) actually represents the OTS item as three separate PID controllers that deal with the feed water flow, the coal feeder rate and the air flow. These controllers output three eponymous variables: Feed Water Flow (F_wf), Coal Feeder Rate (C_fr) and Air Flow (Air_f); these three variables, together with two external variables (Coal Quality and Steam Load) constitute the parameters which determine the behaviour of the boiler system. There are also several internal variables generated by the smart sensors; some of these, together with the configuration set-points, provide the inputs to the PID controllers. Table 1 lists all of the variables used in the model.

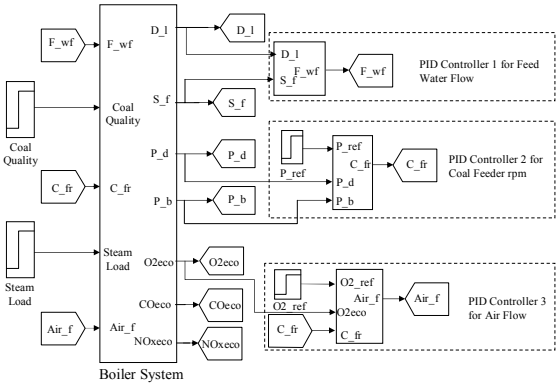

Figure 2. Simulink Model of the Boiler System with PID Controllers

Table 1. Variables used in the model

Variable	Representation	Variable	Representation
Coal Quality	Coal quality, ton per hour	D_l	Drum level
Steam Load	Steam Load, fraction of pure combustibles	S_f	Steam flow
F_wf	Feed water flow	P_d	Steam pressure / drum
C_fr	Coal feeder rate	P_b	Steam pressure / bus
Air_f	Air flow (controlled air)	O2eco	O2 concentration at economizer
P_ref	Bus pressure set-point	COeco	CO concentration at economizer
O2_ref	O2 set-point	NOxeco	NOx concentration at economizer

2.4. Variable Monitoring

Simulink scopes and other display blocks enable us to develop modelling components that observe the intermediate results while the simulation is running. In our experiments we can monitor and display a total of 15 variables, comprising all of the variables listed in Table 1 (except for the two set-points), plus three internal variables which represent two internal air flows and one internal steam flow. The simulation time for all of our experiments is set to 12000 steps. Some monitoring results are presented in Figure 3. In particular, this chart demonstrates the behaviour of the three PID outputs and two external inputs of the boiler system when at step 2000 the steam load is increased, and at step 5000 the coal quality changes: in both these scenarios the boiler system returns to steady operation reasonably soon.

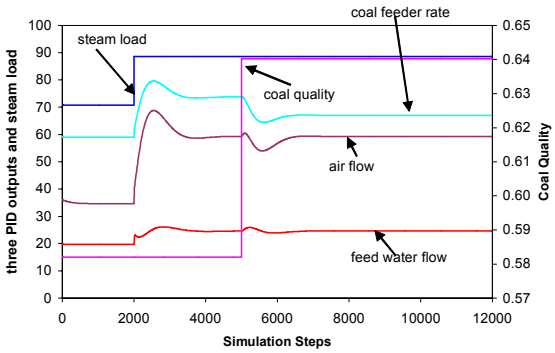

Figure 3. Normal Performance of the Boiler System with PID Controllers

2.5. Properties of the Boiler System and the PID Controllers

In this section we summarise the information which we collected to guide us in developing the protective wrappers. The basic boiler specification provides information on steam flow, bus pressure, output temperature and coal calorific value. As the OTS item (the PID controller(s)) is treated as a black box, any information about its properties must be deduced from the interface or from relevant sources where available. In an ideal world the system designer would have a complete and correct specification of the boiler system, the PID controller and the ROS. Unfortunately, we only had access to limited information about the boiler system and the ROS (which is typical for many practical situations). From an investigation of the boiler model and information acquired from all available sources, we have formulated the following description.

