Certification of Software and Hardware

Kiptoo A. Kiprop
kiprop@uni-koblenz.de

04.08.2005

Seminar: Formal Methods for Fun and Profit
Lecturer: Prof. Beckert

Institute of Computer Science
University of Koblenz
Summer Semester 2005

Abstract

This term paper, was written during the seminar - Formal Methods for Fun and Profit - in the summer semester of 2005. It deals with the certification of software and hardware issues in general and the application of formal methods therein in particular. In the beginning, a general introduction to the issues, definitions and correlations will be discussed. Three examples of system verification methods shall also me mentioned.

Contents

1 Introduction

Common abbreviations

The following abbreviations will be often used in this paper:

Abbrev.	Meaning
CC	Common Criteria
EAL	Evaluation Assurance Level
PP	Protection Profile
SFP	Security Function Policy
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functions
TSP	TOE Security Policy

High security if not totally guaranteed security is an ultimate human desire. Irrespective of which life circumstance, security play a central role: be it driving, choosing a residential area, using the internet or even just eating. Having the feeling of being secure (even if the danger is just around the corner), is a bonus to the living standard and the general mood. Being at ease with one's environment is a worthy course and this applies to security too.
In this paper, an effort is thus made to discuss about the security issues concerning hardware and software in the information and communication technology (ICT). ICT sector is one of the fastest growing sectors worldwide, be in the economy, education, health, name them. ICT is creeping slowly into day to day life and the question of its influence is not being asked anymore, rather; is the technology secure? Can one fully trust matters of life and death to it? What's at stake if something goes wrong? These doubts are even reinforced by some cases of software malfuction, be they real or imagined - think of the pentium-bug of 1994, hacked databanks, numerous stolen PINs. We shudder to imagine a situation where the new Airbus 380 software fails to work over the Atlantic - fully boarded (though a possible failure of the software for example through a runtime error has been categorically excluded. See the current article [10]). Now, these are the kind of scenarios that would rather stay in the world of dreams. On the other hand, the reality calls for measures that spare us all the trouble.But is absolute security possible? What has been undertaken so far? What is still viable? and these are big issues.
Let's stick to the ICT sector. Various checks and balances have been invented and implemented to guarantee security, even if with average results, for absolute security may still remain elusive. One of the

procedures is the certification:

1.1 Certification

Certification is the act of conferring legality, sanction, formal warrant or the act of granting credit or recognition (here with respect to institution that maintains suitable standards). Other synonyms are accreditation, enfranchisement, empowerment and authorization.
That was the common definition. To drive the certification definition home, let's use an example. The computer display monitor: Someone or some people e.g. a government, a society or a consumer foundation, come up with certain requirements in form of technical rules and standards (with some defined scale) that a monitor manufacturer has to fulfil in order to be allowed to sell his products. The monitor shall for example not be smaller than 14 inches, shall not emit certain rays or shall not be brighter than a certain value. If the manufacturer fulfils all these and many other given rules, then he can make money, else he will have to pack and go begging. A software or hardware product (from here henceforth only referred as product) is therefore certified if some requirements or criteria are fulfilled.
But why certify at all?
One of the reasons is to preserve the human health or even life. The monitor example above emitting dangerous rays could be an health risk for the user even in short term. Another important reason is to ensure the quality and reliability of the product. This applies most especially where product flaw, however small it may be, could easily become a disaster. Such an high risk could be a car control system based on software: if somewhere in the code a division by zero (or an overflow) is undertaken, further secure control over the car is not guaranteed anymore - this on a full highway in full speed. Other reasons may of course be disguised for example financial or status interests (if the poor guy might become a business rival). This could be for example in order to be admitted to a certain expert club or to get some prestigious accreditation.
Certification for IT systems especially has not been around for long. Just like many IT system products, certification has its roots in the military sector, to be exact, the USA military. The US Department of Defence first published a catalogue known as Trusted Computer System Evaluation Criteria in 1983. Its purpose was to provide technical hardware, firmware and software security criteria and associated technical methodologies to support the automatic data processing system security policy as well as evaluation and certification.
Certification of IT systems is the duty of government departments

which were founded especially for this. In Germany for example is for example the Federal Department for Security in the Information Technology (BSI). In the USA the National Institute of Standards and Technology and the National Security Agency. Certification of IT systems will follow the fulfilment of certain requirements: But how are these requirements defined? Who stipulates them? This leads us directly to the Common Criteria.

