2014 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) TECHNICAL SESSION
AUGUST 12-14, 2014 – NOVI, MICHIGAN

SOFTWARE SAFETY AND SECURITY BEST PRACTICES: A CASE STUDY FROM AEROSPACE

Paul Skentzos
DornerWorks, Ltd.
Grand Rapids, MI

ABSTRACT

Software safety and security flaws are costly. Defects found in software systems after they are deployed have always been costly to fix. However, the importance placed on software developed today as a key technology for functionality and control of hardware results in even higher costs when defects and errors cause loss of materiel, and in some cases, personnel. Serious safety and security flaws have ramifications that often go beyond tangible dollar amounts or data mishap issues, such as trustworthiness. Safety has always been a major focus for the aviation community, where engineers follow strict practices that adhere to Federal Aviation Administration (FAA) guidelines. Security is a more recent concern. We have found that processes used for safety can often be applied to security.

In this paper we describe the aviation community's DO-178 processes for safety and how they might be tailored to the land vehicle community. We will use the development of our hypervisor as a case study of how we built a system using best practices for both safety and security processes.

INTRODUCTION

As customers demand more features in smaller, lighter, and cheaper systems, engineers have started replacing federated systems, or separate computing devices for individual features, with integrated systems running on a common computing platform. The result is increased complexity in order to achieve lower size, weight, and power (SWaP) in integrated systems. Engineers are writing and modifying more software than ever before and with that come increased risks of failure relating to safety and security. One need not search long to find examples of safety related failures. More recent examples highlight the risks to a company's reputation and, potentially, bottom line [1].

Security of systems has only recently started making headlines as various components now have the capability to be connected to the internet and other external devices. Two examples highlight the potential danger. In 2013, a security researcher demonstrated how one could remotely attack and take full control of an aircraft [2]. Then in 2014, Defense Advanced Research Projects Agency (DARPA) funded researchers showed how they could easily take complete control of practically any automobile and disrupt the steering, braking, and other critical functionality [3]. Both of these cases highlight that failures in security could result in catastrophic failures of safety.

This paper will describe security and some of the processes the government looks for in a secure system, but will be limited here to showing how following guidelines for safety can lead to a more secured system. We will briefly discuss aviation safety in the context of DO-178. We will discuss the automotive safety specification, ISO 26262, as a representative example of how DO-178 easily maps to other safety specifications. Finally, all of this will be looked at in the context of the development of the ARINC 653 Real-Time Linux on Xen (ARLX) hypervisor developed by DornerWorks using established DO-178 safety guidelines.

A BRIEF HISTORY OF SECURITY

Ensuring security has become more complex with large, inter-connected systems. Common security considerations include ensuring that unsecured and secured data remain properly separated, and keeping personnel information from unauthorized access. The Department of Defense (DoD), the National Institute of Standards and Technology (NIST), and the National Security Agency (NSA) created the Trusted Computer System Evaluation Criteria (TCSEC) that eventually became a part of Common Criteria (CC).

COMMON CRITERIA

The Common Criteria consists of a set of requirements that when followed, provides some assurance that the implementation of the computer security product has been conducted in a rigorous and repeatable manner at a level necessary for the target of use. Specifically, information assurance (IA) products can be certified in accordance with the Common Criteria. In the United States, the National Information Assurance Partnership (NIAP) performs Common Criteria evaluations.

The CC is used as the baseline for all government security certifications. Independent testing laboratories conduct the testing and provide certification. The CC certification does not guarantee security, but it does ensure that what the organization says about the security attributes of the system is true. This philosophy can be seen in DO-178 safety certifications that will be discussed later.

SEPARATION KERNEL PROTECTION PROFILE

Any system that is certified under the Common Criteria must conform to a Security Target (ST) that may be compliant with a Protection Profile (PP). The US Government has a Protection Profile for Separation Kernels in Environments Requiring High Robustness. This is often referred too as the Separation Kernel Protection Profile or SKPP. The SKPP is intended to isolate and separate partitions and control information flow between different security domains. The SKPP must prove that there are no channels for information flow between domains other than data flow that is explicitly defined.

