Model Driven Security Accreditation (MDSA)  
For Agile, Interconnected IT Landscapes

Ulrich Lang, PhD  
ObjectSecurity LLC  
530 University Ave  
Palo Alto, CA, 94301, USA  
+1-650-515-3391  
ulrich.lang@objectsecurity.com

Rudolf Schreiner  
ObjectSecurity Ltd.  
St John's Innovation Centre, Cowley Road  
Cambridge CB4 0WS, UK  
+44-1223-420252  
rudolf.schreiner@objectsecurity.com

ABSTRACT
Assurance accreditation of agile, interconnected IT landscapes is a great challenge, and is currently often cited as one of the show-stoppers for the adoption of modern IT architectures (e.g. agile, model-driven, process-led SOA and Cloud) in mission critical domains. This paper presents Model Driven Security Accreditation (MDSA), a novel approach for automating large parts of the compliance and assurance accreditation management processes (e.g. Common Criteria) to achieve reduced cost / effort, and increased reliability / traceability. MDSA is related to Model Driven Security (MDS), an approach that automatically generates fine-grained technical security rules from intuitive, generalized security policy models. MDSA automatically analyzes and documents two main compliance aspects: 1) Does the actual security match with the stated requirements? MDSA is a system and method for managing and analyzing security and information assurance requirements in reusable models, and for (mostly) automating the verification of the traceable correspondence between functional models, security models, and requirements models. 2) Do any changes impact the current accreditation? MDSA automatically identifies changes to any aspect of the “system of systems”, and evaluates whether changes impact the current accreditation and whether manual corrections and re-accreditation are required.

Categories and Subject Descriptors

General Terms
Documentation, Measurement, Security, Verification

Keywords
Model-driven security, accreditation, Common Criteria, compliance, model-driven security accreditation, model-driven compliance, model mapping, model merging.

1. INTRODUCTION
Conventional security and compliance accreditation of IT systems currently involves a human security evaluator who documents evidence and verifies whether the IT system is in line with the requirements by using different methods, for example testing or formal analysis. The security requirements for example state that all data communication has to be protected for integrity (no modification of data) and confidentiality (no disclosure of data). The human evaluator then conventionally checks manually whether the system really meets these requirements.

When model-driven, process-led software development approaches are used, a good portion of the evidence is already documented in well-defined models. This is because Model Driven Software Engineering (MDE) [20] uses standardized modeling language like UML, BPMN or SysML, or Domain Specific Languages (DSLs) to describe the parts of a software system as models, for example, of functional aspects like data formats, services, interfaces, interactions or sequences of actions. In MDE it is possible to directly generate large parts of the IT systems from the models, for example source code and configuration files. Similarly, it is possible to generate security enforcement from security models using Model Driven Security (MDS) [7], which is directed to the automatic generation of machine enforceable security rules from generic, technology-independent security policies expressed close to human thinking (i.e. which are potentially not machine enforceable). MDS applies the concepts of MDE to security. Using MDE (and assuming that the model transformations work as intended), there is a high probability that IT systems match the functional models. Similarly, using MDS, there is a high probability that IT systems security matches the security models (“traceability”).

It is also possible to analyze to models in order to evaluate that the software meets specific requirements. This is often critical, because information security is not only about implementing security across IT systems and applications according to security requirements. It is often also necessary to demonstrate the level of confidence that IT system security complies with the stated security requirements. This is called “compliance” or “information assurance” (and involves “evaluation” and “accreditation” processes) and can involve many factors, for example operational environments or business domains.

Civilian and government compliance examples include best practices, laws and regulations, e.g. ITILv3 / ISO2700x, ISMS, COBIT for security management; privacy legislation, healthcare legislation such as HIPAA, payment card processing such as PCI, safety standards and regulations such as ISO 26262, DO 178B or...
EN50128 for safety, accounting / auditing regulations such as Sarbanes-Oxley.

More rigorous government information assurance examples include the “Common Criteria” (CC) [1] standard ISO/IEC 15408, a framework in which computer system users can specify their security requirements, vendors can then implement and / or make claims about the security attributes of their products, and testing laboratories can evaluate the products to determine if they actually meet the claims. In other words, CC [1] provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous and standard compliant manner. Common Criteria evaluations are performed by “accredited” human evaluators and organizations. Evaluations are performed on computer security products and systems called “Target Of Evaluation” (TOE). The “evaluation” serves to verify claims made about the target's security features.

