

Software Engineering Conference Russia 2006

Formalization of SW Interface Standards and Automatic Construction of Conformance Tests

Speaker:

Vladimir Rubanov,
Project Manager,
Institute for System Programming of the RAS

Co-authors:

Prof. A.K. Petrenko, A. Khoroshilov, V.Kuliamin
ISP RAS

Keywords

Interface standards, testing, requirement formalization, formal modeling, UniTESK

Abstract

Modern software systems usually consist of many components from different vendors, and a lot of individual software engineers or even organizations take part in their construction. Due to this fact, ensuring interoperability of those parts and reliability of the system as a whole is very challenging. The well-known way to solve these problems is enforcement of *software interface standards*.

The idea of interface standards is rather clear - to make possible for different software systems working together through standardized interfaces without hard restrictions on the internal implementations of them. So, interoperability is ensured without damping individual creativity and corporate innovation potential. This approach works well if the standard defines the functionality of the corresponding interfaces clearly, unambiguously, and precisely. However, looking at the current software standards shows that this is a rare case. Historically, they were developed under the market pressure to ensure some interoperability taking into account conflicting interests of many software vendors. Some of the current software and telecommunication standards appeared about 20-25 years ago, and, of course, with each revision they become clearer and more consistent. Most of ambiguities of the first versions were removed, but newer additions and elaborations required by technological progress still may have unclear and equivocal statements.

The way out of this situation proposed by many software engineering researchers and practitioners is *formalization of standards*. Formal re-statement of standard's requirements discloses contradictions and ambiguities, but exacts a lot of hard work. Nobody expects that inaccuracy and inconsistency will be removed from standards at once. But by starting formalization, we at least become aware of real problems and can make practical suggestions on their solutions. Formalization definitely makes standards more useful, but alone it does not give instruments of their enforcement in practice. Developers of real-life systems need some tools that help them ensure conformance to standards. So, practically useful formal description of standard requirements should be supplemented with corresponding *test suites* for checking conformance to these requirements.

The approach described in this paper uses a framework for conformance testing based on automatic test generation from formal specifications. Solid framework makes conformance test construction more rigorous, and therefore helps to enforce quality of the systems implementing the standard. However, practical use of conformance test suites brings into account additional engineering issues.

- *Requirements traceability*. For software engineers and their managers the real source of requirements is the standard's text. Formal specification is an additional document that should clearly demonstrate its correspondence to the standard. The same holds for the tests derived from them - they should be traced to some requirements stated in some sections of the standard. This really helps conscientious development of standard implementations.

- *Component-wise treatment of standard.* Real-life standards are complex, as well as real-life software. Some decomposition techniques are required for adequate treatment of them. Specification formalism used should support definition of separate components of the standard, should allow their consideration in isolation, but also provide means for describing them as a whole.
- *Change management.* Standards and their implementations are not fixed entities - they are changing. These changes should have adequate support in the process of specification development, test derivation and translation. Lack of this support quickly makes the specifications and tests useless.
- *Configurations.* Real-life standards describe parameterized systems, having a lot of configuration options that significantly influence their functionality.

Such options should be also supported in specifications and tests. All those problems should have suitable solutions in the framework that aspires to provide an adequate infrastructure for enforcing software standards.

This paper presents a candidate approach based on UniTESK technology of automated test construction from formal specifications of requirements. The technology was developed by ISP RAS and has being used for more than 10 years in research and industrial projects. We describe preliminary results of applying this approach and the technology to formalizing requirements and creating a conformance test suite for Linux Standard Base, an industrial standard on core API interfaces of Linux operating system.

To stimulate activities in enforcing open standards, ISP RAS donates the results of the project and all the tools needed to deal with them to open source community. We hope that members of this community can be attracted to challenging projects for applying the approach and the tools for other open standards such as embedded and real-time OS specifications and some widely used programming language run-time libraries.