Formal Requirements Capturing using VRS system

Alexander Letichevsky[1], Alexander Kolchin[1], Oleksandr Letychevskyy jr.[1], Stepan Potiyenko[1], Vlad Volkov[1] and Thomas Weigert[2]

[1] Glushkov Institute of Cybernetics, Ukraine
[2] Missouri University, USA

We present system VRS (Verification of Requirements Specifications) designed for development of formal specification and verification. This system has been developed by VRS Kiev group during last 10 years to support requirements capturing in Motorola and then Uniquesoft projects. As its input language VRS uses parameterized MSCs (Message Sequence Charts) with pre- and postconditions interpreted on the states of an environment with inserted agents. Such small MSC we call : *Basic Protocols*. Semantically basic protocol can be considered as a statement $\forall x(\alpha \rightarrow < u > \beta)$ of some kind of dynamic logic [2]. In this statement x is a (typed) list of parameters, α and β are precondition and postcondition, correspondingly, and u is a process defined by the MSC diagram.

There are three main methods of verification in VRS supported by appropriate tools. The system provides static requirement checking on the base of automatic theorem proving; symbolic and deductive model checking, and generation of traces for testing with different coverage criteria.

Concrete trace generator (CTG) of VRS system provide checking the reachability of properties, detecting deadlocks, non-determinisms, safety violations, unreachable requirements, usage of uninitialized attributes and admitted range attribute overflow on a base of a concrete model of formal requirement specification in the form of basic protocols. The problems above are solved by means of generating traces reachable from the initial state of a model. The generated traces can also be used for test generation.

Symbolic Trace Generator (STG) deals with the same iput data as for CTG. The difference is that an abstract model of a system specified by a set of basic protocols is considered instead of a concrete one. The state of a model has the form $\gamma[u_1...u_n]$ where γ is a formula over attributes which coincides with the attributed label of the environment state and $u_1...u_n$ are agents inserted into environment. The deductive system is used for checking the applicability of basic protocol and predicate transformer is used to obtain the next state.

Static requirement checking (SRC) tools allow to solve verification problems without generating traces and exploring the state space. The deductive system is based on the prover for the first order predicate calculi with equality extended by some special provers and solvers. THe last ones support proving and solving linear numerical constraints for integers (Presburger algorithm) and for reals (Fourier-Motzkin algorithm), proving and solving formulas for enumerated and symbolic data types. There is a possibility to generate invariants then SRC technique can be improved by iterative steps in sense that if F is not safety then VRS can generate an invariant G, improve safety property to $F\&G$, and check safety repeatedly.

In the last years this approach has been successfully applied to the problems of the verification of requirement specifications [1, 3] for about 30 distributed concurrent systems from different subject domains including Telecommunications, Telematics, distributed computing and others.

References

[1] A. Letichevsky Jr. V. Volkov S. Baranov V.Kotlyarov T. Weigert. A.Letichevsky, J. Kapitonova. Basic protocols, message sequence charts, and the verification of requirements specifications. *Computer Networks*, (47):662–675, 2005.

[2] Dexter Kozen David Harel and Jerzy Tiuryn. *Dynamic Logic*, page 400, 2000.

[3] V. Kotlyarov A. Letichevsky S. Baranov, C. Jervis and T. Weigert. Leveraging uml to deliver correct telecom applications. *In L. Lavagno, G. Martin, and B.Selic, editors. UML for Real: Design of Embedded Real-Time Systems. Kluwer Academic Publishers, Amsterdam*, 2003.