This is commonly done through a manual process that involves a number of documents: (1) A “Protection Profile” (PP) identifies security requirements for a class of security devices; (2) A “Security Target” (ST) identifies the security properties of the TOE. It may refer to one or more PP's. The TOE is evaluated against the SFRs and SARs (see below) established in its ST, no more and no less. This allows vendors to tailor the evaluation to accurately match the intended capabilities of their product. (3) “Security Functional Requirements” (SFRs) specify individual security functions which may be required by a product, e.g. authentication or access control. (4) “Security Assurance Requirements” (SARs) - descriptions of the measures taken during development and evaluation of the product to assure that the claimed security functionality works as intended. The evaluation process tries to establish the level of confidence that may be placed in the product's security features through quality assurance processes; (5) “Evaluation Assurance Level” (EAL) - the numerical rating EAL1-EAL7 describing the depth and rigor of an evaluation. Higher EALs do not necessarily imply "better security", they only mean that the claimed security functionality of the TOE has been more extensively validated.

2. CHALLENGE: AGILITY & REUSE

Assurance accreditation of agile, interconnected IT landscapes is a great challenge, and is currently often cited [5, 2] as one of the show-stoppers for the adoption of modern IT architectures (e.g. SOA) in mission critical domains such as defense. The problem is that the Common Criteria methodology was originally developed to accredit the assurance of closed and static systems which may be reconfigured within certain and well defined limits, but do not evolve dynamically. It was cheaper and easier to avoid agile changes beyond what was originally provisioned for, thus hindering progress and incremental adoption, evolution, improvement, and optimization.

However, today’s IT architectures support agile interconnected networked (“distributed”) applications to meet changing business demands and evolve over the whole system life time. Example architectures include Service Oriented Architecture (SOA), Web 2.0 and mash-ups, Cloud Computing, SaaS / PaaS, Grid Computing. Examples for agile application development include model-driven, process-led software development and integration (e.g. Model Driven Architecture (MDA), Model Driven Software Engineering (MDE), and executable Business Process Management (BPM)). An example of an agile application aspect is application interactions, such as SOA service interactions and BPM interactions.

Security measures are a critical enabler for such IT architectures and also need to support agility. There are many efforts to use model-driven approaches for non-functional system aspects to improve for example (1) the safety and security of systems, e.g. using the abovementioned Model Driven Security (MDS) for agile security policy management, (2) the assessment of risks, (3) evaluation and accreditation, and (4) compliance with legal and regulatory requirements.

Conventional compliance / security / accreditation methods often fail for such dynamically changing (“agile”) IT systems. This is because it is necessary to document that the changing system always meets the defined security and compliance requirements. Changes may impact the security properties of the system in such a way that the system does not comply with the required level of compliance / assurance anymore. In such a case it is necessary to “re-accredit”, i.e. to analyze (i.e. “re-evaluate”) the impact of the changes and potentially mandate corrections to the IT systems. In addition, new vulnerabilities discovered in the system may need to be taken into account. This is not a modification of the system, but a change to what it is known about the system. These newly discovered vulnerabilities might have an impact on the accreditation or not. Today, all this is a time-consuming, manual process which is not sufficiently fast and cheap to support the agility of today’s IT systems. It includes documenting and processing requirements (using informal text which hinders automation); collecting and documenting evidence; analyzing the evidence to identify whether the IT system and its IT security meet the documented requirements; and manual corrections.

Unlike MDSA, the conventional manual evaluation process does not normally tie into automated, verifiable processes such as Model Driven Software Engineering (MDE) and / or Model Driven Security (MDS). And even if a model-driven approach to development and security is used, compliance related information is conventionally simply tagged to the related model elements in the functional model itself. This has several disadvantages: Compliance information is not described in a generic, application / platform independent reusable way. Instead it has to be described as a single, large model at the low abstraction layer of the functional system model and bound to a specific application, and it is not possible to easily associate specific sets of model elements (e.g. “All information flow”) with a given compliance element (e.g. “All information flow over public network has to be protected”), or associate / aggregate model elements describing accreditation related information.

3. SOLUTION: MDSA

Assurance accreditation needs to be improved to meet those challenges. Through automation and the use of model-driven approaches as described below, the Model Driven Security Accreditation (MDSA) [6] approach and software outlined in this paper enables the cost-effective, low-effort, and reliable / traceable accreditation of agile, interconnected IT landscapes based on model-driven, process-led approaches. The MDSA toolset (together with usage processes and methodologies) could be deployed as part of an integrated high-assurance development and operation tool-suite, technology stack, and methodology (e.g. for SOA).
MDSA is based on a combination of several related concepts, incl.: model-driven, process-led software development, model-driven security, model-driven compliance; automatic documentation of supporting evidence; automatic change detection & change acceptance; and manual re-accreditation decision support (all described in detail below).

MDSA automatically analyses and documents two main aspects:

- Does the actual security match with the stated requirements? MDSA is a system and method for managing and analyzing security and information assurance requirements in reusable models. In particular, MDSA (mostly) automates the verification of the traceable correspondence between the functional models, the security models, and the requirements models, whereby the correspondence indicates that compliance / security / accreditation requirements defined in the requirement model match with security objectives implemented by controls defined by the security implementation model (i.e. the enforced technical security rules). MDSA can also check some consistency aspects of the requirements model, e.g. detect some requirements conflicts.

