INTERNATIONAL STANDARD

ISO/IEC 27034-6

First edition
2016-10-01

Information technology — Security techniques — Application security —

Part 6: Case studies

Technologies de l'information — Techniques de sécurité — Sécurité des applications —

Partie 6: Études de cas

Reference number
ISO/IEC 27034-6:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 27, *IT Security techniques*.

A list of all parts in the ISO/IEC 27034 series can be found on the ISO website.

Introduction

0.1 General

There is an increasing need for organizations to focus on protecting their information at the application level. A systematic approach towards increasing the level of application security provides an organization with evidence that information being used or stored by its applications is being adequately protected.

ISO/IEC 27034 (all parts) provides concepts, principles, frameworks, components and processes to assist organizations in integrating security seamlessly throughout the life cycle of their applications.

The application security control (ASC) is one of the key components of this document.

To facilitate the implementation of ISO/IEC 27034 (all parts) application security framework and the communication and exchange of ASCs, a formal structure should be defined for representing ASCs and certain other components of the framework.

0.2 Purpose

The purpose of this document is to provide examples of security guidance for organizations to acquire, develop, outsource and manage security for their specific applications through their life cycle.

0.3 Targeted Audiences

0.3.1 General

The following audiences will find values and benefits when carrying their designated organizational roles:

a) domain experts.

0.3.2 Domain experts

Domain experts contributing knowledge in application provisioning, operating or auditing, who need to

a) participate in ASC development, validation and verification,

b) participate in ASC implementation and maintenance, by proposing strategies, components and implementation processes for adapting ASCs to the organization's context, and

c) validate that ASCs are useable and useful in application projects.

Information technology — Security techniques — Application security —

Part 6: Case studies

1 Scope

This document provides usage examples of ASCs for specific applications.

NOTE Herein specified ASCs are provided for explanation purposes only and the audience is encouraged to create their own ASCs to assure the application security.

2 Normative references

There are no normative references cited in this document.

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 27034-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

4 Abbreviated terms

ASC application security control

ASLC application security life cycle

ASLCRM application security life cycle reference model

ONF organization normative framework

5 Security guidance for specific applications

5.1 General

Guidelines play an important role for companies trying to implement any best practice or ISO standard because they instruct how to institutionalize the practices or rules and, sometimes, the guidance is based on common examples.

Companies benefit from this guidance as it demonstrates, as a practical example, how to structure ASCs for specific applications using the recommended XML data structure defined in ISO/IEC 27034-5-1 and for the implementation of the Organizational Normative Framework.

5.2 ASC example: Java code revision for mobile applications

5.2.1 General

Code review seams trivial but when an application is built from thousands of lines of code, it may be unproductive and/or too expensive to revise everything.

This example presents an ASC designed by a fictive organization called *ORGANIsation Inc.* This ASC implements the security activity of code review.

5.2.2 Purpose

The purpose of 5.2 is to provide an intuitive description of an example ASC named "Code Review" for an organization developing Java mobile applications. For the sake of brevity and readability, a simplified subset of the ASC is presented in English only (language="EN"), but the ASC requirements defined in ISO/IEC 27034-5 allows any object in an ASC to be described in any characters sets, as presented by the Table A.1.

5.2.3 Context

ORGANIsation Inc. is an international organization developing Java mobile applications for its own use and on behalf of its clients. ORGANIsation software development offices are located in Montreal, Vancouver and Moscow. For this reason, ORGANIsation's policy is that any development documentation, guideline or training should be available in English, French and Russian languages.

ORGANIsation's implementation strategy for ISO/IEC 27034 (all parts) prioritizes the design of ASCs for reducing security vulnerabilities in Java mobile code. The ORGANIsation ONF committee mandates the Application Security Department (ASD) to design and submit Java code review ASCs.

5.2.4 ORGANIsation Information classification guidelines

ORGANIsation utilizes approved internal classification guidelines for classifying the information into four levels:

a) restricted;

b) confidential;

c) secret;

d) top secret.

5.2.5 Levels of trust included in the ORGANIsation ASC Library

ORGANIsation had previously conducted an organization-wide security risk assessment, for the purpose of which it divided its applications into six categories according to their impact on organizational risk. Following this, domain experts mandated by the ONF committee decided to use those six categories as a template for defining ORGANIsation's application levels of trust. An informal definition along with a descriptive label for each level of trust is given in Table 1.

Table 1 — ORGANIsation's application levels of trust

Level of trust	Name	Description
0	Baseline	All ORGANIsation's applications shall comply with this Level of Trust.
1	Isolated — Local network only	This Level of Trust is appropriate for applications used on isolated corporate networks, with no connection to external networks.
2	Low — Internet, public information only	This Level of Trust is appropriate for Internet-facing applications sharing public information without any privacy concern.
3	Medium — Internet, corporate users	This Level of Trust is appropriate for Internet-facing, transactional applications used by corporate users, allowing access to corporate services, user files and/or transactions under $5 000.
4	High — Secure transactions and privacy protection over Internet	This Level of Trust is appropriate for Internet-facing, transactional applications, used by corporate users, allowing access to user private information and/or transactions from $5 000 to $25 000.
5	Private	This Level of Trust is appropriate for transactional applications requiring highly secure transactions, privileged access and/or secure critical storage. Access to critical information and/or transactions over $25 000 is authorized.

5.2.6 Outcome

The application security department was mandate to select and acquire an automatic source code review tool suitable for the Java language, with user-configurable review rules. After analysis of vendor propositions, the department selected a tool named "Efficient-Reviewer version 2.2".

At the end of this project, version 1.0 of five ASCs were developed and implemented.

Table 2 — ORGANIsation's ASCs for code review

ID	Name	Level of trust	Description
ORGANIsation-ASD-042	Code Review	— Baseline — Isolated – Local network only — Low – Internet, public information only — Medium – Internet, corporate users — High – Secure transactions and privacy protection over Internet — Private	This ASC is used to help developers to perform a code review control for JAVA applications.
ORGANIsation-ASD-043	Code Classification	— Baseline — Isolated – Local network only — Low – Internet, public information only — Medium – Internet, corporate users — High – Secure transactions and privacy protection over Internet — Private	Classify all Java classes in the packages needed by the application. Any class should inherit its classification from the highest-classified information it processes.

Table 2 *(continued)*

ID	Name	Level of trust	Description
ORGANIsation-ASD-044	Basic Automatic Code Review	— Baseline — Isolated – Local network only — Low – Internet, public information only	This ASC is used to help developers to implement a code review control for Java applications by providing an automatic source code security review process for Java classes classified as "Strategic" and "Critical".
ORGANIsation-ASD-045	Advanced Automatic Code Review	— Medium – Internet, corporate users — High – Secure transactions and privacy protection over Internet — Private	This ASC is used to help developers to implement a code review control for Java applications by providing an automatic source code security review process for all of the application's Java classes.
ORGANIsation-ASD-046	Manual Code Review	— High – Secure transactions and privacy protection over Internet — Private	This ASC is used to help developers to implement a code review control for Java applications by providing a manual source code security review process for Java classes classified as "Strategic" and "Critical".

NOTE The ASC with ID "ORGANIsation-ASD-042" is the root of the code-review ASC hierarchy.

5.2.7 ORGANIsation stakeholders involved in these ASCs

For each of these ASCs, the following responsibilities were determined.

NOTE This subclause consist consist of informal descriptions of ASC data elements, followed by the formal description of same using XML notation.

Table 3 — Names and responsibilities of Java code review ASCs stakeholders

Role/responsibility	Name	Notes/ORGANIsation directives
Author	Jules Verne	
Owner	Douglas Adams	— M. Adams requested to start numbering ORGANIsation's ASCs from 42.
Creation Request:	Herbert George Wells	— A PDF version of the creation request, explaining why this ASC is required by the organization will be included in each ASC. — The PGP signature of M. Wells will be required to seal the ASC. — The date when this activity will be completed shall be specified.
Design	Jules Verne	
Validation	Arthur C. Clarke	
Development	Frank Herbert	
Verification	Ray Bradbury William Gibson	— The security activity and the measurement and verification activity shall both be verified in both languages.
Approval	Robert Heinlein	— The PGP signature of M. Heinlein will be required to seal the ASC.
Final Owner approval	Douglas Adams	— The PGP signature of the owner is required to seal the ASC.

Table 3 *(continued)*

Role/responsibility	Name	Notes/ORGANIsation directives
Published for training	Isaac Asimov	
Active	Mary Shelley	
Expired	Not defined.	

Additional information recorded in each ASC includes coordinates of each actor (department, e-mail address, phone number, physical address) and the completion date for each activity.

The ASC structure allows each version of an ASC to be electronically signed for integrity purposes. ORGANIsation directives in Table 3 specify which electronic signatures are mandatory. This provides assurance that critical stages of the ASC's life cycle were performed and verified according to ORGANIsation's policy.

In Figure 1, the cloud-shaped region illustrates the part of the ASC protected by electronic signatures.

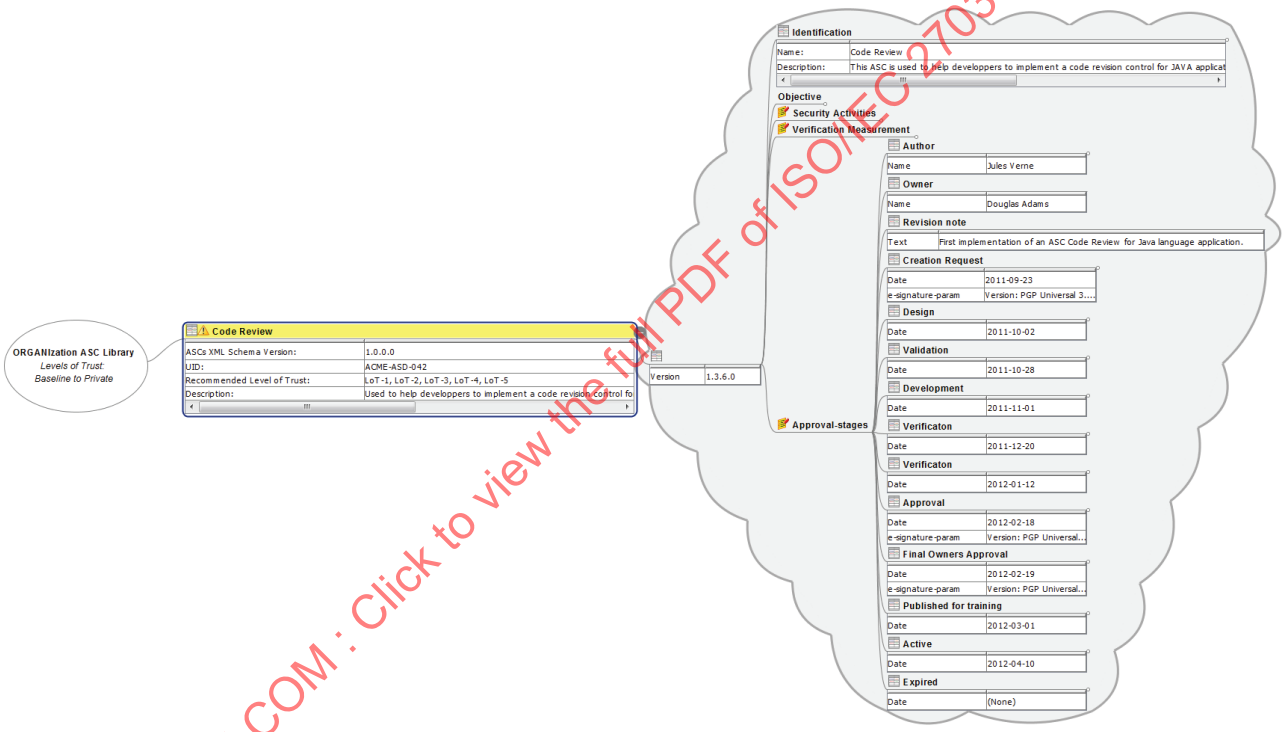

Figure 1 — Integrity scope of stakeholder's electronic signatures

See Table A.2.

"ASC ORGANIsation-ASD-042 - Code Review" is a "Head ASC" and does not contain any security activity or verification and measurement activity. Instead, it refers to four children ASCs, which are required to implement code review in the organization's Java development process. Figure 2 illustrates this concept of ASC hierarchy. It is also to be noted that the "Head ASC" omits some of the mandatory ASC attributes defined by ISO/IEC 27034-1:2011, Figure 6. These will be provided by the child ASCs.

