Why is it so Hard to Predict Software System Trustworthiness from Software Component Trustworthiness?

Jeffrey Voas  
Cigital, Inc.  
voas@cigital.com

Much of the work from the past 10 years into Component Based Software Engineering (CBSE) and Component Based Development (CBD) has dealt with Functional Composability (FC). FC is concerned with whether $F(A) \xi F(B) = F(A \xi B)$ is true (where $\xi$ is some mathematical operator), i.e., whether a composite system results that has the desired functionality given that the system is created solely by joining A and B.

But increasingly, our community is discovering that FC, even if it were a solved problem (using formal methods, architectural design approaches, model checking, etc.), is still not mature enough for other serious concerns that arise in CBSE and CBD. These concerns stem from the problem of composing "ilities". "ilities" are non-functional properties of software components and define characteristics such as security, reliability, fault-tolerance, performance, availability, safety, etc.

The problem stems from our inability to know \textit{a priori}, for example, that the security of a system composed of two components, A and B, can be determined from knowledge about the security of A and the security of B. Why? Because the security of the composite is based on more than just the security of the individual components. There are numerous reasons for this and, here, we will just look at the factors of component performance and calendar time.

As an example, suppose that A is an operating system and B is an intrusion detection system. Operating systems have some level of authentication security built into them, and intrusion detection systems have some definition for the types of event patterns that likely warn of an attack. Thus the security of the composition clearly depends on the security models of the individual components. But even if B has a worthless security policy or flawed implementation, the composite can still be secure. How? By simply making the performance of B so poor that no one can log on, i.e., if the intrusion detection system is so inefficient at performing an authentication, then in a strange way, security is actually increased. And if the implementation of B’s security mechanism is so unreliable that it disallows all users access, even legitimate ones, then strangely, security is again increased. While these last 2 examples are clearly not a desirable way to attain higher levels of system security, both do actually decrease the likelihood that a system will be successfully attacked.

And if we again use our same example of A as an operating system and B as an intrusion detection system, and this time we assume that A provides excellent security and B provides excellent security, we must accept the fact B’s security is a function of calendar time. The reason for this is simply that new threats and ways to “break in” are always being discovered. So even if you could create a scheme such as $\text{Security}(A) \xi \text{Security}(B) = \text{Security}(A \xi B)$, $\text{Security}(B)$ is clearly a function of which version of B is being composed and what recent new threats have arisen.

So the question then comes down to: "which "ilities", if any, are easy to compose? The answer is that there are no "ilities" that are easy to compose and some are much harder than others. Further, there are no widely accepted algorithms for how to do so. We just demonstrated this problem for security. But note that the same holds true for others such as reliability. For reliability, consider a 2-component system in which component A feeds information in B and B produces the output of the composite. And assume that both components are reliable. So what can we assume about the composite’s reliability? While it certainly suggests that the composite system will be reliable, it must be recognized that components (which were tested in isolation for their individual reliabilities) can suddenly behave unreliably when connected to other components, particularly if the isolated test distributions did not at all reflect the distribution of transferred information after composition. Further, there can be component behaviors that are termed as "non-functional", that cannot be observed nor manifest themselves until after composition occurs. Such behaviors can undermine the reliability of the composition. And finally, if one of the components is simply the wrong component although highly reliable, naturally the resulting system will be useless.