- Do any changes impact the current accreditation? MDSA automatically identifies changes to any aspect of the “system of systems” (i.e. functional, security, or information assurance requirements across multiple layers), and evaluates whether changes impact the current accreditation, and whether manual corrections and re-accreditation are required. This also includes the analysis of impacts of security vulnerabilities discovered during the life cycle of the system.

Model Driven Security Accreditation (MDSA) can be defined as follows: MDSA enables “agile accreditation” in a way that is cost-effective, low-effort (i.e. partly automated), and reliable / traceable. MDSA especially enables agile accreditation for agile, interconnected IT landscapes based on model-driven, process-led application development and deployment approaches, and on standard middleware and runtime platforms (e.g. SOA). MDSA allows the automated, formalized assignment of “undistorted” Common Criteria assurance requirements to IT landscape specific technical assurance control objectives in functional system specifications. Both are expressed as formalized models and are automatically and traceably matched. Using model-driven security (MDS), the technical assurance control requirements are then automatically transformed into concrete technical IT controls enforcement & monitoring at runtime. In addition, the traceable correspondence between technical security implementation and the information assurance requirements is analyzed and checked. MDSA also documents Common Criteria “supporting evidence” based on all available design-time system / security models, system / security artifacts, system / security model transformations, and runtime system / security incident logs. Furthermore, MDSA enables the automated analysis whether changes to or newly discovered knowledge about an agile IT landscape impact its security properties, and whether the accreditation is still valid. The goal of MDSA is to automatically check whether IT systems security meets its assurance accreditation requirements, and to check the impact of changes (incl. system, security, requirements, and newly discovered vulnerabilities) on the accreditation. Based on so-called “change policies”, MDSA decides whether particular system re-configurations are within scope of the current accreditation (thus enabling a level of IT agility) or whether manual corrections and re-accreditation are required. MDSA also allows to assess the impact of newly discovered security vulnerabilities, e.g. weaknesses in crypto algorithms or buffer overflows in libraries, on one system or multiple systems as part of an Accreditation Management System (AMS), a central database of fine grained accreditation information. If manual re-accreditation is required, MDSA also acts as a decision support tool.

The following flowchart and component diagram illustrate the main steps of the MDSA architecture:

![MDSA Architecture Flowchart](image)

The MDSA architecture includes one or more the following parts (implemented as software tools):

- **Requirements Definition**, which allows the modeling of the security requirements and risks of a system in platform and middleware independent Domain Specific Languages
- **Risks Modeling**, which allows the modeling of risks, vulnerabilities and attack trees in platform and middleware independent Domain Specific Languages
- **Requirements Model Merging & Mapping**, which automatically, semi-automatically or manually maps “undistorted” (i.e. system-independent, generalized) models of accreditation / risk requirements into concrete technical assurance controls for the particular IT landscape, using MDSA’s so-called “model merging & mapping” approach
MDSA’s Model Driven Compliance (MDC) feature, the first part of MDSA (top half of Fig. 1) automatically checks that the security of the IT systems matches the stated requirements. These stated requirements can often be predetermined. MDSA starts when a human user or a machine initiates the execution. Next, MDSA reads a number of models:

- one or more functional models, which describe the functional parts of the IT systems (e.g., services, components, applications, host machines) and how they relate (e.g., SOA service interactions, BPM workflow interactions). The models are defined using meta-models to allow automated processing. The meta-models of the models used are also read together with the models (for the sake of simplicity, this is not shown in the figures). Functional models should not contain any security or requirements model elements, because of the disadvantages outlined in the introduction and background section. However, it is possible (but not recommended) to also include security and requirements model elements in the functional models.

- one or more security models that describes the security relevant parts of the IT systems. Such security models should be defined in a generic (i.e. independent of the specifics of other models such as the functional model) fashion, although definitions are possible (but not recommended) that directly relate to the particular functional model(s).

- one or more mapper model(s) that describe the relation between the elements of the functional model(s), the security model(s), and the requirements model(s). The mapper models enable users to define generalized security model(s) and requirement model(s) independently of the particular functional model(s). The mapper models link model elements across several models and meta models and thus enable model re-use, flexibility, technology-independence, and abstraction layer independence.

- one or more requirements models, which define the compliance, accreditation and security requirements (again, technology-independent and generalized).

MDSA then analyzes and checks that the IT system security implementation traceably corresponds to the stated requirements. Taking into account the mappings defined in the mapper model(s) (and the functional, security, and requirements models) MDSA relates as many model elements as possible across all read models. The sum of all related model elements across all read models forms a merged model (spread across several models) that contains all the information and all correspondences between model elements. MDSA then analyzes the correspondence of the functional model(s) and security implementation model(s) elements with the requirement models. MDSA decides according to the results of this correspondence analysis whether the results identify the correspondence or if the results identify that there is no correspondence. For example, if the compliance model states that communication between two services has to be protected for confidentiality and integrity it is checked whether the communication is sufficiently protected, e.g. that encryption and a secure hash are applied. MDSA also documents the results of this analysis as evidence, and makes it available to other systems or to a human user.