1.2 Common Criteria

The purpose of Common Criteria (CC) is to develop standard collection of the necessary requirements. These requirements are on the other hand called the Protection Profiles (PP)(1.2.1). The CC is a conglomeration of other existing standards. These criteria are: the European (ITSec - Information Technology Security Evaluation), Canadian (CTCPec - Canadian Trusted Computer System Evaluation Criteria) and U.S. American (TCSec - Trusted Computer System Evaluation Criteria). TCSec is otherwise known as the "Orange Book". TCSec specifies the criteria the Department Of Defense uses in evaluating the security of a product. The assessed features include the security policy, marking, identification, accountability, assurance, and continuous protection of the system. Based on the assessment, the security of the system is classified into one of four hierarchies, with A providing the most security and D providing minimal or non-existent security. Each hierarchy has a number of levels as well. The "Red Book" was published to provide subsidiary information to enable the Orange Book principles to be applied in a network environment. The Red Book was initially published as the Trusted Network Interpretation (TNI) of the Trusted Computer System Evaluation Criteria.
The CC has been designed in such a way that it is flexible enough to allow a bridging to the existing national schemes of security evaluation, certification and accreditation. The first CC version was published in 1996 and the second more extensive version adopted by the International Standards Organisation (ISO) in 1998.[8]
CC therefore stands for the requirements for the IT security of a product or system under unique categories of functional requirements and assurance (1.2.3) requirements. Here is a summary of these requirements:

- The CC functional requirements define the desired security behaviour. These requirements are grouped into classes which are very general, but all members of a class having the same focus. There are eleven functionality classes so far. These are: Audit, Cryptographic support, Communications, User Data Protection,

Identification and Authentification, Security Management, Privacy, Protection of the TOE Security Funcions, Resource Utilization, TOE Access as well as Trusted Path/Channels.
The catalogue of functional components provide guidance on the organization of the requirements for new parts to be included in the ST. The functional requirements are expressed in classes, families and components. Within these classes, there can be inter-dependencies. An example is the dependence of Data Protection class has on the correct Identification and Authentification of users in order to be effective.[8][9]

- The CC security assurance requirements form the countercheck for the security measures that they are effective and correctly implemented. The too are grouped into classes and members of a class just as those in functional requirements share a common focus. There are eight assurance classes: Configuration Management, Delivery and Operation, Development, Guidance Documents, Life Cycle Support, Tests, Vulnerability Assessment and Assurance Maintenance.
  The EAC categorization into classes is similar to that of the functionality classes above. Each class has a number of member families, each family having an emphasis area but at the same time sharing security objectives with the others in the class. These classes are a summary of security requirements and all members of a class have a similar focal point.

At this point, it would be appropriate to expound on a few important terms which play a central role and thus used frequently:

- Target of evaluation (TOE): an IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation. The TOE objectives, requirements and summary specification of security function and assurance measures together form the primary inputs to Security Target - ST (1.2.2)- used by evaluators as basis for evaluators. TOE therefore defines assets to protect.
- TOE Security Functions (TSF): A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.
- TOE Security Policy (TSP): A set of rules that regulate how assets are managed, protected, and distributed within a TOE.

The diagram (1) is a summary of the correlation between the CC components. All efforts are directed at the Target Of Evaluation:

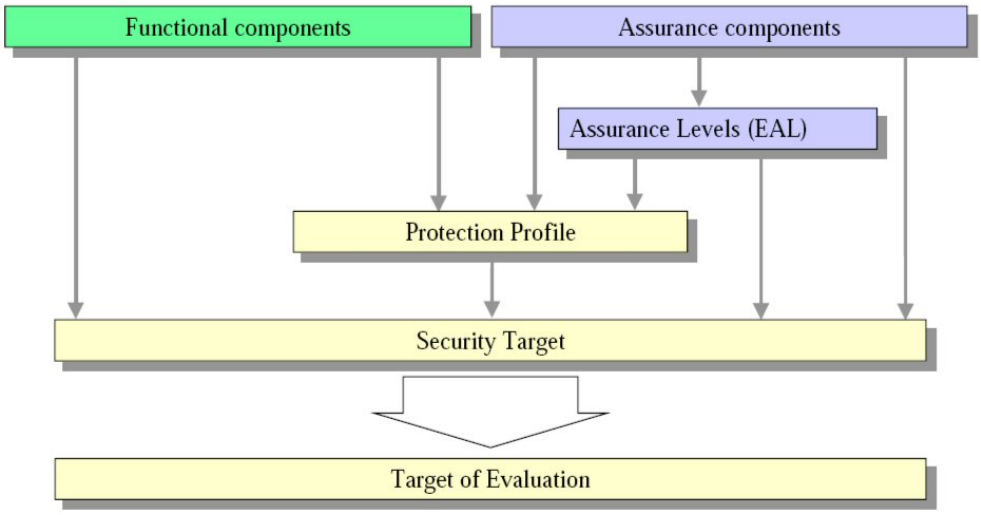

Figure 1: The CC components

But what kind of people are involved and in what capacity? What is their funtion here? In a certification scenario, three parties play key roles. All the three are the main users of CC:

