Protective Wrapper Development: A Case Study

Tom Anderson, Mei Feng, Steve Riddle, Alexander Romanovsky

School of Computing Science
University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, UK

Abstract. We have recently proposed a general approach to engineering protective wrappers as a means of detecting errors or unwanted behaviour in systems employing an OTS (Off-The-Shelf) item, and launching appropriate recovery actions. This paper presents results of a case study in protective wrapper development, using 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 constraints, and oscillations.

1 Introduction

There are many economical reasons why integration of Off-The-Shelf (OTS) components into systems with high dependability requirements (including the safety-critical ones) is becoming a viable option for system developers (see, for example, [1]). The main obstacle is that employing such components can undermine overall system dependability unless special care is taken. Considerable evidence supports the judgement that complex systems built using OTS components could have a higher risk of failure. This is due to a number of reasons:

- OTS components are typically aimed at a mass-market and are often of a lower quality than bespoke components;
- OTS components are seldom intended for the specific setting and environment in which they are employed - consequently a system in which an OTS component is integrated may misuse or misinterpret it;
- information about the COTS item which the system integrator has at his/her disposal is often incomplete, ambiguous or, even, erroneous;

We take a pragmatic view in developing our approach by accepting that, in spite of all efforts to improve the quality of OTS components and of the system in which they are to be integrated, their use will be a source of failure. The solution we are advocating is a defensive strategy of employing specialised fault tolerance techniques during system integration.

1.1 Protective Wrappers

Fault tolerance [2] as a general means for improving overall system dependability has been an area of very active research and development over the last 30 years. Many fault tolerance techniques proceed by employing redundant software in some form; for example, recovery blocks, N-version programming, exception handling [3, 4]. The main phases of providing fault tolerance are error detection, error diagnosis and error recovery [5]. At the first phase an erroneous state is identified as such; after that error diagnosis is used to examine the damaged area to allow it to be replaced by an error-free state during error recovery. Employing proper system structuring techniques (e.g. classes, processes, layers, modules) is vital for achieving fault tolerance because it allows erroneous information to be contained within structural units. Our work aims to apply these general fault tolerance techniques in the context of integrating OTS items into complex systems.

Component wrapping is an established technique used to intercept data and control flow between a component and its environment [6]. Typical applications include adding data cacheing and buffering, simplifying the component interface, and providing transparent component replication.

In a component-based system development the OTS items are natural units of system structuring. Unfortunately, as we have explained above, they usually do not provide enough assurance of correct behaviour. In previous work [7] we proposed the development of *protective wrappers* as part of system integration activities. A protective wrapper is a piece of redundant, bespoke software intercepting all

information going to and from the OTS item. Such a wrapper may detect errors or suspicious activities, and initiate appropriate recovery when possible.

A wrapper is a piece of software and, clearly, may contain software defects itself. Deploying a wrapper to perform protection functions obliges us to take considerable care over issues such as relative complexity and common-mode failures between the wrapper and the wrapped component. For this reason, wrappers 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. Information required for wrapper development is obtained by analysing several sources of information [7], including:

- Specification of the OTS item behaviour, as provided by both the item designers and the integrated system designers. The latter characterises the behaviour the system designers require of the OTS item in order to integrate it with the system.
- "Erroneous" behaviour of the OTS item, for example a known failure to react to stimuli as specified by the item designers (these may be known, for example, from testing or from previous experience in using the OTS item), or behaviour which the system designer especially wants to protect against.
- Specification of the correct behaviour of the ROS with respect to the OTS item.

1.2 Case Study

In this paper we report the results of applying the proposed approach to developing a protective wrapper for an Off-The-Shelf PID (Proportional, Integral and Derivative) controller. It is our intention to demonstrate how the approach could be applied in practice. The results of this study will be used to aid the development of a generic approach for wrapping OTS items.

Rather than conduct an experiment with protective wrapping in the real world environment, with all the associated cost and potential damage to equipment and life, we have employed software models of the PID controller and of the steam boiler system in which it is to be integrated. Employing software models of the controller and the boiler system is an active area of R&D carried out by many leading control product companies (including Honeywell [8]). We believe that the decision to use a third-party model adds credibility to our results.