MDSA can also provide the generated and documented results of the correspondence analysis to e.g. a decision support tool that helps human accreditors to correct non-correspondence between requirements and actual security.

MDSA’s trust in the accreditation partly stems from the fact that in model-driven, process-led software development approaches there is a high degree of confidence that the operational software traceably matches with the functional models, because the running software has been automatically generated from the functional models (using Model Driven Engineering, MDE). In the same way, there is also a high degree of confidence that the security implementation traceably matches with the security implementation models (because MDS is used) and the
requirements (because MDSA’s MDC is used). Therefore, the input models have a very high probability to match with the actual implementation and operation of the running IT systems, especially when application are not just developed, but also deployed in a controlled fashion as part of MDE (e.g. using orchestration approaches like BPEL or deployment models like in the CORBA Components Model).

The other main part of MDSA (bottom half of Fig. 1), the evidence collection and change detection / analysis, has the following purposes:

- automatically document comprehensive evidence about the functionality of the IT systems, about their security implementation, and about the security requirements;
- automatically detect changes and their implications (including consequential implications) to the IT systems and their security across all layers, including the model layers, the model transformations layer, and the IT systems / security layer (see Fig. 2), and document those changes; and
- evaluate based on “change policies” whether a change voids the current compliance / assurance certification.

To achieve this, MDSA collects various information (referred to as “evidence”) about the current deployment and operation of the information technology (IT) systems and about their security and accreditation across all layers, including the model (functional and security / compliance) layers, the model transformations layer, and the IT systems / security layer. Such information includes e.g. system design; security policy model; detailed functional description; formal security requirements; how security requirements are met; compliance / accreditation requirements (incl. vulnerabilities, threats, threat agents, controls, control objectives); model transformations; generated low-level security rules and the IT infrastructure; runtime events; development life-cycle (e.g. software development guidelines document); and information about newly discovered vulnerabilities.

MDSA processes all the collected evidence and restructures it into a form that can be compared to previously collected evidence, i.e. in a form that is consistent, repeatable, and without repetition of information. Such a form can be referred to as “normalized” evidence. Evidence is normalized if it is exactly in the same syntax and order, even if the evidence has been collected in a different order or with repetition or in a different syntax or by a different collection method. Normalization can be achieved in different ways, including well-known mapping and sorting techniques.

MDSA then reads previously collected, normalized and stored evidence versions. The stored evidence reader could for example read the previous evidence (i.e. the evidence collected before the current evidence) for the purpose of comparing it with the current evidence to detect current changes. In other examples, the stored evidence reader could also read any other previously collected, normalized and stored evidence, for example to analyze the impact of changes over time. The stored evidence reader could also read several versions of previously collected, normalized and stored evidence, for example, to analyze changes that happened at a particular point in the past. MDSA then identifies the differences between the different read normalized evidence versions, and stores evidence that contains all identified changes in a normalized form, as well as information about explicitly flagged changes.

If MDSA detected changes (including indirect consequential changes), MDSA reads a “change policy” which includes one or more change policy rules that define which changes do or do not impact the compliance / assurance level. It uses generic modeling and meta-modeling methods and concepts to support the specification of flexible/extendable change policies. MDSA then evaluates the change policy rules for the detected changes. For example, if a normalized change evidence element indicates a change of an interaction between two networked applications such as SOA services, the change policy evaluator will search for a particular interaction related change policy rule. For example, a rule could state that interaction changes do not impact the compliance, security or assurance level if the security level on both sides of a Multi-Level Security (MLS) controlled interaction remain the same (e.g. “secret” interaction of one “secret” application with another “secret” application). MDSA also stores the evaluation results, including, the changes, the particular rules applied, and the results. If MDSA detected that change policies were violated, it provides the stored evaluation results to a human user, e.g. within a decision support tool that helps the human user to manually carry out necessary corrections and re-accredit the compliance / assurance level of the IT system.

It is important to also mention that the described MDSA is applicable to all model-driven approaches in general, because model-driven approaches inherently provide the means to produce the required traceable evidence about models and system artifacts, and consequently the trust in the accreditation. Also, the concepts of MDSA can be applied to achieving both civilian compliance (e.g. government, critical infrastructure, healthcare, finance, utilities etc.) and government accreditation.

4. MDSA CONCEPTS

The Model Driven Security Accreditation (MDSA) approach outlined in this document is an innovative, patent-pending combination of several related pre-existing and novel concepts approaches (rather than one individual concept) – which enable agile accreditation because of the automation achieved.

