Safety, Security, and Software Reuse: 
A Model-Based Approach

Olaf Kath\textsuperscript{1}, Rudolf Schreiner\textsuperscript{2}, and John Favaro\textsuperscript{3}

\textsuperscript{1} ikv++ technologies ag  
Dessauer Str. 28/29 – 10963 Berlin, Germany  
kath@ikv.de

\textsuperscript{2} ObjectSecurity Ltd.  
St John’s Innovation Centre, Cowley Road – Cambridge CB4 0WS, UK  
rudolf.schreiner@objectsecurity.com

\textsuperscript{3} Intecs SpA  
via E. Giannessi, 5y – I-56121 Pisa, Italy  
John.Favaro@intecs.it

Abstract. With the move to distributed, component based systems involving reuse of components and services, emergent, system-wide properties, including safety and security in particular, are becoming increasingly difficult to guarantee. Model based techniques constitute a promising approach to guarantee safety and security in systems built with reusable components. The key elements in this approach are correctness and certifiability by construction, and separation of concerns. A safety-oriented development process compliant to the ISO DIS 26262 standards is presented. A model driven approach to security accreditation of service-oriented architectures is also presented.

Keywords: reuse, safety, security, service oriented architecture, modeling, domain specific languages, certification, accreditation.

1 Introduction

Modern component based and service oriented systems take advantage of reuse to attain numerous benefits in terms of productivity and cost savings. Instead of developing complete applications and systems from scratch, now they are implemented by assembling existing and newly developed components and services, saving costs and time to market. But these benefits are offset by a number of complications when such systems have safety and security related characteristics. (Although safety and security are two separate domains, in this paper we will treat them together whenever possible.)

There are at least two principal issues in the relationship between software reuse and safety/security.

- Reuse is closely associated with the achievement of flexibility (see, for example, [1] for an economic analysis of flexibility in reuse). But flexibility
is anathema to the common concepts of security and safety assurance for monolithic systems, which rely on fixed, rigid structures that are amenable to the kinds of analyses carried out in safety and security evaluation, certification and accreditation.

- Both safety and security are *emergent* properties of a system [2], and therefore it is extremely difficult to assure component parts of the system individually. Normally, assurance takes place at overall system level.

The “local” nature of reusable components versus the “global” nature of safety and security leads to several problems.

- The creator of a component is unlikely to know *a priori* the precise safety or security context in which his component will be used, making it difficult to build in the necessary properties.
- The creator of a component is more likely to be an expert in software engineering than in security or safety.
- It is not always clear whether it is desirable to build safety and security properties into components, because it couples functional and nonfunctional properties, making reuse more difficult.

Ours is a two-pronged approach to the problem:

- Decoupling functional and nonfunctional properties through a principle of *separation of concerns*;
- Model based techniques for separately defining security policies and safety certification procedures.

## 2 Separation of Concerns in Component Based Systems

The concept of systems development using components and services has existed as long as the concept of reuse. Nevertheless, up until recent years the idea of component and service-based development was essentially “in the small.” In particular, component and service-based development took place within the context of monolithic, static architectures – whether of single programs (e.g. the programs of an office automation suite) or of entire systems (including operating systems, or large scale, distributed applications like Air Traffic Control or military C4ISR systems).

But the advances in architecture made over the last decade have brought component-based development “in the large,” including distribution over networks. As a specific example, Service Oriented Architecture (SOA) is essentially an application of component-based development with a concept of a “service as a component”. Other variations such as Software As A Service (SaaS) are similar.

(As an aside: these new component based systems have often not been recognized as variations on the theme of component based development, and consequently, their authors are unaware of results from the reuse community that are applicable, including design for optimal reusability.)
Such new types of component-based systems share a number of challenging characteristics: they are distributed; there is no longer a clear separation between development time and runtime; the overall configuration may be dynamic and change in flexible ways over the whole system life cycle; there may be multiple patterns of communication to meet the functional requirements; different parts of the overall systems might have different safety and security requirements; the safety and security requirements might change over the system requirements, for example in order to reflect new legal or compliance requirements. The reason for this type of dynamic, agile reconfiguration of systems is to react to new business processes and needs.