Information from the documentation available to us is as follows:

- Output temperature 540 deg C
- Coal calorific value 16-18 MJ/kg
- Steam load 50-125 ton/hour
- Coal quality is measured as a fraction of pure combustibles (where pure = 1; actual value about 0.55-0.7)
- Three controlled outputs (F_wf, C_fr, Air_f) are given as a percentage

Information obtained by analysing the interface and by investigating the simulated model:

- Set-point of bus pressure ranges from 0 to 20 (usual value about 9.4)
- Set-point of O2 concentration at economiser ranges from 0 to 0.1 (usual value about 0.03)
- Internal variables input to PID controllers:

- Drum level: output value between -1 and +1 (usual value close to 0)
- Steam Flow: 0 to 125
- Bus pressure: 0 to 20
- O2 concentration at economiser: 0 to 0.5

3. Requirements for a Protective Wrapper

In the previous section we presented an outline characterisation of the boiler system, as deduced from the model and other sources. In the following sections we consider the errors which could arise from integrating an OTS PID controller in the system, in order to derive requirements for a protective wrapper. To simplify our modelling perspective, we adopted the following assumptions in respect of an implementation of the system:

- The value of each variable can be checked instantaneously through microprocessors. In particular, we assume that the values of input and output variables of the PID controller are available instantaneously. This (highly) simplifying assumption enables us to illustrate the method for protective wrapper development without regard to issues relating to response times.
- The wrapper program can be inserted into the control system, either by a partial hardware implementation which intercepts the physical connections, or purely in software. There are, of course, significant issues involved in deciding on the implementation of a protective wrapper, but we do not address these in this paper.

In order to clarify the requirements for a protective wrapper, it is necessary to form a view of what the PID controller and the ROS should, and should not, do at the interface between them. This view can be formulated as a collection of *Acceptable Behaviour Constraints (ABCs)* [7] defined from the perspective of the systems integrator. Once defined, these ABCs can be thought of as contracts [11] which a system designer could use as the basis for defining a protective wrapper, which can then embody relatively conventional mechanisms for error detection, containment and recovery [2].

4. Safe Boiler Operation

Many aspects of the operation of the boiler and control system, such as the flow of gases, fuelling, pressures and levels, could lead to a failure of some type. However, some features are much more significant in terms of safety; in a steam boiler, the drum level is a key parameter. This parameter represents the quantity of water in the boiler more accurately than a direct measurement of the water level, due to changes in mass caused by differences in temperature. By monitoring and controlling the drum level we can maximize steam quality and maintain the proper water quantity to prevent damage to the boiler. Too low a level could expose the water tubes to heat stress and damage; too high a level could allow water to go over the steam header, exposing the steam turbines to corrosion and damage [12, 13]. Steam pressures on the drum and the bus are the two other key variables, since they indicate the balance between the supply and demand for steam. The consequences of excessively high steam pressures are obvious and explosive. Thus, any deviation from normal values of steam pressure and drum level must be corrected immediately, whereas abnormal values of the other variables can be tolerated for a time period (which must be defined by the system designer).

We classify the detectable variables into two loops, the control loop and the safety loop, which operate as follows:

- an alarm from the safety loop will shut the system down;
- alarms from the control loop are processed on-line and either resolved or, if this is unsuccessful, the safety loop is triggered.

The wrapper envelopes the PID controllers as shown in Figure 4: it monitors the values of variables which go into and come out from the PID controllers. Three variables – the drum level and the steam pressures on the drum and bus – are classed as belonging to the safety loop, and all other variables (PID outputs to the boiler (via the ROS), set-points and other input variables) belong to the control loop.

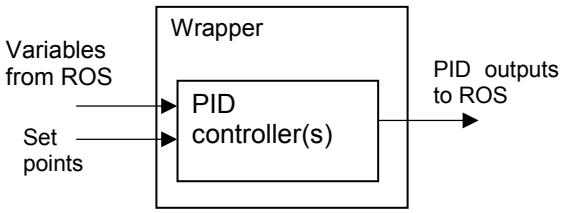

Figure 4. Variable categories around the PID controller(s)

5. Error Recovery

Error recovery transforms a system state that contains errors into an error free state. The transformation typically takes the form of either *backward* or *forward* error recovery [2]. Backward error recovery returns the system to a previous (assumed to be correct) state; typically, the techniques used are application-independent and often operate transparently for the application (e.g. atomic

transactions and checkpoints). Forward error recovery aims to move the system into a correct state using knowledge about the current erroneous state; this recovery is application-specific by its nature. The most general framework for achieving forward recovery is *exception handling* [3]. It is not difficult to see that backward error recovery is not generally applicable in dealing with OTS items [4].