Pre-existing concepts: Some of the concepts used for MDSA are pre-existing, i.e. have been developed independently of the accreditation challenge to solve other challenges: Model-driven, process led software development, and model-driven security:

- Model-driven, process-led software development and integration approaches, including Model Driven Architecture (MDA), Model Driven Development (MDD), and executable Business Process Management (BPM). MDD (and also “Model Driven Integration, MDI) and BPM (for SOA orchestration) are already being adopted and are forecasted to be mainstream by 2011-2012, driven by large vendor pushes [19] (e.g. Microsoft, SAP, IBM, Oracle). This is highly relevant to agile, interconnected IT landscapes, such as Service Oriented Architecture (SOA), or Cloud / SaaS / PaaS / Web 2.0 architectures.

Model Driven Security (MDS) applies the reasoning behind MDA to security policy & compliance management. It is realistic to forecast MDS adoption to “piggyback” on the adoption of model-driven, process-led approaches (e.g. as a product add-on such as ObjectSecurity’s OpenPMF). MDS makes agile policy management manageable in model-driven, process-led
environments. MDS also makes it possible - among other benefits – to automate the adjustment of security policies whenever the IT environment gets modified or reconfigured, which reduces the administration overhead to a minimum and improves assurance by minimizing human errors. Model-driven security (MDS) is a critical component of future Information Assurance (IA) architectures, esp. for agile IT environments such as SOA. It primarily tackles the problems “where do the fine-grained, contextual security policy rules come from, and how do they match with business intent”. MDS can be defined [7] as the tool supported process of modeling security requirements at a high level of abstraction, and using other information sources available about the system, for example the applications functional models (produced by other stakeholders). These inputs, which are expressed in Domain Specific Languages (DSL) or using generic modeling languages (e.g. UML) and frameworks (e.g. MODAF [17], DoDAF [18]), are then analyzed to automatically generate enforceable security rules with as little human intervention as possible. These rules are then enforced across the entire IT environment (e.g. through local enforcement points integrated into the middleware or at a domain boundary) or in the security infrastructure, for example at firewalls. Conflicts, for example when several different policies form the input into model driven security, can be detected by sorting and comparing rules. The local enforcement points also deal with the monitoring of security compliance relevant events. Model-driven security also includes the run-time security enforcement of the policy on the protected IT systems, dynamic policy updates and the integrated monitoring of policy violations.

**Novel concepts:** The proposed MDSA approach also introduces a number of novel (i.e. independently developed by the authors) concepts:

- Domain Specific Languages for accreditation criteria and risk models
- Formalized models of accreditation criteria
- Semi-automatic and tool assisted rapid requirements-to-controls-mapping using systems and accreditation models,
- Graphical User Interfaces (GUIs) for model merging
- Automatic and tool assisted system and assurance analysis and evaluation,
- Automatic generation and execution of system assurance tests,
- Automatic merging of test results into accreditation documentation,
- Automatic generation of supporting evidence documentation,
- Change detection,
- Re-accreditation analysis based on change policies,
- Model-driven security policy generation,
- Runtime control enforcement & monitoring,
- Manual re-accreditation decision support,
- Model-driven / process-led development,
- Agile / interconnected IT landscapes.

MDSA’s first part, the Model Driven Compliance (MDC), is based on the same model-driven approach as MDS, but applied to mapping risk-management objectives and controls management objectives to concrete technical controls for the particular IT landscape for which accreditation (or compliance) need to be achieved. This includes (1) the formalization (in models) of the different artifacts involved (such as assets / vulnerabilities / risks / control objectives / controls assignment) and the relation between the different aspects and the system specification. It also involves (2) an automatic or semi-automatic process to map control objectives to concrete controls for each system artifact and to resolve conflicts (by exploiting known interdependencies). And (3) the automatic checking that the controls traceably correspond to the stated requirements. MDC is based on a correspondence between the elements of:

- Risk models (describing for example assets, vulnerabilities and control objectives) or compliance models (describing for example assets and compliance regulations/modules to apply) on one hand, and
- Design/runtime models (describing functional artifacts and control implementations) on the hand side.

The second part of MDSA deals with automatic documentation and change detection / analysis: Thanks to the usage of model-driven, process-led development and model-driven security, most of the high-level / low-level system and security specifications are available in a formalized, normalized form that can automatically be turned into supporting evidence and system documentation. Furthermore, the model transformations for development and security enable the evidence documentation that there is a traceable correspondence between the high-level and low-level specifications and software artifacts. Run-time evidence (especially security incident monitoring) is also readily available and useful for ongoing analysis and improvement / optimization. The formalized evidence allows the detection of changes and their concrete impact across the IT landscape. Change policies, which express which changes do not alter the accreditation / compliance targets, can then be applied to automatically evaluate & accept whether changes impact the current accreditation (thus enabling agility).