The fundamental problem for development for and with reuse in such environments is illustrated by the well-known “3C’s Model” [4]. This model states that the three principal issues in the creation and use of reusable components are the following:

- Concept – the abstract semantics of the component
- Content – the implementation of the component
- Context – the environment in which the component is used

It is the third issue – context – which is the most problematic in safety and security related systems. On the one hand, the more context awareness is included in the component, the more the possible uses of that component are restricted. The problem of context is normally addressed through the concept of separation of concerns. The different aspects of a component are kept as independent of each other as possible (e.g. the functional aspects and the non-functional aspects) in order to maximize the reuse potential of a component. On the other hand, it is virtually impossible to separate entirely concerns in the context of safety/security related systems due to the system-wide nature of those properties.

This dilemma of separation of concerns versus system-wide properties is currently being addressed in safety and security related development standards that seek an appropriate compromise between the need for an independent development capability encouraging component reuse on the one hand, and the need for a system-wide certification capability on the other hand. An example of such initiatives is the ISO DIS 26262 standard [5] for the automotive sector. In this standard, the concept of Safety Element out of Context (SEooC) is introduced. The essential characteristic of this and similar concepts is that the safety-related requirements applicable to this component are first assumed and the component is developed (according to the mandated safety-related development process) to satisfy the assumed requirements. This includes verification of the correct implementation of the assumed requirements.

When the full system is subsequently built (under the same applicable safety related development process), the component enters into the full system validation process according to the system requirements. In particular, the assumed safety requirements according to which the component was developed are matched against the actual safety requirements on that component in the context of the full system. These actual requirements will be the result of a full, upstream hazard and risk assessment, determination of Automotive Safety Integrity Level for the component (called an “item” during full system development), etc.
Naturally, this doesn’t completely resolve the reuse problem, because there is no \textit{a priori} guarantee that the assumed requirements out of context will match up with the actual requirements in context. But in practice that will often indeed be the case, and it will make it possible to achieve greater degrees of reuse.

Another problem specifically associated with component-based development for safety-related systems is that not only do components have to be developed out of context, as discussed above, but also at specific “safety integrity levels” – whereby this translates into a set of development best practices prescribed at each of these safety integrity levels. Thus the “out of context” problem is compounded with yet another problem: \textit{out of context}, we don’t know (yet) at which safety integrity level a component must be developed, because it is only known \textit{in context}. Here, too, some standards are beginning to address this problem with some proposed solutions. Again, we use the ISO DIS 26262 standard as an example, with its introduction of the concept of \textbf{ASIL Decomposition}. The principle of ASIL (Automotive Safety Integrity Level) Decomposition allows a developer to replace a component with ASIL at level $n$ to be replaced by the same component with ASIL at level $n-1$ plus a safety function with ASIL at level $n-1$, as long as it can be demonstrated that the component and safety function are independent.

The relationship of the ASIL Decomposition technique to reusable safety-related components is thus the following: if a component has been developed, tested, and verified out of context at a specific (automotive) safety integrity level, but it then turns out that in the context of the overall system a higher safety integrity level is specified for that component, it may be possible, if the above conditions are satisfied, to reuse that component anyway, by adding a suitable, independent safety function, rather than having to re-develop that component from scratch.

The examples of Safety Element Out of Context and ASIL Decomposition from the ISO DIS 26262 standard illustrate the fact that at least some industry standards are finally beginning to address in concrete ways – even if only partially – the problem of reuse in safety-related software.

\section{Model Based implementation of Separation of Concerns}

There are two major problems related to the construction of safety and security related systems:

- Correct construction of the system according to the safety requirements and/or security policies;
- Accreditation and certification of the system – that is, providing evidence to authorities according to accepted standards that the system does indeed comply with its safety requirements and/or security policies.

The model based approach to development is gaining in popularity because large parts of the system can be generated from models. In addition to saving time and cost and improving code quality, this approach, which combines aspects of generative and component oriented reuse, brings in a much needed aspect of globality to the
component oriented software development process. That is, it captures to a greater extent the overall system properties that are not captured by the individual components of the system.

But there are stumbling blocks in using the model based approach. Security related systems provide an illustration: often, the security policy information is embedded into the system's functional model, as annotations, for example. But this ties the security policy to the actual model and its meta model, diminishing its flexibility. In order to satisfy the principle of separation of concerns, the security policy has to be defined in a separate model, using an appropriate meta model (aka Domain Specific Language) tailored to the problem. As an additional benefit, this approach leads to reusable models of security policies.

