Protective Wrapping of Off-the-Shelf Components

Meine van der Meulen[1], Steve Riddle[2], Lorenzo Strigini[1], and Nigel Jefferson[2]

[1] Centre for Software Reliability, City University, London, U.K.
{mjpm, strigini}@csr.city.ac.uk
[2] School of Computing Science, University of Newcastle upon Tyne, U.K.
{steve.riddle, n.p.jefferson}@ncl.ac.uk

Abstract. System designers using off-the-shelf components (OTSCs), whose internals they cannot change, often use add-on "wrappers" to adapt the OTSCs' behaviour as required. In most cases, wrappers are used to change "functional" properties of the components they wrap. In this paper we discuss instead *protective wrapping*, the use of wrappers to improve the dependability – i.e., "non-functional" properties like availability, reliability, security, and/or safety – of a component and thus of a system. Wrappers can improve dependability by adding fault tolerance, e.g. graceful degradation, or error recovery mechanisms. We discuss the rational specification of such protective wrappers in view of system dependability requirements, and highlight some of the design trade-offs and uncertainties that affect system design with OTSCs and wrappers, and that differentiate it from other forms of fault-tolerant design.

1 Introduction

As building "component-based" software systems becomes more common, it becomes more often necessary to combine existing off-the-shelf (*OTS* for brevity) components – hardware as well as software – that were not necessarily designed to work together. *Wrapping* is a popular, often cost-effective technique for integrating pre-existing components into a system. When designing a new system, ad hoc "wrappers" are developed, i.e. new, small components that will be interposed between the others, reading and sometimes altering the contents of the communications they exchange. Wrapping has the advantage of not requiring detailed knowledge of the internal structure of the components being wrapped.

In most cases, wrappers are used to adapt the functionality of a component to the requirements set for it by the system's design: they often perform simple functions like translation between the argument formats used by two communicating components. In this paper we look instead at the use of wrappers for improving dependability. We call such wrappers *protective* wrappers. Protective

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council through project DOTS (Diversity with Off-The-Shelf Components), grants GR/N23912/01 and GR/N24056/01.

X. Franch and D. Port (Eds.): ICCBSS 2005, LNCS 3412, pp. 168–177, 2005.

wrapping is a way of structuring the provision of standard fault tolerance capabilities, like error detection, confinement and recovery, plus the less common capability of *preventing* component failures, in a component-based design where dependability is a concern. We wish to clarify how these wrappers can be rationally specified, the trade-offs facing system designers (simply "designers" for the rest of the paper), and the peculiarities of this form of fault-tolerant design, compared to the general case.

When designing a system with off-the-shelf components (OTSCs), it is often the case that an OTSC's functionality, and even more often its dependability, is insufficiently documented. Both these deficiencies are threats to system dependability: wrong assumptions about how an OTSC is intended to behave lead to system design faults; optimistic assumptions about an OTSC's probability of behaving as intended may lead to overestimating the dependability levels achieved by the chosen system design. Wrapping can help a designer to compensate for this lack of information.

Wrapping for dependability has been addressed by other authors. Wrappers are used to transform or filter unwanted communications that may cause failures. Fault injection may be used to identify such failure-causing values [7, 3, 5]. Wrappers are proposed to protect OTS applications that do not deal properly with kernel-raised exceptions, by transforming these into other exceptions or error return codes [7]; or to protect OTS kernels against inappropriate requests ([3]; here, an extended notion of wrappers is proposed that can access the kernel's internal data). In [5], the goal is automatic protection of library components against failure-causing parameter values, submitted by accident or malice. In [4], wrappers protect name servers from receiving unverifiable requests. A somewhat general approach to wrappers for common security concerns is described in [6].

Most of this previous work assumes that a good knowledge can be gained about which communications will cause OTSC failure. We have argued for a more general view of protective wrapping [9], to take into account the fact that this knowledge is usually deficient, the specification of the OTSC may be incomplete, and designers need to be concerned with failures of both the OTSC and the rest of the system. Here, we discuss issues of design, verification and quantitative dependability trade-offs that arise in protective wrapping.

In the rest of this paper, Section 2 introduces terminology and an illustrative example. Section 3 introduces the specifications of components in relation to system-level requirements, including those concerning fault tolerance. Sections 4 and 5 discuss the options for the actual semantics of wrappers, i.e. the cues that can trigger their intervention and the forms of these interventions. Section 6 sets the previous discussion of wrapper specifications in the context of probabilistic system dependability requirements and discusses the important design trade-offs that arise. Our conclusions follow.

2 System Model and Example

Throughout this paper, we will use a simple example to clarify the concepts introduced. The example system (Fig. 1) is a water boiler. We focus on a single