Evidence about the detected change and the identified crossing of change policy boundaries, as well as the other evidence mentioned above, accreditors can be presented in a consolidated manner to human accreditors through a decision support tool. Such a tool can also act as an expert system and provide concrete contextual guidance for the particular situation identified. It includes both intentional changes and the analysis of the impact of newly discovered knowledge about the system, e.g. vulnerabilities in libraries, services or components.

**4.1 Additional considerations**

1) **Cost-Benefits Justification:** Model-driven, process-led approaches are sometimes still criticized because of the perceived extra effort they require to specify models. This criticism is generally debatable because good software needs sound architectural design. The criticism is specifically not valid for IT landscapes that need to be accredited, because accreditation requires structured system documentation anyway, and software modeling is currently accepted best practice to specify system
functionality. In fact conceptually, modeling systems does not actually add to the total cost of policy management. This is because if security administrators have to manually specify detailed technical security rules, they are effectively also specifying the security related aspects of the system specification within their policy administration tool. Model-driven security simply re-uses this information (which often make up the greater part of security policy rules) from models specified by other stakeholders (and / or tools) who know applications and workflows better anyway (i.e. application developers / integrators, and process modelers). This argument shows that, even after only a short while in operation, the total cost of MDS is very likely to be significantly lower than traditional policy management without MDS. In addition, the quality of protection is also improved, reducing the risk of security incidents, and also their damage and the related costs for response. The same argument applies to MDSA. SOA and related modern IT architectural approaches are also sometimes criticized because of the added cost and the added software complexity they introduce. However, that up-front investment needs to be compared to the cost of not adopting modern IT architectural approaches, such as (1) maintenance cost / effort explosion, (2) integration hurdles / costs, (3) costs because re-use is not possible, (4) lost business opportunities because the lack of IT agility prevents the offering improved services to the business side etc., and (5) investments are siloed and stove-piped, and thus hard to “future-proof”.

2) Blurred Boundary – Design Time vs. Runtime: Traditional accreditation is done at the system lifecycle stages prior to operational runtime. The primary purpose in general is to approve a system’s assurance for its intended operational use. The implied assumption is that systems do not change after accreditation, and if they do, manual re-accreditation is necessary. However, today’s agile, interconnected IT landscapes blur that boundary between design time and runtime: IT landscapes dynamically evolve, i.e. dynamic system changes can occur frequently during runtime (e.g. BPM workflow re-orchestration). This blurs the clear “waterfall” process and the boundary between design-time and runtime which traditional Common Criteria assumes. MDSA therefore needs to collect and analyze both design time (i.e. about the system, the security policy, the accreditation requirements, the “traceability” of the inner workings of the model transformations) and ongoing at runtime (i.e. evidence about runtime activity of the IT landscape). Runtime analysis is typically not part of traditional accreditation (e.g. Common Criteria).

3) Need for Risk- and Control-Based Approaches: A pure risk-based approach or a pure control-based approach to accreditation (and civilian compliance) is not enough [9]. A combination of both approaches, and a traceable mapping between risk-based and control-based approaches is needed, i.e. the identification of risks, and its (automatic) mapping to controls. MDSA’s MDC therefore partly acts as a bridge between a risk-based approach and MDS, which is currently purely control-based (i.e. concerned with how to concretely implement high-level control objectives across the IT landscape).

4) “Traceable Correspondence”: Because MDSA’s evidence documentation & accreditation support is based on models used to build / deploy IT landscapes and security policies, MDSA inherently ensures to a high degree of confidence that the accreditation traceably corresponds to the actual IT landscape (i.e. ToE / ST documentation inherently and traceably matches with reality). MDSA also takes runtime incident monitoring into account, which enables continuous accreditation compliance monitoring.

5) Separation of Concerns: A particular feature of MDSA is that it allows the separation of concerns between involved stakeholders. In particular, thanks to its model-merging concept, it allows for example unclassified, general-purpose Common Criteria “change policy” modeling and MDSA toolset development and the accreditation of platforms and unclassified services and components, while the application of the generic models and tools to specific (classified) systems during accreditation is done by cleared personnel. Both views can be linked using the abovementioned model mapping and merging technique. The general concept of using modeling techniques for a security analysis of systems is not a new approach, it was for example proposed by Jürjens in SecUML, where UML annotations were used to add security related information to UML models [4]. Unlike MDSA (and MDS), such other approaches have the disadvantage that they directly couple the security information to the functional model of the application and also to the meta model describing it. In addition, such approaches allowed to add security related information to the individual elements of the functional model, but did not allow expressing relationships between security annotations or to cluster elements with the same security attributes. An obvious solution would be an extension of the functional meta model. Unfortunately, this raises many issues. For example, in contradiction to the concept of separation of concerns it again couples functional and non functional aspects in a single model, it makes already very complex functional meta models like UML even more complex, and it does not allow reuse of security information in different meta models. To solve this, the authors’ MDSA (and MDS) approach separates functional and security models, both described in their own, adequate meta models.