A related case study in steam boiler control specification was examined at the Dagstuhl seminar on Methods for Semantics and Specification [9]; the seminar was run as a competition to show the strengths and weaknesses of particular formal methods. Rather than adapt this specification to our needs, we chose to use a model developed within a research project conducted by Honeywell for their control products [8]. This model simulates a real controller and the controlled steam boiler system, enabling us to investigate the effect of wrapping with a more representative model than the idealised specification employed at Dagstuhl. In the course of our work we extended the Honeywell 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 outlines the categories of errors to be detected and tolerated at the level of the wrapper. The next section outlines design and implementation of the wrapper to detect these categories of error. Section 5 concludes the paper by summarising the results, discussing the limitations of our approach, and indicating avenues for future work.

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 or sampled time, as well as a hybrid of the two. 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 are modelling is shown in Fig. 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.

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 devices 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.

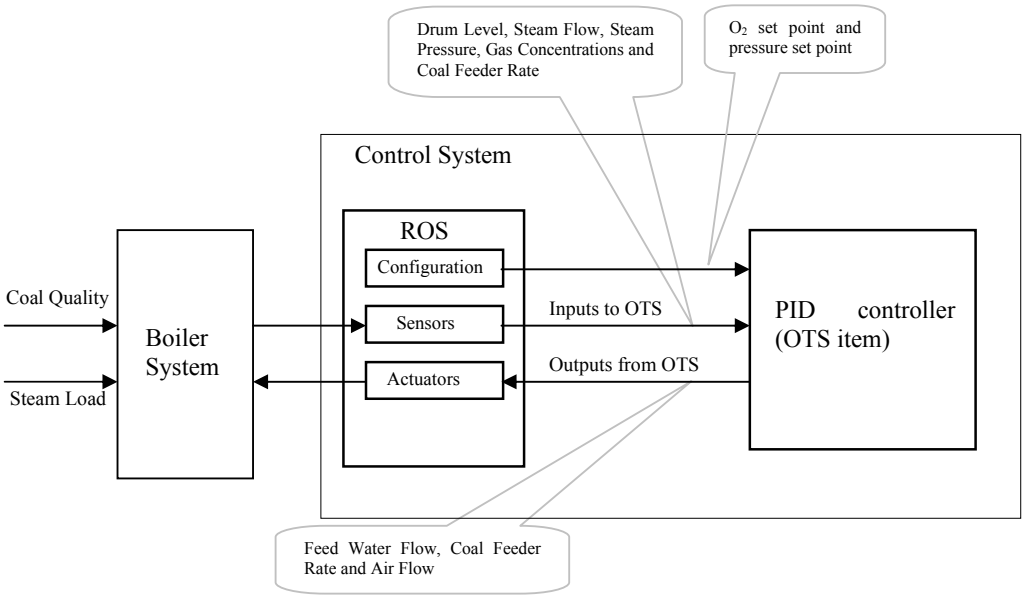

Fig. 1. Boiler System and Control System (including the PID Controller)

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 system we are simulating.

2.3 The Simulink Model

The Simulink model (shown in Fig. 2) actually represents the OTS item as three separate PID controllers that handle 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.