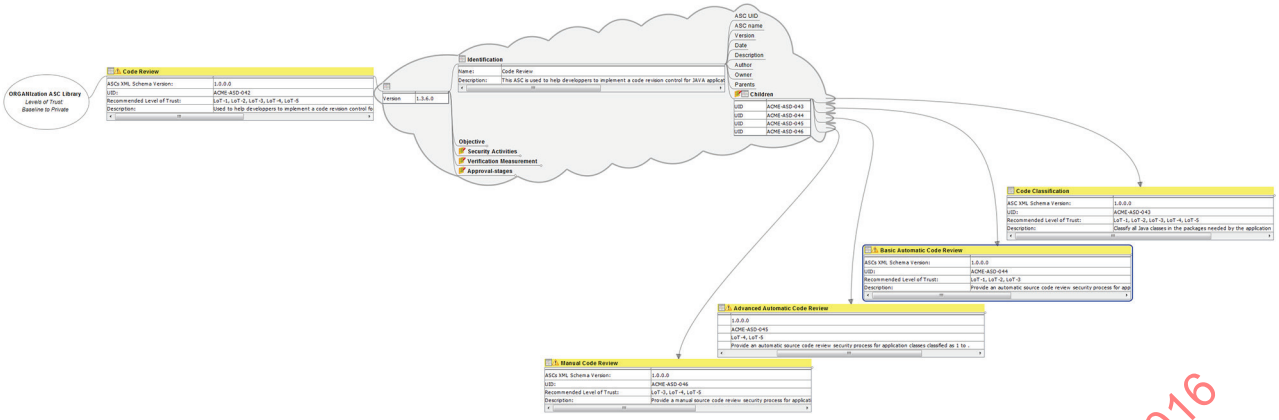

Figure 2 — Code review ASC graph

See Table A.3.

5.2.8 Descriptions of sample ASCs

5.2.8.1 General

This subclause describes the head ASC (Code review) and its four children ASCs developed and implemented in the ORGANIsation ASC Library.

5.2.8.2 ASC ORGANIsation-ASD-042: Code review

ASC	Code Review
ASC UID	ORGANIsation-ASD-042
Identification	
ASC UID	ORGANIsation-ASD-042
ASC name	Code Review
Version	1.3.6.0
Date	2016-01-04
Description	This ASC is used to help developers to perform a code review control for JAVA applications.
Author	Jules Verne Application Security Department ORGANIsation inc. 1234 Street ave W, Beautiful city, Quebec, Canada Email office: JVernes@ORGANIsation.com Phone office: +1.234.567.8901

Owner	Douglas Adams Application Security Department ORGANIsation inc. 1234 Street ave W, Beautiful city, Quebec, Canada Email office: DAdams@ORGANIsation.com Phone office: +1.109.876.5432
Parents	None
Children	ORGANIsation-ASD-043 ORGANIsation-ASD-044 ORGANIsation-ASD-045 ORGANIsation-ASD-046

See Table A.4.

Approval-stages	(See the ASC approval-stages XML example in 5.2.7.)

Objective			
Objective description	Top-level ASC whose objective is to group the various leaf ASCs related to code review in Java.		
Requirements addressed	-- Content removed for simplification --		
Assigned Levels of trust	0, 1, 2, 3, 4, 5		
Context of use	Technological context		
Levels of trust range	**Level of Trust**	**Name**	**Description**
	0	Baseline	All ORGANIsation's applications shall comply with this Level of Trust.
	1	Isolated – Local network only	This Level of Trust is appropriate for applications used on isolated corporate networks, with no connection to external networks.
	2	Low – Internet, public information only	This Level of Trust is appropriate for Internet-facing applications sharing public information without any privacy concern.
	3	Medium – Internet, corporate users	This Level of Trust is appropriate for Internet-facing, transactional applications used by corporate users, allowing access to corporate services, user files and/or transactions under $5 000.
	4	High – Secure transactions and privacy protection over Internet	This Level of Trust is appropriate for Internet-facing, transactional applications, used by corporate users, allowing access to user private information and/or transactions from $5 000 to $25 000.

	5	Private	This Level of Trust is appropriate for transactional applications requiring highly secure transactions, privileged access and/or secure critical storage. Access to critical information and/or transactions over $25 000 is authorized.
	(See Table 2)		
Pre-conditions	-- Content removed for simplification --		

See Table A.5.

Security Activity	None.
Verification Measurement	None.

5.2.8.3 ASC ORGANIsation-ASD-043: Code classification

ASC	Code Classification
ASC ID	ORGANIsation-ASD-043
Identification	
ASC UID	ORGANIsation-ASD-043
ASC name	Code Classification
Date	2015-12-25
Description	Classify all Java classes in the packages needed by the application. Any class should inherit its classification from the highest-classified information it processes.
Version	2.6.1.1
Author	-- Content removed for simplification --
Owner	-- Content removed for simplification --
Parents	ORGANIsation-ASD-042
Children	None

Approval-stages	(See the ASC approval-stages XML example in 5.2.7.)

Objective	
Objective description	Define the scope of the code review.
Requirements addressed	Business requirements: ORGANIsation Development guidelines v2.1, Section 5.6 – Application components classification.
Assigned Levels of trust	0, 1, 2, 3, 4, 5
Levels of trust range	(See Table 2)
Pre-conditions	-- Content removed for simplification --

See Table A.6.

Security activity	
Name (what)	Classify classes and packages
Description	Identify and categorize the application's Java classes and packages.
Target information group	Application Data (see ISO/IEC 27034-1:2011, 6.3)
Target information sub-group	Development documentation
Target information group name	Application's Java code architecture
Outcome general description	Categorized classes and packages information merged in the application's Java code architecture documentation.
Supporting expert ressource	Orson Scott Card ORGANIsation inc. Email: Orson.Scott.Card@ORGANIsation.com
Complexity	COMPLEX
Complexity description	This activity should be performed by someone able to identify, from the application architecture documents, what information is manipulated by each Java class and to identify security risks that may threaten sensitive information.
Global estimated effort (how much)	— Average of 1 h to classify and document 10 Java classes. — Average of 15 h to update the Application Security Risk Analysis.
Role (who)	APPLICATION ARCHITECT
Responsibility	RESPONSIBLE
Required qualifications	1. Passed an examination on the ORGANIsation Java coding best practices. 2. Minimum 5 years experience in Java Development. 3. Active CSSLP Certification.
Pre-condition	— The application classes and packages identification section of the Application design document is completed. — A list of categorized information groups involved by the application already exists.
Security activity description (how)	Classify the application classes to be developed or maintained in this project. 1. Identify contexts, roles, and information involved with the application module. Ref.: ORGANIsation Development guidelines v2.1. 2. Realize or update the Application Security Risk Analysis. 3. Classify all classes in the packages needed by the application in the Application Class Classification section of the Application design document. Ref.: ORGANIsation Code Classification Guide, v1.4, and Application Class Classification section – Template v2.3.
Localization (where)	— Application development environment

Moment (when)	AFTER: The detailed application architecture is completed.
Supporting documents	1. ORGANIsation Development guidelines v2.1.PDF. 2. ORGANIsation Code Classification Guide, v1.4.PDF. 3. Application Class Classification section – Template v2.3.RTF.
Artefacts (what)	1. Document: The Application Module Classification section, in the Application design document, describing for all application modules, their class classification value from “Not critical” to “Critical”. 2. Document: The classification method, templates and examples are described in the classification Guide, referenced within this ASC.

See Table A.7.

Verification measurement	
Name (what)	Classes and packages classification verification.
Description	Verify if the application’s Java classes and packages produced or modified were adequatly categorized.
Target information group	Application Data (See ISO/IEC 27034-1:2011, 6.3)
Target information sub-group	Application development documentation
Target information group name	Application’s Java code architecture
Outcome general description	These two documents are produced or updated: the Application security risk analysis and the Application design document.
Supporting expert ressource	Ray Bradbury Email: Ray.Bradbury @ORGANIsation.com
Complexity	COMPLEX
Complexity description	This activity should be done by someone who is able to validate, from applicaton architecture documents, what information is manipulated in every Java classes and to validate security risks that may threaten sensitive information.
Global estimated effort (how much)	— An average of 6 h to validate the contexts and revise the risk analysis and an average of 10 min to approve/reject each class and package. — Around 12 h/project.
Activity specification	— Classify application’s classes and components. Ref.: ORGANIsation Code Classification Guide, v1.4
Role (who)	APPLICATION SECURITY ARCHITECT
Responsibility	ACCOUNTABLE and RESPONSIBLE
Qualifications required	1. Passed an examination on the ORGANIsation Java coding best practices. 2. Passed an examination on the secure application architecture. 3. Active CSSLP Certification.
Pre-condition	— The Application’s Java code architecture documentation includes a complete categorized classes and packages information.

Verification-measurement activity description (how)	Revise and approve Application Module Classification section, in the Application design document: 1. validate the contexts, roles and information involved with this module; 2. revise the application risk analysis; 3. for each class, reject or approve the classification and mark accordingly.
Localization (where)	— Application development environment
Moment (when)	Before coding.
Supporting documents	1. ORGANIsation Code Classification Guide, v1.4.PDF
Artefacts (what)	1. Updated Application security risk analysis document. 2. Approved Application Module Classification section, in the Application design document.

See Table A.8.

5.2.8.4 ASC ORGANIsation-ASD-044: Basic automatic code review

ASC	Basic Automatic Code Review
ASC ID	ORGANIsation-ASD-044
Identification	
ASC UID	ORGANIsation-ASD-044
ASC name	Basic Automatic Code Review
Version	1.3.0.0
Date	2015-12-25
Description	This ASC is used to help developers to implement a code review control for Java applications by providing an automatic source code security review process for Java classes classified as “Strategic” and “Critical”.
Author	-- Content removed for simplification --
Owner	-- Content removed for simplification --
Parents	ORGANIsation-ASD-042
Children	None
Approval-stages	(See the ASC approval-stages XML example in 5.2.7.)
Objective	
Objective description	Defines the activities (and their verification) involved in basic automatic code review.
Requirements addressed	1. This control is required for compliance with PCI-DSS 2.0, section xx.xx.
Assigned Levels of trust	0, 1, 2
Context of use	Technological context
Levels of trust range	(See Table 2)
Pre-conditions	Java classes in the packages needed by the application are categorized.
Security Activity	
Name (what)	Automatic code review (basic)
Description	Run an automatic code review only on Java classes classified as “Strategic” and “Critical”.
Target information group	Application data
Target information sub-group	Source code

Target information group name	Java class and packages
Outcome general description	Code review reports produced by the tool.
Supporting expert ressource	Team leader
Complexity	MEDIUM
Complexity description	The developer shall verify the version of the rules file loaded in the tool before running the automatic review on the Java code.
Global estimated effort (how much)	An average of 20 min per class.
Role (who)	DEVELOPER
Responsibility	RESPONSIBLE
Required qualifications	1. More than 3 months experience in Java programming. 2. Passed an examination on CR tools utilization. 3. Passed an examination on the ORGANIsation Java coding best practices.
Pre-condition	The application “Efficient-Reviewer version 2.2” is installed and configured on the user’s station.
Security activity description (how)	For all java classes classified as “Strategic” and “Critical”, developed by the developer, run the security code review tool as a unit test activity. 1. After the developer has completed coding and compiling the class without any error. 2. Start the security code review tool and load rules file XXXY-053-JAVA. 3. Perform the automatic code review. 4. Save the report produced by the code review tool. 5. Analyze the report and correct all errors and warnings detected. 6. Generate a single hash code from the Java code and the produced report files. 7. Check-in the hash code and the report with the Java code.
Localization (where)	Developper workstation
Moment (when)	BEFORE: APPLICATION LAYER, REALIZATION STAGE, DEVELOPMENT ACTIVITY AREA, CODING ACTIVITY, COMPILE CODE.
Supporting documents	1. File: XXXY-053-JAVA.rules
Artefacts (what)	1. A security report without any error or warning. 2. A check-in including the verified code, the report produced by the tool and the hash code of these two files.
Verification Measurement	
Name (what)	Basic automatic code review verification
Description	Check the reports produced by the code review tool for all classes that were classified as “Strategic” and “Critical” for this application project module
Target information group	Application data
Target information sub-group	Source code
Target information group name	Signed Java class and packages Verification tools report

Target information group description	Once the code review is completed, the tool will sign the source code and save the signature in the report. The source code file, the report and the signature will have to be verified.
Outcome general description	Target code modules are verified or rejected. When verified, they are migrated to the "preliminary tests" environment.
Supporting expert resource	Team leader
Complexity	EASY
Complexity description	Make sure the tool report is in the same folder of its related source code before starting the verification.
Global estimated effort (how much)	Average of 4 h per migration.
Role (who)	AUDITOR
Responsibility	RESPONSIBLE
Required qualifications	Should have followed the ORGANIsation components migration training.
Pre-condition	The application module "Efficient-Reviewer version 2.2 – Hash code validation" shall have been installed configured on the source code server for the project.
Security activity description (how)	Check the reports produced by the code review tool for all classes that were classified as "Strategic" and "Critical" for this application project module and migrate them in "preliminary tests" environment. With the Code Revision tool "Efficient-Reviewer version 2.2" 1. Identify packages to migrate. 2. For each package a. verify that each class has a log report, and b. verify that each log report contains no error or warning. If so, i. Verify that the hash code associated with the package file and the report file produced with the code revision tool is correct. ii. If so, set the flag associate with the package file as "Verified" in the versioning system. If not, iii. Set the class Flag associate with the package file as "Rejected" in the versioning system. iv. Send an email to the programmer. 3. Are all package files flagged as "Verified"? If so, a. Migrate the module, and b. Send an email to the project manager. If not, a. Cancel the migration, and b. Send an email to the project manager.
Localization (where)	Development environment, project source code server
Moment (when)	BEFORE, APPLICATION SUPPLY LEVEL, TRANSITION STAGE, MODULE MIGRATION IN TEST ENVIRONMENT.
Artefacts (what)	List of application project packages in this module, with their updated flag status.

