

SecDSVL: A Domain-Specific Visual Language To Support Enterprise Security Modelling

Mohamed Almorsy and John Grundy
Centre for Computing and Engineering Software and Systems
Swinburne University of Technology
Melbourne, Australia
[malmorsy, jgrundy]@swin.edu.au

Abstract-Enterprise security management requires capturing different security and IT systems' details, analyzing and enforcing these security details, and improving employed security to meet new risks. Adopting structured models greatly helps in simplifying and organizing security specification and enforcement processes. However, existing security models are generally limited to specific security details and do not deliver a comprehensive security model. They also often do not have user-friendly notations, being complicated extensions of existing modeling languages (such as UML). In this paper, we introduce a comprehensive Security Domain Specific Visual Language (SecDSVL), which enables capturing of key security details to support enterprise systems security management process. We discuss our SecDSVL, tool support and the model-based enterprise security management approach it supports, give a usage example, and present evaluation experiments of SecDSVL.

Keywords-component: Domain Specific Visual Language; visual modelling tools, model-based security management

I. INTRODUCTION

Managing enterprise security involves capturing, enforcing, monitoring and improving deployed applications' security [1]. The wide-adoption of complex IT systems and new hosting paradigms, such as cloud computing and SOA platforms that rely on outsourcing enterprise assets, increase the complication of the enterprise security management process. Ordinary enterprise security management efforts suffer from tools that provide only limited help in capturing, modeling, and enforcing security requirements, difficulty in maintaining consistency of security specifications across different IT systems, and the need for repeated security updates. In this paper, we focus on how to help security experts in modeling details of their enterprise security management process by the use of a novel domain-specific visual language - SecDSVL.

Many security modeling efforts do exist, including secureUML [2], UMLsec [3], KAOS [4], and i* [5]. Most focus on either early stage security requirements engineering (security requirements elicitation) or design time requirements modeling (mapping security requirements to system entities such as components, classes, methods,). Efforts focusing on comprehensive security engineering – from security goals down to security functions – do exist [6]. However, they neither support enterprise security architecture details nor integration of systems with operational environment security.

Moreover, all of these efforts result in models tightly coupled to system models (UML models), making it hard to deliver consistent updates and sharing of models among different enterprise deployed software systems. Current efforts in security modeling thus lack a comprehensive approach for capturing the wide variety of security details existing in a given enterprise. These include security objectives, risks and threats, requirements, security architecture, and security controls and mechanisms. Few support traceability between security properties i.e. traceability between security objectives to requirements and requirements to security controls.

To address this issue we introduce a new, comprehensive security domain-specific visual language, SecDSVL. SecDSVL helps in capturing the wide variety of security details that arise during the enterprise security management process and maintains traceability between differing levels of abstraction of these security details. This helps security engineers in reasoning about security requirements and controls completeness and conformance – i.e. to make sure that security objectives have been iteratively and incrementally refined to their realization security controls. Moreover, it helps in change impact analysis. Thus given a modification on an abstract entity e.g. security objectives, requirements, architecture, etc., we highlight other security entities impacted by such a change. SecDSVL is used to capture enterprise IT systems' security details and UML models are used to capture IT systems' details. Enterprise security engineers then work with these two models to map security onto target systems' entities. Finally, the specified security is realized using an automated security controls' configuration and integrated with enterprise IT systems using system interceptors and dynamic Aspect-Oriented Programming (AOP) that enables injecting code at arbitrary system entities at runtime [7]. Thus, one integrated enterprise security model, SecDSVL, captures all enterprise security details and these are mapped onto different IT systems at different granularity levels. Updates to SecDSVL specifications are dynamically reflected on running systems.

Section II presents a motivating example for our research, identifies key research challenges, and reviews key efforts in security management and engineering. Section III describes our general model-based security management approach that we use in modeling and realizing enterprise security. Section IV introduces SecDSVL as a part of our approach that helps

in modeling enterprise security. Section V presents a usage example. Section VI describes SecDSVL evaluation details. Section VII discusses key strengths and weaknesses and areas for further research.

II. MOTIVATING EXAMPLE

Consider the scenario where an organization, Swinburne University, is deploying different IT systems including: (i) an HR system to manage staff and student details and recruitment – called “Galactic system”. Galactic also supports student management; (ii) a Timetable Management System organizes labs, classes and lectures – called “Allocate+”; and (iii) Swinburne portal and networked file systems. Swinburne security administrators are very busy in managing all these systems taking into account the number of users including administration, staff, researchers, and students that have access to such systems. The Swinburne file network is publicly accessible to help their staff to work and access their resources from home or even overseas. Swinburne is planning to be ISO27000 certified. This requires applying an extensive security management process, including conducting risk analysis for the Swinburne operational environment including their IT systems. They then have to integrate the identified and deployed security controls with their IT systems. Currently, this security management process is done manually.

Key Challenges: security management is a recurring process that should be revised with the change of enterprise business objectives or security risks. The security management process includes capturing various security details and systems’ details; an enterprise may operate a huge number of IT systems. Maintaining consistency of security enforced on such systems is a very complicated task.

Key Requirements: a security specification approach should support capturing early stage security goals and objectives, requirements, architectures, controls and maintaining traceability between such concepts. It should help in abstracting these details and facilitating the enforcement, integration, and monitoring of such security and system details.