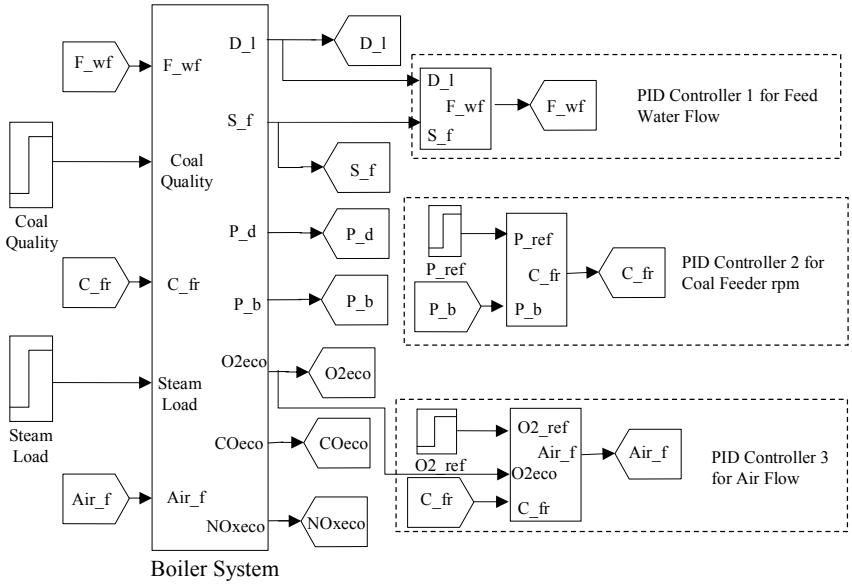

Fig. 2. Simulink Model of the Boiler System with PID Controllers

2.4 Variable Monitoring

Simulink scopes and other display blocks enable us to develop modelling components that monitor the intermediate results while the simulation is running. In our experiments we can monitor and display a total of 15 variables, comprising all 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 Fig. 3. This particular 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.

Table 1. Variables used in the model

Variable	Representation
Coal Quality	Coal quality, ton per hour
Steam Load	Steam Load, fraction of pure combustibles
F_wf	Feed water flow
C_fr	Coal feeder rate
Air_f	Air flow (controlled air)
P_ref	Bus pressure set-point
O2_ref	O2 set-point

Variable	Representation
D_l	Drum level
S_f	Steam flow
P_d	Steam pressure /drum
P_b	Steam pressure / bus
O2eco	O2 concentration at economizer
COeco	CO concentration at economizer
NOxeco	NOx concentration at economizer

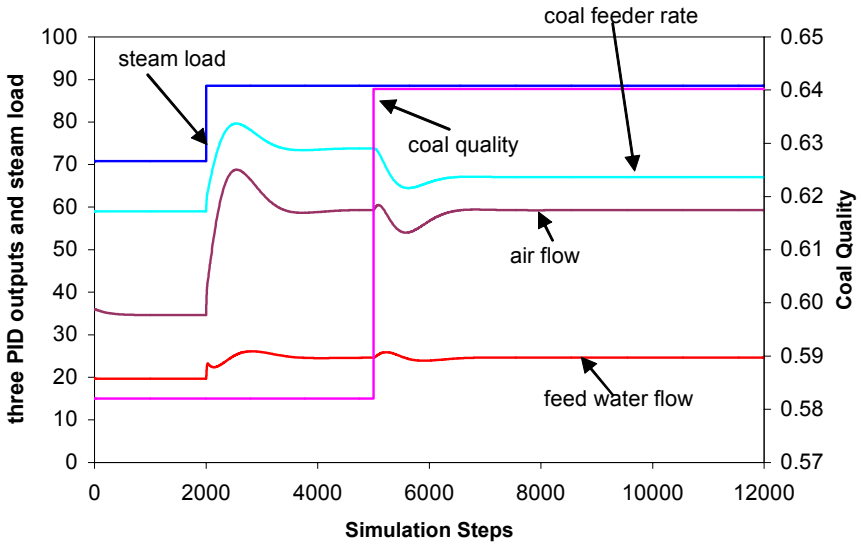

Fig. 3. Normal Performance of the Boiler System with PID Controllers

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 will 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:

- 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 each 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 (actual value about 9.4)
- Set-point of O2 concentration at economiser ranges from 0 to 0.1 (actual value about 0.03)
- Internal variables input to PID controllers:
  - Drum level: output value between -1 and +1 (actual 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 specification of the modelled boiler system, as deduced from the model and other sources. In this section we consider the errors which could arise from

integrating an OTS PID controller in the system, in order to derive the requirements for a protective wrapper. We make the following assumptions:

- The value of each variable can be detected instantaneously through microprocessors. In particular, we assume that the values of input and output variables of the PID controller are detected 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. We are not concerned at this stage with the details of this implementation.

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 can, and cannot, do at their shared interface. 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 would employ conventional mechanisms for error detection, containment and recovery [2].

3.1 Types of cues, and examples

For our case study, Table 2 provides a list of error symptoms (cues) and associated actions, following a structural analysis of the possible errors detectable at the interface between the ROS and the PID controller. Since the OTS PID controller is a black box item we can only reason about errors concerning the inputs and outputs to the PID from the ROS. This gives four groups for the cues, as shown in first column of the table.