5.2.8.5 ASC ORGANIsation-ASD-045: Advanced Automatic Code Review

ASC	Advanced Automatic Code Review
ASC ID	ORGANIsation-ASD-045
Identification	
ASC UID	ORGANIsation-ASD-045
ASC name	Advanced Automatic Code Review
Date	2015-12-25
Description	This ASC is used to help developers to implement a code review control for Java applications by providing an automatic source code security review process for all of the application's Java classes.
Version	1.0.0.0
Author	-- Content removed for simplification --
Owner	-- Content removed for simplification --
Parents	ORGANIsation-ASD-042
Children	None
Approval-stages	(See the ASC approval-stages XML example in 5.2.7.)
Objective	
Objective description	Defines the activities (and their verification) involved in advanced automatic code review.
Requirements addressed	This control is required for compliance with PCI-DSS 2.0, section xx.xx.
Assigned Levels of trust	3, 4, 5
Levels of trust range	(See Table 2)
Pre-conditions	Java classes in the packages needed by the application were classified.
Security Activity	
Name (what)	Automatic Code Review (advanced)
Description	Run a full automatic code review on all Java classes written by the developer.
Target information group	Application data
Target information sub-group	Source code
Target information group name	Java class and packages
Outcome general description	Code review reports produced by the tool.
Supporting expert ressource	Team leader
Complexity	MEDIUM
Complexity description	The developer shall verify the version of the rules file loaded in the tool before running the automatic review on the Java code.
Global estimated effort (how much)	An average of 20 min per class.
Role (who)	DEVELOPER
Responsibility	RESPONSIBLE
Required qualifications	1. More than 3 months experience in Java programming. 2. Passed an examination on CR tools utilization. 3. Passed an examination on the ORGANIsation Java coding best practices.

Pre-condition	Application “Efficient-Reviewer version 2.2” is installed and configured on the user’s station.
Security activity description (how)	For all Java classes developed by the developer, run the security code review tool, as a unit test activity. 1. After the developer has completed coding and compiling the class without any error. 2. Start the security code review tool and load rules file XXXY-053-JAVA. 3. Perform the automatic code review. 4. Save the report produced by the code review tool. 5. Analyze the report and correct all errors and warnings detected. 6. Generate a single hash code from the Java code and the produced report files. 7. Check-in the hash code and the report with the Java code.
Localization (where)	Developper’s workstation
Moment (when)	BEFORE: APPLICATION LAYER, REALIZATION STAGE, DEVELOPMENT ACTIVITY AREA, CODING ACTIVITY, COMPILE CODE.
Supporting documents	1. File: XXXY-053-JAVA.rules
Artefacts (what outcomes)	1. A produced security report without any error or warning. 2. A check-in including the verified code, the report produced by the tool and the hash code of these two files.
Verification Measurement	
Name (what)	Advanced automatic code review verification
Description	Check the reports produced by the code review tool for all classes that were classified as “Strategic” and “Critical” for this application project module
Target information group	Application data
Target information group name	Application data
Target information sub-group	Source code
Target information group description	Signed Java class and packages Verification tools report
Target information group classification	Once the code review is completed, the tool will singed the source code and save the signature in the report. The source code file, the report and the signature will have to be verified.
Outcome general description	Target code modules are verified or rejected. When verified, they are migrated to the “preliminary tests” environment.
Supporting expert ressource	Team leader
Complexity	EASY
Complexity description	Make sure the tool report is in the same folder as its related source code before starting the verification.
Global estimated effort (how much)	Average of 10 h per migration.
Role (who)	AUDITOR
Responsibility	RESPONSIBLE
Required qualifications	Should have received the ORGANIsation components migration training.

Pre-condition	The application "Efficient-Reviewer version 2.2 – Hash code validation module" is installed and configured on the user's station.
Security activity description (how)	Check the reports produced by the code review tool for all classes for this application project module and migrate them in "preliminary tests" environment. With the Code Revision tool " Efficient-Reviewer version 2.2" 1. Identify packages to migrate. 2. For each package a. verify that each class has a log report, and b. verify that each log report contains no error or warning. If so i. Verify that the hash code associated with the package file and the report file produced with the code revision tool is correct. ii. If so, set the flag associate with the package file as "Verified" in the versioning system. If not i. Set the class Flag associate with the package file as "Rejected" in the versioning system. ii. Send an email to the programmer. 3. Are all package files flagged as "Verified"? If so a. Migrate the module b. Send an email to the project manager If not a. Cancel the migration b. Send an email to the project manager
Localization (where)	Development environment, project source code server
Moment (when)	BEFORE, APPLICATION SUPPLY LEVEL, TRANSITION STAGE, MODULE MIGRATION IN TESTS ENVIRONMENT.
Artefacts (what outcomes)	List of application project packages in this module, with their updated flag status.

5.2.8.6 ASC ORGANIsation-ASD-046: Manual code review

ASC	Manual Code Review
ASC ID	ORGANIsation-ASD-046
Identification	
ASC UID	ORGANIsation-ASD-046
ASC name	Manual Code Review
Version	3.5.0.2
Date	2014-09-07
Description	This ASC is used to help developers to implement a code review control for Java applications by providing a manual source code security review process for Java classes classified as "Strategic" and "Critical".
Author	-- Content removed for simplification --
Owner	-- Content removed for simplification --
Parents	ORGANIsation-ASD-042
Children	None

Approval-stages	(See the ASC approval-stages XML example 5.2.7)
Objective	
Objective description	Defines the activities (and their verification) involved in manual code review.
Requirements addressed	This control is required for compliance to the ORGANIsation Critical Application Security Policy, section 12.6.
Recommended Level of trust	4, 5
Context of use	Technological context
Levels of trust range	(See Table 2)
Pre-conditions	Java classes in the packages needed by the application are classified.
Security Activity	
Name (what)	Manual code review
Description	Execute a manual code review only on Java classes classified as “Strategic” and “Critical”.
Target information group	Application data
Target information sub-group	Source code
Target information group name	Java class and packages
Outcome general description	Code review reports produced by the reviewer.
Supporting expert ressource	
Complexity	DIFFICULT
Complexity description	
Global estimated effort (how much)	An average of 14 h per module.
Role (who)	PROGRAMMERS TEAM LEADER
Responsibility	RESPONSIBLE
Required qualifications	1. Passed an examination on the ORGANIsation Java coding best practices 2. Minimum 5 years experience in Java development 3. Active CSSLP certification
Pre-condition	The latest version of the application “PGP Desktop Home” is installed and configured on the user’s workstation.
Security activity description (how)	1. Retrieve the template for the manual code review report (attached to this ASC). 2. Retrieve the best practice guide for Java secure programming (attached to this ASC). 3. Identify classes that have been classified. 4. For each strategic and critical class: a. verify that the code conforms with the ORGANIsation Java secure programming best practices;

	b. complete the code review report, using the template; c. flag class as either “verified” or “rejected”; d. sign the class and associated report; e. check-in the class and associated report. 5. If any class was flagged as “Rejected”, a. open correction requests for rejected classes, and b. send a list of rejected classes to the programmer.
Localization (where)	Developer workstation
Moment (when)	AFTER: A developer checks-in a Java module.
Supporting documents	1. Template of manual code review report, v5.6.docx. 2. ORGANIsation Java secure programming best practices guide v2012.PDF.
Artefacts (what)	1. Checked-in classes, signed by reviewer, with updated status, with code review report. 2. Correction requests 3. List of rejected classes
Verification Measurement	
Name (what)	Manual code review verification
Description	Check the manual code review report for all classes that were classified as “Strategic” and “Critical” for this application project module.
Target information group	Application data
Target information sub-group	Source code
Target information group name	Application data
Target information group description	Signed Java class and packages, Manual code review report
Outcome general description	Target code modules are verified or rejected.
Supporting expert ressource	
Complexity	MEDIUM
Complexity description	
Global estimated effort (how much)	15 min per application component
Role (who)	AUDITOR
Responsibility	RESPONSIBLE
Required qualifications	1. Passed an examination on the ORGANIsation Java coding best practices. 2. Minimum 5 years experience in Java development. 3. Active CISA certification.
Pre-condition	The last version of the application “PGP Desktop Home” shall have been installed and configured on the user’s station.
Security activity description (how)	Verify that all strategic and critical classes were manually reviewed as required. 1. Identify packages to migrate. 2. For each package:

	a. For each strategic and critical class, verify that the checked-in class is flagged as "Verified" and that the class and associated report has been signed by the authorized code reviewer; b. Do all classes in this package satisfy the above criteria? If so, i. flag the package as "verified", and ii. send an email to the project manager. If not, i. flag the package as "rejected", and ii. send an email to the project manager.
Localization (where)	Development environment, project source code server
Moment (when)	BEFORE, APPLICATION SUPPLY LEVEL, TRANSITION STAGE, MODULE MIGRATION IN TESTS ENVIRONMENT.
Artefacts (what)	1. A list of categorized application components required to be reviewed, and 2. the outcome of the manual code review activity for each one of them, including a. the name of the reviewer, b. the review date and time, c. approval, and d. the verifier's public key to validate that the source code was correctly signed to ensure its integrity.

5.3 Case study: Developing ASCs to address the issue of privacy for two countries

5.3.1 General

The regulatory context lists and documents any law or regulation, in any of the organization's business locations that could impact application projects. It includes laws, rules and regulations of the countries or jurisdictions where the application is developed and/or deployed and/or used.

Regulatory matters are delicate issues, especially if conflicting laws from multiple nationalities or cultures are in the scope of an application. Once the regulatory context of an application is defined, potential conflicting regulations may be detected and addressed.

5.3.2 Purpose

The purpose of this subclause is to provide an example of a strategy for implementing ASCs to address conflicting privacy issues for two countries.

5.3.3 Context

ORGANIsation Inc. is an international organization providing a service of on-line medical records for which customers are health clinics. The application has an automated process for deleting medical records after 5 years of inactivity.

ORGANIsation has found new customers in country A and country B.

In Step 1 of the ASMP, ORGANIsation established the regulatory context for the application, by having legal experts identify and analyse both countries' relevant laws and regulations. It was discovered that there was a conflict between some requirements of country A's and country B's privacy laws.

Specifically, one article of country A's privacy law states that a person's private information shall be kept for at least 10 years after its last use, while one article of country B's privacy law states that a person's private information shall be securely deleted at most 1 year after its last use.

For ORGANIsation's customers, the security risk of not complying with existing laws generates two security requirements:

a) ensure that a person's private information is kept for at least 10 years after its last use in country A;

b) ensure that a person's private information is securely deleted at most 1 year after its last use in country B.

ORGANIsation needs to demonstrate to its customers that the application meets these requirements.

ORGANIsation decides to address these requirements by implementing the following ASCs.

a) **ASC 1**: implement a process for securely deleting a medical record and any related information. This is a "head" ASC that serves as a parent to the following two children ASCs.

   1) **ASC 1.1**: Develop, implement and verify secure delete processes.

      i) **When:** DURING detailed architecture (Application supply layer, Realization/Development/ Elaboration).

      ii) **Security Activity:** This ASC should describe at least the following security activities:

         — develop and implement a secure delete process for a medical record in the application database for a specific customer;

         — develop and implement a secure delete process for files related to a medical record for a specific customer, on the application's file servers;

         — develop and implement a secure delete process for a medical record in the archived database and archived files for a specific customer;

      iii) **Verification Measurement activity:** This ASC should describe activities for verifying that each security activity above was correctly implemented.