Key Efforts: existing security management standards such as ISO27000 [1] and NIST-FISMA [8] define security management processes and phases that should be conducted and the steps to carry out in each phase. However, they do not provide any toolset to conduct such processes. On the other hand, existing research efforts in security management tend to focus on specific phases of the security management process. POSTIF: Policy-based security management [9], Ontology-based management [10], and model-based security management [15] efforts focus on automating the security enforcement (configuration) of heterogeneous security solutions using security policies, rather than how or what to capture in terms of security objectives, requirements, controls details. OCTAVE [11] approach focus on identifying possible threats, existing infrastructure vulnerabilities, and security plans to mitigate identified risks. CORAS [12] approach focuses on security risk modeling including assets, threats, vulnerabilities and risks with their likelihood and impacts.

Both approaches do not help in capturing security requirements, architecture and controls, relations between security and systems entities. Early-stage security engineering approaches focus on security requirements elicitation and capturing at design time. KAOS [4] was extended to capture security requirements in terms of obstacles to stakeholder’s goals. Obstacles are defined in terms of conditions that when satisfied will prevent certain goals from being achieved. Secure i* [5] focuses on identifying security requirements through analysing relationships between users, attackers, and agents of both parties. Secure Tropos [13] introduces two categories of goals to help capturing security requirements and trust goals: hard goals reflect system functional requirements and soft goals reflect security requirements. These approaches do not cover security modelling at system design stages (only security requirements elicitation).

Later-stage security engineering approaches typically focus on security engineering during system design. Both early and later stage approaches lack a complete security model that captures security details and abstraction levels. Misuse cases [14] capture use cases that the system should not allow and may harm the system operation or security. UMLsec [3] extends UML with a profile that provides stereotypes to be used in annotating design elements with security intentions and requirements. UMLsec provides a comprehensive UML profile but it was developed mainly for use during the design phase. UMLsec has stereotypes for predefined security requirements only (secrecy, secure dependency, critical...). secureUML [2] provides a meta-model to design role-based access control (RBAC) policies of target systems. Golnaz et al [15] introduce vulnerability-centric security requirements engineering approach. The proposed approach introduced some of the key security concepts including vulnerability, attack, and countermeasures as a complete solution – i.e. given an existing vulnerability V, an attack may exploit it to attack the system. This requires deploying countermeasures to mitigate such vulnerabilities. Thus, this approach does not capture all other relevant security concepts. Such security engineering approaches are tightly coupled with system design models, do not help in enterprise security architecture development, and do not provide a comprehensive security model (a model that captures all necessary security details).

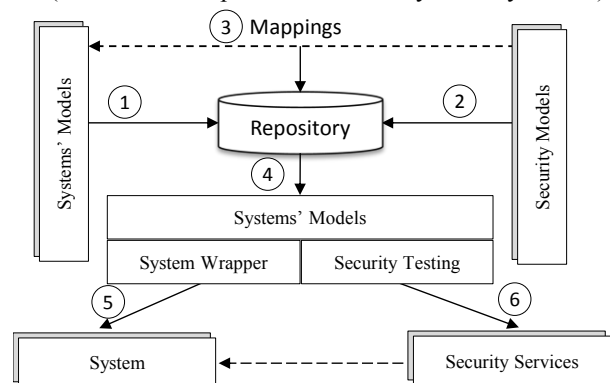


Figure 1. Our Model-based security management approach

III. MODEL-BASED SECURITY MANAGEMENT

Our security management approach is based on model-based security management, as shown in Figure 1. Security and systems' details are captured as abstract models and then realized by automating deployment, configuration, and integration of the security controls within target enterprise IT systems.

System models (1) are developed and delivered by software vendors using UML models. A system model should capture system delivered features, system architecture and main components, system classes, system deployment details. Enterprise security models (2) are gradually developed and frequently updated by enterprise security engineers/administrators using SecDSVL. A security model should capture security objectives, security requirements, enterprise security architecture and design, security controls operated. Unlike many other approaches, we use a separate DSVL-based security model that captures security information at varying levels of abstraction. Our rationale is to allow security specialists to be able to effectively model security properties, requirements and techniques without the need – or distraction - of system-specific requirements or design models.

Both models – system and security – are then weaved together by security administrators (3) while specifying security to be enforced on target systems' and their entities. Models and mappings are maintained in a shared repository and used by the security kernel (4) to integrate security controls deployed in the operational environment within the target IT system entities (5), and testing security controls' integration with target entities (6), as specified in the security mappings. A many-to-many mapping between the systems' model and security models is supported. Thus, one or more security concepts (security objectives, requirements and/or security controls) are mapped to one or more system model entities (system-level, feature-level, component-level, class-level and/or method-level entities). Whenever a high-level security concept such as security objective is mapped to a system or system entity, it implies that all related security requirements and controls are also mapped on the same entity. This helps avoid security specification inconsistency problems arising from wrongly configured or updated entities because of newly defined security specifications. Once a modification is applied to a modelled security concept, this update is reflected on other systems that use this concept. The security kernel is responsible for integrating security solutions as specified in the security-system mappings (5). This is achieved using static weaving AOP or even at runtime using dynamic-AOP. The former is more efficient as security code (to call security solutions) is already integrated with system code. However, the latter is more dynamic i.e. changes in security specified to be enforced on the target system can be realized at runtime [8]. In this paper we focus mainly on the SecDSVL as a security modeller we introduce to capture security details. The rest of the framework is published in different parts including model-driven security engineering at runtime - MDSE@R [7] and adaptive cloud computing security management [16].