The second column classifies the type of error we are concerned with, and the third column gives an example of each type of error. The recovery action given in the fourth column is a suggested action which a protective wrapper could be designed to launch; we do not claim that these illustrative actions are the most appropriate in each case. The cue highlighted in bold is selected as an example for further discussion.

Table 2. Cues for protective wrapper

	Types of cues	Examples	Actions
Errors in PID inputs w.r.t. ROS output constraints	Illegal output from ROS (according to ROS specification)	An output from ROS is disconnected from PID	Shutdown
	Output from ROS is detectably erroneous	ROS sampling rate suddenly exceeds the normal rate	Alarm
	Output from ROS is illegal w.r.t. the system designer's specification of system operation	A ROS output is outside the envelope of values anticipated by the system designer	Alarm
Errors in PID inputs w.r.t. PID constraints	Input to PID is illegal (according to PID specification)	Set-point values are mis-configured and violate the PID specification	Alarm
	Input to PID that is suspect	The measured derivative of a PID input exceeds the maximum level for which it has been tested	Alarm
	Input to PID which is known to be untrustworthy	**A PID input (or its derivative) is close to the boundary specified for the PID, at a level which is known to create problems**	**Alarm**
Errors in PID outputs w.r.t. PID constraints	Illegal output from PID (according to PID specification)	An output from PID is disconnected from ROS	Shutdown
Errors in PID outputs w.r.t. ROS input constraints	Illegal input to ROS (according to ROS specification)	The PID controller changes its rate of processing and sends messages to ROS too frequently	Alarm
	Input to ROS is illegal w.r.t. the system designer's specification of system operation	A PID output is outside the envelope of values anticipated by the system designer	Alarm

The example, “Input to PID which is known to be untrustworthy”, is illustrated in Fig. 4. Steam load is kept constant at 70.8 ton per hour during the operation. After 5000 simulation steps, the coal quality is increased from 0.5 to an artificial value just under 3, and as a consequence some inputs to the OTS would approach untrustworthy values which are close to the boundary of the PID controller's practical specifications. The curves shown on Fig. 4 converge to a steady state, but if the overshoots of the initial oscillations were of greater magnitude than the specification of the boiler system permits, or the oscillations kept going longer than the boiler system can support, the situation would be regarded as critical.

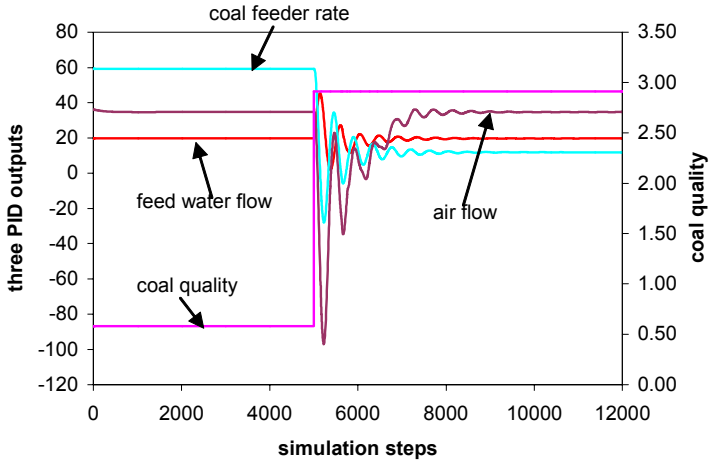

Fig. 4. Input to the PID for which it is known to be untrustworthy: a PID input exceeds the boundary specified for the PID controller

A more extreme version of this example is shown in Fig. 5. Here, the coal quality (not shown on the Figure) is increased unrealistically to almost 4.5 at step 1000, with steam load constant as before. The curves are no longer convergent, leading to a dangerously unstable situation.