   2) **ASC 1.2**: Perform secure delete processes implemented in ASC 1.1 and verify their correct execution.

      i) **When:** DURING the use of the application (Application supply layer, Utilization and maintenance/Utilization).

      ii) **Security Activity:** This ASC should describe at least the following security activities:

         — perform secure delete processes when needed by the application.

      iii) **Verification Measurement activity:** This ASC should describe activities for verifying that the secure delete processes are actually being performed correctly by the application.

b) **ASC 2**: Implement a process for correctly selecting medical records for deletion according to the relevant regulations. This is a "head" ASC that serves as a parent to the following two children ASCs.

   1) **ASC 2.1**: Develop, implement and verify a process for correctly selecting medical records for deletion according to the relevant regulations.

      i) **When:** DURING detailed architecture (Application supply layer, Realization/Development/ Elaboration).

      ii) **Security Activity:** This ASC should describe at least the following security activities:

         — add to the clinic profile, information identifying the country where it is located;

— add to each medical record a field identifying the clinic owning the record;

— add to each medical record a field identifying the time of last use;

— define and implement deletion rules related to each country;

— modify in the application the existing selection process so that it applies the relevant deletion rule for each medical record.

iii) **Verification Measurement activity:** This ASC should describe activities for verifying that each security activity above was correctly implemented

— In particular, have legal experts validate the deletion rules for each country.

2) **ASC 2.2**: Perform selection process implemented in ASC 2.1 and verify its correct execution.

i) **When:** DURING the use of the application (Application supply layer, Utilization and maintenance/Utilization).

ii) **Security Activity:** This ASC should describe at least the following security activities:

— perform selection process when needed by the application.

iii) **Verification Measurement activity:** This ASC should describe activities for verifying that the selection process is actually being performed correctly by the application.

5.4 Case study: Integration of third-party ASCs

5.4.1 General

Developing the necessary ASCs for an organization's application security initiative *in-house* can be a complex and lengthy undertaking requiring specialized knowledge and resources. The process can be significantly facilitated if, instead of in-house development, third-party ASCs are acquired, adapted and integrated into the organization's ONF.

An open and portable exchange language for ASCs (as recommended in ISO/IEC 27034-5-1) is a prerequisite for the integration and adaptation of third-party ASCs.

5.4.2 Purpose

The purpose of this case study is to illustrate how third-party ASCs can be adapted and integrated into an organization's application security program. The case study also emphasizes the importance of including the range of levels of trust in the definition of the ASCs.

As a way to simplify this case study, some clauses are defined as "[omitted]" but these clauses should be defined in any ASC as presented in the recommended ASC XML structure in ISO/IEC 27034-5-1.

5.4.3 Context

ORGANIsation Inc. ("ORGANIsation") is a company engaged in the business of online brokerage for exchange traded securities, such as futures, bonds and stocks. ORGANIsation currently maintains 80 applications, among which 25 applications are Internet facing. Twelve months ago, ORGANIsation has initiated an application security program. Currently the ONF consists of 13 ASCs assigned to 4 levels of trust (depicted in the Table 4).

Table 4 — ORGANIsation's level of trust table

Level of trust	Description
(i)	Internet facing applications processing confidential information
(ii)	Internet facing applications processing only non-confidential information
(iii)	Internal applications processing confidential information
(iv)	Internal applications processing only non-confidential information

Recent changes to the regulatory context require ORGANIsation to update its ONF so as to mandate different types of security testing for all their applications. In order to accelerate the process of updating the ONF, ORGANIsation decides to purchase two ASCs from third-party ASC vendor and pen-test specialist Hackus Inc. ("Hackus").

For the "Third-party ASC integration" use case, Figure 3 indicates how two example ASCs (called ASC1 and ASC2) are transferred from two different sub-suppliers to the acquirer ("ORGANIsation").

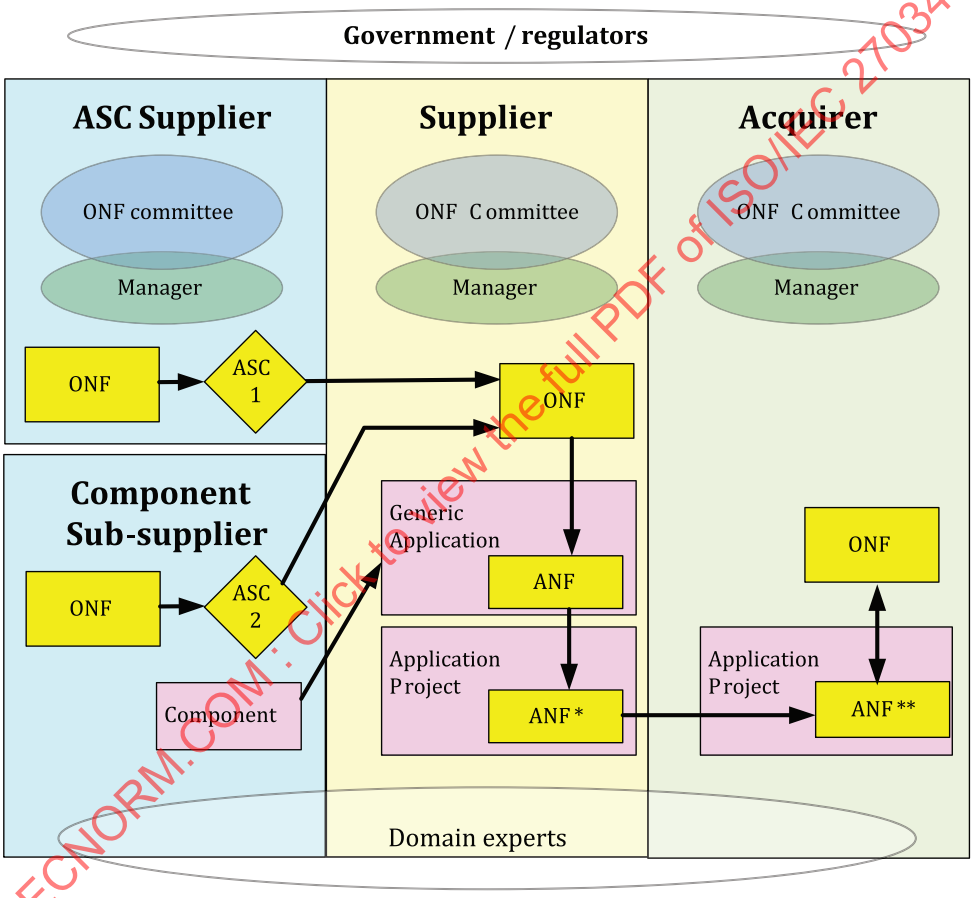

Figure 3 — ASC supply chain example

First, Figure 3 indicates that each involved company has its own ONF. As part of an "Application Project" the Supplier provides a customized version of his "Generic Application". The ASCs of the "Generic Application" are in their own (application specific) ANF. These ASCs are a subset of the Supplier's ONF. The Supplier's ONF receives ASC1 from a specialized security company, called "ASC Supplier" and ASC2 from one of its (software) component sub-suppliers, called "Component Sub-supplier". While the purpose of ASC1 and ASC2 is for use with the "Generic Application" as part of "ANF", they are first integrated into Supplier's ONF. The customer specific "ANF*" may contain additional/different controls as compared to the generic "ANF". Together with the "Application Project" as a whole, ANF* is also transferred to the Acquirer. At the Acquirer its security configuration may be updated, resulting in ANF**. The new ASC1 and ASC2 that made their way into ANF** are also registered in the Acquirer's ONF. The arrow between

the Acquirer's ONF and the Application Project's ANF is bidirectional, as additional (e.g. site-specific) ASCs of the Supplier's ONF may also apply for the new application project.

The path along which ASC1 is transferred in Figure 3 can be summarized as: ONF of "ASC Supplier" → ONF of Supplier → ANF of Supplier → ANF* of Supplier → ANF** of Acquirer → ONF of Acquirer. The Acquirer can obtain an ASC3 directly from the ASC supplier, which is not explicitly indicated in Figure 3. According to the purchase agreement, Hackus exports both ASCs in form of an "ASC Package" in XML format consistent to the Schema described in ISO/IEC 27034-5-1. Fragments of both ASCs are given below. Since the purpose of 5.4 is to illustrate how third-party ASCs may be integrated into an organization's ONF, the security activities as well as the verification activities of the ASCs are omitted.

ASC 1.1: Automatic Vulnerability Scanning

When:

— DURING (PERIODIC 12 months) (Application supply layer/Operation/ Utilization and Maintenance)

Recommended Levels of Trust:

— Level "Blue"

Level of Trust Scale:[1)]

— Level "Blue": Applications that do not process, store or access confidential information

— Level "Red": Applications processing, storing or accessing confidential information

Security activity: [omitted]

Verification activity: [omitted]

ASC 1.2: Penetration Testing

When:

— DURING (PERIODIC 12 months) (Application supply layer/Operation/Utilization and Maintenance)

Recommended Levels of Trust:

— Level "Red"

Level of Trust Scale:[1)]

— Level "Blue": Applications that do not process, store or access confidential information

— Level "Red": Applications processing, storing or accessing confidential information

Security activity: [omitted]

Verification activity: [omitted]

According to the Level of Trust Scale provided in both ASCs, Hackus uses two levels of trust ("Red" and "Blue") whereas ORGANIsation uses a Level of Trust Scale consisting of four levels of trust (i, ii, iii, iv). In order to integrate both ASCs into ORGANIsation's ONF, it is essential perform a mapping between Hackus' and ORGANIsation's level of trust ranges. Table 5 shows such a mapping.

1) In this example, Hackus associated the colour "Blue" with lower risk, while it associated the colour "Red" with a higher risk in its Level of Trust Scale.

Table 5 — Hackus and ORGANIsation's level of trust alignment table

Hackus	ORGANIsation
Blue	(ii), (iv)
Red	(i), (iii)

The mappings were identified by consulting the level of trust definitions provided by ORGANIsation and Hackus (within the ASCs). Based on the mappings, it becomes evident that ASC "Automatic Vulnerability Scanning" should be integrated in ORGANIsation's ONF under levels of trust (ii) and (iv) and that ASC "Penetration Testing" should be integrated under levels of trust (i) and (iii).

5.5 Case study: Using the ASLCRM to facilitate implementation of ASCs by different development groups inside an organization

5.5.1 General

The application security life cycle reference model (ASLCRM) defines a "development methodology agnostic" reference list of activities and roles covering various stages and aspects of the application life cycle. The ASLCRM provides a common ground for all ASCs and helps organizations to apply the same ASCs across projects, development teams, and development methodologies.

5.5.2 Purpose

The purpose of this case study is to illustrate the importance and the utility of the ASLCRM. It will be demonstrated that ASCs are applicable in different development context by referencing development activities of the ASLCRM.

5.5.3 Context

ORGANIsation Inc. ("ORGANIsation") is a nationwide banking and insurance corporation. It employs more than 40 000 employees and maintains several in-house IT development groups using different development methodologies, such as "Agile" and "Heavyweight". A recently inaugurated application security initiative has as its objective to integrate security systematically into all development projects, following any development methodology.

To this end, ORGANIsation has developed 18 ASCs covering different aspects of application security, such as training, code review, testing and certification. An example ASC ("Penetration Testing") is depicted in the Table 6. For the sake of brevity, the ASC is shown in abbreviated and simplified form.

Table 6 — ORGANIsation's ACS for penetration testing

Title	Penetration Testing
Description	A penetration test (pentest) is a controlled security test that aims at exploiting a system's vulnerabilities in order to gain (unauthorized) access. It is carried out in a controlled environment by a specialized security expert (ethical hacker) following specific rules of engagement. Each penetration test results in a detailed report listing the found vulnerabilities (exploits) and their criticalities.
Levels of Trusts	The ASC is applicable for applications with the following levels of trust: — critical; — high.
Security Activity	
Complexity	High Typically, a penetration test consists of multiple modules where each module aims at testing a particular aspect or component of the system. For each module, multiple testing sessions may be required.

Table 6 *(continued)*

Title	Penetration Testing
Specification	**Task 1 – Request and Planning**: The project manager contacts the security group to request and schedule a penetration test for the system under development. **Execution Moment:** (1.1.1.2.14 – PLAN_QUALITY) **Resource Allocations**: Project Manager **Task 2 – Scope and Rules of Engagement:** The security architect collaborates with the application owner and project manager to define the scope of the test (according to the security requirements), as well as the rules of engagement. **Execution Moment**: 2.1.3.3.3 (DEFINE_THE_SECURITY_SPECIFICATION) **Resource Allocations:** — Security Architect — Application Owner — Project Manager **Output:** — Test plan — Rules of engagement **Task 3 – Testing:** The tester carries out the penetration test in accordance to the test plan and the rules of engagement. After the completion of the test, the tester reviews the test results with the application architect and the application owner in order to compile a detailed action plan on how to mitigate the found vulnerabilities. **Execution Moment:** 2.1.3.3.4 (TEST_SOLUTION) **Resource Allocations:** — Tester — Application architect — Application owner **Output:** Vulnerability Report, Action Plan
Verification Measure	[omitted]

As depicted, the security activity of the ASC consists of three tasks. Each task is attributed a description, execution moment, resource allocation and expected output (if applicable). While the description and the expected output are specified informally, the indications of the execution moment and resource allocations are specified relative to the ASLCRM.