IV. SECDSVL

SecDSVL provides a “mega-model” – a mega-model is a set of multiple related models with interrelationships - to be used by security engineers in managing enterprise IT systems' security. To develop SecDSVL, we studied the existing security management standards (NIST-FISMA [8], ISO27000 [1], steps are summarized in Figure 2 and explained in the next subsection), and Common Criteria [17] in details and come up with a comprehensive meta-model covering security management details, as shown in Figure 3. We then applied the physics of notations principles [18] in developing our SecDSVL notations, as we discuss below.

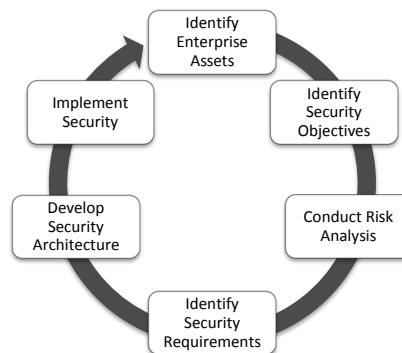


Figure 2. Standard security management process

A. Security Management Domain Model

Below we go through the main tasks/steps conducted in any security management process. In each task, we identified a set of concepts that need to be modeled, a set of relations to be captured, and logical groupings of these concepts into models.

Identify Enterprise Assets; the first step in a security management process is the identification of existing enterprise assets to be secured. We capture enterprise assets using an *asset model*. This captures all enterprise assets along with categories i.e. information system, physical asset or business value - and interrelationships. We use UML models to capture IT systems' detailed descriptions.

Identify Security Objectives; enterprise top management defines key security goals, objectives, and losses of breaching assets' security. These objectives are captured in a *security objectives model*. Objectives may be specified per asset– i.e. objectives are mapped to the corresponding asset, or enterprise-wide thus a given security objective could be mapped to many assets. Possible relationships between objectives include: composition – a security objective is made up of sub-objectives; dependency – a security objective depends on other security objectives in order to be satisfied/achieved. Availability, integrity, confidentiality, and accountability are key security objective categories [8]. Each security objective has importance level indicating the impact of a failure to meet such objective because of a security breach. A security objective may have different realization strategies: preventive, detective, and recovery.

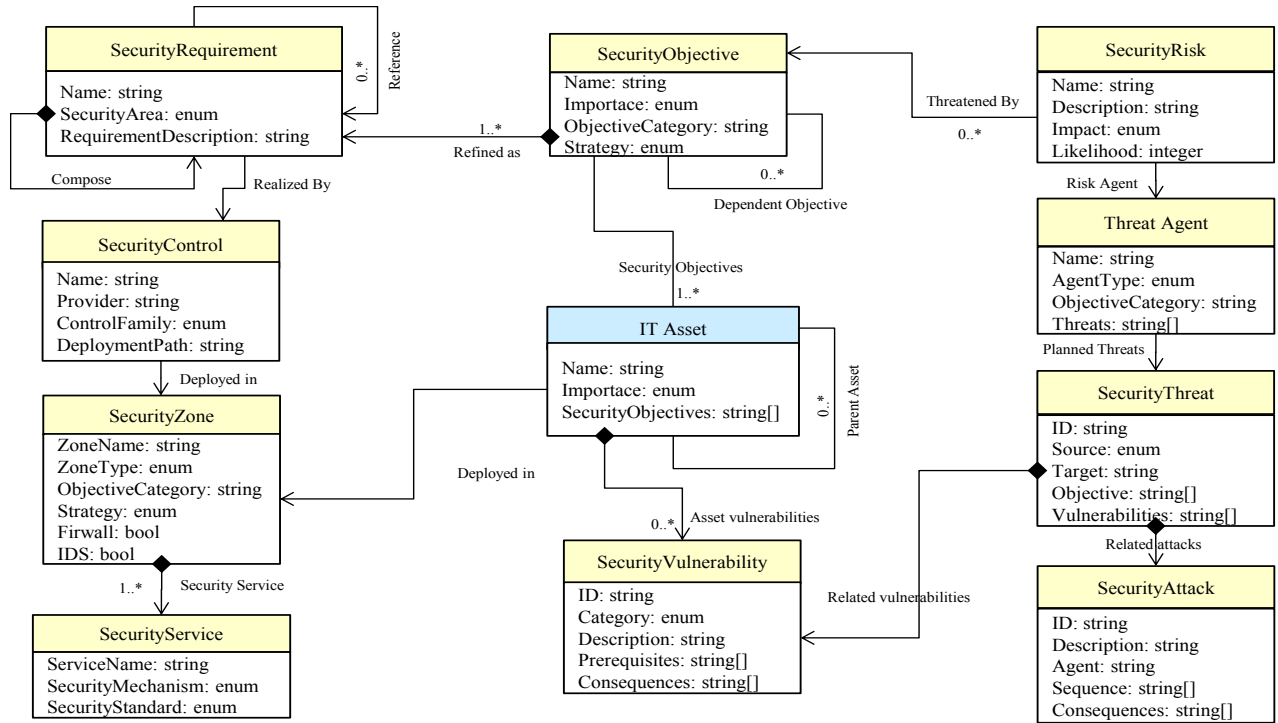


Figure 3. SecDSVL meta-model

Conduct Risk Analysis; security engineers conduct risk analysis on every enterprise asset to identify possible threats, likelihood, and impact of such threats. This may include performing vulnerability analysis to identify inherent flaws that may be exploited by threat agents. Such security risk details are captured in a *security risk model*. This may be developed per asset – i.e. for each enterprise asset we develop a risk model capturing possible risks, threats and attacks on this given asset, or an enterprise-wide risk model capturing all possible risks, threats and attacks and map such items onto different assets. Threats are linked to source threat agents and to exploited vulnerabilities.

