does not include failure states explicitly; rather essentially a reward is assigned to each state (with the value of the reward representing the probability of the system failing in that state), where the system's reliability is computed as a Markov reward function [25]. However, the system failure description is still limited, assuming that the system fails when any (active) component fails. [28] provides a somewhat richer description of system failures, where a reliability model includes backup components that can provide services when the primary component fails; the system fails when the primary component and all backup components fail. However, this approach is not capable (without significant changes) of describing other notions of system failure, e.g., an OR-type relationship (the system fails when $Comp_A$ or $Comp_B$ fails). Such notions of system failure can be described within SHARP, by changing the way we compute combination reliability in Section 4.3.5.

Some existing approaches make use of scenario models [10, 22, 30], but they assume a sequential system, with the exception of [22] as described above. For example, in [30] system reliability is defined as the weighted sum of scenario reliabilities. The weights represent the probabilities that each scenario occurs, with the assumption that one scenario is active at a time. This is not the case in our work: in a concurrent system, it is possible to have concurrency within a scenario, as well as multiple scenarios and/or multiple instances of the same scenario running simultaneously. Moreover, [30] assumes that the probabilities of each scenario occurring are known, which is also not the case in our work.

7. CONCLUSIONS

We presented SHARP, a scalable framework for predicting reliability of concurrent systems. SHARP models concurrency by allowing multiple instances of system scenarios to run simultaneously. We overcame inherent scalability problems by leveraging scenario models and using an approximate hierarchical technique that allowed generation and solution of smaller parts of the overall model at a given time. Our experimental evaluation showed that SHARP's scalability, which is missing from existing techniques, is achieved without significant degradation in the prediction accuracy.

8. ACKNOWLEDGEMENTS

This work is supported by the NSF (award numbers 0509539, 0920612, and 0905665).

9. REFERENCES

[1] F. Baskett et al. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2), 1975.

[2] B. Boehm. Software engineering economics. *IEEE TSE*, 10(1), 1984.

[3] L. Cheung et al. Early prediction of software component reliability. In *ICSE'08*.

[4] L. Cheung et al. SHARP: A scalable approach to architecture-level reliability prediction of concurrent systems. In *QUOVADIS'10*.

[5] R.C. Cheung. A user-oriented software reliability model. *IEEE TSE*, 6(2), 1980.

[6] V. Cortellessa et al. Early reliability assessment of uml based software models. In *WOSP'02*.

[7] R. El-Kharboutly et al. UML-based methodology for reliability analysis of concurrent software applications. *I. J. Comput. Appl.*, 14(4), 2007.

[8] S. Gokhale. Architecture-based software reliability analysis: Overview and limitations. *IEEE TDSC*, 4(1), 2007.

[9] S. Gokhale and K. Trivedi. Reliability prediction and sensitivity analysis based on software architecture. In *ISSRE 2002*.

[10] K. Goseva-Popstojanova et al. Architectural-level risk analysis using UML. *IEEE TSE*, 29(3), 2003.

[11] K. Goseva-Popstojanova and S. Kamavaram. Software reliability estimation under uncertainty: Generalization of the method of moments. In *HASE 2004*.

[12] K. Goseva-Popstojanova and K. Trivedi. Architecture-based approaches to software reliability prediction. *Intl J. Computer & Mathematics with Applications*, 46(7), 2003.

[13] A. Immonen and E. Niemela. Survey of reliability and availability prediction methods from the viewpoint of software architecture. *Software and Systems Modeling*, Jan 2007.

[14] I. Krka et al. Synthesizing partial component-level behavior models from system specifications. In *ESEC/FSE 2009*.

[15] I. Krka et al. A comprehensive exploration of challenges in architecture-based reliability estimation. *Architecting Dependable Systems*, 6, 2009.

[16] J. Magee and J. Kramer. *Concurrency: State Models And Java Programs*. John Wiley & Sons, 2006.

[17] S. Malek et al. Reconceptualizing a family of heterogeneous embedded systems via explicit architectural support. In *ICSE'07*.

[18] C. D. Meyer. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Review*, 31(2), 1989.

[19] OMG. UML 2.2 specification, 2009.

[20] M. Reiser and S. S. Lavenberg. Mean value analysis of closed multichain queueing networks. *J. ACM*, 27(2), 1980.

[21] R. Reussner et al. Reliability prediction for component-based software architectures. *J. of Systems and Software*, 66(3), 2003.

[22] G. Rodrigues et al. Using scenarios to predict the reliability of concurrent component-based software systems. In *FASE 2005*.

[23] R. Roshandel et al. A Bayesian model for predicting reliability of software systems at the architectural level. In *QoSA 2007*.

[24] R. Roshandel, B. Schmerl N. Medvidovic, D. Garlan, and D. Zhang. Understanding tradeoffs among different architectural modeling approaches. In *WICSA 2004*.

[25] W. Stewart. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, 2009.

[26] R. Taylor, N. Medvidovic, and E. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. Wiley, 2009.

[27] S. Uchitel et al. Incremental elaboration of scenario-based specifications and behavior models using implied scenarios. *ACM TOSEM*, 13(1), 2004.

[28] W. Wang et al. Architecture-based software reliability modeling. *J. of Systems and Software*, 79(1), 2006.

[29] J. Whittle and P.K. Jayaraman. Synthesizing hierarchical state machines from expressive scenario descriptions. *ACM TOSEM*, 19(3), 2010.

[30] S. Yacoub et al. Scenario-based reliability analysis of component-based software. In *ISSRE'99*.