Protective wrappers offer a structured approach for incorporating fault tolerance measures in systems with OTS items. In previous work [1] we demonstrated how error detection can be developed in a wrapper by cyclically checking for each type of possible error identified during the analysis phase. We characterise these errors into three distinguishable types: (a) signal not available, (b) signal violating specifications, and (c) unacceptable signal oscillations. When the wrapper detects an erroneous situation it immediately initiates recovery action by classifying the error and invoking a corresponding exception handler. Three possible recovery actions are suggested here.

- Handler1:Reset the signal to a standard normal value and send an alert to the operators.
- Handler2:
  - Wait Δt, if error resolved, take no action;
  - Otherwise, send an alarm to the operators and wait ΔT, if error resolved, take no action;
  - Otherwise, invoke handler 3.
- Handler3: Shutdown the system and send an alarm to the operators.

In Handler2, the delay times Δt and ΔT would be determined by the wrapper designer after consulting the system specification. In the Simulink model we took Δt as 500 steps and ΔT as 1500 steps, representing reasonable values for a genuine industrial application.

Analysis of the error types for different signals then enabled us to define a recovery strategy, which is discussed in the following subsection, and then illustrated in Figure 5.

5.1. Recovery Strategy

In the case of an erroneous situation detected for a set-point value, whether it is missing, out of specification or oscillating is of secondary interest. The wrapper is aware of the appropriate range of set-point values, and Handler1 provides appropriate recovery by forcing the input to a suitable value, and alerting the operators (since the mistake could be theirs, or an internal corruption).

The situation is rather different when a PID output value is detected as being erroneous, but the same response can be made; either by adopting standard operating output values, or by storing a recent history and using a smoothed average, the wrapper can apply Handler1 to impose a valid output signal which should result in stable, though suboptimal, performance from the boiler system. It would be possible to differentiate between the three error types in terms of determining the signal value to be imposed, but we have not exploited this in our simple demonstrator.

Now consider inputs to the PID controllers which are not set-points, and are not in the safety loop. When one of these is detected as erroneous there is little point in feeding a fixed value to the PID, since this will not reflect the actual conditions monitored by the ROS. However, given that there is no immediate safety concern, the optimistic strategy of "wait and see" may be successful; indeed for a short interval it may not be appropriate even to alert the operators, since the phenomenon may be completely transient in nature (it would, of course, still be logged for an off-line report). If the problem persists an alarm report to the operators may enable them to cure the problem, but if not an eventual shut-down is inevitable. So the wrapper can deploy Handler2; again, although differentiation of the response in line with error type is possible, we have not exploited this option.

Lastly, consider the variables in the safety loop: drum level, drum and bus steam pressures. Detecting an erroneous condition on one (or more) of these variables implies a risk to safety, and the natural response is to shut the boiler system down, despite the economic consequences. Handler3 provides this response, but to illustrate error category differentiation in our model we invoke Handler2 in the particular case of an oscillating value. The justification is that although a missing signal value or an out of specification value indicates a clear and present risk of accident, an oscillating value *within specification* does not necessarily pose the same immediate threat – the oscillations may die out or an operator response could stabilise the situation. If the oscillations persist then Handler3 will still be brought into play. Of course, in a real boiler system this strategy would only be acceptable if justified by a safety case.

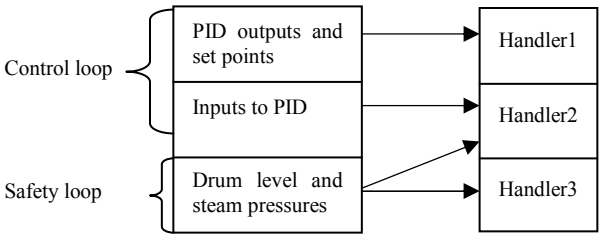

Figure 5. Recovery action implemented by exception handlers

5.2. Exception Handling

When any of the errors above is detected an exception is raised. Error diagnosis is performed to select the appropriate handler, depending on which variable caused the exception to be raised. The diagnosis is straightforward for variables in the control loop: in the safety loop, we differentiate between error categories.

