Applying Research Results in the Industrial Environment: the Case of the TRIO Specification Language\textsuperscript{1}.

Dino Mandrioli\textsuperscript{2}

Abstract

There are almost universal complaints that too much research effort goes wasted and never finds application in the industrial world. The complaints are raised symmetrically both by the academia and by the industrial world. This situation becomes even more frustrating in the case of application of formal methods to software engineering: despite formal methods are advocated as a useful tool to enhance software quality from more than thirty years, it is still quite controversial whether or not they can really have an impact on the industrial software development.

This paper addresses the above issue on the basis of author's experience. This experience has been developed for several years in the field of real-time systems. The core of the research is a formal specification language for real-time systems, TRIO, which is an extension of temporal logic. The language has been enriched by supporting tools and methods and has been applied to real industrial projects of increasing complexity.

In this paper, a short introduction to the TRIO language is provided; then the experience on its practical application is briefly reported; finally comments are given on the most important "lessons learned" from the above experience. It is argued that, on the basis of this and similar experiences, new enthusiasm and useful indications for hoping into greater success of formal methods can be generated.

1. Introduction

There are almost universal complaints that too much research effort goes wasted and never finds application in the industrial world. The complaints are raised symmetrically both by the academia and by the industrial world. This situation becomes even more frustrating in the case of application of formal methods to software engineering: despite formal methods are advocated as a useful tool to enhance software quality from more than thirty years, it is still quite controversial whether or not they can really have an impact on the industrial software development.

On the one hand, the literature reports many examples of unsatisfactory software quality. This inconvenience is most serious when software is embedded in safety-critical systems, such as avionics systems, plant control systems, etc. In these cases, it has been often reported that a software failure caused major disasters: see, e.g., the well documented case of the THERAC 25 accidents [LT]. In principle, the use of formal techniques promises major improvements in the overall quality of such products (see e.g. [MMM] for an in depth analysis of the THERAC 25 case). These promises have also been verified in a fairly rich set of prototypal applications to real-life cases.

On the other hand, still much skepticism is raised against a wide and generalized adoption of formal methods in practice. Usual complaints are about the excessive amount of mathematical skill that is needed to apply them; about the unsuitability of

\textsuperscript{1}Work partially supported by CNR and partially by ENEL.
\textsuperscript{2}Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza L. Da Vinci 32, 20133, Milano, Email: mandriol@elet.polimi.it
pure mathematical formalisms to catch many aspects that matter in practical applications; even the impact on final product correctness is often challenged.

This paper addresses the issue of practical usefulness of formal methods in the development of computer-based applications on the basis of author's experience. This experience has been developed for several years in the field of real-time systems. Author's group has been composed from the very beginning by researchers and "users" (i.e., industrial experts of the application domain). The core of the research consists of a formal specification language for real-time systems, TRIO, which is an extension of traditional temporal logic.

The language has been enriched by supporting tools (editors, interpreters, test-case generators, verifiers and, in a more general sense, manuals, seminars, and methods for its use) and has been applied to real industrial projects of increasing complexity. The three "processes", namely theoretical language development, tools construction, and application to real-life cases, have been and are being performed in parallel producing mutual feedbacks.

The paper is organised as follows: Section 2 provides a short introduction to TRIO: this includes a survey of language features and a brief description of related tools and methods; Section 3 reports on major past and present applications of TRIO to industrial projects. Section 4 reports on main lessons learned from such experiences. In Section 5 a few conclusions are derived: it is argued that, if we are able to learn from previous failures, new enthusiasm can be generated and useful indications can be obtained for hoping into greater success of formal methods in the future.

The presentation style of this paper is of tutorial type: only general principles and main features are addressed, avoiding technicalities. The interested reader will be referred to a more appropriate literature for a deep analysis of technical details. A little background on mathematical logic and on object-oriented terminology is assumed.

2. TRIO's Basics

TRIO [GMM] (acronym for the Italian words "Tempo Reale ImplicitO") is now a fairly general purpose specification language, although it is rooted in the application field of real-time systems – which are often highly demanding in terms of safety-critical requirements. TRIO's evolution has been driven by the following guidelines and major objectives:

• Attention was initially focused on the "high phases of the life-cycle" which are universally acknowledged as the most critical ones. It has always been kept in mind, however, that, eventually, a complete development methodology covering the whole life-cycle was needed.

• Requirement specifications should be as abstract as possible, i.e., should not bias – nor should they be biased by – implementation choices. This is mainly important in an application environment whose products often have a life of tens of years – and, therefore, must survive tumultuous technological evolution – and where often the product implementation is performed by a different organization than the one which defines requirements and uses the product.

For this reason a specification language based on mathematical logic rather than on some kind of abstract machine has been chosen.

• A specification language to be used in practice must join the precision and rigor that are typical of mathematical formalisms with other factors that have a major impact

---

3Our main industrial partner is ENEL, the Italian agency of energy.