Identify Security Requirements; security requirements describe the actions to be taken by enterprise security engineers in order to mitigate or avoid identified threats. Security requirements are captured in a *security requirements model*. For each specified security objective and identified security risk, we may define a set of security requirements e.g. “the system should not grant access to a resource X unless the user is authorized by the user name and password”. NIST [2] have 18 Security requirements’ families including audit, cryptography, and authentication, privacy, etc.

Develop Enterprise Security Architecture; Security architects use the security requirements model to develop a *security architecture model*. This captures planned behavior and structure of enterprise security. It shows how and where security controls and enterprise assets are positioned in the given enterprise architecture.

Moreover, it captures how these security controls are integrated with other enterprise assets. A security architecture model includes identifying security zones (domain) in the

enterprise operational environment including uncontrolled, controlled, restricted, and managed zones. Security architects define security services that will be deployed or used in every zone. These include authentication, authorization, cryptography, audit, etc.

Implementing Security; security administrators specify security controls that realize security services specified in the security architecture model. These security controls are captured in a *security controls model*. This model covers security controls location, configuration parameters, etc.

Traceability; each security entity in this security management process should retain tractability details to its high-level (parent) entities and to the lower-level (realization) entities. This helps in managing consistency and validating completeness of security models – i.e. validating that security controls defined retain the necessary trace back information to existing security requirements and not redundant. It also confirms that all security objectives, requirements and threats are realized by specific control(s). Figure 3 shows the meta-model of SecDSVL based on the tasks, concepts, and relations discussed above.

B. SecDSVL Notations

Given the security management concepts as summarized in SecDSVL meta-model Figure 3, we used Moody’s Physics of Notations (PON) principles [18] in designing our SecDSVL for the above models. Below we summarize the key PON principles and how PON were applied on SecDSVL.

The PON Principles. This includes a set of principles that should be taken into consideration when designing a visual language. These notations help improving language usability

including: **Semiotic clarity principle** focuses on assessing notations redundancy, overload, deficit, and excess. **Perceptual discriminability principle** focuses on discriminability between different symbols. **Semantic transparency principle** focuses on the relation between visual notations and semantics of the concept. **Complexity management principle** focuses on managing complex diagrams. **Cognitive integration principle** focuses on visual and conceptual integration between diagrams. **Visual expressiveness principle** focuses on improving notations expressiveness using visual variables such as color, shape, size, brightness. **Graphic economy principle** focuses on minimizing number of graphical symbols. Some of these principles are related to modeling language such as complexity management, cognitive integration and graphic economy. Others focus on individual concepts' properties such as semiotic clarity, visual expressiveness, and perceptual discriminability. Given the discussed security concepts, relations and PON principles, we have developed our SecDSVL physical notations, as shown in Table 1. Below, we discuss the design decisions we made during the development of these visual notations for SecDSVL and our selections and reasons behind these decisions. Usually icon selection does not have much (if any) scientific justification - it is often more of an art rather than science [13]. However, we tried to find a careful balance between expressiveness and simplicity when selecting SecDSVL notations and their combinations to form our DSVLs.

PON and SecDSVL. We discuss how we use the PON principles when selecting SecDSVL notations.

Enterprise Assets: we use the “puzzle” shape to capture enterprise assets. An asset “puzzle” is a part of the enterprise assets “image”. Each puzzle has pop-outs to capture provided capabilities and pop-ins representing required capabilities. Puzzle color visual variable captures asset criticality {high, moderate, low} in {red, grey, and white}.

Security Objectives: we use the “star” shape to reflect stakeholders' dreamed objectives. We use a set of visual variables to visualize a security objective attributes. The star line color is used to capture objective importance e.g. highly important - red, moderate – grey and low - white. Shading is used to capture security strategy – e.g. preventive, detective, recovery, or any. Fill color is used to capture objective category – i.e. confidentiality - black, integrity - blue, availability – green, and accountability - orange.

Security Threat: we use an “oval” shape with an extra icon to represent a security threat. Icon style is used to reflect threat type, as defined in Microsoft STRIDE [19] – spoofing – stick man, tampering – down arrow, repudiation - + icon which means repeat, information disclosure – up arrow, all can read, and DOS – x icon which means not available.

Threat Agent: we use “sticky man” shape to capture threat agent concept. Color variable reflects agent type – i.e. external attacker – red, or malicious insider – black – usually unseen or unexpected in terms of security breach.

Security Vulnerability: we use “bug” shape to visualize security vulnerabilities. Bug color is used to reflect discovered vulnerability category including: input validation (such as SQL Injection), output validation (Cross Site Scripting - XSS), processing (race conditions), or hosting service related vulnerability such as Cross Site Request Forgery - CSRF.

Security Attack: we use “Bomb” shape to capture security attacks. Bomb color reflects different attack objectives as discussed in the security objective concept above.

Security Requirement: we use “document” shape to capture security requirements. Document color is used to capture security requirement family – e.g. access control, cryptography, identity management, etc. which is linked to the security objective category.

Security Zone: we use “Big box” shape to represent security zone. Box color is used to represent zone security level – e.g. uncontrolled, controlled, secured, and managed.

