JOURNAL OF OBJECT TECHNOLOGY
Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering. ©JOT, 2002

JOURNAL OF OBJECT TECHNOLOGY
Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering. ©JOT, 2002

Vol. 0, No. 0, Z

Componentware for Critical Systems

Jan Jürjens[1]
Software & Systems Engineering, Dep. of Informatics, TU München, Germany

We give an overview over the challenges and some solutions for using componentware for building critical systems.

The high quality development of critical systems (be it dependable, security-critical, real-time, or performance-critical systems) is difficult. Many critical systems are developed, fielded, and used that do not satisfy their criticality requirements, sometimes with spectacular failures.

Systems whose correct functioning human life and substantial commercial assets depend on need to be developed very carefully. Systems that have to operate under the possibility of system failure or external attack need to be scrutinized to exclude possible weaknesses.

Part of the difficulty of critical systems development is that correctness is often in conflict with cost. Where thorough methods of system design pose high cost through personnel training and use, they are all too often avoided.

On the other hand, there is an increasing interest in the use of componentware due to a possibility for savings from potential reuse.

This raises the question whether the component-ware approach can be used fruitfully in the critical systems area.

Beyond the general arguments about savings by reuse, the componentware approach offers an interesting opportunity for high-quality critical systems development that is feasible in an industrial context.

- If reusable critical components can be identified, they can be developed at a high standard of quality, and possibly even certified by official authorities.
- If a suitable methodology for component-based construction of critical system exists, these critical components can be employed safely and securely within the system context.

Under these two provisos, the componentware approach has the potential not only to reduce costs, but at the same time to increase the quality of critical systems software. This observation prompts some challenges one has to overcome to exploit this opportunity, which include the following:

- Adaptation of an appropriate notion of component to critical system application domains.
- Correct use of critical components in the system context and the application domain.
- Conflict between flexibility and level of criticality guarantees when defining components.
- Improving tool-support for component-based development of critical systems.

Surprisingly few attempts have thus far been started to address these challenges, explicitly dealing with component-based development of critical systems: [MV99] points out some problems with using components-off-the-shelf (COTS) software for security-critical systems. On the tool-side, the CASE tool AutoFocus [SH99] based on the component-based design methodology in [Bro99] has been used for critical systems development for example in [GHJW03]. Component-based approaches to developing critical systems with the Unified Modeling Language can be found for example in [Jür02, Jür04, Jür03]. [GHD00] considers components in the context of specifying safety-critical embedded systems with Statecharts and Z. The integration of mixed-criticality software components is treated in [DS99]. [May02] considers testing component-based safety critical software. [JH03] treats the verification of requirements specifications using composition and invariants. Several of the contributions in [JCF$^+$02] also consider components. To be able to use an component-based approach to developing critical systems, one needs knowledge on to what extent criticality requirements are preserved by composing system parts. For example, [McL96, Jür00] examine composability of security requirements.

REFERENCES

[Bro99] M. Broy. A logical basis for component-based systems engineering. In M. Broy and R. Steinbrüggen, editors, *Calculational System Design*. IOS Press, 1999.

[DS99] B. Dutertre and V. Stavridou. A model of noninterference for integrating mixed-criticality software components. In *International Working Conference on Dependable Computing for Critical Applications (DCCA)*, San Jose, CA, January 1999.

[GHD00] W. Grieskamp, M. Heisel, and H. Dörr. Specifying Safety-Critical Embedded Systems with Statecharts and Z: An Agenda for Cyclic Software Components. *Science of Computer Programming*, 2000.

[GHJW03] J. Grünbauer, H. Hollmann, J. Jürjens, and G. Wimmel. Modelling and verification of layered security protocols: A bank application. In *Computer Safety, Reliability, and Security (SAFECOMP 2003)*, volume 2788 of *Lecture Notes in Computer Science*, pages 116 – 129, Edinburgh, GB, 23-26 September 2003. Springer-Verlag.

[JCF$^+$02] J. Jürjens, V. Cengarle, E. Fernandez, B. Rumpe, and R. Sandner, editors. *Critical Systems Development with UML*, number TUM-I0208 in TUM technical report, 2002. UML'02 satellite workshop proceedings.

[JH03] R. Jeffords and C. Heitmeyer. A strategy for efficiently verifying requirements specifications using composition and invariants. In *European Software Engineering Conference/ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003)*, Helsinki, Finland, September 1-5 2003.

[Jür00] J. Jürjens. Secure information flow for concurrent processes. In C. Palamidessi, editor, *CONCUR 2000 (11th International Conference on Concurrency Theory)*, volume 1877 of *Lecture Notes in Computer Science*, pages 395–409. Springer-Verlag, 2000.

[Jür02] J. Jürjens. UMLsec – Presenting the Profile. In *Sixth Annual Workshop On Distributed Objects and Components Security (DOCsec2002)*, Baltimore, MD, March 18-21 2002. OMG. Half-day tutorial.

[Jür03] J. Jürjens. Developing safety-critical systems with UML. In P. Stevens, editor, *UML 2003 – The Unified Modeling Language*, volume 2863 of *Lecture Notes in Computer Science*, pages 360 – 372, San Francisco, Oct. 20–24 2003. Springer-Verlag. 6th International Conference.

[Jür04] J. Jürjens. *Secure Systems Development with UML*. Springer-Verlag, March 2004. To be published.

[May02] J. May. Testing the reliability of component-based safety critical software. In S. Thomason, editor, *20th International System Safety Conference*, pages 214–224. System Safety Society, August 2002.

[McL96] J. McLean. A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, 1996.

[MV99] G. McGraw and J. Viega. Why cots software increases security risks. In *ICSE Workshop on Testing Distributed Component-Based Systems*, May 1999.

[SH99] Bernhard Schätz and Franz Huber. Integrating formal description techniques. In *[WWD99]*, pages 1206–1225, 1999.

[WWD99] Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors. *World Congress on Formal Methods in the Development of Computing Systems (FM'99), Volume II*, volume 1709 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.