It is interesting to note that conformance to the SKPP does not ensure a secured product. "... *conformance to this protection profile, by itself, does not offer sufficient confidence that national security information is appropriately protected in the context of a larger system in which the conformant product is integrated.*" [4]

In light of this and other issues such as the high cost of certification and increased complexity of systems, the NSA has deprecated the SKPP [5]. While the NSA is no longer supporting certification of the SKPP, they continue to support the sound design for security-critical systems.

FORMAL METHODS

Formal Methods use mathematical logic to model and verify requirements of computing systems. Formal methods is required as part of the common criteria certification for Evaluation Assurance Level (EAL) of six (6) or higher. There are seven levels of EAL and are described in the table below.

EAL	Definition	Explanation
1	Functionally	Applicable to

	Tested	systems where threats to security are not viewed as serious
2	Structurally tested	Requires a low to moderate level of independently assured security
3	Methodically tested and checked	Requires a moderate level of independently assured security and thorough investigation of TOE
4	Methodically designed, tested, and reviewed	Applicable when moderate to high level of independently assured security is required
5	Semi-formally designed and tested	Applicable where a high level of independently assured security in a planned development and a rigorous development approach is needed
6	Semiformal verified design and tested	Applicable to the development of security TOEs for application in high risk situations.
7	Formally verified design and tested	Applicable to TOEs with tightly focused security functionality that is amenable to extensive formal analysis

Table 1: EAL Levels

As one might image the costs for increased levels of EAL increase with the associated level of complexity and completeness required for the classifications. The picture below shows the cost range depending on the level or EAL certification.

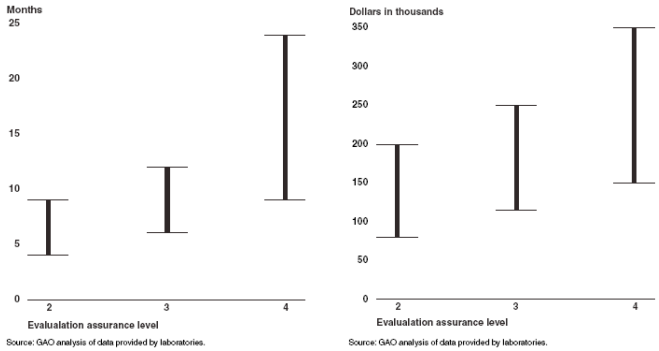

Figure 1:EAL Certification Costs [6]

Formal methods are very good at addressing security requirements, specifically requirements of the "shall not" type. "shall not" requirements are difficult in general because it is impossible to thoroughly test a negative requirement. The safety world has traditionally focused on testing to prove adequacy. Therefore, where safety concerns itself with "shall" requirements, security and formal methods concerns itself with "shall not" requirements.

BRIEF HISTORY OF DO-178

Avoiding aircraft accidents has always been a priority of the aviation community. However the increased use of software in aviation systems resulted in the need for a set of industry accepted standards for airworthiness requirements that resulted in the RTCA/DO-178 specification, first released in 1982.

COMPARISON TO OTHER SAFETY MODELS

Figure 2 below describes the design flow mapping of DO-178 to ISO 26262. Given the maturity of DO-178, one will find that DO-178 safety processes can be tailored to a number of other security processes such as ISO 26262. As one can see in Figure 3, the artifact output from DO-178 can map to those from other safety guidelines, in this specific case, ISO 26262.