Errors detected by the protective wrapper can be caused by malfunctioning of the OTS item, by faults arising in the ROS, or by misinterpretation between the OTS item and the ROS (Figure 1). In the PID case study considered here, the exception handlers implemented in the protective wrapper always act at the level of the integrated system, which constitutes the exception handling context [4]; they can send an alert or alarm signal to the operator, replace an erroneous value with an alternative "normal" value, await a natural rectification, or (when safety requires it)) shut the system down.

Clear separation of the normal and abnormal system behaviour by employing exception handling in the wrapper facilitates the integration of the OTS PID into the composite system [4].

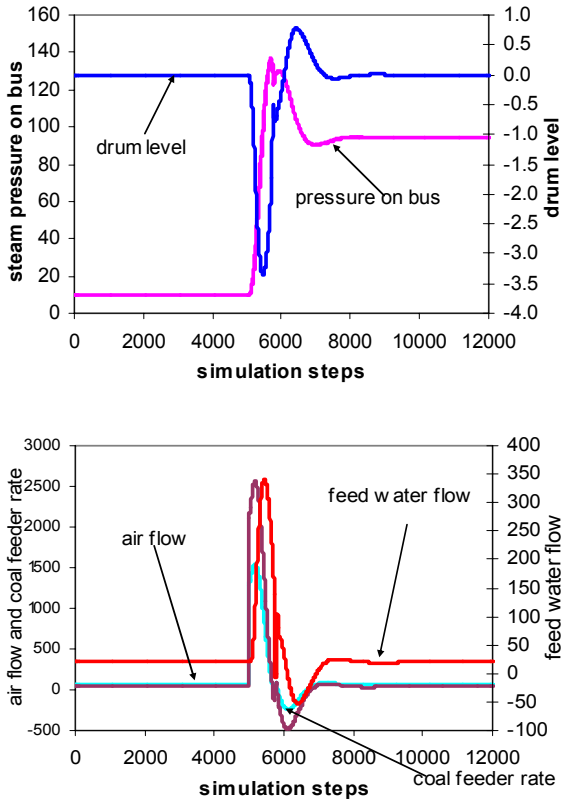

Figure 6. Erroneous pressure set-point – boiler system with no wrapper

6. Implementing the Recovery Actions

This section illustrates the operation of recovery when an error occurs, with the wrapper implemented in the MATLAB model. The example presented below applies error recovery by invoking Handler1. The error was introduced by artificially simulating an incorrectly valued pressure set-point. At step 5000 the operator "accidentally" recalibrates the pressure set-point to be 94 instead of the normal value of 9.4.

Figure 6 shows the boiler system behaviour with no wrapper protection; note that the bus steam pressure is superimposed over drum steam pressure, so only one pressure variable is actually displayed.

The first chart of Figure 6 shows that the faulty pressure set-point results in a huge drop in the drum level followed by a peak at too high a value before returning to a level close to normal. Similarly, the three PID outputs shown in the second chart jump up beyond their specified levels after step 5000 but return to normal performance by step 7000. Very much more serious is the steam pressure, which the first chart shows rising and remaining at an excessive level.

Figure 7 shows the behaviour of the boiler system with wrapper protection active.