This form of indirection allows the ASC to be (re)used in different development methodologies using different activities and roles. For this purpose, it is merely required to map the concrete roles and activities (of a particular development methodology) to their counterparts of the ASLCRM. Example mappings for "ORGANIsation Agile" and "ORGANIsation Heavyweight" development methodologies are given in Tables 7 and 8.

Mapping of the life cycle activities:

Table 7 — ASLCRM activities alignment with ORGANIsation's existing activities

ASLCRM	ORGANIsation agile	ORGANIsation heavyweight
PLAN_QUALITY	Plan Increment	Define Project Plan
DEFINE_THE_SECURITY_SPECIFICATION	Create Backlog	Define Supplementary Requirements Specification
TEST_SOLUTION	Execute Test Cases	Execute Security Testing

Mapping of the roles:

Table 8 — ASLCRM roles alignment with ORGANIsation's existing roles

ASLCRM	ORGANIsation agile	ORGANIsation heavyweight
Project Manager	Scrum Master	Project Manager
Security Architect	Security Champion	Security Architect
Application Owner	Solution Owner	Application Owner
Tester	Penetration Tester	Security Tester

With these mappings, the ASC development team and domain experts do not need to know about the activities and roles in the "ORGANIsation Agile", "ORGANIsation Heavyweight" or any other specific methodology. They only need to know about the ASLCRM.

The application development groups using the "ORGANIsation Agile" methodology don't need to learn a new methodology, but only need to know the mapping of their methodology with the ASLCRM. They do not need to know about the activities and roles in the "ORGANIsation Heavyweight" or any other specific methodology. The same goes for other development groups.

5.6 Case study: Implementation of third-party ASCs in a secure development life cycle process

5.6.1 General

One characteristic of a well-managed company can be demonstrated by its reaction to adversity. Developing necessary ASCs efficiently and effectively in response to an identified attack or to identify vulnerabilities is one such example. Developing appropriate ASCs can be a challenging task. This is particularly true if there is no established secure development framework and associated ASC library. As expressed in 5.4, developing an appropriate set of ASCs in-house can be a complex and lengthy undertaking requiring specialized knowledge and resources and is likely not an appropriate course of action for the attack scenario. In this situation, an organization may choose to incorporate third party ASCs to help facilitate a quick and efficient response to the immediate threat and exposure.

5.6.2 Purpose

The purpose of this case study is to illustrate how third-party security controls for development can be adapted and integrated into an organization's application security program as a direct response to a targeted attack.

Follow on activities for full ISO/IEC 27034 (all parts) adoption may include:

a) update third-party security controls for development to ASC structure

   1) update security controls to ASC's Security activities;

   2) update security controls to ASC's Verification Measurement activities;

b) map the software development processes to the ASLCRM.

This case study presents only the initial step to incorporate ISO/IEC 27034 compliant ASCs into an organization's development processes.

5.6.3 Context

In 2008, a large company in the United States received targeted website attacks which resulted in moderate internal database damage. More importantly, the website hack provided a platform for a botnet to use the compromised website to spread malware to unsuspecting Internet visitors.

Once the breach was discovered and the botnet code identified, the company took immediate action to correct the faulty web code and begin what turned out to be a yearlong process of changing the culture of application development throughout the company. Within a matter of days, the company's Chief

Information Officer (CIO) convened meetings with IT managers from all disciplines and developed a plan to secure the company's programming code. After much discussion, they decided an internal team of skilled programmers would be more effective developing solutions because of their intimate knowledge of the company and the belief they possessed sufficient knowledge about writing secure code.

A team of the company's best developers and computer security specialists was selected to address the code vulnerability problem. This secure development (SD) team was charged with responsibility for identifying new processes to ensure that security was a feature of all development work and not just treated as an afterthought. The new SD team members represented several coding language disciplines, web development, database administration, and computer security. All members were considered the best in the company within their respective specialty and were knowledgeable about securing code.

The CIO instructed the SD team to develop recommendations for new processes that would result in the most secure code development possible. The team was to also identify tools necessary to assist developers in their jobs, recommend how to implement the new secure coding processes, identify training needs and eliminate all critical code vulnerabilities within one year.

The developed SD process is still being employed by the company largely as designed over 5 years ago.

5.6.4 Preparation phase (1.00)

During the preparation phase activities occurred to organize the secure development process, equip developers with the knowledge of secure coding techniques, acquire tools to identify code vulnerabilities, provide opportunities to share secure programming techniques, publish SD standards, publish a policy requiring compliance with the secure development process and develop metrics to measure code security status.

Table 9 — ASC Activities and outcomes for the preparation phase

Preparation phase	ASC activities	ASC outcomes	Responsible roles
1.10 Administrative Controls	1.11 Secure Development Policy created and communicated to all development staff	— Published SD policy — Individuals understanding of policy and its implementation is verified.	— CIO — SD Team — Security section — IT managers
	1.12 Personal goals added to annual performance evaluations for each person involved with development to enhance compliance with SD standards	— HR performance forms are annotated with new personal goal for complying with SDL. — Personnel evaluations include adherence to SD standards.	— CIO; — IT managers; — Team supervisors — Code developers
	1.13 Acquire and install code vulnerability scanning utility tools.	— Installation and SD team training completed and recorded prior to general IT department training.	— CIO; — SD team — Active Directory administrators — Supply Chain dept. — Network infrastructure department

Table 9 *(continued)*

Preparation phase	ASC activities	ASC outcomes	Responsible roles
	1.14 Quick Reference (QR) tip sheet (double sided, legal size, laminated) for each code developer with company's SD standards and secure coding tips from external organizations, such as OWASP.	— QR-SD sheet Designed. — QR-SD tip sheet is published and laminated. — QR-SD tip sheet is distributed to developers. — Periodic verification that code developers had the QR-SD tip sheets available for reference during development is recorded.	— SD Team — Company Publishing department — CIO — Security section
	1.15 Develop metrics for measuring progress in code vulnerability elimination status.	— Vulnerability numbers identified by code scanning utilities are charted. — Documented review and acceptance of metrics by CIO and Development managers. — Weekly and monthly publishing of vulnerabilities identified by scan utility.	— Security section — SD team — CIO — Development managers
	1.16 Establish SD portal for centrally storing standards, training, SD Team meeting minutes, metrics reports, and sharing security techniques.	— Portal for SD sharing is established. — SD portal is populated and maintained with standards, SD team meeting minutes, training, secure development references, team sharing of secure development methods, metrics reports.	— SD Team — Development managers — Team supervisors — Code developers
	1.17 Develop internal training program for all code developers.	— Minimum 3 h training program presentations are scheduled. — Training for all code developers in geographically dispersed IT development sites is scheduled.	— SD Team — CIO
	1.18 Acquire training by external subject matter experts in website secure coding.	— Vendor is selected. — 3 day intense training for web developers is scheduled and conducted.	— SD Team — CIO — Supply Chain — Code developers — Team supervisors — Development managers
Training All software development team members received appropriate training in secure development techniques	1.21 Basic Concepts: The SD Team developed the 3 h basic security training program for all IT development staff. Each technical member of a project team is expected to be conversant in concepts in the following subsections.	— Training attendance recorded/maintained either within the IT department or the company's training department. — 100 % staff participation at all training achieved	— SD Team — Development managers — Team supervisors — Code developers

Table 9 *(continued)*

Preparation phase	ASC activities	ASC outcomes	Responsible roles
	— Secure design: Attack surface reduction; Defense in depth; Principle of least privilege; Secure defaults; Positive Security Model. — Threat Modeling: Overview of threat modeling; Design to a threat model; Coding to a threat model; Testing to a threat model. — Secure Coding topics: OWASP Top Ten vulnerabilities; Integer arithmetic errors; Buffer overrun; Managed code issues (Microsoft .NET/Java). — Security Testing topics: Security testing vs. Functional testing; Risk assessment; Test methodologies; Test automation. — Privacy topics: Types of privacy data; Privacy design best practices; Risk analysis; Privacy development best practices; Privacy testing best practices.	— Makeup training session conducted for developers who could not make the scheduled training — Developers who are assigned web development have completed required additional training in website development security	
	1.22 Training to run the code scanning utility and correctly interpret results	— 100% attendance for training documented. — Makeup training session scheduled for code developers who missed scheduled training — Training on operation and interpretation of code scanner output completed	— SD Team — Development managers — Team supervisors — Code developers

Table 9 *(continued)*

Preparation phase	ASC activities	ASC outcomes	Responsible roles
	1.23 Advanced Security Concepts: additional advanced training was encouraged including: — Security design and architecture — User interface design — Security bugs in detail — Security response processes — Implementing custom threat mitigation	— Training attendance recorded/maintained either within the IT department or the company's training department — Staff participation was recorded in personnel files	— Development managers — Team supervisors — Code developers
	1.24 Continuing education: Each developer is required to attend at least one training course each year to remain current on secure coding methods	— Training attendance was recorded/maintained either within the IT department or the company's training department — Staff participation was recorded in personnel files	— Development managers — Team supervisors — Code developers
	1.25 Personal education: Developers who work in various disciplines are encouraged to read the following publications: — Writing Secure Code, Version 2 (ISBN: 0-7356-1722-8) — Latest OWASP Developers Guide (http://www.owasp.org/index.php/Category:OWASP_Guide_Project) — OWASP Top Ten (http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)	— Training attendance was recorded/maintained either within the IT department or the company's training department — Staff participation recorded in personnel files	— Development managers — Team supervisors — Code developers

5.6.5 Requirements phase (2.00)

The need to consider security "from the ground up" is a fundamental tenet of secure system development. While many development projects produce "next versions" that build on previous releases, the requirements phase and initial planning of a new release or version offers the best opportunity to build secure software.

The requirements phase is when the company has the opportunity to consider how security feature requirements will be integrated into the development process, identify key security objectives and otherwise maximize software security while minimizing disruption to plans and schedules. This is also when consideration is given to how the security features and assurance measures of the software will integrate with other software. (Interfacing with other software is a crucial consideration for meeting users' needs to integrate individual products into secure systems.)

While some security feature requirements will be identified in response to threat modeling, user requirements may also dictate the inclusion of security features in response to customer demand. Security feature requirements will also be raised by the need to comply with industry standards or regulations and should be recognized and reflected in requirements as part of the normal planning process.

Table 10 — ASC activities and outcomes for the requirements phase

Requirements phase	ASC activities	ASC outcomes	Responsible roles
2.10 Initial Security Risk Assessment	2.11 Initial Security Risk Assessment: developer prepares risk assessment and shall justify any unauthorized settings or security setting changes — Setup Questions: operating system modifications or ACL changes? — Attack Surface Questions: does code use elevated privileges or have any Internet connections? — Mobile Code Questions: is the code developed using fine-grained security technologies, such as .NET or Java? Are ActiveX controls used? — Security Feature: does the code utilize existing security mechanisms or does it contain its own security controls? Is cryptography employed?	— Documentation developed indicating that every method and property on each ActiveX control were reviewed to determine safety. — A formal documented threat model was required and recorded for the following situations: — the application has a networking interface; — the application has kernel-mode and user-mode interaction; — non-administrators interact with higher-privileged processes; — the application is a security feature for other code.	— Development managers — Team supervisors — Code developers
2.2 Privacy Impact Rating	2.21 Privacy data descriptions are: — Anonymous — PII — Sensitive PII	— Documentation showing analysis of the type of data to be handled by the code was recorded. — Privacy threshold/ ranking is established and documented.	— Team Supervisors — Code developers — Security section
	2.2 Components determined to be at the highest privacy ranking were subjected to thorough privacy analysis, often involving privacy experts, to make sure that the application did not leak private data or violate any privacy laws or regulations.	— Documentation showing analysis of the level of privacy ranking for each program and module, if 2.21 shows any data more sensitive than Privacy threshold is developed.	— Team Supervisors — Code developers — Security section

5.6.6 Design phase (3.00)

The design phase identifies the overall requirements and structure for the software and is perhaps the most important phase in secure development because this stage sets the security tone for any software development. From a security perspective, the key elements of the design phase are: define security architecture and design guidelines; document the elements of the software attack surface; conduct threat modeling. All actions in the design phase are documented as part of the computer system permanent documentation.

1) Define security architecture and design guidelines: Define the overall structure of the software from a security perspective, and identify those components where correct functioning is essential to security. Identify design techniques, such as layering (organization of software into well-defined components that are structured as to avoid circular dependencies among components), use of strongly typed language, least privilege, and minimization of attack surface, that apply to the software globally. Specifics of individual elements of the architecture will be detailed in individual design specifications, but the security architecture identifies an overall perspective on security design.