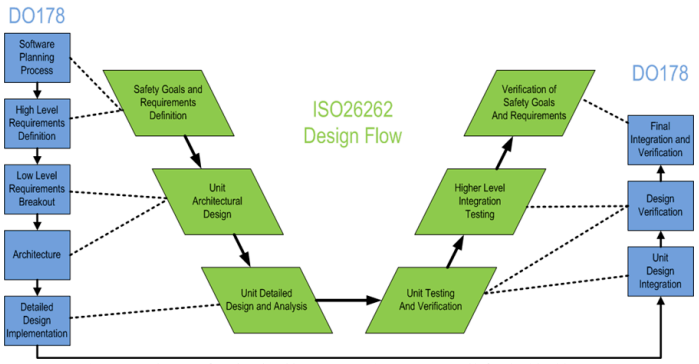

Figure 2: DO-178 Comparison to ISO 26262

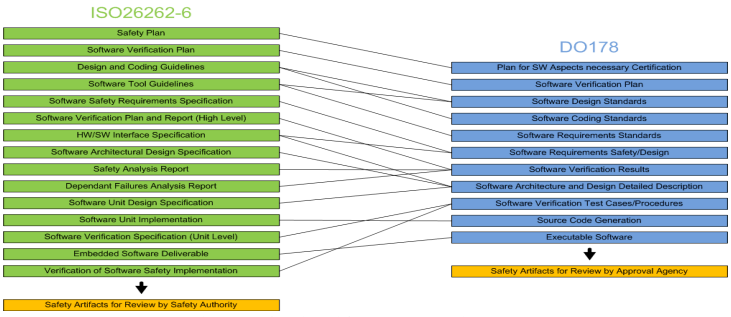

Figure 3: Artifact Mapping

ARINC 653 REAL-TIME LINUX ON XEN

ARINC 653 Real-time Linux on Xen or ARLX is the Xen based, open-source; type 1 hypervisor DornerWorks developed with both internal funding and SBIR funding from the US Navy and DARPA. Figure 4 shows the general architecture of the ARLX Hypervisor.

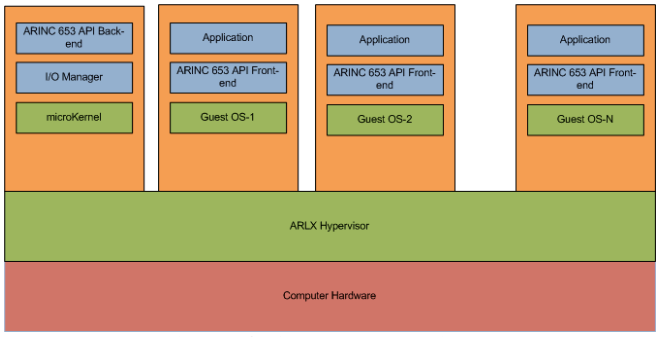

Figure 4: ARLX Hypervisor

ARLX is developed for systems that require a high degree of safety and security. Safety is achieved through following DO-178C processes for level A as tailored by DornerWorks and by implementing the ARINC 653 software partitioning specification that provides deterministic use of computer resources. Strictly following the ARINC 653 standard and then judiciously applying formal methods analysis to the target of evaluation gives the system an initially good level of security due to the restrictions that ARINC 653 places on the computer resources.

SAFETY PROCESS FOR ARLX

As part of the Navy SBIR, DornerWorks proposed that we would develop a safe and secure hypervisor along with all the necessary processes and artifacts that one would need to truly validate safety and security. In the development of these plans and standards, we showed how they map directly back to the DO-178C standard.

DO-178C has five safety levels shown in the table below.

DO-178 Level	Definition	Description
A	Catastrophic	Prevents

		continued safe flight or landing, many fatal injuries
B	Hazardous/Severe	Potential fatal injuries to a small number of occupants
C	Major	Impairs crew efficiency, discomfort, or possible injuries to occupants
D	Minor	Reduced aircraft safety margins, but well within crew capability
E	No Effect	Does not affect the safety of the aircraft at all

Table 2: FAA DO-178C Criticality Levels

DO-178 requires five plans and three development standards.

- Plans
  - Plan for Software Aspects of Certification (PSAC)
  - Software Development Plan (SDP)
  - Software Verification Plan (SVP)
  - Software Configuration Management Plan (SCMP)
  - Software Quality Assurance Plan (SQAP)
- Development Standards
  - Software Requirements Standards (SRS)
  - Software Design Standards (SDS)
  - Software Coding Standards (SCS)

Inspections of these artifacts are conducted in events called Stages of Involvement (SOI). The SOI is the opportunity for the Designated Engineering Representative (DER) to inspect artifacts and for the development and quality assurance teams to respond and make correction to artifacts, documents, and processes. SOIs are normally conducted on systems and not on specific software tools. Since there was no program of record that ARLX was being implemented on, mock SOIs were conducted. That is, SOIs were handled as if there was a complete system, although this was not the case.

The planning documents were tailored specifically for the ARLX project. These documents are described below. The development standards are already established standards that DornerWorks practices and has documented for all projects at DornerWorks. The development standards will not be discussed further except to note that the company has established practices for coding, design, and requirements.

PLAN FOR SOFTWARE ASPECTS OF CERTIFICATION

The PSAC is the top-level safety certification plan created by the developer and agreed upon with the certifying authority.