- Consumers: Apply the results of CC evaluation and are part of the decision making whether a product or system fulfil the security needs. These needs may be a result of both risk analysis and policy direction. The consumers can use results to decide on different systems or products through comparison. They further specify the security functionality of the product in terms of protection profiles agreed upon and and select the level of evaluation assurance from a defined list of seven levels (1.2.3) independently.
- Developers: Design, formulate and develop the security specifications for TOEs. They react to expressed or perceived consumer needs. The developers use the CC to evaluate their products or systems, and thus identify the security requirements to be fulfilled by them. They can also use CC as a basis for their claim that the TOE meets the identified demands through stipulated security functions and assurances to be evaluated. Every TOE requirement is contained in the security target. The CC further puts down the security functions that a developer includes in the TOE.
- Evaluators: Oversee and determine whether a TOE effectively meets security functions claimed by the developers using func-

tional requirements. the CC describes the set of general activities the evaluator has to carry out and the security functions on which to implement the actions. However, CC does not specify steps to be taken in performing the actions.

1.2.1 Protection Profiles

PP is a definition of a set of IT security requirements and aims for a category of TOEs which are implementation independent. The requirements are directed to a category of products and systems which are developed to meet certain similar user requirements for security. A PP is designed to be used more than once and to define the requirements to be applied effectively to meet the identified objectives. The concept of PP is further aimed at to support functional standard and as an help in formulating acquiring specifications. A PP should therefore be user oriented and should avoid prompting the user to refer to other documents which might not be within reach. In simple english, PP says what the system is supposed to do. The other important function of PP is the evaluation rating. This is the chance for the absolute experts to check and decide if the system really meets requirements specified in PP.[9]

1.2.2 Security Target

Another important structure of CC is the Security Target. An ST contains the security requirements of an identified TOE and states the functional and assurance security measures offered by that TOE to meet the laid down requirements, in other words, it is a basis over which an evaluation is performed. It defines the target of evaluation, the environment, the threats, assets to protect, security objectives and assumptions. It contains the IT security objectives and requirements of a specific identified TOE and defines the functional and assurance measures offered by that TOE to meet stated requirements. A ST may demand conformity of one or more PPs, and forms the basis for an evaluation.[9]

1.2.3 Evaluation Assurance Levels (EALs)

Evaluation assurance can be informally described as trustworthiness or reliability. EALs seek to reach a balance between the level of assurance and the cost and feasibility of reaching that level of assurance. The CC style points on separate approaches of assurance in a TOE at the end of the evaluation, and of maintenance of that assurance during the operational use of the TOE.

The CC contains a set of defined assurance levels constructed using components from the assurance families. Other groupings of components are not excluded. To meet specific objectives an assurance level can be augmented by one or more additional components.The levels are intended partly to provide backward compatibility to source criteria and to provide internally consistent general purpose assurance packages. The levels' hierarchy increases with the increasing assurance. EAL1 is the entry level and upto EAL4 the rigour and detail is increased but no special security techniques. At the top level 5 to 7, there are compelling limitations to meeting the stipulated requirements due to developer and evaluator activities' costs.[6]

- EAL1

- Functionally tested.
This level is used where some assurance of correct operation is needed, but the security risk is not seen as high. It makes an evaluation of the TOE available to the customer. Its evaluation should be possible without the help of the developer of the TOE and should be for minimal outlay. The user should be able to test the consistency in comparison to its documentation and should provide protection against indentified threats. The level covers only functionality i.e. that a system will work in an environment.

- EAL2

- Structurally tested.
This level requires the developer's assistance especially on the provision of design information and test results. This should however not demand more from the developer as it is should in the market. There should also be no substantial increase in investment and costs. EAL2 is mostly used where the user or developer needs a low to moderate level of security assurance when the complete development record is not nearby. In this level, the security mechanism are checked but only moderately but less stricter than level 3.

