NIMBUS: A Tool for Specification Centered Development

Mats P.E. Heimdahl, Michael W. Whalen, Jeffrey M. Thompson
University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455, USA.

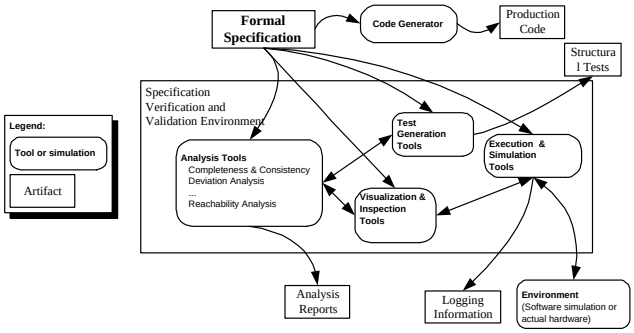

Figure 1. The NIMBUS Environment.

Assurance that a formal specification (system specification or software specification) possesses desired properties can be achieved through (1) manual inspections, (2) formal verification of the desired properties, or (3) simulation and testing of the specification. To achieve the high level of confidence in the correctness required in a safety-critical system, all three approaches must be used in concert. We have developed an specification language, called RSML^{-e}, and an environment, called NIMBUS, which provides support for all these activities (Figure 1). The three V&V techniques fill complementary roles within the validation and verification process. Manual inspections and visualization provide the specification team, customers, and regulatory representatives the means to informally verify that the behavior described formally matches the desired "real world" behavior of the system. Formal analysis is helpful to determine if the specification possesses desirable properties. Simulation and testing helps the analyst to evaluate and address poorly understood aspects of a design, improves communication between the different parties involved in development, allows empirical evaluation of design alternatives, and is one of the more feasible ways of validating a system's behavior.

RSML^{-e} [4] is a fully formal, synchronous, data-flow language semantically similar to Lustre [1], SpecTRM-RL [3], and SCR [2]. NIMBUS supports large-scale, distributed simulation of specifications through communications over Microsoft's distributed COM or OMG's CORBA. For example, in a Rockwell Collins Inc. simulation of portions of a flight deck, 20 different simulations controlling flight guidance, flight management, instrumentation and the external environment were concurrently executed in a real-time simulation on two machines. Through the use of standard communications protocols, support is provided for simulations involving input files, other software tools (such as Microsoft Office or Simulink), and actual hardware devices.

Through automated translation to the model checker NuSMV and the theorem prover PVS, a wide range of analysis capabilities are provided. These capabilities include both verification of safety and liveness properties on specifications, and also new analysis techniques such as deviation analysis. Deviation analysis, determines how a specification behaves given expected sensor and actuator errors and delays. In addition, we have used PVS to help determine root causes of mode confusion in specifications, attempting to compare "mental models" of systems under consideration with the actual specification.

Finally, we have a proven correct code generator, extensive support for automated structural test-case generation from specifications up to an MC/DC level, and the ability to generate statistical tests according to a usage profile, allowing our toolset to significantly reduce the amount of effort required for developing certain safety-critical systems. NIMBUS provides a developer a mature, extensive toolkit of services for writing and verifying specifications. NIMBUS is also open for extension and will be released for research purposes in conjunction with RE'03.

References

[1] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language lustre. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.

[2] C. Heitmeyer, J. K. Jr., B. Labaw, M. Archer, and R. Bharadwaj. Using abstraction and model checking to detect safety violations in requirements specifications. *IEEE Transactions on Software Engineering*, 24(11):927–948, November 1998.

[3] N. G. Leveson, M. P. Heimdahl, and J. D. Reese. Designing Specification Languages for Process Control Systems: Lessons Learned and Steps to the Future. In *Seventh ACM SIGSOFT Symposium on the Foundations on Software Engineering*, volume 1687 of *LNCS*, pages 127–145, September 1999.

[4] M. W. Whalen. A formal semantics for RSML^{-e}. Master's thesis, University of Minnesota, May 2000.