

Issues in Object Orientation and Software Safety

John Favaro

Consulenza Informatica, Via Gamerra 21,
56123 Pisa, Italy
jfavaro@tin.it

Abstract. The object oriented approach to software development has become the most popular paradigm for software development today. Particular claims are made about its contribution to software reusability, through characteristics such as encapsulation, inheritance, and polymorphism. But it is claimed by many that those very characteristics make it infeasible to create verifiable safety critical systems. **Keywords:** safety, object oriented, functional, testing.

1 Introduction

Just last week I had some safety engineers complain to me that they are finding it extremely difficult if not impossible to perform safety analysis and ensure safety in systems with [object oriented] software.
- N. Leveson [1]

Given the popularity of the object oriented paradigm today, and in particular its support for some of the most valued principles in software engineering, such as strong encapsulation, modularity, and reuse, it generally comes as a surprise to discover that object orientation is looked upon with skepticism by many in the safety-critical software community.

The skepticism is not based upon unfounded theoretical considerations, but upon practical experience in the real world of projects. As the quote above reflects, safety engineers have often found that the proven techniques that have been applied for years to the verification of safety-critical systems developed with traditional, procedural approaches could only be applied with much difficulty or not at all to systems developed with object oriented approaches.

This is cause for concern in the software reuse community, since enhanced reuse is one of the advantages claimed by promoters of object orientation.

2 Object Orientation, Requirements, and Traceability

According to Leveson, the problem starts already at the requirements and design levels. This problem can be seen, for example, in the way that software is envisioned in the CENELEC 50128 standard for railway systems software. Following traditional

practice, the standard foresees partitioning of the system first into components, and then at a lower level into modules. The requirements are then allocated onto these components and modules.

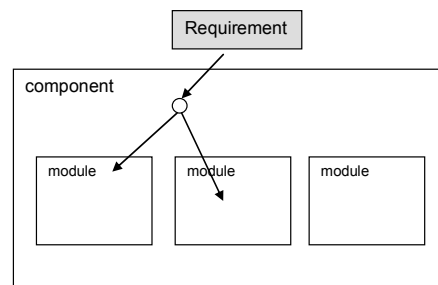


Figure 1: Components and modules in CENELEC 50128

Requirements are generally allocated in a kind of tree structure down through the levels of component and module, and the tree is generally not very broad, so that it is relatively easy to trace requirements down to where they actually are implemented in the code. But Leveson reports that object orientation changes the nature of this traceability:

OOD [Object Oriented Design] results in the opposite of what we used to call cohesion, i.e., putting all the information related to a particular function in one module so that it becomes easier to read and to change. The goal in design was to reduce coupling and increase cohesion. OOD spreads out the operations related to a function – potentially throughout all the objects – and therefore it is much harder for an application expert or safety engineer who is trying to determine whether a particular function (which may be implemented using methods from dozens or hundreds of objects), as implemented, is both correct and safe. [1]

The way in which object orientation spreads out the allocation of requirements onto objects essentially requires another approach to traceability. Currently, one approach used is to trace requirements through the use cases and collaborations among objects. One organization I am aware of traces requirements down to responsibilities of objects.

3 Object Orientation, Concurrency, and Safety

Another issue is concurrency. This message posted on a forum on safety critical software introduces the problem:

... a lot is written about the design of safety-critical systems with object-oriented methods. I really wonder if object-oriented programming is applicable to the de-

velopment of technical safety-critical systems. Very important features of those systems are, in my opinion, concurrency and timing constraints. Features that are not systematically supported by object-oriented programming languages. Concurrency is still a more procedure-oriented feature of languages like C++ and Java. Time-critical operations are very problematic, because of dynamic memory management. Probably it is possible to overcome the first problem by some construction, but is synchronisation of a concurrent and distributed system without deterministic time response possible? Is determinism possible without the loss of the dynamic features of object-orientation?