5. MDSA PROTOTYPE

MDSA is currently at the concept exploration phase (i.e. patented, and prototypes built). The goal of the current prototype was the practical evaluation of the overall concepts of MDSA in a simplified, but still realistic environment, with special focus on the integration of MDSA into MDD and MDS. The authors extended an existing MDS/MDD tool chain by an additional accreditation model describing vulnerabilities, threats and required security functionality. From this new model and the already existing models of the MDD/MDS tool chain, the functional model and the compliance model defining the high level security, we are able to generate the security configuration and, in parallel, the related accreditation evidence, e.g. whether the high level security policy is in accordance to the required security functionality. An analysis of changes is not yet part of the prototype, but experiences from another projects related to safety demonstrated that the approach of change policies is feasible.

As the authors have learned in the past, many solutions in different domains work very well for demo applications with the complexity of HelloWorld. Unfortunately, the evaluation of a technology based on such trivial demo applications says little about the technology’s suitability for solving the complex problems of real world applications. Therefore, a non trivial demo was chosen for the MDSA prototype, namely ObjectSecurity’s
SimulateWorld / SWIM system, the prototype of a secure System Wide Information Management (SWIM) system based on a distributed, component based simulation of air and ground traffic around San Francisco airport.

The SimulateWorld / SWIM system was initially developed for the evaluation of security systems in complex, heterogeneous environment [8], and uses different communications paradigms like request / reply and information flow and also different middleware technologies:

![Figure 2: Part of the SWIM Prototype Model](image)

The SimulateWorld / SWIM prototype consists of multiple components implementing specific simulators (aircrafts, ground vehicles, ships), a central simulation infrastructure to establish a shared “world” and the SWIM application on top of it. The complete simulation infrastructure is implemented in C++ using the SecureMiddleware CORBA Components Model implementation as runtime platform with Qdeo [14] as CCM implementation, MICO [10] as underlying ORB and the ObjectWall [11] IIOP proxy for domain boundary traversal. On top of the simulation infrastructure, our SWIM prototype is located. It consists of components for information exchange and management, Controller Working Positions and other displays. This SWIM prototype is implemented mostly in C++ and Java, using different middleware technologies, for example the OMG Data Distribution Service (DDS), a Java Messaging Service (JMS) implementation and the Advanced Message Queuing Protocol (AMQP) for information flow, and Web Services / EJB (implemented with Glassfish). The Glassfish Java application server also implements data persistency and also provides a web interface to some information, for example airport flight data.

For the protection of most parts of the system ObjectSecurity’s OpenPMF [12] policy management framework is used. It allows the central definition and management of security policies in the distributed systems, and also the central monitoring of policy violations.

For the development of the system, which mainly involved the generation of software artifacts and of the security policies, the authors used Model Driven Development (MDD) and Model Driven Security. The MDD / MDS / MDSA tool chain is implemented using the Eclipse Modeling Framework (EMF) and OpenArchitectureWare (OAW). We use the.ecore meta model to describe all meta models and proprietary Domain Specific Languages for modeling the system. Currently, the models are edited using the standard EMF reflective editor. For the model transformations (M2T) we used the Xtext language of the OAW framework.

During the implementation and evaluation of the MDSA prototype the authors had two main objectives: Firstly check whether it is possible to model compliance and security requirements, and to check whether they are met in the system of the full life cycle. And secondly, check whether it is possible to do fine grained security evaluations using model driven techniques. An additional goal was to keep the prototype as independent as possible from the functional meta models and the target application, in order to allow the platform independent definition of vulnerabilities, risks, and compliance and security requirements.

Separation of Models and Concerns: To achieve the abovementioned model separation, the authors developed an ElementMapper. The ElementMapper supports the flexible mapping of elements of different models described in different meta models at the same or at different layers of abstraction. This decouples the different models, allows the “translation” of terms and descriptors and supports clustering of elements. It allows the description of different views of the system at the right level of abstraction and using the right term for specific aspects and views. Therefore, the ElementMapper allows the definition of the optimal vocabulary for the description of the system for a given task. It also allows adding attributes to the model elements of the mapper model and the definition of relationships between the elements. The mapping algorithm supports one to one, one to many or many to one mappings and is based on flexible, declarative descriptors supporting logical operations of arbitrary model elements and meta model elements. The mapping is also bi-directional, allowing a bi-directional navigation between models.

In the prototype, the authors defined the mapping mainly based on the meta types of the functional model. Therefore, all elements of the same meta type are treated in the same way. It is also possible to choose other model elements like names or attributes as well, and also logical combinations of them. This allows the definition of mappings for individual elements, for example specific services, or to cluster elements. It is used in the Attack Tree Analysis (ATA) model, where we define a mapping based on component instances. This allows to express security, compliance, accreditation or risk related information for specific instances of the system. In our prototype we define a simple attack tree model for the “Radar” component instance.