- EAL3

- Methodically tested and checked.
EAL3 enables a reliable developer to reach maximum assurance from positive security engineering at the design stage without having to make extensive changes to existing and properly functioning development practices. The level comes into action where the developers and users need a medium level of independent security assurance and require a proper investigation of the TOE and its development without extensive re-engineering. Many systems have achieved the rating of this level for example Linux Server v. 8. see section (2.1).

- EAL4

- methodically designed, tested and reviewed.
This level is the one we will most likely meet in IT systems. Operating systems like Windows 2000, Linux Server v.9, and NetWare have achieved its rating. This is because it is the first level that proves that a system is safe. The level enables the developer to maximise assurance obtained from positive security engineering based on good commercial development practices. These practices may not extensive expert knowledge, skills and other resources. This level might be the highest that can be applied to existing industry and at the same time economically sound. The developer and user require a medium to high level of independently assured security in normal goods' TOEs and the presence of a willingness to carry the extra costs arising from re-engineering.

- EAL5

- semi-formally designed and tested.
This level introduces the formal models to the assurance evaluation, even though to a preliminary extent. It enables the developer to gain maximum assurance from security engineering on the basis of strict commercial development practices. The level further uses the development environment controls and understandable TOE configuration management including automation and evidence of security procedures to provide assurance. EAL5 extends EAL4 by requiring semi-formal design descriptions, the whole implementation, a more structured architecture, among others.

- EAL6

- semi-formally verified design and tested.
The high assurance provided through EAL6 enable developers produce a high standard TOE for protecting high value assets against significant risks. The level is hence applicable for developing security TOEs for application in high risk situations where the value of the assets to be protected justifies the extra costs. EAL6 provides assurance by an analysis of the security functions using functional and complete interface specification, guidance documentation, high and level design of the TOE and a structured presentation of the implementation to understand the security behaviour. More assurance is reached through a formal model of the TOE security policy, semiformality of the functional specification, high and low level design and a semiformal demonstration of contact between them. No system has yet attained level 6 rating.

- EAL7

- Formally verified design and tested.
The last and highest level of evaluation is directed at evaluation of extremely high risk situations and or the high value of assets is not an hurdle for spending more time and carrying more costs for the security. However, practical application of this level is at the moment limited to TOEs with rigidly focused security functionality that is liable to extensive formal analysis. The formal model in this level is improved by a formal presentation of the functional specification and high level design showing correlation. Proof of developer White Box[1] testing is a must, as well as complete independent confirmation of developer test results. EAL7 therefore extends EAL6 through requirement of more comprehensive analysis using formal representations and correspondence and comprehensive testing. The level is limited to very specific systems with very specific security functionality. There is however no system yet, that has been evaluated on this level.

Figure (2) summarizes the EALs above. The Assurance Classes, Families and their corresponding components are illustrated The numbers denote the component. The assurance families marked red are the ones that apply semiformal (or formal) models in the verification and are all to be found in the development assurance class. An example of how to read the table: One of the assurance components of EAL7 is the ADV_SPM.3. This is the formal TOE security policy model (TSP) in the development class. The numbers denote the component, in line with its hierarchy in the level family. Component numbers stressed in bold signify the relationship between components within a family. Bolding convention calls for the bolding of all new requirements. The other components (with semi-formal or formal aspects) are:
- In EAL5: ADV_FSP.3 (semi-formal functional specification), ADV_HLD.3 (semi-formal high-level design), ADV_RCR.2 (semi-formal representation correspondence demonstration) and ADV_SPM.3 (formal TOE security policy model).
- In EAL6: ADV_FSP.3 (semi-formal functional specification), ADV_HLD.4 (semi-formal high-level explanation) and ADV_LLD.2 (semi-formal low-level design).
- In EAL7: ADV_FSP.4 (formal function specification), ADV_HLD.5 (formal high-level design), and ADV_RCR.3 (formal correspondence

[1]White Box testing also known as clear box, glass box or structural testing is used in computer programming, software engineering and software testing to check that the outputs of a program, given certain inputs, conform to the internal design and implementation of the program. The term White Box indicates that the tester closely examines the internal implementation of the program being tested.

demonstration). The high-level design decomposes the system into

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and operation	ADO_DEL		**1**	1	**2**	2	2	**3**
	ADO_IGS	**1**	1	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Guidance documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life cycle support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_SOF		**1**	1	1	1	1	1
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Figure 2: The EAL summary

modules, or subsystems, which provide the functionality described in the functional specification. The low-level design provides a specification of the internal workings of each module.