The PSAC is normally one of the first documents written and submitted to the DER. This is done in order to ensure that the project has established compliant processes and other appropriate project items so that the product's safety assurance can be demonstrated by appropriate objective evidence.

The PSAC for the ARLX program was formally reviewed internally and then submitted to the DER for her review. The DER made comments on PSAC that were corrected or explained during the first SOI.

SOFTWARE DEVELOPMENT PLAN

This document is written for the developers working on the project so that they understand how the development should proceed. In the SDP for the ARLX project we documented specific procedures and processes that we would follow,

including the software lifecycle model we used. Specifically we described the documents that discussed the standards we would follow such as for software requirements and software design. We also discussed project specific coding standards in the document.

The software lifecycle was also discussed in terms of planning, configuration management, quality assurance, verification of the software, and how we would address compliance and document approval for deviation of some issues from the DER inspections.

Lastly we described the development environment that engineers would be expected to use while on the project.

Additionally, much of this information was documented on our internal wiki for everyone to view. The use of the wiki was to document some of these plans in more of a "how-to" mechanism making it easier for engineers to consistently refer to the plan and make comments on lessons learned while following the plan.

SOFTWARE VERIFICATION PLAN

The SVP is the document used to describe the verification of the software and includes inspection, analysis, and test processes.

In the SVP for ARLX, we described how we would ensure verification independence as required by DO-178. This was accomplished through the use of peer reviews and ensuring that the peer reviewing the material did not actually write the article under review.

The SVP also discussed the methods for testing and analysis, tools used to verify software, and since we were implementing the ARINC 653 partitioning standard, how we would verify the space and time partitioning correctness of the software.

The process names that are used internally to DornerWorks were often different from the DO-178C processes names. We had to ensure that we were following the DO-178C processes, so we provided a mapping of the DO-178C process names to the DornerWorks process name. This information is shown in Table 3 below and was documented in the SVP.

DO-178C Process Name	DornerWorks Process Name
Software Planning Process	Project Planning
Software Requirements Process	Requirements Definition (HLRs)
Software Design Process	Design & Development (Architecture & LLRs)
Software Coding Process	Design & Development (Code)
Integration Process	Design & Development (Integration)
Software Verification Process	Verification & Validation
Software Configuration Management Process	Configuration Management
Software Quality Assurance Process	Quality Assurance
Certification Liaison Process	N/A

Table 3: Process Mapping

SOFTWARE CONFIGURATION MANAGEMENT PLAN

The purpose of the SCMP is to establish the Software Configuration Management (SCM) related policies and methods to be adopted and implemented during the lifecycle development of the program.

The SCMP was written for the ARLX system and is fully implemented for development of the program. The SCMP documents the plan of how the software artifacts associated with this project are identified, configured, controlled, archived, and tracked.

The SCMP specifically discusses the various configuration management tools and how each tool will be used. In the case of ARLX, we mentioned using the open source tools Subversion and Mercurial (and eventually git) for tracking documents and code. One thing to note about this is that we started out using Mercurial since that was the source code control tool of choice for the

Xen community. However, the Xen community formally changed to the git source code control tool and similarly, we did do. This required a modification of the SCMP document because of this change. We also tracked issues and problem reports using a commercial product call Jira. The document also describes how parts numbers are established in addition to change management.

SOFTWARE QUALITY ASSURANCE PLAN

The SQAP documents software lifecycle processes and their output for assurance that the objects set fourth in the planning documents are satisfied, deficiencies are detected, evaluated, tracked, and resolved, and that the software product and software lifecycle data conform to the certification requirements.

For the ARLX project, the software quality assurance activities are documented in the SQAP and in Table 4.

Software Quality Assurance Activities
Prepare and Maintain the Software Quality Assurance Plan (or delegate)
Change Authority
Review planning documents (PSAC, SDP, SVP, SQAP and SCMP)
Audit reviews of HLRs, Architecture, LLRs, & Source Code
Audit reviews of High-Level Verification Cases and Procedures, Low-Level Verification Cases and Procedures, and verification results
Audit execution of High-Level Verification Cases and Procedures & Low-Level Verification Cases and Procedures
Perform Software Conformity Review
Submit certification artifacts to the customer and/or Certification Authority

Table 4: Software QA Activities

DornerWorks has a full time QA manager who is responsible for QA at the company and projects on which DornerWorks performs.