Security Service: we use “process” shape to capture security services. Process color reflects service security category. We use the same coloring scheme as for security requirements.

Security Control: we use the “guard” shape to represent security controls. We use the same coloring schema as used in the security requirements.

Traceability between models is realized using drag-and-drop of model elements – e.g. dragging a security objective *Obj* (defined in a security objectives model) to a security requirement *Req* (in a security requirements model) creates a link between these objects and updates *Req* objectives list to include objective *Obj*.

Table 1. Summary of key SecDSVL Notational Elements



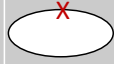







Concept	Physical Notation	Visual Variables and Concept attributes binding
Asset		{Colour, asset criticality} : { Red, Grey, White} {Pop outs, Provided capabilities} {Pop ins, Required capabilities}
Security Objective		{Line Colour, Importance} : {Red, Grey, White} {Shading, Strategy} : { vertical, horizontal, diagonal} { Fill Color, Objective Category} : {Black, Blue, Green, Orange}
Security Threat		Relations with other entities including threat agent, vulnerabilities, and assets. {Icon, threat agent objective} { x, ↓, ↘, ↑, ⚡ }
Threat Agent		{ Colour, Agent Type} :{ Red, Black}
Security Vulnerability		{ Colour, Vulnerability Category} :{ Red, Black, Green, Yellow}
Security Attack		{ Colour, Attack Objective} :{ Red, Black, Green, Yellow}
Security Requirements		{ Colour, Requirement Family} :{ Red, Black, Green, Blue, Orange, etc}
Security Zone		{Colour, Zone Security Level} : { Yellow, Grey, Magenta, White}
Security Service		{Colour, Service Category} :{ Red, Black, Green, Blue, Orange, etc}
Security Control		{ Colour, Control Family} :{ Red, Black, Green, Blue, Orange, etc}