2 Examples of certified products

First of all, what are formal methods?
Formal methods are methods that apply modelling, calculation, prediction in the specification, ideas and techniques from arithmetical or

formal logic to specify and reason about hardware and software systems. Formal methods therefore help to ensure that a system fulfil the specifications which (if fulfilled) later used to decide if the system is to be certified for the corresponding level or not. Just to stress the difference between formal methods and security catalogues (e.g. the CC), the catalogues are used to stipulate the specifications and the processes (e.g. formal methods) of how these specifications should be evaluated. In other words the catalogue is just a long list of specifications which has to be fulfilled. To find out if they are fulfilled, the processes are put in action.
Before even applying formal methods which are represented in the higher evaluation levels (EAL5 to EAL7) in product evaluation, the product has to pass through the lower levels first. As we will find out, just about all products never get through to the rigorous evaluation with the high assurance levels. Formal specifications have a number of advantages which could be of importance if applied e.g.
- they may be analysed mathematically and finally demonstrating their consistency and completeness.
- they may be used to guide the tester (evaluator) of the system component in identifying appropriate test cases.
- they may be processed using software tools which may in turn make possible the animation of the specification to provide a software prototype.

All theory is grey. Having discussed a criteria (CC) that leads to certification with many of its features, it is now time to come back down to reality. What is gained from all the theory above? Which systems are being certified at which level? Why is an higher degree of confidence needed and for which systems?

2.1 Example 1

To demonstrate this a concrete product example might do the trick: SUSE LINUX Enterprise Server v.8 (SLES8) [1] sponsored by the IBM Corporation. (SLES v.9 has also been evaluated and obtained an EAL4 rating, Same level attained by Microsoft Windows 2000. See further down).
According to the manufacturer, the Server v.8 has been evaluated and obtained an EAL3 rating. In this level the following activities must have been undertaken: independent confirmation of a selected sample of developer tests results, search for weakness justified by developer, development environment control and TOE configuration management. Just for a start: for someone looking for a secure server, this

evaluation still be less elaborate: it has only be methodically tested and checked, nothing more. There is no code re-engineering, no interruption of development process, but more costs.
The TOE was, the operating system, running and tested on the hardware and firmware specified in the security target which were; the evaluation assurance level of EAL3. The test plan was limited to the areas enforcing the secure operation of SLES8. The design of the test cases was only to verify the correct operation of security related user programs, database files and system calls. In fact the author himself explicitly mentions that the testing for system availability in a stress environment was beyond the scope of the evaluation [1] - and this should be what a test is actually for (how the systems handles serious errors). In spite of this shortcoming, the system passed through the rest of the evaluation objectives successfully and works well in a normal environment. On the formal methods application, the product is not anywhere nearby. It would have to undergo EAL4 testing (which SLES9 already underwent) and then perhaps from there think about EAL5.
There are a number of reasons why a product like SLES8 may not be evaluated in the higher levels soon especially with the application of formal methods although some of the reasons could at the same time be advantages of evaluating systems with the formal methods:

1. Consumers would like to see security advantages fast, concrete, cheap and as simple as possible. These advantages could be for example guaranteed higher security, stable systems and the possibility of adding more components (e.g. new application software) without having to seal new security holes. With formal methods, this might be a huge task, since it involves discrete mathematics for specification. This will require more engagement of more developers who will then in turn might be more expensive. On the other hand, this will require very versed consumers, calm enough to go through the specification jungle of denotation and proofs.
2. To achieve a higher security, the system features and components has to be kept to the minimum. It is well known that the more components and features a system has, the more prone it becomes to attacks. In other words, the wider the war front, the more thinner the defence (or the more the soldiers needed). The more components would mean that there is more to evaluate, the costs would generally rise and the formal specifications to evaluate such a system might be more extensive. The problem is exemplary for MS-Windows systems which have been burdened with so much features that it is almost impossible to detect a

weak point. Else the user or customer would have to accept a system with very few features if any at all (a lot of doubt if this would ever happen).

3. To produce high quality code or highly secure systems that will still fully function in the business environment, the developer will need a lot of time and resources. This demand a great amount of finances which the manufacturers are trying to save in the first place. Therefore a compromise has to be reached between profit and security. The developers at the same time stand under great pressure to deliver (functioning) goods at a stipulated time. This might not only be impossible but also dangerous to health. Working long hours or overtime does not really contribute to secure and high quality products. Employing more developers could be a solution but this comes with more costs which will be in contradiction to the saving target. If more developers can deliver high quality goods with certification in EAL5+, the only remaining problem would be to find a market which will absorb the products. The manufacturers could therefore take the advantage of the high quality to market the products. This could work out since due to the fact that the products tested in EAL4 have had a good market.