SECURITY PROCESS FOR ARLX

ARLX is built with safety and security from the ground up. DornerWorks used two companies to aid in formal methods analysis. The goal of security in ARLX is to support a DoD MILS environment and the need to protect data up to the level of Top Secret/Sensitive Compartmented Information. During the Phase I portion of the SBIR funding, the company Galois was commissioned to analyze a scheduler for the hypervisor that implemented the ARINC 653 standard. They performed this analysis using manual methods. It was costly in both actual money and time to perform. It was determined that we needed a quicker, more cost effective method to run formal analysis on ARLX that supported the open source business model on which we built ARLX.

The advantage DornerWorks had with establishing some level of security was due to including ARINC 653 in the ARLX system. ARINC 653 is an aviation software specification for partitioning computer hardware in space and time for the purpose of enhancing its function safety. With this open standard, one can develop multiple applications on the same hardware with different DO-178 software safety levels. ARINC 653 has very strict requirements for memory, processor and device I/O usage. This fits very nicely in a security paradigm where information flow needs to be strictly enforced.

DornerWorks contracted with Rockwell Collins (RC) to perform the formal analysis on ARLX during the next phase of the project. RC has developed a tool called the Data Flow Logic (DFL). The DFL is a domain specific language for use in specifying and verifying information flow properties of secure systems, implemented as an extension to the GCC compiler. The DFL is automated and as a result reduces costs significantly. The analysis performed with the DFL is consistent with activities associated with Common Criterial EAL6 and above and DO-178 Level A certification.

DornerWorks worked with Rockwell Collins to establish a target of evaluation (TOE) and in determining what the security domains and

resulting security policy would be. Security domains are the objects of policy statements.

DornerWorks and Rockwell Collins specified and analyzed the scheduling subsystem of ARLX. The classification process is time consuming and requires an understanding of the system security policy and an intimate knowledge of the implementation. We minimized the overhead of this work and focused on the subset of the scheduling TOE.

Out of this work, the security policy was created and five security domains were established:

- ARLX_INIT – read only data used to initialize the rest of the system
- ARLX_CONFIG – configuration data that is written during initialization
- ARLX_XEN – the state of the Xen hypervisor
- ARLX_DOM0 – the state of the Xen Dom0 (or control) domain
- ARLX_DOMU[i] – the state associated with the guest domains

A graphical picture of the ARLX security policy, showing that the flow is one way, is shown below.

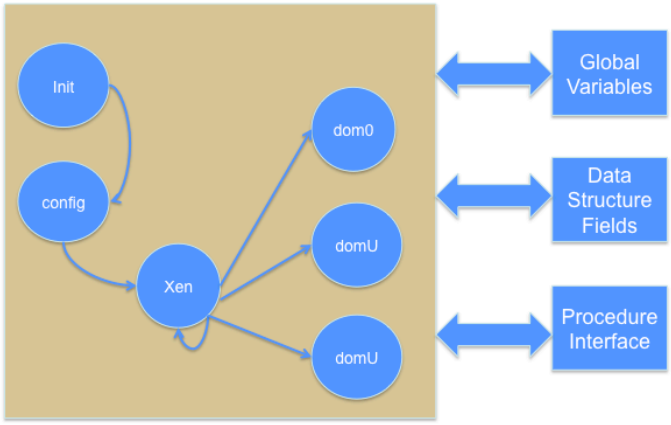

Figure 5: ARLX Security Policy

After the policy was established, the DFL tool was used to analyze the TOE source to ensure that the data flows as indicated in the policy. Exceptions and failures to the rule are documented in the report. Some of the source code included assembly language that was not analyzed by the DFL since it is not able to analyze assembly language at this time.

We received a final report from Rockwell Collins that presented what was done during the evaluation. The final report separated its findings into exceptions and failures. Exceptions were issues that need to be further analyzed since they could either be a real issue or explainable through manual analysis. Failures were issues that the DFL tool determined were issues going against the security policy. The final report also stated the results and gave recommendations for what could be done to further enhance security.