The essential non-determinism of many aspects of object orientation, from polymorphism to dynamic allocation of objects, has been a problem under study for some time. The approach I am most familiar with concerns the definition of restrictive profiles of both programming languages and modeling languages. A typical example is the HRT-UML methodology and supporting toolset. These support an approach to develop systems that exhibit predictable hard real-time behavior, principally by restricting the modeling and programming elements to a subset defined by the so-called Ravenscar profile. [3] The approach is considered effective enough that the European Space Agency is promoting it for use in the development of embedded onboard systems. [2]

The ASSERT project of the European Commission is doing research on an approach to developing reusable components (called “containers” in their terminology) with provably guaranteed non-functional characteristics such as schedulability and fault containment. [4]

4 Reuse, Dead Code, and Safety

Reuse is an important design goal for many OO programs. However, the requirements, design, and code of a reusable component might contain more functionality than required by the system being certified, which raises concerns about dead and deactivated code. Based on DO178B, the European Software Standard ECSS-Q-80-01 requires that dead code be *removed* and analysis performed to assess the effect and need for re-verification. In addition, it requires deactivated code to be verified (analysis and test) to prove that it cannot be inadvertently activated. Hence, there is a trade-off between the benefit of reuse and the cost of additional verification of deactivated code or removal of dead code. A specific concern raised in the issue list is the difficulty of identifying dead code and deactivated code in O-O programs. Dead or deactivated code can result when (a) methods of a class are not called in a particular application; (b) methods of a class are overridden in all subclasses; or (c) attributes of a class are not accessed in a particular application. For example, no instances of a superclass might be used if all of the subclasses override a particular method.

4 Conclusion: what is a complex system?

Leveson writes:

The problem I have found in using object-oriented design in control systems is that the design becomes more complex and *much* harder to validate or verify for safety. I have looked at dozens of OO designs of control systems, and I find them incredibly more complicated than those based on functional decomposition. I think I can now explain why. A mechanical engineering professor here at MIT defines "complexity" as the degree to which the structural decomposition of the system differs from the functional decomposition. OOD structurally decomposes the system (according to objects) but the functionality gets spread throughout and gets far away from the structure. That probably also explains why I see engineers struggling to use object oriented designs and making objects like "navigation" (a function, not an object). OOD fanatics tell me it is just because those engineers do not know how to do object-oriented design, but I think there is a much deeper reason. They seem to find it much more natural to design these types of systems using functional decomposition. That naturalness translates into easier to understand and review, easier to design without errors, easier to analyze to determine whether the system does what the engineer wants and does it safely. The problem is that I want to analyze functions, not objects. I also think that the best and most natural design method for control systems is functional decomposition. ... What I don't understand is why so many people nowadays think there is only one way to design all systems. A master craftsman has a whole toolkit and not only can use them all but knows when it is best to use each tool. An amateur has only a few tools and uses them all for everything. Same with great architects. There are always tradeoffs in any complex design effort, and different designs will optimize the different desired qualities. A great designer knows how to create a design that will optimize the most important qualities for that system. If building a screen editor, I would be the first to use an OOD. But when designing a control system, I would use a more appropriate design. The truly great software designer does not use OOD for every design (that is the sign of an amateur) but can select a design approach that is the best for that particular system. [1]

These observations still hold true to a great degree today. Object orientation in safety critical systems is still confronted with a number of thorny issues, and the safety critical community is understandably very conservative about allowing systems to be constructed with object oriented technology without a clear vision of how traditional techniques for ensuring safety are mapped onto these systems. On the other hand, the popularity of object technology is so great and growing that we can expect that over the next few years a number of the outstanding issues will be resolved.

References

1. Leveson, N., 2002, message to Safety-Critical Mailing List, archived at the following URL: <http://www.cs.york.ac.uk/hise/safety-critical-archive/2002/0203.html>.
2. Mazzini S., Favaro J., Puri S., Bavaro M., “Software Methodological and Tool Support for Embedded Control Systems”, Proceedings of the “Adaptability Techniques for Control System Software” session of the IFAC 2005 Conference.
3. Mazzini S., Puri S., Stragapede A., “A UML2 Profile for Hard Real-Time Modeling”, Proc. of the DASIA Conference, 2005.
4. ASSERT Online, <http://www.assert-online.net/>