4. Many consumers lack the expertize (or cannot afford) to service or repair the highly complex systems which might become necessary in order to continually check the system if it fulfils the security requirements over the time. Security demands that there be a regular check of systems and when need be updates or other measures be undertaken. They will therefore go for the next best. This problem may even be intensified by poor documentation of the code especially in the middle levels. On the other hand, the formal methods in the higher levels should by its nature be properly documented which may in turn be of assistance in understanding the system. But since it is rare (if at all there exist) to find a system that has been evaluated with higher methods, this advantage does not come to use. The consumer will try to understand the problem or issues at hand and the system documents should then be helpful. At the end, product may have been evaluated and certified, but for a user, it may still be a mystery.

With the above reasons and others, certification might still remain as interest for a few. And indeed as long as certification has not become compulsory, there will not be much to show of it. On the other hand, there are examples of system certification that are compulsory. These

laws are made by a government for example or regulated by a certain ministry. Here is an example of a certifying body [11]: The State of Schleswig-Holstein has founded a department for data security. This is an independent body (financed by the State itself) which has duties like data security in commercial sector, consulting on data security, development of new technologies for data security and protection, and in the case of this paper; issuing of a seal of approval for IT products. On the other hand, they countercheck the security complaints from the citizens and inform the manufacturer of the product security flaws, carry out assessment of products and even advice people on our to set up computer systems. Once again to the seal of approval: for now, the seal is issued only for IT products. After a system has been evaluated, this seal is issued for the product which has passed some stipulated security requirements. Though this certification is not yet compulsory, the market has shown that people go for the products with this seal, meaning that the more they are bought, the further, the manufacturers check closely on security and at the same time getting good financial returns. If therefore, such a certifying body is made compulsory nation- or even worldwide, by say a government or groups of them, the products will surely be more secure. If the certifying body uses a criteria like CC with EALs above, then we can get closer to the application of formal methods for evaluation.

2.2 Example 2

Here is a further example of a certified product: Microsoft Windows 2000. The evaluation was financed by Microsoft and was undertaken against the Control Access Protection Profile (CAPP) [6]. The software attained a EAL4 rating, meaning that formal methods did not come to application. EAL4 is the first level that proves that a system is safe, as the vendor was willing to fix problems in the development process of the product to achieve this rating. EAL 4 incurs cost to the vendor, re-engineering is possible if flaws are found. There is therefore no measurable evaluation of software especially the code - no evaluation of the software itself is required at all. In spite of not having been evaluated and tested in a high level (EAL5-EAL7), Windows 2000 could still be said to have a good functionality. For anyone who does not require very high assurance, the system can still fulfil the minimum of security.
Finally, do we stand a chance of experiencing full formal specification at work in evaluation? I could not come across a system has already been evaluated with full formal specifications in EAL5 to EAL7 (perhaps except the Smart Card VM below in EAL5). Well, as an expert

[10] in the field of verification freely admitted, we are still far away from this. On the other hand, smaller systems (with less components and less program complexity e.g. cars) could be evaluated formally in the near future. There is hope however that some impossibilities now are the assignments of tomorrow. It could even be the governments' duty (or international bodies) to stipulate regulations of which systems must go through which level of evaluation, not only defining the criteria for evaluation. Further, a certifying body could determine which level different system types must be evaluated in, for example all operating systems in EAL4+ (I could not yet confirm at which level a certifying body e.g. a government demand for operating systems).

2.3 Example 3

The last example: Java Card Virtual Machine (JCVM) developed by the Sun Microsystems. A JCVM is a surrogate to the Smartcard which is an integrated system that is mostly used to supply security to an information system. A Smartcard is used to secure data storage and authentification. They are widely used in the banking and communication sector and might be extended to eCommerce and eIdentity among other branches. Smartcards may run on platform independent virtual machines and interaction with systems occur via Application Programming Interfaces.
JCVM has been evaluated and obtained a EAL4 and EAL5+ rating. TOE: processor chip and IC for software (drivers). Evaluation level 5 is about semiformal design and testing and as described above, involves search for vulnerabilities and resistance to moderate potential attacks, semi-formal demonstration of correspondence and functional specification and high level design among others. In other words, it is a gain of confidence from EAL4.
Since JCVM is based on a collection of applets (small programs written in Java language) which together form a closed system, what happens if the part on which one (internal) applet depends on is defective? Will the problem spread to other applets or even be detected anyway? To overcome this problem, JavaCards try to prevent data leaking and references passing from applet to applet through some form of dynamic security mechanism named firewall, which controls the object sharing. Every time an access to a resource is required, the *firewall* checks it. If this is not allowed, the firewall returns a security exception. For these questions (and perhaps many others) could there be an answer in formal specification?
JavaCards try to improve security and increase reliability through the application of formal specification and verification of the source code.