Addressing the findings in the final report is an important task since failures and exceptions might be positively explainable after manual analysis. Analyzing the report was a useful effort, in our case, for two reasons. One, we found that in only a single case, the DFL analysis reported a failure. The procedure it failed on was called ioapic_guest_write(). It turns out that this function assigns an integer to a bit-mapped structure and was not in the TOE. It was good experience to go back to review the analysis and take a visual look at the source code to determine the failure. The other case was in analyzing the exceptions. As an example, we had two exceptions in the function a653sched_do_schedule(). The first exception was due to the fact that we had inline assembly in the code and the DFL tool is not capable of analyzing the assembly language sections. These need to be done manually at this point. The other exception in the function indicated that the function return value may contain information from a local variable allocated on the call stack. This is significant to the DFL tool since the function returns a stack-allocated structure whose fields are individually populated in the body of the function. If there is slack in the structure implementation, the slack regions of the structure would never get initialized. It is unlikely that there is slack in this particular structure, but the DFL tool was unable to determine that for sure and as a result manual

analysis will need to be done to ensure that this not a potential issue.

CONCLUSION

Software is being used on more systems that involve moving people. It is, therefore, becoming more important and even a requirement in some areas, such as aviation, to have an established software development process that can result in certified systems. Following safety and security processes from the beginning of a project helps to deal with errors early.

A lack of emphasis on safety becomes readily apparent when bugs are discovered. Unfortunately the result of finding problems once a product is released to the public causes consumers to question how an organization could have released such a product. Safety requires well-established processes and engineers who are trained in thoses processes and have an attention to detail in making sure that the product development is safety focused.

Safety has always been critically important in the aviation industry. Formal aviation safety criteria have matured much over 30 years. The process for developing a system under the guidelines of DO-178C are time consuming and result in higher software development costs, however the costs are necessary to achieve the level of safety the aviation world has come to expect.

This case study discussed safety in the context of aviation, specifically DO-178C. We introduced the ARLX hypervisor and described how ARLX was developed using the aviation safety specification. We also introduced security in the context of the Common Criteria and how DornerWorks used the ARINC 653 standard as the architecture for the formal methods analysis. We described the documents created throughout the safety process. Lastly we described security steps that we followed in creating security artifacts for ARLX. The military and civilian markets that DornerWorks plans to introduce the ARLX hypervisor will demand the safety and security artifacts described in this case study. ARLX is focused on safety critical systems and proven safety and security can only be achieved through processes developed from established guidelines.

REFERENCES

[1] Henderson, P. and Lienert, P. 2014, *GM safety crisis grows with recall of 3 million more cars for ignition issues* [Online]. Available at: http://www.reuters.com/article/2014/06/17/us-generalmotors-idUSKBN0ER2Q220140617 [Accessed: 22 June 2014]

[2] Storm, Darlene 2013, *Hacker uses an Android to remotely attack and hijack and airplane* [Online]. Available at: http://blogs.computerworld.com/cybercrime-and-hacking/22036/hacker-uses-android-remotely-attack-and-hijack-airplane [Accessed: 25 June 2014]

[3] Greenberg, Andy 2014, *DARPA-Funded Researchers Help You Learn To Hack A Car For A Tenth The Price* [Online]. Available at: http://www.forbes.com/sites/andygreenberg/2014/04/08/darpa-funded-researchers-help-you-learn-to-hack-a-car-for-a-tenth-the-price/ [Accessed: 25 June 2014].

[4] U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03, June 2007. Available at: https://www.niap-ccevs.org/pp/pp.cfm?id=pp_skpp_hr_v1.03/&CFID=26203994&CFTOKEN=f7667ce3bc6ae71f-40F73C94-97C5-F776-C2540D8048D9BA22 [Accessed: 30 June 2014]

[5] Carol Saulsbury Houck, Director, NIAP. Email sent to affected commercial partners. Available at: https://www.niap-ccevs.org/pp/pp.cfm?id=pp_skpp_hr_v1.03/&CFID=26203994&CFTOKEN=f7667ce3bc6ae71f-40F73C94-97C5-F776-C2540D8048D9BA22 [Accessed: 30 June 2014

[6] "Range of completion times and costs for Common Criteria evaluations at EAL2 through EAL4", US government report GAO-06-392, From Wikimedia Commons, the free media repository, Creative Commons License, http://en.wikipedia.org/wiki/Evaluation_Assurance_Level#mediaviewer/File:Common_Criteria_evaluation_costs.gif [Accessed: 6 July 2014]