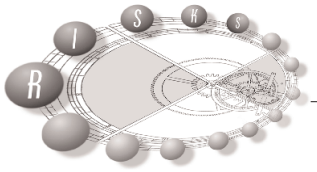

Wilhelm Hasselbring, Matthias Rohr, Jürgen Taeger, and Daniel Winteler

Liability Risks in Reusing Third-Party Software

Reuse of well-tried components promises cost-effective construction of high-quality software systems. With this approach, third-party software components are usually integrated into complex software systems. The risks of using such components of different suppliers are both technical and judicial. Some technical problems were discussed by Peter G. Neumann in the July 2006 "Inside Risks" column. Here, we discuss emergent judicial concerns as they apply in Europe.

From the judicial point of view, a risk exists for system vendors to be held liable for malfunctions that are not self-induced, without being able to take recourse to the supplier delivering the faulty component. The resulting problems are evident in the context of German legislation. These problems apply throughout the European Union, due to an EU council directive concerning liability for defective products (No. 85/374/EEC of July 25th, 1985).

The reason for the additional liability risks when reusing third-party software components is this: The integrator/vendor of systems composed of several software components from different suppliers is in general liable to his customers as well as to the general public for any malfunction of the whole system. If the product does not function properly, the customer will confront the vendor—for example, to receive a refund or to seek a discount. Even worse, if the system is safety critical, the customer might sue the vendor for damages. The same is true for any injured persons who are not the contractual partner of the vendor. The vendor might have to pay huge sums because of damage to goods or people. The customer and the injured party have the advantage that an exact localization of the fault is not necessary for their claim. They merely have to provide evidence that a malfunction of the product caused the damage. The vendor then has little chance of warding off the claims. In particular, according to the European directive 85/374/EEC, no contractual derogation is permitted as regards the liability of the producer in relation to the injured person; all producers involved in the production process could be made liable, insofar as their finished product, component part, or any raw material supplied by them was defective.

After having paid for damages, the system vendor might consider taking recourse to one of his suppliers, or more precisely, the one supplier that delivered the defective component. In contrast to the injured party, the vendor now has the problem that he must localize the fault within his component-based system. He can point his finger to a supplier only if he can prove that this supplier's software component failed and contains the responsible fault as root cause of the failure. Fault diagnosis and fault localization in complex component-based systems present a real challenge, because they can require the complete reconstruction of the error-propagation process among all components and the identification of the conditions that activated the root cause of the failure.

An additional difficulty for software forensics is the transient state, which is usually lost at the time of system failure. The error propagation process within a system composed of software components does usually not create "visible" indications about single component failures. In contrast, error propagation processes in hardware usually create physical indication toward the fault locations. For instance, if a laptop burns, it should be possible to prove that a particular hardware component such as the battery burned first (although this does not necessarily mean that this component was responsible).

Component specifications are used to formalize provided services, and may define what is required from the execution environment or by other components in order to provide correct service. These component specifications could be part of requirements specifications and even of formal contracts between system vendor and component suppliers. However, such specifications are often ambiguous, incomplete, and neglecting precise metrics for quality properties. Conversely, failures may be caused by improper protocols governing interactions among multiple components; thus, the system composer may be responsible for failures himself. Here, technical and judicial issues meet. In our interdisciplinary Graduate School TrustSoft (www.trustsoft.org), we integrate research in Computer Science and Law to address these problems.

Wilhelm Hasselbring (hasselbring@informatik.uni-oldenburg.de) is a professor of software engineering and chair of the graduate school TrustSoft at the University of Oldenburg, Germany. **Matthias Rohr** (matthias.rohr@informatik.uni-oldenburg.de) is a computer science Ph.D. student in the graduate school TrustSoft at the University of Oldenburg. **Jürgen Taeger** (j.taeger@uni-oldenburg.de) is a professor of law and supervisor in the graduate school TrustSoft at the University of Oldenburg. **Daniel Winteler** (daniel.winteler@uni-oldenburg.de) is a law Ph.D. student in the graduate school TrustSoft at the University of Oldenburg.

PAUL WATSON