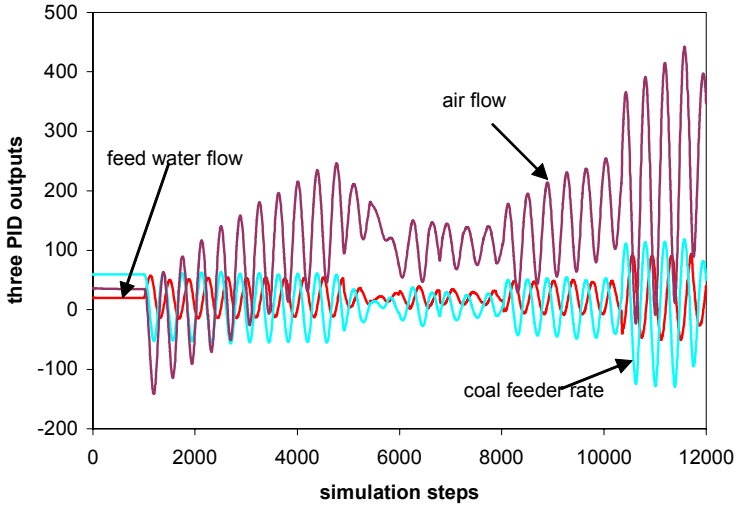

Fig. 5. Input to the PID for which it is known to be untrustworthy: a more extreme example leading to dangerous oscillations

3.2 Summary Analysis

Errors may occur anywhere in the boiler system or the PID. However, a protective wrapper for the OTS PID controller can only check for error conditions as cues at the PID/ROS interface. In the previous subsection we have characterized these cues in terms of their sources, but the wrapper can only detect them by their behavioural attributes. We have therefore placed the cues into three distinguishable categories:

1. Unavailability of inputs/outputs to/from the PID controller
   Either the PID controller crashes, resulting in no output from the PID controller to the ROS (and beyond), or the boiler system or ROS (or some connections between these and the PID) is disrupted, so that inputs to the PID controller are unavailable.
2. Violation of specifications of monitored variables
   Set-points to the PID controller, or any monitored variable, violate their specification.
3. Oscillations in monitored variables
   Monitored variables, and their derivatives, take on excessive and rapidly changing values.

This categorization of error types informs the design of a protective wrapper, which is addressed in the next section.

4 Design and Implementation of a Wrapper

4.1 Design of Wrapper

In this section we address the design of a wrapper to be implemented using MATLAB functions.

The main function of the wrapper is to cyclically check for each type of error identified in the previous section (3.2). It uses two sub-functions:

- *no_signal_alarm*, to check for absence of a signal after a given period of time
- *check_oscillate*, to check whether oscillating variables revert to a stable state before a maximum number of oscillations.

The "given period of time" and "maximum number of oscillations" referred to here should in general be determined after consulting the specification of the system. Since this information was not available, we have set the number of steps for detecting absence of signal to 1000 (the simulation itself runs for 12000 steps), as this serves to illustrate the protective wrapper design.

The wrapper implements a number of ABCs based on the properties (discussed in Section 2.5) of input and output variables F_wf, C_fr, Air_f, D_l, S_f, P_b, P_ref, O2eko and O2_ref, with an addition of two more ABCs stating that the signals cannot be lost and that the variables cannot oscillate beyond the maximum number of oscillations. (we view these as dangerous conditions which have to be prevented). If the signals do not violate any of the ABCs they go through the wrapper unchanged otherwise an alarm is raised, or the system is shut down.

Thus, in this case we are adopting an elementary monitoring and alerting strategy, as an obvious starting point on the way to the development of a more general engineering methodology for wrapper design. Our approach, as outlined in Section 3 above, is to derive wrapper requirements from a consideration of possible interface errors classified by their symptoms (cues); this leads to the formulation of a set of ABCs. We touch on more general issues in [7] but clearly there is considerable scope for further research.

4.2 Wrapper Implementation

Although there are many pragmatic considerations to be addressed in order to achieve an effective implementation of wrapper technology, the choice of the appropriate mechanism will be largely determined by the environment. A more detailed discussion is presented in [7].

4.3 Example

We now illustrate the operation of the wrapper for the error discussed in Section 3.1 (Fig. 4), due to oscillation of signals. Fig. 6 demonstrates the case where the boiler system can only withstand three oscillations in signals. This is the same situation as was presented in Fig. 4.