Figure 4. Examples of the assets model and Galactic system description model

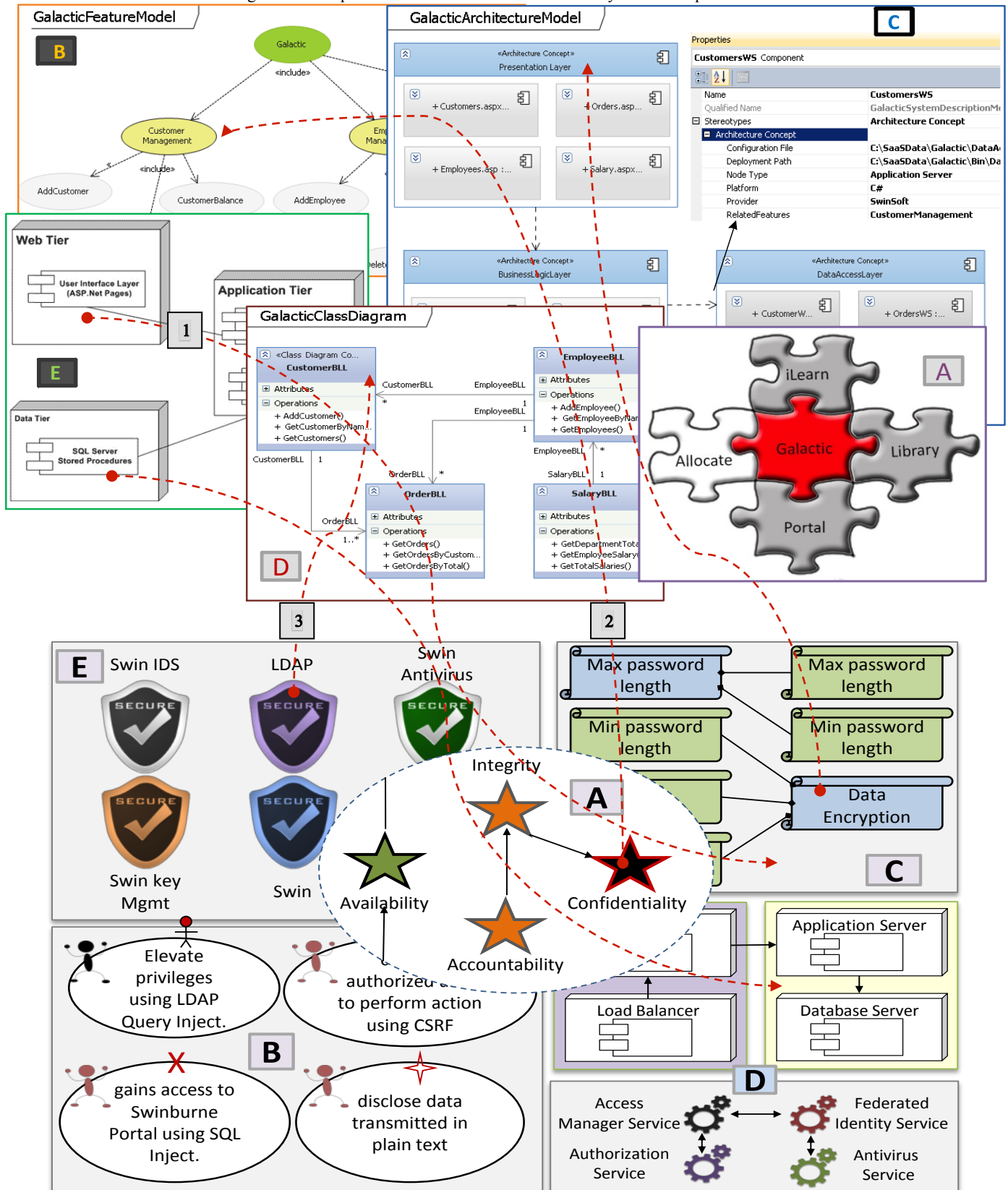


Figure 5. Examples of Swinburne SecDSVL Models

V. USAGE EXAMPLE

We illustrate how security engineers use our SecDSVL in managing their enterprise IT systems' security. We highlight the involved stakeholders and their roles and expected outcomes in each step. We use our motivating example from Section II, where Swinburne University has a set of deployed IT Systems that need to be secured. We illustrate this usage example using screen dumps from our Visual Studio plug-in based SecDSVL toolset. This toolset provides an integrated environment for modeling systems (UML and UML profiles), security (SecDSVL), weaving system and security models, and realizing specified security.

Modeling Enterprise IT Systems: Swinburne security engineers identify and model existing IT systems that need to be secured. These details are captured in Swinburne assets model. Figure 4-A shows existing IT systems and connections between them. We see from assets' colors that Galactic is a critical asset, allocate is low, others are moderate. For each asset, Swinburne then develops, if not delivered by system vendor, a detailed system description model (Figure 4). This model captures key details of the system including system features (Figure 4-B), system architecture (Figure 4-C), system classes (Figure 3-D), and system deployment details (Figure 4-E). We developed a UML profile that captures dependences and relations between system entities e.g. system features to realization components, and system components to realization classes.

Modelling Swinburne Security Details: Swinburne security administrators develop and revise the enterprise SecDSVL model. In this scenario, Swinburne security administrators document security objectives that must be satisfied (Figure 5-A) including confidentiality, integrity, availability. This model should be repeatedly revisited to incorporate changes in Swinburne security objectives. Security administrators then use security objectives to identify security threats, vulnerabilities, and risks related to a given asset or enterprise-wide that violate these objectives (Figure 5-B).

Next, Swinburne administrators develop their security requirements model (Figure 5-C), including "authenticate user" and "data encryption" requirements. Swinburne security engineers then develop a detailed security architecture model including services and security mechanisms to be used in securing enterprise assets (Figure 5-D). In this example we show the different security zones that cover Swinburne networks and allocation of IT systems to zones. The security architecture also shows security services, security mechanisms and standards to be deployed. Swinburne security engineers finally specify the security controls (i.e. the real implementations) for the security services modeled in the security architecture model (Figure 5-E).

In this example it includes SwinIDS Host Intrusion Detection System, LDAP access control and SwinAntivirus. These are used to realize the security requirements and security architecture as previously specified. Each model

keeps track of related security entities. Each security control maintains traceability information to parent models' entities.

Weaving Systems and Security Models: after developing the enterprise assets model and system description models, and the SecDSVL model, Swinburne security engineers map security attributes (in terms of objectives, requirements and controls) to Swinburne assets e.g. Galactic system (in terms of features, components, classes). This is achieved by drag and drop of security attributes to systems entities. SecDSVL support different granularity levels – i.e. may define security on component level or even on method level. All models and defined mapping are maintained in models repository to be used by security enforcement point. The dashed red lines between Figures 3 and 4 show security to system mappings, such as (1) placement of deployment nodes within security zones; (2) security objectives that should be met on different components; and (3) security solutions mapped to deployment node or system entities, as shown in links between Figure 3 and Figure 4.

Table 2. Comparison between SecDSVL and existing efforts

Tool	1	2	3	4	5	6	7	8	9	10
Misuse case	X	X	√	√	√	X	X	X	X	X
UMLsec	X	X	X	X	•	X	•	X	X	•
SecureUML	X	X	X	X	•	X	•	X	X	•
KAOS	X	√	•	√	√	X	X	X	X	X
i*	X	√	•	√	√	X	X	X	X	X
Tropos	X	√	•	√	√	X	X	X	X	X
CORAS	√	•	√	√	X	X	X	X	•	•
OCTAVE	√	•	√	√	X	X	X	X	•	X
FISMA	√	√	√	√	X	X	X	√	√	X
ISO27000	√	√	√	√	X	X	X	√	√	X
SecDSVL	√	√	√	√	√	√	√	√	√	√
[1] Assets Model, [2] Security Objective, [3] Security Threats and Vulnerabilities, [4] Security Risks, [5] Security Requirements, [6] Security Architecture, [7] Design, [8] Security Controls, [9] Traceability, [10] Mapping between systems and security models										
√: Fully Supported • : Partially supported X : Not supported										

Table 3. Assessing SecDSVL against PON principals

PON Principal	Attribute	Found %
(1) Semiotic Clarity	Symbol deficit	0%
	Symbol redundancy	0%
	Symbol overload	0%
	Symbol excess	0%
(2) Perceptual Discriminability	Shapes Discriminability	100%
	Textual differentiation	0%
	Redundant Coding	0%
	Perceptual Popout	100%

(3) Semantic Transparency	Icons	90%
(4) Complexity Management	Modularization	Yes
	Hierarchy	Yes
(5) Cognitive Integration	Conceptual Integration	100%
	Perceptual Integration	Yes
(6) Visual Expressiveness	Use of colour	100%
	Choice of Visual Variables	60%
	Range of Visual Variables Values	60%
	Textual Vs Graphical Encoding	100%
(7) Graphic Economy	Number of graphical symbols	100%