Security and Compliance Models and Evidence Generation: The Security Model describes the high level security policy of the system. It is used for the generation of the low level security rules and configurations. The Compliance Model defines the Assets, Actors and Actions in the system and also the ComplianceModules they have to be compliant to in a generic, platform independent way. A ComplianceModule, for example InformationFlowIntegrity or ServiceImplementationProtection, consists of one or more Controls like Authentication, transport layer encryption or OpenPMF AccessControl and the security
of trust in the general approach of MDS/MDSA can be established. At the first glance, this seems to be difficult, esp. because the transformations depend on the functional meta model used and some parts of the transformations, e.g. the model merger, are somewhat complex and hard to understand. The more complex the functional model, the more complex and harder to understand the transformation. This especially plays an important role for very complex meta models like MODAF. In the current prototype, the transformations were manually written, which required a lot of experience, care and sound testing to establish trust in the correctness of the transformations and their results.

Vulnerabilities can only be exploited under specific conditions. For example, a ThreatAgent without access to a service cannot exploit a buffer overflow in the implementation of this service. Vulnerabilities are countered by one or more Controls. Controls are equivalent to the CC Security Functional Components (SFC) including for example authentication, message protection or audit functionality. The definition of the controls describes the functionality and which security objective this functionality enforces. For the sake of simplicity, we used a simple, flat structure of controls without classes and families. In addition, it is also possible to make assumptions about the system and its environment, e.g. to state that specific vulnerabilities are not relevant.

The MDSA model transformations now compare, using fully automatic matching, the required and the provided security functionality under consideration of specific assumptions, in order to check whether the security objectives are met. From this matching process, evidence and documentation are generated, similar to large parts of a Common Criteria evaluation report.

Risk Models and Risk Analysis: Risk models are used to model risk related information about a system, for example for Attack Tree Analysis. This includes both information about assets, threat agents, vulnerabilities, security objectives, controls and their functionality and the objectives they are able to enforce, and attack trees. The related models are defined for the runtime platform, e.g. the middleware, and the application on top of it, for example attack trees to break the access control or the message protection of the system.

From the functional, compliance and attack tree models we were able to generate fine grained risk analysis documents describing the risk of the system in the actual configuration. It also was possible to analyze the impact of newly discovered knowledge about the system, e.g. about new security vulnerabilities.

The attack tree models were very complex, but it was possible to reuse the middleware related parts for different applications based on the same middleware. This greatly simplified the attack tree analysis process, because today's applications strongly depend on underlying runtime platforms and their security mechanisms. Only a small part of the code (expressed in Lines Of Code, LoC) and therefore also of the possible attacks is application specific.

Automatic Generation of Model Transformations: A key part of MDS/MDSA are the model transformations that implement the generation of security policies and accreditation evidence. Only if it is possible to prove that the transformations are correct and that the transformation generating the policy behaves exactly like the generation generating the accreditation evidence, a sufficient level of trust in the general approach of MDS/MDSA can be
6. RELATED WORK

There is some related publicly available scientific research where model-driven security is applied to accreditation [3, 15]. Thanks to its integration with commercially available, well-established model-driven approaches, it is possible that the proposed MDSA approach is closer towards an implementable, practical solution than much of the previous scientific work. Furthermore it is possible that classified work on model-driven security accreditation approaches exist, but the authors are not aware of any. No products seem to be publicly available today to support agile accreditation in the proposed way for agile, interconnected IT landscapes such as SOA.

7. CONCLUSION

Assurance accreditation of agile, interconnected IT landscapes is a great challenge. This paper presented Model Driven Security Accreditation (MDSA), a novel approach for automating large parts of the compliance and assurance accreditation management processes, e.g. for Common Criteria. MDSA uses model-driven approaches to automatically analyze and documents two main aspects related to compliance and information assurance accreditation: 1) Does the actual security match with the stated requirements? 2) Do any changes impact the current accreditation? MDSA enables the cost-effective, low-effort, and reliable / traceable accreditation of agile, interconnected IT landscapes with applications built, operated, and secured using model-driven, process-led approaches (Model Driven Software Engineering, MDE, and Model Driven Security, MDS). The presented prototype implementation shows that model-driven approaches as used in MDSA can automate a large part of the conventionally manual evaluation and accreditation process (e.g. Common Criteria). The prototype shows that the MDSA automation approach saves human effort and supports dynamic changes to the (model-driven, process-led) IT landscape.

8. ACKNOWLEDGMENTS

Our thanks to UK Ministry of Defence (MoD) Centre for Defence Enterprise (CDE) for funding ObjectSecurity for the development of a MDSA concept exploration document deliverable. Thanks also to David Chizmadia and John Mullen from Promia, Inc who provided useful suggestions early on during the concept exploration; and to Prof. Dieter Gollmann (TU Hamburg-Harburg) for reviewing a late draft of this paper.

9. REFERENCES