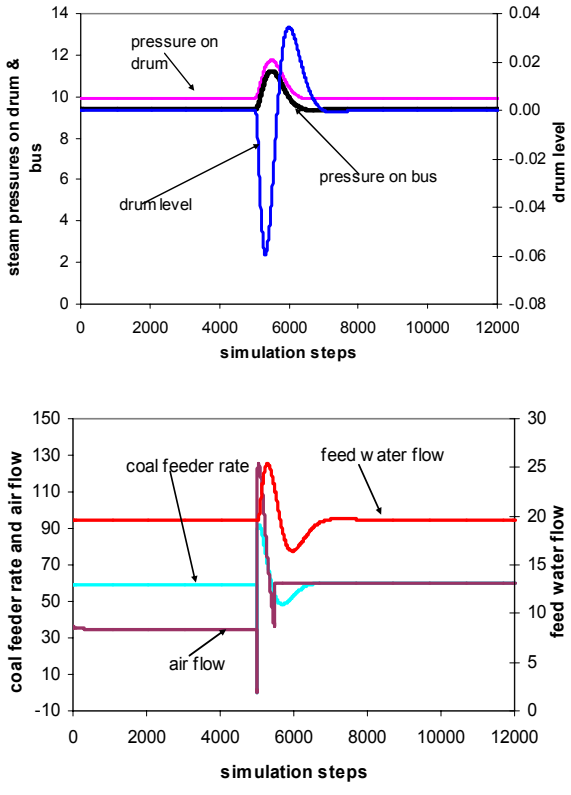

Figure 7. Erroneous pressure set-point – error recovery by Handler1

The wrapper detects the incorrect value of the set-point and invokes Handler1. Consequently the set-point is very quickly corrected, and the impact on system behaviour and on the other variables is greatly reduced. From the two charts in Figure 7 we see that although there are some alterations to the variables when the error occurs, the changes are actually quite minor, and lie within the range of the specifications for the system.

7. Discussion and Conclusion

7.1. Generic Error Recovery Strategy

Since an OTS item that has been integrated into a system is treated as a black box, only the inputs to and outputs from the OTS are available to be monitored for error detection. The inputs in a control system can be partitioned into two groups, where the first group consists of configuration variables (often under operator control as set-points), while the second group provides dynamic status information from the system under control. A wrapper providing protection in the system can monitor these two groups of inputs, and the outputs, and can attempt to distinguish different categories of erroneous behaviour. As three simple base categories we proposed: missing value, out of range value, and oscillating value.

In response to a detected error situation, over these groups of variables and categories of error, the protective wrapper can apply an exception handling framework to attempt to recover from the error. Simplistic generic recovery strategies that we have considered are: do nothing, alert the human operators, change variables to normal values, and stop the system. In our boiler system example we decided that erroneous outputs from the OTS could be over-ridden by the wrapper; in effect, the wrapper will take over the role of the OTS in erroneous situations, but can only provide a standard set of normative outputs. In the same way, the wrapper can over-ride configuration variable inputs to the OTS, particularly when these are input set-points. Our Handler1 is used to over-ride erroneous values.

We need to analyse in more detail the other input variables delivered to the OTS by the ROS, since forcing a change here could only have a very indirect influence on the variable itself, via the OTS and its outputs back to the controlled system. One important aspect is with respect to safety and we used this to apply a hierarchy of recovery. For variables with a direct impact on safety we "recover" by shutting the system down (Handler3); for other variables, not in the safety loop, we first wait to see if the error is transient, then to see if human intervention will achieve recovery, and if that too is unsuccessful then the system must be shut down in any case (Handler2).

Wrapper design for any system would proceed by analysis of the state space of variables, the errors that could be detected, the damage assessment that could be conducted, and the recovery strategies that could be devised, bearing in mind the implications on system operations from both a mission (economic) and a safety perspective.

7.2. Scope of the Wrapper

The essential characteristic of a protective wrapper is that *all* inputs to and outputs from the wrapped component are accessible to and modifiable by the wrapper. It might be argued that no other system variables should be accessible to the wrapper, since otherwise the intuitive image of "wrapping the component" would be distorted. We feel this is unnecessarily restrictive. If the system designer believes that improved performance of the wrapper can be achieved by utilising information from elsewhere in the system this should not be prohibited by an artificial limitation. When the wrapped component is an OTS item it may be very unlikely that the wrapper could make any effective use of internal state information within the OTS component, but valuable insight may perhaps be gleaned from variables in the ROS that are not visible to the component. An example of this is present in our case study. The drum steam pressure is not actually utilised by the PID controller (although it is made available), so it is debateable whether or not it constitutes an input. We gave the wrapper access to this variable without hesitation, since we suspect that any practising engineer would do the same.

7.3. Complexity of the Wrapper

A wrapper inserted into a system as a protective component performs an important role in improving the reliability of the integrated system. In discharging this role it is clearly essential that the wrapper does not itself contribute to an increase in failing behaviour from the overall system. Ideally, the wrapper should introduce no faulty behaviour itself, and should capture and rectify all faulty behaviour it encounters. Perhaps the best general guidance here is that the most reliable designs will usually be those that are simplest. The OTS PID controllers may, of necessity, involve highly complex algorithms to achieve the optimised boiler performance that is their goal. But in designing an effective wrapper we are likely to find that a simple and effective recovery strategy will outperform something overly sophisticated. First, a simple design is more likely to be implemented correctly, and second,

a more complex strategy may have unforeseen interactions with the control environment and these could easily detract from effectiveness.