2) Document the elements of the software attack surface. Given that software will not achieve perfect security, it is important that only features that will be used by the vast majority of users be exposed to all users by default, and that those features be installed with the minimum feasible level of privilege. Measuring the elements of attack surface provides the development teams with an ongoing metric for default security and enables them to detect instances where the software has been made more susceptible to attack.

3) Conduct threat modeling. Threat modeling is conducted at a component-by-component level. The assets that the software shall manage and the interfaces by which those assets can be accessed are modeled to identify threats that can do harm to each asset and the likelihood of harm being done (an estimate of risk). Countermeasures are identified that mitigate the risk, either in the form of security features, such as encryption, or in the form of strictly controlled functioning of the software that protects the assets from harm.

Table 11 — ASC Activities and outcomes for the design phase

Design phase	ASC activities	ASC outcomes	Responsible roles
3.10 Define security architecture	3.11 Secure design principles — Positive security model (whitelist what actions are allowed) — Minimize Attack Surface Area (minimize input points) — Asset classification (protect what is being touched) — Secure By Default (security enabled throughout execution) — Principle of Least Privilege for any action within the code) — Defense in depth (multiple controls inhibit vulnerability exploitation) — Fail securely (failure follows the same execution path to disallow the operation) — Security by obscurity *never* allowed since it is not security 3.12 Separation of Duties (system administrators can set policies within the code but cannot log into the user interface as administrator) 3.13 Reducing Exposure — Compartmentalization (contain damage if something does go wrong) — Privilege separation (separate user sessions from areas only accessed with elevated rights)	— Documentation developed that security architecture analysis occurred during initial design phase. — Verification that SD standards have been followed in the design is recorded.	— Team Supervisors — Code Developers — Development Managers

Table 11 *(continued)*

Design phase	ASC activities	ASC outcomes	Responsible roles
	— Input Validation (define what data format and range is acceptable and define reaction to invalid data entry)		
3.20 Threat Risk Modeling/ Data Flow Diagrams	3.21 Identify Security Objectives, what needs to be protected: — Personal identity — Compliance — Reputation — Financial — Service Level Agreement — Privacy 3.22 Application overview — Components — Data Flows — Trust Boundaries 3.23 Create application overview — Draw the end-to-end deployment scenario — Identify technologies (servers, databases, web interfaces) — Identify application security mechanisms 3.24 Decompose the Application — Identify trust boundaries — Identify data flows — Identify entry points — Identify exit points 3.25 Utilize Threat Risk Rating (STRIDE) — Spoofing identity: Users shall not be able to act as another user. — Tampering with data: Users can change any data delivered to them. The application shall carefully check any data received from the user to identify if it is applicable. — Repudiation: Application shall have adequate repudiation controls, such as access and transaction logs.	— Program notes and team documents demonstrating that security architecture analysis occurred during design phase are recorded. — Verification by Team Supervisors and Development Managers that SD standards were followed in the design is completed.	— Team Supervisors — Code Developers — Development Managers

Table 11 *(continued)*

Design phase	ASC activities	ASC outcomes	Responsible roles
	— Information Disclosure: Applications shall include strong controls to prevent user ID tampering and minimize information stored by a browser. — Denial of Service: Applications should minimize abuse by a denial of service attack. — Every facet of the application shall be implemented to perform as little work as possible. — Elevation of Privilege: All actions shall be gated through an authorization matrix to ensure that the only the right roles can access privileged functionality.		

5.6.7 Implementation phase (4.00)

During the implementation phase, the development team codes, tests and integrates the software. Steps taken to remove security flaws or prevent their initial appearance during this phase are highly leveraged and they significantly reduce the likelihood that security vulnerabilities will make their way into the final version of the software that is released to users. The results of threat modeling provide particularly important guidance during the implementation phase. Developers pay special attention to ensuring the correctness of code that mitigates high-priority threats and focus their testing on ensuring that such threats are in fact blocked or mitigated.

The elements of the SDL that apply in the implementation phase are:

1) Apply coding and testing standards. Coding standards helped developers avoid introducing flaws that could lead to security vulnerabilities. For example, the use of safer and more consistent string handling and buffer manipulation constructs can help to avoid the introduction of buffer overrun vulnerabilities. Testing standards and best practices help to ensure that testing focuses on detecting potential security vulnerabilities rather than concentrating only on correct operation of software functions and features.

2) Apply security-testing tools including fuzzing tools. “Fuzzing” supplies structured but invalid inputs to software application programming interfaces (APIs) and network interfaces to maximize the likelihood of detecting errors that may lead to software vulnerabilities.

3) Apply static analysis code scanning tools. Tools can detect some kinds of coding flaws that result in vulnerabilities, including buffer overruns, integer overruns and uninitialized variables.

4) Conduct code reviews. Code reviews supplement automated tools and tests by applying the efforts of trained developers to examine source code and detect and remove potential security vulnerabilities. They are a crucial step in the process of removing security vulnerabilities from software during the development process.

Table 12 — ASC activities and outcomes for the implementation phase

Implementation phase	ASC activities	ASC outcomes	Responsible roles
4.10 Prepare code and verify security	4.11 Apply company SD coding standards during development — Submit new code to code vulnerability scanning tests — Correct any deficiencies identified — Conduct detailed threat modeling — Peer review of code — Encourage developers to frequently test new code for vulnerabilities (even several times daily) in order to correct any found code vulnerability early in development — Enable logging of all activity surrounding data touched by the code.	— Frequent submission of in-progress code to vulnerability detection utility is recorded. — Weekly reports from code vulnerability scanning utility to verify that work is being examined are recorded — Version control to en.sure vulnerabilities are corrected on subsequent scans is implemented.	— Team supervisors — Code developers — Development managers
	4.12 Testing of code modules with code vulnerability utility	— Security section runs weekly and monthly reports from vulnerability utility for updating the metrics spreadsheet are recoded. — Metrics tracking vulnerabilities from weekly reports from code testing utility are maintained. — High vulnerability test results and plotting to graphic display of progress are recorded.	— Security section — CIO — Team supervisors — Code developers — Development managers
4.2 Database Security	4.21 General security — Database administrators are solely responsible for database schema and structure — Identifiable user accounts are required for access to any database data — Transactions entering or modifying data within a database shall only be accomplished using the authorized program transaction function — All transactions and activity within a database will be captured in audit logs	— Documentation demonstrating that only identifiable user accounts can interact with data in database is developed. — Documentation demonstrating that DBAs cannot modify or delete database data is developed. — Documentation that only DBAs can create or modify database schema. — Verification that all changes or viewing of data in a database are logged is developed. — Verification that the database log file cannot be altered and is stored in a location that neither DBAs nor developers can access is recorded.	— Database Administrators — User account Administrators — Team supervisors — Code developers — Development managers — Security section

Table 12 *(continued)*

Implementation phase	ASC activities	ASC outcomes	Responsible roles
	4.22 Password security — User account authentication system will control user and service account passwords — Service account passwords (used by program code only with no local login capability) are expired annually — Database user accounts expired in 45 days synchronizing with the user account authentication system and other services accounts	— Documentation developed demonstrating that only user accounts can access data within a database. — Documentation developed of any service accounts rights and settings. — Documentation demonstrating that all accounts meet password expiration rules.	— Database Administrators — User account Administrators — Team supervisors — Code developers — Development managers — Security section
	4.23 Accounts — Accounts accessing databases are controlled by the user account administrator section — Programmers are not allowed to utilize generic accounts for accessing databases — Use of stored procedures is required as no direct database access is allowed — Bind variables are required to avoid SQL injection vulnerabilities	— Documentation developed that only user accounts using program transaction code can access data within the database. — Documentation developed of any service accounts rights and settings. — Documentation developed that direct access to data is not available but only through stored procedures.	— Database Administrators — User account Administrators — Team supervisors — Code developers — Development managers — Security section

5.6.8 Verification phase (5.00)

The verification phase is the point at which the application is functionally complete, enters user functional beta testing, and is subject to additional pre-production security tests. During this phase, while the application is undergoing beta testing, the team conducts a “security push” that includes security code reviews beyond those completed in the implementation phase, as well as focused security testing. This security push will be measured through the source code vulnerability utility and conduct of a manual code review.

It is important to note that code reviews and testing of high priority code (code that is part of the “attack surface” for the application) are critical to several parts of the SDL. For example, such reviews and testing should be required in the implementation phase to permit early correction of any problems and identification and correction of the source of such problems. They are also critical in the verification phase when the product is close to completion.

In this example, this ASC was design to only address high priority code.

Table 13 — ASC activities and outcomes for the verification phase

Verification phase	ASC activities	ASC outcomes	Responsible roles
5.10 manual code review	5.11 Conduct peer review — Authorization — Access Control — Input Validation — Error Handling — Session Management — Form Keys or Frequent Session Rotation (for Cross Site Request Forgery – CSRF-defense) — Proper Application Logging — Privacy	— Record developed and verified that manual code review has been conducted by team members other than the original developer. — Record developed and verified that SD standards have been complied with. — Record developed and verified that no HIGH vulnerabilities are identified in the final version of code.	— Team supervisors — Code developers — Development managers
5.20 Transactional analysis	5.21 Conduct transactional analysis — Define all the input points including data input validation from external sources — The path the input takes in the application — Any output resulting from the input received. — Cookie or state information passed between the client and server — Dynamic and static data flow analysis including where and when are variables set and how the variables are used — Error handling — Logging/auditing — Cryptography — Session management, logon/logoff	— Documentation that transactional analysis has been completed and verified.	— Team supervisors — Code developers — Development managers

5.6.9 Release phase (6.00)

During the release phase, the application should undergo a final security review (FSR). The goal of the FSR is to answer one question. “From a security viewpoint, is this application ready to deliver to customers?” The FSR is conducted prior to application completion, depending on the scope of the application. The application shall be in a stable state before the FSR, with only minimal non-security changes expected prior to release.

The FSR is not simply a pass/fail exercise, nor is the objective of the FSR to find all remaining security vulnerabilities in the software. Rather, the FSR gave the team and the company’s top management an overall picture of the security posture of the software and the likelihood that it will be able to withstand attacks after it has been released to customers. If the FSR finds a pattern of remaining vulnerabilities, the proper response is not just to fix the vulnerabilities found, but to revisit the earlier phase and take other pointed actions to address root causes (e.g. improve training, enhance tools).

Table 14 — ASC activities and outcomes for the release phase

Release phase	ASC activities	ASC outcomes	Responsible roles
6.10 Final Security Review	6.11 Document FSR findings — Identify root cause of any significant vulnerabilities. — A threat model is required for the following situations: — the application has a networking interface; — the application has kernel-mode and user-mode interaction; — non-administrators interact with higher-privileged processes; — the application is a security feature for other code.	— Documentation developed that development managers have reviewed FSR and functional testing and approved release. — Peer review of vulnerability scan results is conducted and recorded. — Every method and property on each ActiveX control were again reviewed for safety and verification is recorded. — If the code is a new product, then an additional thorough security design review is conducted and results recorded.	— Team supervisors — Code developers — Promotion controller — Development managers

5.6.10 Sustainment, support and servicing phase (7.00)

The ongoing support of production computer systems is required in order to meet changes in user needs, regulatory changes, periodic audit finding remediation, and changes in security threats. Even if circumstances to not dictate any changes to programming code, the company established a schedule to periodically rescan systems to ensure that new threats cannot affect existing software programs. This phase also establishes the goal that all individuals involved in software development should be expanding and updating their technical knowledge of security issues and techniques.