Since the applet language is simple and API relatively small, the application of formal methods is simplified. One way of formalizing the specifications of JavaCard for a CC evaluation is by the use of the B-Method. The B-Method applies semi-formal and formal models which are components of the high level evaluation to specify, design and code high risk systems. B-Method covers the whole system lifecycle i.e. from specification to executable code. A refinement (way of reformulating machine data and operations by introducing more concrete information which in turn causes machine expansion) process to obtain the implementation of the B specification. Every refinement level must be internally correct towards its abstraction. B models are independent from one another and represent independent processes. One model can include several machines linked together using special clauses. As for the JCVM model, there are five modules used, i.e. the *dispatcher*, the *interpreter*, the *firewall*, *java stack, exception manager* and the *memory*.[12]

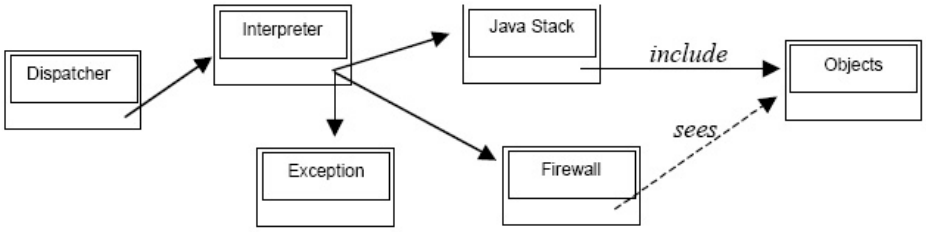

Figure 3: Architecture of the JCVM B machines

In B-Method, the modeling policy is specified and documented in the INVARIANT clause of a B machine (clause has for example, the states description, transitions restrictions, e.t.c.). The characteristics contained in this clause must be respected after the initialization of the machine variables and after each operational call. A model cannot be proved if this condition is not true (not fulfilled). The B model of the dynamic security policy (set of functional requirements documented in a CC document - Security Policy Model or SPM) should not contain any operation since it is a set of predicates to be verified. In EAL5, of formal model of the TOE Security Policy (TSP) shall be provided in ADV_SPM.3. The developer shall also prove the correspondence between TSP and the functional specification. This correspondence must establish that the security fucntions are consistent and complete with respect to the model, that the model contains the rules and characteristics of all policies of the TSP that can be

modelled as well as that the model itself is consistent and complete. The formal model required by an EAL5 evaluation should therefore not specify how things are done but rather what is done. Therefore the machines of the policy model only contain the definition set of variables, the initialization of these variables and the INVARIANT to be fulfilled by the variables.

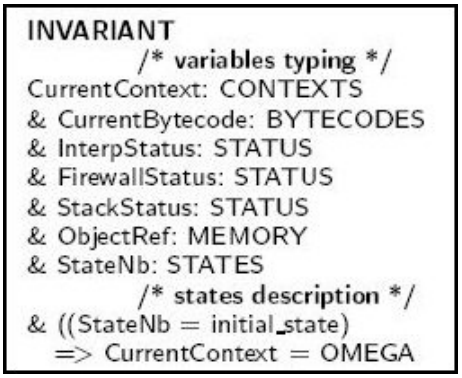

Figure 4: Example of an INVARIANT definition

The functional requirements in SPM are explained by analyzing the security functions and mechanisms that are involved at runtime. The JCVM runtime policy is not really based on semi-formal model but used to describe exactly what shall happen. In other words, the formal model contains one machine that mathematically describes all possible paths in the runtime graph. In the formal model, the different states of the graph must be associated to concrete data of the system. A system can in turn be in two entities: the bytecode *interpreter* and the applet *firewall*. The *interpreter* processes the current bytecode and then calls the *firewall* if the execution requires access authorization to be delivered. A *firewall* requires information on the runtime current context and some attributes of the accessed object (context, type, e.t.c.).
In the semi-formal model, the *interpreter* action are separated into two, i.e. obtaining parameters from stack (associated to *java stack*) and the bytecode execution itself (associated to *interpreter*). In the EAL5 for JCVM, the relations between the security functions and the policy are not formally required, rather in the EAL7 evaluation, which JavaCard is yet to reach.
The TSP design in EAL7 provides the formal definition of the policy to be fulfilled by the TOE model. The FSP component imposes the establishment of a model of each security function. In the model, function interfaces (mechanisms and environment) shall be defined