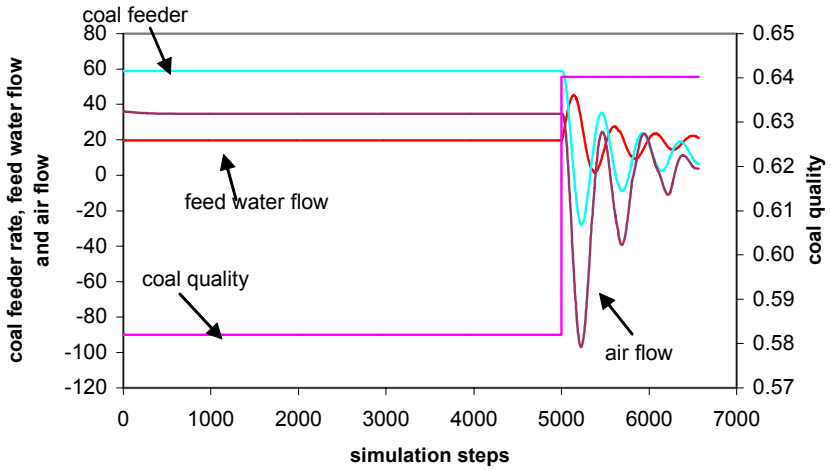

Fig. 6. Demonstration for wrapper working on signal oscillations

A maximum peak plus one minimum peak is counted as one oscillation. After the forth oscillation was generated, the wrapper shut down the boiler system.

5 Conclusions

This paper has investigated the types of error which can occur in the modelled boiler system (Section 3.2) and then addressed the design of a wrapper to detect these errors.

We would like to emphasise that this work focuses on error detection and error recovery preventing systems from failure rather than on fault diagnosis and fault treatment [2]. The protective wrappers detect erroneous information going to and coming from the PID controller and perform recovery actions (raising alarms or shutting the system down). In real systems employing PID controllers such errors can be clearly caused by various reasons (faults), such as design faults in the PID or in the system employing it, mismatches between these two entities (in which case it may be impossible to identify the fault location), failures of the underlying hardware, etc.

The errors were categorised as:

1. Unavailability of signals to/from PID controllers, either through the controllers themselves not working or a fault in the boiler system
2. Violation of limitations specified for variable(s), due to a fault in the boiler system or mis-configured set-points
3. Oscillations in the values of some variable(s)

A simple wrapper to check for these errors and take appropriate action has been designed and demonstrated (Section 4.2) for a representative example.

Use of Simulink to program the wrapper has some disadvantages. Simulink is not as expressive as conventional object-oriented programming languages, such as C^{++}, as it is specifically designed to allow mathematical modelling and analysis without the full range of general purpose programming features. Models of systems are assembled from a limited library of "blocks". Users can define their own blocks using "S-functions", but these are constrained by means of a template which users must adhere to. In spite of this limitation, Simulink is still a practical and intuitive platform to demonstrate and investigate industrial process systems, such as the steam boiler system considered in this paper.

Use of modelling and simulation is commonplace in the design of protection systems, and for this reason we have not felt that the approach taken here is unrealistic. However, a further piece of work will be to consider a more "real-world" scenario. This would in turn require an extended wrapping strategy which takes account of a wider range of actions to be taken if an error has been detected.

In addition, future work will develop more generic monitoring activities using a combination of error detection and fault injection techniques, to measure the effect a wrapper has on the overall system dependability. Other avenues could include investigating the potential use of wrappers to capture behaviour that has not been explicitly specified.

Acknowledgements

This work is supported by the UK EPSRC project DOTS: *Diversity with Off-The-Shelf Components* (www.csr.ncl.ac.uk/dots). A. Romanovsky is supported by the European IST project DSoS: *Dependable Systems of Systems*.

We are grateful to Vladimir Havlena for sharing with us some of the results of his research, and to Prof. Lorenzo Strigini for comments on an early draft of this paper.

References

1. IEEE Computer, "Special issue on COTS", *IEEE Computer*, 31(6), 1998.
2. P. A. Lee, T. Anderson, *Fault Tolerance: Principles and Practice*, Wien - New York, Springer-Verlag, 1991.
3. M. R. Lyu. *Software Fault Tolerance*. John Wiley and Sons, 1995.
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, 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, p. 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.