Table 15 — ASC activities and outcomes for the support and servicing phase

Support and servicing phase	ASC activities	ASC outcomes	Responsible roles
7.10 Verify security	7.11 Scheduled scans of production code — Quarterly updates to vulnerability scanning utility may indicate new vulnerabilities can be identified. — Company SD standard requires quarterly re-scan of applications that interact with Sensitive PII or financial data.	— Updates applied to the scanning utility are recorded. — Required quarterly code re-scans are completed and recorded. — A new vulnerability scan is conducted for any changes to production code.	— Security section — SD Team — Team supervisors — Code developers — Development managers — Promotion controller
	7.12 Third Party Auditors — At least annual audits of network infrastructure, databases, and applications including scans of Internet facing applications. — Compliance audits, such as SOX, NACHA, other regulatory organizations.	— Audit report developed indicating found weaknesses. — Document developed of comparison of audit report to in-house vulnerability remediation tracking to identify any unrecognized weaknesses. — Tracking of remediation and correction of audit findings recorded.	— 3rd party IT auditors — Security section — CIO — Development managers — Team supervisors
	7.13 User change requests or functional failures — Document and verify security of feature changes requested by users. — New features requested by users may require updates to the threat analysis model.	— Any changes to production code results in a new vulnerability scan.	— Team supervisors — Code developers — Promotion controller

Table 15 *(continued)*

Support and servicing phase	ASC activities	ASC outcomes	Responsible roles
	7.14 Attack detection — Where technically capable, applications will be written with special code linking the vulnerability and attack detection utility which helps detect and protect application attacks at runtime with the appropriate auditing information. — Linkage to threat detection monitoring will be done to ascertain trends in attack data.	— Verify and record that proper linkage code has been embedded in production software. — Verify and record that threat monitoring of applications is operating properly.	— Team supervisors — Code developers — Development managers — Security section
7.20 External changes	7.21 Monitor external requirements or changes — Regulatory changes — New critical vulnerabilities in the wild that potentially affect company code	— Verify and record that a functioning communications channel exists for identifying external changes that could affect IT support and/or new security requirements.	— CIO — Development managers — Security section — Business users — Legal dept.
7.30 Maintaining security knowledge	7.31 Continuing education: Each developer is required to attend at least one secure coding training course each year to remain current on secure coding methods and new threats.	— Training attendance is recorded and maintained either within the IT department or the company's training department. — Staff participation is recorded in personnel files.	— Development managers — Team supervisors — Code developers

Annex A
(informative)

XML examples for case studies in 5.2

These XML examples are presented here to help the audience to develop and communicate ASC across and/or between organizations, as define in the ISO/IEC 27034-5-1.

Table A.1 — XML example of an ASC name written in three languages

```
<?xml version="1.0" encoding="UTF-8"?>
<asc:asc-package xmlns:asc="http://iso.org/ISO27034/ASC-structure" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance" xml-asc-package-schema-version="1.0.0.0">
 <asc:package-content>
  <asc:package-identification>
   <!-- Content removed for simplification -->
  </asc:package-identification>
  <asc:asc xml-asc-schema-version="1.0.0.0">
   <asc:content>
    <asc:identification>
     <asc:uid>ORGANIsation-ASD-042</asc:uid>
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>Code Review</asc:text>
      </asc:localized-information>
      <asc:localized-information language="FR" country="CA" organization="ORGANIsation">
       <asc:text>Révision de code</asc:text>
      </asc:localized-information>
      <asc:localized-information language="RU" country="RU" organization="ORGANIsation">
       <asc:text>Анализ кода</asc:text>
      </asc:localized-information>
     </asc:name>

 <!-- Content removed for simplification -->


</asc:asc-package>
```

Table A.2 — XML example of ASC approvers and their respective signatures subset

```
<?xml version="1.0" encoding="UTF-8"?>
<asc:asc-package xmlns:asc="http://iso.org/ISO27034/ASC-structure" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance" xml-asc-package-schema-version="1.0.0.0">
 <asc:package-content>
  <asc:package-identification>
   <!-- Content removed for simplification -->
  </asc:package-identification>
  <asc:asc xml-asc-schema-version="1.0.0.0">
   <asc:content>
    <asc:identification>
     <asc:uid>ORGANIsation-ASD-042</asc:uid>
```

Table A.2 *(continued)*

```
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>Code Review</asc:text>
      </asc:localized-information>
     </asc:name>
     <asc:version number="1.3.6.0" date="2016-01-04" life-cycle-stage="ACTIVE"></as-
c:version>
     <!-- Content removed for simplification -->
    </asc:identification>
    <asc:objective>
     <!-- Content removed for simplification -->
    </asc:objective>
    <asc:security-activity>
     <!-- Content removed for simplification -->
    </asc:security-activity>
    <asc:verification-measurement>
     <!-- Content removed for simplification -->
    </asc:verification-measurement>
   </asc:content>
   <asc:approval-e-signatures>
    <asc:approval-stage>
     <asc:date>2011-09-23</asc:date>
     <asc:approval-stage-type>CREATION_REQUEST</asc:approval-stage-type>
     <asc:approver>
      <asc:name>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Herbert George Wells</asc:text>
       </asc:localized-information>
      </asc:name>
      <asc:coordinate location-name="Office">
       <asc:organization>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>ORGANIsation inc.</asc:text>
        </asc:localized-information>
       </asc:organization>
       <asc:department>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Application Security Department</asc:text>
        </asc:localized-information>
       </asc:department>
       <asc:emails>
        <asc:email type="Office">JVernes@ORGANIsation.com</asc:email>
       </asc:emails>
       <asc:country>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Canada</asc:text>
        </asc:localized-information>
       </asc:country>
      </asc:coordinate>
     </asc:approver>
```

Table A.2 *(continued)*

```
     <asc:approver-e-signature>
      <asc:e-signature-param>HGWells@ORGANIsation.com</asc:e-signature-param>
      <asc:e-signature-param>Version: PGP Universal 3.2.0 (Build 1950)</asc:e-signa-
ture-param>
      <asc:e-signature-param>Charset: us-ascii</asc:e-signature-param>
      <asc:e-signature-data>wsBVAwUBT06tfp/JsGz ... fwymKtSR63wb7QQ===x0gO</asc:e-signa-
ture-data>
     </asc:approver-e-signature>
    </asc:approval-stage>
    <asc:approval-stage>
     <asc:date>2012-01-11</asc:date>
     <asc:approval-stage-type>VALIDATION</asc:approval-stage-type>
     <asc:approver>
      <asc:name>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Arthur C. Clarke</asc:text>
       </asc:localized-information>
      </asc:name>
      <asc:coordinate location-name="Office">
       <asc:organization>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>ORGANIsation inc.</asc:text>
        </asc:localized-information>
       </asc:organization>
       <asc:department>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Application Security Department</asc:text>
        </asc:localized-information>
       </asc:department>
       <asc:emails>
        <asc:email type="Office">ACClarke@ORGANIsation.com</asc:email>
       </asc:emails>
       <asc:country>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Canada</asc:text>
        </asc:localized-information>
       </asc:country>
      </asc:coordinate>
     </asc:approver>
    </asc:approval-stage>
    <asc:approval-stage>
     <asc:date>2012-05-10</asc:date>
     <asc:approval-stage-type>DEVELOPMENT</asc:approval-stage-type>
     <asc:approver>
      <asc:name>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Frank Herbert</asc:text>
       </asc:localized-information>
      </asc:name>
      <asc:coordinate location-name="Office">
```

Table A.2 *(continued)*

```
      <asc:organization>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>ORGANIsation inc.</asc:text>
       </asc:localized-information>
      </asc:organization>
      <asc:department>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Application Security Department</asc:text>
       </asc:localized-information>
      </asc:department>
      <asc:emails>
       <asc:email type="Office">FHerbert@ORGANIsation.com</asc:email>
      </asc:emails>
      <asc:country>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Canada</asc:text>
       </asc:localized-information>
      </asc:country>
     </asc:coordinate>
    </asc:approver>
   </asc:approval-stage>
   <asc:approval-stage>
    <asc:date>2012-09-07</asc:date>
    <asc:approval-stage-type>VERIFICATION</asc:approval-stage-type>
    <asc:approver>
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>Ray Bradbury</asc:text>
      </asc:localized-information>
     </asc:name>
     <asc:coordinate location-name="Office">
      <asc:organization>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>ORGANIsation inc.</asc:text>
       </asc:localized-information>
      </asc:organization>
      <asc:department>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Application Security Department</asc:text>
       </asc:localized-information>
      </asc:department>
      <asc:emails>
       <asc:email type="Office">RBradbury@ORGANIsation.com</asc:email>
      </asc:emails>
      <asc:country>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Canada</asc:text>
       </asc:localized-information>
      </asc:country>
     </asc:coordinate>
```

Table A.2 *(continued)*

```
    </asc:approver>
   </asc:approval-stage>
   <asc:approval-stage>
    <asc:date>2012-09-17</asc:date>
    <asc:approval-stage-type>VERIFICATION</asc:approval-stage-type>
    <asc:approver>
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>William Gibson</asc:text>
      </asc:localized-information>
     </asc:name>
     <asc:coordinate location-name="Office">
      <asc:organization>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>ORGANIsation inc.</asc:text>
       </asc:localized-information>
      </asc:organization>
      <asc:department>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Application Security Department</asc:text>
       </asc:localized-information>
      </asc:department>
      <asc:emails>
       <asc:email type="Office">WGibson@ORGANIsation.com</asc:email>
      </asc:emails>
      <asc:country>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Canada</asc:text>
       </asc:localized-information>
      </asc:country>
     </asc:coordinate>
    </asc:approver>
   </asc:approval-stage>
   <asc:approval-stage>
    <asc:date>2012-10-07</asc:date>
    <asc:approval-stage-type>APPROVAL</asc:approval-stage-type>
    <asc:approver>
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>Robert Heinlein</asc:text>
      </asc:localized-information>
     </asc:name>
     <asc:coordinate location-name="Office">
      <asc:organization>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>ORGANIsation inc.</asc:text>
       </asc:localized-information>
      </asc:organization>
      <asc:department>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
```

Table A.2 *(continued)*

```
          <asc:text>Application Security Department</asc:text>
         </asc:localized-information>
        </asc:department>
        <asc:emails>
         <asc:email type="Office">RHeinlein@ORGANIsation.com</asc:email>
        </asc:emails>
        <asc:country>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Canada</asc:text>
         </asc:localized-information>
        </asc:country>
       </asc:coordinate>
      </asc:approver>
      <asc:approver-e-signature>
       <asc:e-signature-param>RHeinlein@ORGANIsation.com</asc:e-signature-param>
       <asc:e-signature-param>Version: PGP Universal 3.2.0 (Build 1950)</asc:e-signa-
ture-param>
       <asc:e-signature-param>Charset: us-ascii</asc:e-signature-param>
       <asc:e-signature-data> Gz86uwqAQgcAp3fe ... B45vjfqO4Vq/woF</asc:e-signature-data>
      </asc:approver-e-signature>
     </asc:approval-stage>
     <asc:approval-stage>
      <asc:date>2012-10-17</asc:date>
      <asc:approval-stage-type>OWNERS_FINAL_APPROVAL</asc:approval-stage-type>
      <asc:approver>
       <asc:name>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Douglas Adams</asc:text>
        </asc:localized-information>
       </asc:name>
       <asc:coordinate location-name="Office">
        <asc:organization>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>ORGANIsation inc.</asc:text>
         </asc:localized-information>
        </asc:organization>
        <asc:department>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Application Security Department</asc:text>
         </asc:localized-information>
        </asc:department>
        <asc:emails>
         <asc:email type="Office">DAdams@ORGANIsation.com</asc:email>
        </asc:emails>
        <asc:country>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Canada</asc:text>
         </asc:localized-information>
        </asc:country>
       </asc:coordinate>
```

Table A.2 *(continued)*

```
     </asc:approver>
     <asc:approver-e-signature>
      <asc:e-signature-param>DAdams@ORGANIsation.com</asc:e-signature-param>
      <asc:e-signature-param>Version: PGP Universal 3.2.0 (Build 1950)</asc:e-signa-
ture-param>
      <asc:e-signature-param>Charset: us-ascii</asc:e-signature-param>
      <asc:e-signature-data>bgHi0LLo+OyTx9T4uGCyx ... A09CKT4aIsmvtOFLvtuB</asc:e-signa-
ture-data>
     </asc:approver-e-signature>
    </asc:approval-stage>
    <asc:approval-stage>
     <asc:date>2012-11-06</asc:date>
     <asc:approval-stage-type>PUBLISHED_FOR_TRAINING</asc:approval-stage-type>
     <asc:approver>
      <asc:name>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Isaac Asimov</asc:text>
       </asc:localized-information>
      </asc:name>
      <asc:coordinate location-name="Office">
       <asc:organization>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>ORGANIsation inc.</asc:text>
        </asc:localized-information>
       </asc:organization>
       <asc:department>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Application Security Department</asc:text>
        </asc:localized-information>
       </asc:department>
       <asc:emails>
        <asc:email type="Office">IAsimov@ORGANIsation.com</asc:email>
       </asc:emails>
       <asc:country>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Canada</asc:text>
        </asc:localized-information>
       </asc:country>
      </asc:coordinate>
     </asc:approver>
    </asc:approval-stage>
    <asc:approval-stage>
     <asc:date>2013-03-06</asc:date>
     <asc:approval-stage-type>ACTIVE</asc:approval-stage-type>
     <asc:approver>
      <asc:name>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Mary Shelley</asc:text>
       </asc:localized-information>
      </asc:name>
```