and described. In EAL7, there is a formal presentation of high-level design which is a refinement of the whole specification in a modular way. The HLD, LLD and FSP formal model components must be consistent and justify that it is an accurate and complete specification of the TOE requirements. The correspondence between HLD and LLD shall be semi-formal based on the identification of a total bijection between data and states of TSF. The RCR (representation correspondence) shall be formal. EAL7 thus requires a demonstration that the SFs (which are formally specified and designed) enforce the security policy through the proof of the formal model containing both the policy and the functions.

Which systems (very high risk) now could be good candidates to be evaluated in high levels? Of course it would be a major security step forward if all systems with fairly high to very high risk are evaluated in spite of the reasons above. Very complex and high risk products like the atomic reactor control systems, banking systems, space rockets or military systems could then undergo the rigorous evaluation. After all, it is in everybody's interest to have products which have been certified under well-known and trusted criteria.

3 Conclusion

This paper dealt with the common verification of software and hardware for example through application of formal methods in the highest levels of evaluation assurance (EAL5-EAL7) if any.
All said and done, the legible question at this point would be: to buy a certified product or not. Here the customer or the user for that matter will have to decide alone. The decision may be simplified by factors like:
- after assessing own needs, the requirements could be determined.
- decide on the product one is most sure it will meet the needs.
- the costs factor.
- some or many products might have a compulsory certification e.g. from a government body (see the Schleswig-Holstein example [11]).
The consumer would just have to know what the product was tested for and under which criteria the product was tested or evaluated for and of course if the testing body is trustworthy.

Formal methods or not formal methods?
The consumer (or developer for that matter) has to assess the benefits of the formalisation (which again may demand a certain level of know-how). The CC also shall require justifications for the methods

and tools that have been used and this is more work. The ultimate goal is to increase the level of confidence that the security is provided, but it still remains open if it is possible to quantify the improvement. The formal specification is not really a panacea for high security. They are man-made and could respectively have flaws which may not be detected, therefore evaluating a systems with wrong tools. But there is no other way out of better security, if not to just hope that nothing bad will happen (it does though), maybe we would be better off by improving the formalisation infrastructure (though with disadvantages above) rather than to design more powerful proof techniques. This may include:
- Integration with semi-formal methods (gets us closer to formal methods).
- Methodology for modelisation in the context of the CC and better system structuring.
- Methodology for proving in the large.
- Better communication with non-experts (includes the consumers).
- Better integration between tools and guidelines (documentation) to use them in an appropriate way. This could also include better compatibility integration between security research efforts and the existing systems.

References

[1] Daniel H. Jones (2003).Test Plan for SuSE Linux Enterprise Server V8 EAL3 Security Function Verification, http://ltp.sourceforge.net/docs/EAL3_test_plan_v1.9.htm.

[2] Anthony Hall, Seven Myths of Formal Methods, IEEE Software 6(9), 1990.

[3] Bundesamt für Sicherheit in der Informationstechnik, Common Methodology for Information Technology Security Evaluation, (2004). http://download.commoncriteria.de/Dokumente/cemv2.4r256.pdf.

[4] Certification Report, (2001). http://www.commoncriteriaportal.org/public/files/epfiles/0166a.pdf

[5] Alain Merle. Evaluation des produits suivant les Criteres Communs, (2001). http://www.systemes-critiques.org/merle.pdf

[6] National Security Agency. Controlled Access Protection Profile (1999). Fort George, USA.

[7] Bolignano et al, (2003). Formal Methods in Practice: the Missing Link. A Perspective from the Security Area. http://www.systemes-critiques.org/bolignano.pdf.

[8] Introduction to Common Criteria, (2005). http://download.commoncriteria.de/Dokumente/cemv2.4r256.pdf

[9] Common Criteria for Information Technology Security Evaluation, (1999). http://www.commoncriteriaportal.org/public/files/ccusersguide.pdf

[10] Heise Online, (21.06.2005). Software ohne Fehl und Tadel. http://www.heise.de/tr/artikel/60759.

[11] Center for Data Security Schleswig-Holstein, (05.07.2005). http://www.datenschutzzentrum.de/.

[12] Using B Method to Formalize the Java Card Runtime Security Policy for a Common Criteria Evaluation, (2000). http://www.gemplus.com/smart/rd/publications/pdf/MT00bjav.pdf.