A similar concept holds in safety. The key to combining flexibility and certifiable safety is to have separate and smaller models but to relate them to each other. Small models can be created for local areas, and then linked together for the traceability needed to demonstrate overall properties of safety. For component development “out of context,” as discussed earlier, the developer can assume the safety context and organize development based on that assumption. The user of the component “knows” the context, so he can carry out safety analysis for that context in detail.

4 Concrete example of a model based functional safety process

As a concrete illustration of a development process based upon separate and smaller models that are linked together for traceability, we present the medini analyze process and associated modeling and toolchain for functional safety in the automotive industry according to the requirements of the ISO DIS 26262 discussed earlier. In its main elements, this process is representative of the safety-related processes in a number of industries, e.g. DO 178B for aviation and EN50128 for railway systems.

Figure 1 depicts this process, whereby for purposes of illustration, we show only the detailed safety design and analysis, omitting the previous preliminary safety analysis steps which include requirements elicitation and hazard analysis. In general terms, the detailed safety related design and analysis process has the following three major elements (indicated in the figure with circled numbers):

1. **Creation of the system model.** This encompasses the design of the overall system architecture, including both software and hardware components, which may be reused. It also includes the allocation of functionality to software components and design of the software model. Note that such models may be based upon engineering designs in environments such as Matlab/Simulink or AUTOSAR.

2. **Safety related analyses.** These analyses take a number of forms, according to (automotive) safety integrity levels (ASIL) of the system components, which prescribe the analyses required. Bottom-up, *inductive* analysis techniques include Failure Mode and Effects Analysis (FMEA). Top-down, *deductive* analysis techniques include Fault Tree Analysis (FTA). These analyses are
driven by inputs such as failure rates of the various components. Fault injection techniques can stimulate the analyses with hypothesized fault events. The results of these analyses are gathered for the safety requirements related to safety goals.

3. **Evaluation of analysis results.** The evaluation of the results of the previously carried out analyses against the requirements leads into two important activities. The first is the generation of requirements to system design. The second involves the generation of certification-related documentation. (More will be said about this latter activity in a later section.) These activities will invariably feed back into the safety related analyses for further iterations until the safety related requirements are satisfied.

This safety related process can be (and usually is) carried out manually. It is a laborious process, and furthermore it is brittle, in the sense that any changes to the system could “break” the analyses and certification, necessitating a enormous amount of work to regenerate the necessary analyses and certifications. The important characteristic of the medini analyze approach is the provision of a lightweight chain of individual, “little” tools, each of which supports a particular activity, but which can be linked to each other for full traceability (and, as a consequence, partial automation) throughout the process.

Figure 2 depicts the toolchain that accompanies the process, broken down into the same three major elements introduced above:

1. **Creation of the system model.** For the actual design of the system model, a tool for creating models using the SysML notation is provided. Note that this is not a large, typically expensive commercial off-the-shelf tool from a vendor supporting the entire SysML/UML specification, but rather a custom built tool (in the Eclipse environment) supporting only the part of SysML needed and required – that is, a “little tool.” Similarly, a bespoke tool created within the Eclipse modeling environment supports the allocation of system functionality to components. Not only does this approach of “little tools” keep the cost and functionality streamlined – it also ensures that everything in the process is recorded somewhere in a model, which can then be traced.

2. **Safety related analysis.** Following the approach presented in the last paragraph, bespoke tools are provided to support each specific type of safety analysis. Each tool does no more and no less than its own analysis (e.g. no “combined multipurpose” FTA/FMEA tools). Similarly, specific bespoke tool support is provided for fault injection and simulation.

3. **Evaluation of analysis results.** The payoff of the approach of providing small, bespoke and loosely coupled tools arrives during the evaluation of analysis results, where it is possible to trace back all of the information generated by the analyses through to the design and back all of the way to the safety requirements and goals, tying all of the elements of the process together across the underlying models of the toolchain. The inevitably iterative nature of the design / analysis / evaluation cycle becomes much
more efficient and less error-prone through the ability to link back through the models from effects to causes and subsequent rework of design, repeated analyses, and new evaluation.

The complete linkage among artifacts throughout the design lifecycle makes it possible to guarantee security and safety properties by construction rather than \textit{a posteriori} V&V activities.