Table A.2 *(continued)*

```
      <asc:coordinate location-name="Office">
       <asc:organization>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>ORGANIsation inc.</asc:text>
        </asc:localized-information>
       </asc:organization>
       <asc:department>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Application Security Department</asc:text>
        </asc:localized-information>
       </asc:department>
       <asc:emails>
        <asc:email type="Office">MShelley@ORGANIsation.com</asc:email>
       </asc:emails>
       <asc:country>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Canada</asc:text>
        </asc:localized-information>
       </asc:country>
      </asc:coordinate>
     </asc:approver>
    </asc:approval-stage>
   </asc:approval-e-signatures>
  </asc:asc>
 </asc:package-content>
 <asc:package-editor-e-signature>
  <!-- Content removed for simplification -->
 </asc:package-editor-e-signature>
</asc:asc-package>
```

Table A.3 — XML example of an ASC children definition

```
<?xml version="1.0" encoding="UTF-8"?>
<asc:asc-package xmlns:asc="http://iso.org/ISO27034/ASC-structure" xmlns:x-
si="http://www.w3.org/2001/XMLSchema-instance" xml-asc-package-schema-ver-
sion="1.0.0.0">
 <asc:package-content>
  <asc:package-identification>
   <!-- Content removed for simplification -->
  </asc:package-identification>
  <asc:asc xml-asc-schema-version="1.0.0.0">
   <asc:content>
    <asc:identification>
     <asc:uid>ORGANIsation-ASD-042</asc:uid>
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGAN-
Isation">
       <asc:text>Code Review</asc:text>
      </asc:localized-information>
     </asc:name>
```

Table A.3 *(continued)*

```
     <asc:version number="1.3.6.0" date="2016-01-04" life-cycle-stage="AC-
TIVE"></asc:version>
     <asc:date>2016-01-04</asc:date>
     <asc:description>
      <asc:localized-information language="EN" country="CA" organization="ORGAN-
Isation">
       <asc:text>This ASC is used to help developers to perform a code review
control for JAVA applications.</asc:text>
      </asc:localized-information>
     </asc:description>
     <asc:children>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-043</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="OR-
GANIsation">
         <asc:text>Code Classification</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-044</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="OR-
GANIsation">
         <asc:text>Basic Automatic Code Review</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-045</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="OR-
GANIsation">
         <asc:text>Advanced Automatic Code Review</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-046</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="OR-
GANIsation">
         <asc:text>Manual Code Review</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
     </asc:children>
    </asc:identification>
```

Table A.3 *(continued)*

```
    <asc:objective>
     <!-- Content removed for simplification -->
    </asc:objective>
    <asc:security-activity>
     <!-- Content removed for simplification -->
    </asc:security-activity>
    <asc:verification-measurement>
     <!-- Content removed for simplification -->
    </asc:verification-measurement>
   </asc:content>
   <asc:approval-e-signatures>
    <!-- Content removed for simplification -->
   </asc:approval-e-signatures>
  </asc:asc>
 </asc:package-content>
 <asc:package-editor-e-signature>
  <!-- Content removed for simplification -->
 </asc:package-editor-e-signature>
</asc:asc-package>
```

Table A.4 — XML example of the ASC ORGANIsation-ASD-042: Code review, identification

```
<?xml version="1.0" encoding="UTF-8"?>
<asc:asc-package xmlns:asc="http://iso.org/ISO27034/ASC-structure" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance" xml-asc-package-schema-version="1.0.0.0">
 <asc:package-content>
  <asc:package-identification>
   <!-- Content removed for simplification -->
  </asc:package-identification>
  <asc:asc xml-asc-schema-version="1.0.0.0">
   <asc:content>
    <asc:identification>
     <asc:uid>ORGANIsation-ASD-042</asc:uid>
     <asc:name>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>Code Review</asc:text>
      </asc:localized-information>
     </asc:name>
     <asc:version number="1.3.6.0" date="2013-03-06" life-cycle-stage="ACTIVE">
      <asc:revision-note>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Categorization ASC was added in this version to ensure a homogenous ap-
plication's class classification.</asc:text>
       </asc:localized-information>
      </asc:revision-note>
     </asc:version>
     <asc:date>2016-01-04</asc:date>
     <asc:description>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
```

Table A.4 *(continued)*

```
        <asc:text>This ASC is used to help developers to perform a code review control for
JAVA applications.</asc:text>
       </asc:localized-information>
      </asc:description>
      <asc:author>
       <asc:name>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Jules Verne</asc:text>
        </asc:localized-information>
       </asc:name>
       <asc:coordinate location-name="Office">
        <asc:organization>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>ORGANIsation inc.</asc:text>
         </asc:localized-information>
        </asc:organization>
        <asc:department>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Application Security Department</asc:text>
         </asc:localized-information>
        </asc:department>
        <asc:emails>
         <asc:email type="Office">JVernes@ORGANIsation.com</asc:email>
        </asc:emails>
        <asc:phones>
         <asc:phone type="Office">+1.234.567.8901</asc:phone>
        </asc:phones>
        <asc:street-address>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>1234 Street ave W</asc:text>
         </asc:localized-information>
        </asc:street-address>
        <asc:city>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Beautiful city</asc:text>
         </asc:localized-information>
        </asc:city>
        <asc:provice-state>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Quebec</asc:text>
         </asc:localized-information>
        </asc:provice-state>
        <asc:country>
         <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
          <asc:text>Canada</asc:text>
         </asc:localized-information>
        </asc:country>
       </asc:coordinate>
      </asc:author>
      <asc:owner>
```

Table A.4 *(continued)*

```
      <asc:name>
       <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
        <asc:text>Douglas Adams</asc:text>
       </asc:localized-information>
      </asc:name>
      <asc:coordinate location-name="Office">
       <asc:organization>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>ORGANIsation inc.</asc:text>
        </asc:localized-information>
       </asc:organization>
       <asc:department>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Application Security Department</asc:text>
        </asc:localized-information>
       </asc:department>
       <asc:emails>
        <asc:email type="Office">DAdams@ORGANIsation.com</asc:email>
       </asc:emails>
       <asc:phones>
        <asc:phone type="Office">+1.109.876.5432</asc:phone>
       </asc:phones>
       <asc:street-address>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>1234 Street ave W</asc:text>
        </asc:localized-information>
       </asc:street-address>
       <asc:city>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Beautiful city</asc:text>
        </asc:localized-information>
       </asc:city>
       <asc:provice-state>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Quebec</asc:text>
        </asc:localized-information>
       </asc:provice-state>
       <asc:country>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Canada</asc:text>
        </asc:localized-information>
       </asc:country>
      </asc:coordinate>
     </asc:owner>
     <asc:children>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-043</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Code Classification</asc:text>
```

Table A.4 *(continued)*

```
        </asc:localized-information>
       </asc:description>
      </asc:child>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-044</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Basic Automatic Code Review</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-045</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Advanced Automatic Code Review</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
      <asc:child>
       <asc:ref-asc>ORGANIsation-ASD-046</asc:ref-asc>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Manual Code Review</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:child>
     </asc:children>
    </asc:identification>
    <asc:objective>
     <!-- Content removed for simplification -->
    </asc:objective>
    <asc:security-activity>
     <!-- Content removed for simplification -->
    </asc:security-activity>
    <asc:verification-measurement>
     <!-- Content removed for simplification -->
    </asc:verification-measurement>
   </asc:content>
   <asc:approval-e-signatures>
    <!-- Content removed for simplification -->
   </asc:approval-e-signatures>
  </asc:asc>
 </asc:package-content>
 <asc:package-editor-e-signature>
  <!-- Content removed for simplification -->
 </asc:package-editor-e-signature>
</asc:asc-package>
```

Table A.5 — XML example of the ASC ORGANIsation-ASD-042: Code review, objective

```
<?xml version="1.0" encoding="UTF-8"?>
<asc:asc-package xmlns:asc="http://iso.org/ISO27034/ASC-structure" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance" xml-asc-package-schema-version="1.0.0.0">
 <asc:package-content>
  <asc:package-identification>
   <!-- Content removed for simplification -->
  </asc:package-identification>
  <asc:asc xml-asc-schema-version="1.0.0.0">
   <asc:content>
    <asc:identification>
     <asc:uid>ORGANIsation-ASD-042</asc:uid>
     <!-- Content removed for simplification -->
    </asc:identification>
    <asc:objective>
     <asc:objective-description>
      <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
       <asc:text>Top-level ASC whose objective is to group the various leaf ASCs related
to code review in Java.</asc:text>
      </asc:localized-information>
     </asc:objective-description>
     <asc:requirements-addressed>
      <asc:requirement>
       <!-- Content removed for simplification -->
      </asc:requirement>
     </asc:requirements-addressed>
     <asc:assigned-levels-of-trust>
      <asc:level-of-trust-ref>45F736847</asc:level-of-trust-ref>
      <asc:level-of-trust-ref>76878654</asc:level-of-trust-ref>
      <asc:level-of-trust-ref>9876D54</asc:level-of-trust-ref>
      <asc:level-of-trust-ref>4576825</asc:level-of-trust-ref>
      <asc:level-of-trust-ref>989A67547</asc:level-of-trust-ref>
      <asc:level-of-trust-ref>932564543</asc:level-of-trust-ref>
     </asc:assigned-levels-of-trust>
     <asc:contexts-of-use>
      <asc:context type="Regulatory">TECHNOLOGICAL</asc:context>
     </asc:contexts-of-use>
     <asc:levels-of-trust-range>
      <asc:level-of-trust>
       <asc:level-of-trust-ref>45F736847</asc:level-of-trust-ref>
       <asc:level>0</asc:level>
       <asc:label>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Baseline</asc:text>
        </asc:localized-information>
       </asc:label>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>All ORGANIsation's applications shall comply with this Level of
Trust.</asc:text>
        </asc:localized-information>
       </asc:description>
```

Table A.5 *(continued)*

```
      </asc:level-of-trust>
      <asc:level-of-trust>
       <asc:level-of-trust-ref>76878654</asc:level-of-trust-ref>
       <asc:level>1</asc:level>
       <asc:label>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Isolated – Local network only</asc:text>
        </asc:localized-information>
       </asc:label>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>This Level of Trust is appropriate for applications used on isolated
corporate networks, with no connection to external networks.</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:level-of-trust>
      <asc:level-of-trust>
       <asc:level-of-trust-ref>9876D54</asc:level-of-trust-ref>
       <asc:level>2</asc:level>
       <asc:label>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Low – Internet, public information only</asc:text>
        </asc:localized-information>
       </asc:label>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>This Level of Trust is appropriate for Internet-facing applications
sharing public information without any privacy concern.</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:level-of-trust>
      <asc:level-of-trust>
       <asc:level-of-trust-ref>4576825</asc:level-of-trust-ref>
       <asc:level>3</asc:level>
       <asc:label>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Medium – Internet, corporate users</asc:text>
        </asc:localized-information>
       </asc:label>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>This Level of Trust is appropriate for Internet-facing, transactional
applications used by corporate users, allowing access to corporate services, user files
and/or transactions under 5,000 $.</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:level-of-trust>
      <asc:level-of-trust>
       <asc:level-of-trust-ref>989A67547</asc:level-of-trust-ref>
       <asc:level>4</asc:level>
       <asc:label>
```

Table A.5 *(continued)*

```
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>High – Secure transactions and privacy protection over Internet</asc:text>
        </asc:localized-information>
       </asc:label>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>This Level of Trust is appropriate for Internet-facing, transactional
applications, used by corporate  users, allowing access to user private information and/
or transactions from $5 000 to $25 000</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:level-of-trust>
      <asc:level-of-trust>
       <asc:level-of-trust-ref>932564543</asc:level-of-trust-ref>
       <asc:level>5</asc:level>
       <asc:label>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>Private</asc:text>
        </asc:localized-information>
       </asc:label>
       <asc:description>
        <asc:localized-information language="EN" country="CA" organization="ORGANIsation">
         <asc:text>This Level of Trust is appropriate for transactional applications
requiring highly secure transactions, privileged access and/or secure critical storage.
Access to critical information and/or transactions over $25 000 is authorized.</asc:text>
        </asc:localized-information>
       </asc:description>
      </asc:level-of-trust>
     </asc:levels-of-trust-range>
     <asc:pre-conditions>
      <asc:condition>
       <!-- Content removed for simplification -->
      </asc:condition>
     </asc:pre-conditions>
    </asc:objective>
    <asc:security-activity>
     <!-- Content removed for simplification -->
    </asc:security-activity>
    <asc:verification-measurement>
     <!-- Content removed for simplification -->
    </asc:verification-measurement>
   </asc:content>
   <asc:approval-e-signatures>
    <!-- Content removed for simplification -->
   </asc:approval-e-signatures>
  </asc:asc>
 </asc:package-content>
 <asc:package-editor-e-signature>
  <!-- Content removed for simplification -->
 </asc:package-editor-e-signature>
</asc:asc-package>
```