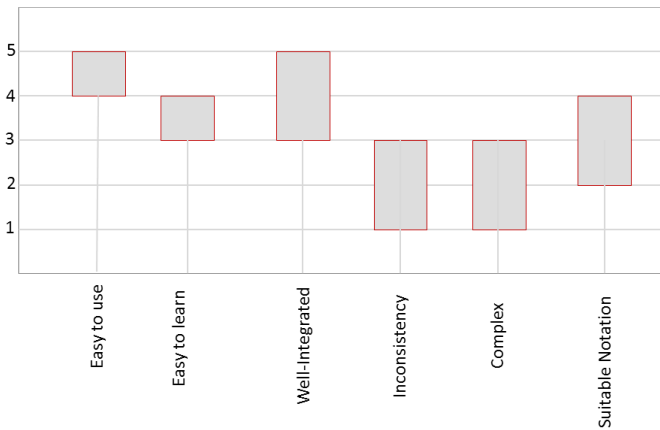


Figure 6. Level of agreement of usability factors

VI. EVALUATION

SecDSVL Comprehensiveness: We have evaluated SecDSVL provided capabilities in capturing different kinds of enterprise security details with different levels of abstractions compared to existing security management, risk management, and security engineering approaches. Table 2 shows SecDSVL compared to a range of other existing security modeling techniques based on their documented capabilities. Rows in the table reflect existing security engineering and modeling efforts. Columns reflect the set of security concepts that we have identified from our analysis of the existing security management standards including NIST-FISMA and ISO27000 (as discussed in the earlier sections). Table 2 shows that none of the existing approaches provides a comprehensive security model that covers every aspect in the security management process. Table 2 shows that security management approaches focus on assets, risks, threats, and mitigation controls. Risk management approaches focus on similar areas except for security controls. Early stage security engineering efforts focus on security objectives and requirements. Later-stage security engineering efforts focus on security requirements, mapping and integration with target systems.

User Evaluation: We carried out a user evaluation to assess SecDSVL usability. We had seven post-graduate researchers to use our developed tools and platform. Five of

them are working in software engineering research. Two of them are working in the security domain. None were involved in the development of SecDSVL. We gave them a one hour training session on the tool. Then, we asked them to: explore SecDSVL notations; read a set of security models developed by SecDSVL; and modify these models with new security needs. We conducted a usability survey to gain their feedback on SecDSVL and modeling toolset. The results show that they successfully understood and updated security models. They give positive feedbacks about the overall approach and the tool usability, and the capabilities in managing systems' security Figure 6 (1: strongly disagree...5: strongly agree).

Figure 1.

ecDSVL and PON: We have assessed SecDSVL against the physics of notations principles. For each principle, we calculate the conformance ratio (number of SecDSVL notations that satisfy a PON principle against total number of SecDSVL notations).

Table 3 shows that SecDSVL design covers most of good principles and successfully avoids bad practices in modelling languages design. (1) SecDSVL has no Semiotic clarity problems which focus on having one-to-one correspondence between concepts and graphical notations. (2) Concepts' shapes are efficiently discriminable – i.e. no overlap or conflict between shapes of different concepts. No text is used to support differentiation between shapes. (3) Icons have been selected to match perceptual expectations as much as we can (still this is a matter of art – e.g. we use start to represent an objective, bug to reflect defect and so on). We got feedback on the security zone (big box) to be replaced with more expressive icon (thus 1 out of 10 need to be revised). (4) Diagram complexity is managed using hierarchical organization (using meta-model relations) and diagrams modularization using loose-coupling diagramming – i.e. every diagram can reflect specific security aspect. This is directly supported by the underlying platform. (5) Navigation between diagrams is also managed by the underlying platform. Moreover, drilling and tracing conceptual concepts throughout the whole SecDSVL is supported. (6) Many visual variables have been used to replace text-based notations including colors, icons, shape line style, shading, and line color. Some people did not like shading and line color as easy to discriminate visual variables (thus 2 out of 5 need to be updated). (7) The number of shapes is efficient compared to existing concepts.

VII. DISCUSSION

We developed SecDSVL to better support the security management process specified in NIST and ISO27000 security management standards. SecDSVL supports asset modeling, security objective modeling, security risk modeling, security requirements modeling, security architecture and design modeling, and security controls modeling. We maintain relationships and traceability between each of these different security models' entities. Tool support is via a Visual Studio

plugin. From our experiments, SecDSVL succeed in capturing most of security details arise during the security management process. SecDSVL concepts can be refined in lower-level concepts (using other sub-models) or in specific concept details e.g. sub-objectives that make up a specific security objective. Users have found that SecDSVL is easy to learn and use, and effective in capturing security details.

A key question with regard to any modeling language is how to decide the right level of abstraction. In SecDSVL, we cover a range of different abstraction levels including security objectives, requirements, risks and threats down to the realization security services and controls. We are currently working on extending security controls with configuration details. Thus, security administrators can fully depend on SecDSVL model in configuring their security controls as well.

Existing security management efforts fail to ensure that modeled security is met by deployed systems. This is usually done manually causing delay in updating IT systems with new security specifications. To overcome this limitation, we connected the security requirements, captured in SecDSVL models, with the enterprise IT systems architectures and security enforcement controls, captured in UML models. Using AOP techniques we succeed in enforcing security specified onto IT Systems. This helps in supporting runtime security updating to mitigate newly discovered vulnerabilities and risks.