Note that this approach of “little models” and single-purpose “little tools” within a Safety Element out of Context process not only permits reuse of the components themselves, but also the reuse of the analyses performed on them in the overall system safety assurance process.

5 Certification and Accreditation

Despite the payoffs in efficiency and correctness described above, even larger payoffs are achieved in the area of certification and accreditation by this approach. Guaranteeing the security and safety properties of a system made with reusable components through model based approaches is one thing, but another is getting it through the certification and accreditation process. The certification process both in terms of security and of safety is generally quite heavyweight, and the documentation load is high. Manual production of documentation, as well as manual verification of the documentation, is extremely labor intensive and error-prone, and one reason why it is difficult to justify flexibility in systems: changes carry too high an economic cost. The flexibility that comes through component composition (think again of service oriented architectures) is therefore canceled out by the high cost of certifying changing systems.

The model based approach is useful here, for automatically generating the certification and accreditation documentation. Not only does that increase efficiency in the production of the documentation (through automation) but it also increases efficiency in the validation of the documentation (through guarantees of the validity of the documentation by construction from the model).

5.1 A concrete example: security accreditation

In order to illustrate the importance of automatic certification evidence and documentation generation, we introduce the concrete example of security accreditation according to the Common Criteria. Using our model-driven approach to the correctness by construction of secure systems, we define high-level security and compliance requirements that are then correctly enforced using our process of Model Driven Security (MDS) technology, which ensures the correctness of security rules and configurations [3].

In theory, the Model Driven Security process should be enough to demonstrate (through correctness by construction) that the system is secure and permit accreditation. This would be extremely important in the context of Service Oriented Architectures, which can be reconfigured in very flexible ways. Thanks to MDS, the
low level policy configuration in the system is guaranteed to match the high level policy requirements and specification even in the face of flexible reconfigurations within the SOA. Unfortunately, this is not enough, in the current state of practice.

First of all, MDS is not able to cover all security relevant parts of the overall system, for example vulnerabilities in runtime and service implementations, e.g. buffer overflows, and it is also not able to directly cover assumptions about the context of the system, for example the runtime environment.

Additionally, accreditors still insist on the paper documentation showing the actual configuration of the system really meets the stated security requirements over the whole system life cycle. It would be impossible to conserve the advantages of automatic reconfiguration and policy assurance with manual documentation production.

Therefore, we have developed Model Driven Security Accreditation (MDSA) to automatically generate the evidence and documentation required by the accreditation scheme, including a full model driven risk analysis and consideration of assumptions. Although this constitutes to a certain degree a conceptual redundancy (effectively two sets of security requirements – one for construction, one for accreditation), in practice it is a solution that makes it possible to provide the flexibility of component-based systems (e.g. SOA) in an environment requiring the standard security accreditation schemes like Common Criteria.

6 Summary and Conclusions

Reuse in component-oriented system development is never easy to achieve, but it is particularly challenging in the presence of system-wide, emergent properties. Two specific such emergent properties are security and safety. We have presented examples of each in this paper:

- Service Oriented Architectures are modern examples of large-scale, component oriented development with the promise of reuse of entire services. They are often seen in large distributed applications that, by their nature, have security-related requirements. But the very flexibility and reconfigurability of these applications tends to “break” the stringent certification and accreditation procedures for their associated security policies, with consequences for cost-effectiveness of reuse.

- Component-oriented architectures such as AUTOSAR [6] for the auto industry strive for reuse of components within a demanding safety-related environment. Concepts such as “Safety Element out of Context” within the ISO DIS 26262 standard help to address the problem of component construction in a safety-related development process, but the integration of such components into the overall safety-related development lifecycle must still be managed in a cost-effective manner in order for the benefits of reuse to be assured.
We have presented our approaches to addressing each of these problems through a combination of the principle of separation of concerns and model-based techniques. By automating key elements of procedures (e.g. Model Driven Security Assessment) and providing full traceability over all phases of mission-critical lifecycles (e.g. Medini Analyze), it is possible to overcome the forces that work against preserving the benefits of component reuse in such demanding environments.

References

5. ISO/DIS 26262 Road vehicles -- Functional safety -- Part 1 to 10
Figure 1. *medini analyze* automotive functional safety process for ISO 26262
Figure 2. *medini analyze* toolchain for safety process support.