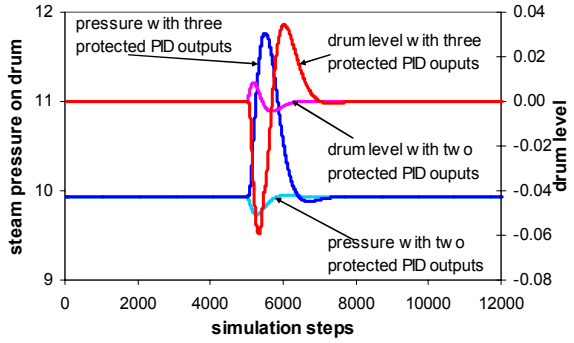

Figure 8. Comparison of effects on performance of boiler system with two kinds of wrapper designs

Fig. 8 shows a comparison of the impact on two of the variables in the safety loop for the boiler system, with two different wrapper designs responding to the situation described in section 6 (the pressure set-point changed from 9.4 to 94 at step 5000). One wrapper applies recovery by using Handler1 to reset all three PID outputs to standard values, while the other wrapper only resets the feed water flow and air flow, leaving the coal feeder rate at its computed value. We can see from Fig. 8 that the more simplistic strategy of only changing two of the variables leads to a swifter and more stable recovery.

This paper has summarised our recent work in the development of protective wrappers as a structured approach to providing error detection and recovery in systems utilising OTS items. This approach embodies error classification and corresponding recovery strategies implemented within an exception handling framework, building on the structure and error detection issues considered in earlier papers [1,7].

8. Acknowledgements

This work was supported by the EPSRC/UK project DOTS: *Diversity with Off-The-Shelf Components*. (http://www.csr.ncl.ac.uk/dots), and has benefited significantly from interaction with colleagues at City University within the project. A. Romanovsky is partially supported by FP6 IST RODIN Project (IST-511599).

An earlier version of the paper was presented at ECOOP 2003.

9. References

[1] T. Anderson, M. Feng, S. Riddle, A. Romanovsky. Protective Wrapper Development: A Case Study. *2nd International Conference on COTS-Based Software Systems (ICCBSS 2003)*. Ottawa, Canada, February, 2003. pp. 1-14

[2] P. A. Lee, T. Anderson, *Fault Tolerance: Principles and Practice*, Wien - New York, Springer-Verlag, 1991.

[3] F. Cristian. Exception Handling and Tolerance of Software Faults. In M. R. Lyu (Ed). *Software Fault Tolerance*. John Wiley and Sons, 1995, pp. 81-108

[4] A. Romanovsky. Exception Handling in Component-Based System Development. *25th Int. Computer Software and Application Conference (COMPSAC 2001)*, Chicago, IL, October, 2001. pp. 580-586.

[5] J.-C. Laprie. "Dependable Computing: Concepts, Limits, Challenges". *Special Issue of the 25th International Symposium On Fault-Tolerant Computing*. IEEE Computer Society Press. Pasadena, CA. June 1995. pp. 42-54

[6] J. Voas. Certifying Off-The-Shelf Software Components. *IEEE Computer*, 31(6), 1998, pp. 53-59.

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

[8] V. Havlena, Development of ACC Controller with MATLAB/SIMULINK. *MATLAB '99*. Praha: VSCHT - Ustav fyziky a merici techniky, 1999, pp. 52-59.

[9] J-R. Abrial, E. Börger, H. Langmaack. *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*. LNCS 1165, Springer Verlag, October 1996.

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

[11] B. Meyer. Programming by Contract. In D. Mandrioli, B. Meyer (eds.), *Advances in Object-Oriented Software Engineering*. Prentice Hall, 1992.

[12] Siemens, Boiler Control Overview, http://www.procidia.com

[13] Minnesota Department of Labor and Industry, Your guide to safer boiler operation, http://www.doli.state.mn. us/pdf/guide2saferboiler.pdf.