A key feedback from the user evaluation practitioners is the suitability of the threat and vulnerability shapes. They recommend using more expressive shapes to support perceptual expectations taking into consideration threats and vulnerabilities' attributes. SecDSVL is currently developed as Microsoft Visual Studio 2010 plug-in using Microsoft Modeling SDK (the tool along with examples can be download from MDSE@R: <http://sourceforge.net/projects/mdse-r/>). We are developing a web-based SecDSVL as a stand-alone web tool. Moreover, we plan to conduct more extensive evaluation using industrial case studies, and automate SecDSVL models generation using security analysis tools such as risk and vulnerability models using vulnerability and risk assessment tools.

VIII. SUMMARY

We have introduced a new, comprehensive security specification domain-specific visual modeling language, SecDSVL. This is used for capturing a variety of security specifications for enterprise security including objectives, risks, threats, requirements, architecture, and enforcement controls. Moreover, it helps in maintaining traceability between these security concepts. SecDSVL is developed using Moody's Physics of notations principles. We performed a comparative evaluation between SecDSVL and existing enterprise security modeling approaches, a user evaluation of SecDSVL notations, and an evaluation against key PON principles. These showed that SecDSVL is effective and efficient for enterprise security management modeling.

IX. ACKNOWLEDGMENT

The authors are grateful to Swinburne University of Technology and the FRST SPPI project for support for this research.

REFERENCES

- [1] International Organization for Standardization (ISO), "ISO/IEC 27000 - Information technology - Security techniques - Information security management systems - Overview and vocabulary," ISO/IEC 27001:2005(E), 2009.
- [2] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in *Proc. of The 5th International Conference on The Unified Modeling Language*, Dresden, Germany, 2002, pp. 426-441.
- [3] J. Jürjens, "Towards Development of Secure Systems Using UMLsec," in *Fundamental Approaches to Software Engineering*. vol. 2029, ed: Springer Berlin Heidelberg, 2001, pp. 187-200.
- [4] A. Lamsweerde, S. Brohez, and e. al, "System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering," in *Proc. of the RE'03 Workshop on Requirements for High Assurance Systems*, Monterey, 2003, pp. 49-56.
- [5] L. Liu, E. Yu, and J. Mylopoulos, "Secure μ^* : Engineering Secure Software Systems through Social Analysis," *International Journal of Software and Informatics*, vol. Vol.3, pp. 89-120, 2009.
- [6] H. Mouratidis, J. Jürjens, and J. Fox, "Towards a Comprehensive Framework for Secure Systems Development," in *Advanced Information Systems Engineering*. vol. 4001, E. Dubois and K. Pohl, Eds., ed: Springer Berlin / Heidelberg, 2006, pp. 48-62.
- [7] M. Almorsy, J. Grundy, and A. S. Ibrahim, "MDSE@R: Model-Driven Security Engineering at Runtime," presented at the Proc. of the 4th International Symposium on Cyberspace Safety and Security Melbourne, Australia, 2012.
- [8] National Institute of standards and technology (NIST), "The Federal Information Security Management Act (FISMA)," Washington: U.S. Government Printing 2002, <http://csrc.nist.gov/drivers/documents/FISMA-final.pdf>, Accessed on August 2012.
- [9] C. Basile, A. Lioy, G. M. Perez, F. J. G. Clemente, and A. F. G. Skarmeta, "POSITIF: A Policy-Based Security Management System," in *Eighth IEEE International Workshop on Policies for Distributed*

- Systems and Networks, 2007. POLICY '07, 2007*, pp. 280-280.
- [10] B. Tsoumas and D. Gritzalis, "Towards an Ontology-based Security Management," presented at the Proc. of 20th International Conference on Advanced Information Networking and Applications - Volume 01, 2006.
- [11] P. Marek and J. Paulina, "The OCTAVE methodology as a risk analysis tool for business resources," presented at the Proc. of The 2006 International Multiconference Computer Science and Information Technology, 2006.
- [12] R. Fredriksen, M. Kristiansen, B. Gran, and K. Stølen, "The CORAS Framework for a Model-Based Risk Management Process," in *Computer Safety, Reliability and Security*. vol. 2434, S. Anderson, M. Felici, and S. Bologna, Eds., ed: Springer Berlin / Heidelberg, 2002, pp. 39-53.
- [13] H. Mouratidis, and P. Giorgini, "Secure Tropos: A security-oriented Extension of the Tropos Methodology," *International Journal of Software Engineering and knowledge Engineering*, 2007.
- [14] G. Sindre, and A. Opdahl, "Eliciting security requirements with misuse cases," *Requir. Eng.*, vol. 10, pp. 34-44, 2005.
- [15] G. Elahi, E. Yu, and N. Zannone, "A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities," *Requir. Eng.*, vol. 15, pp. 41-62, 2010.
- [16] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Adaptable, Model-driven Security Engineering for SaaS Cloud-based Applications," *Automated Software Engineering Journal*, vol. 29, 2013, pp1-38.
- [17] Common Criteria for Information Technology Security Evaluation, "Part 1: Introduction and general model, Version 3.1," 2006, <http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf>, Accessed on August 2013.
- [18] D. Moody, "The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 35, pp. 756-779, 2009.
- [19] M. Abi-Antoun and J. r. M. Barnes, "STRIDE-based security model in Acme," Technical Report CMU-ISR-10-106, Carnegie Mellon Univ., 2010.2010.