Common Criteria for professionals: evaluation processes
Introduction

I have spent the larger portion of my working life doing Common Criteria evaluations and certifications, and helping/coaching/training developers, evaluators and certifiers to do evaluations efficiently. Over time I’ve given a lot of advice in many forms¹ and now I decided it was time to bring the parts around the processes together in this book.

This book takes a step back from the technicalities of the Common Criteria² and goes into the processes in a CC evaluation: the business view on the process and interpersonal view on the play between the roles of the end-user, the developer, the evaluator and the certifier.

As with the CC, these topics are quite broad and interwoven with each other. I’ve gathered them in chapters around common threads in my experience. This way the chapters can be read stand alone while showing the bigger picture.

Whom this book is written for

This book is written for people interested in improving the way a Common Criteria evaluation goes for them and/or want a view on the meta–level processes.

I’m assuming you are working in the roles of evaluator and certifier primarily, although I think experienced developers and consultants will also find it useful to know what happens in the evaluator/certifier side of the process.

I’m assuming you already have a background in Common Criteria (if you are just starting after your training in the formalities of Common Criteria, you should be fine).

Whom this book isn’t written for

This book is not written to guide developers through a Common Criteria evaluation. Should you be looking for such support as a developer, I advise you to seek consultants that have experience with evaluations, preferably in your technology field, to support you. Most of these consultants are part of the established evaluation labs and there are some like me that are independent.

This is also not for absolute beginners, as it is not an introduction book to the Common Criteria. I’m assuming you as a reader have basic knowledge of the Common Criteria, the procedures and its terms. Contrary to many books and white papers on the Common Criteria, I will not be copying in the content of the Common Criteria standard itself wholesale into this book. I consider that unnecessary padding of this book with information you as a reader should already be familiar with, or at least know how to look up.

¹ Some of you may recognize parts of this book from conversations, blog and forum posts, emails, training courses and presentations. What can I say, I’m generous in sharing and efficient with reuse.

² I’m saving that for another book.
If you do not have this level yet, you will most likely still get some valuable knowledge from reading this book, but I advise you to follow an introduction course in Common Criteria first. These are often offered by your local certification body (these are intended for beginning evaluators and tend to be highly focussed on CC-technical aspects), by evaluation labs (these range from also very CC-technical to practical for developers and that lab) and by consultants (these can typically adapted to your requirements). Obviously I provide such training too.

**Disclaimer**

As always, I stand behind my opinions and insights, as being the best way I know to explain and understand the topics. Insights can change though, as my experience grows even further. Your situation might not match the (implicit) conditions under which the described approaches are optimal. You might have completely different opinions (if so: I welcome all constructive discussion with all my heart!³ See the chapter Contact at the end of this book or look on the [Your Creative Solution website](#)).

I do know I have had to simplified things in places because the full complexity is simply too big to describe, and it is the simplified, clearer view that is most useful. Just like our teachers told us “lies to students”⁴ when they explained the earth is completely round, leaving us to discover about deformations of the globe, removing the not-so-relevant details makes the concept clearer. Only after we understood the bigger picture, we could understand and appreciate the small items that are not quite so simple. I also used that training technique in this book. I promise though, the “lies to students” are in my experience and view small items (or I will warn you for them).

As those who know me already understand: I am my own man. The opinions in this book are mine, and may or may not be those of the companies, organisations, colleagues, friends, acquaintances, and/or any sentient or non-sentient entity I have worked for, worked with, will work for/with, communicate with, compete with and/or might have been in the neighbourhood of (or not). If you have issues with something, contact me if you want to discuss it. Please do not project your opinions on my opinions onto these others.

**Dedication**

I dedicate this book to the wonderful sentient entities that have guided, prodded, challenged and otherwise supported me. With a special place of honour for DJ, for starting my growth in this domain by quickly and fully entrusting me with the 'special' CC projects.

---

³ Academic ground rules: for intellectual exchange of opinions, delighting in learning other viewpoints and while respecting each other. Preferably over a beer.

⁴ Adapted from the “Lies to children” part in *The Science of Discworld (2000)*, Terry Pratchett
Saying “pass”: the difficulty of approval ........................................ 28
Careful with previews ........................................................................ 31
Contact ......................................................................................... 32
Bibliography .................................................................................. 33
Setting context

Although this book is not an introduction in the Common Criteria, giving an introduction on the CC is very often the start of a CC evaluation process for you as a professional in the CC. This is the moment where you set the context that will have a large impact on the following process. The beginning is a very delicate time and it pays to take good care of this start.

Explaining the CC and especially the processes can be done in 5 minutes and in 5 days: what is useful for your audience very much depends on where they are at and where they want to be.

In any case, the topics to cover are on the technicalities of the evaluation process to communicate (the subject of the first part of this book) and on the way of interacting (the subject of the second part of this book).

I found it very useful to immediately and explicitly frame CC evaluations as long term, co-operative approach. Not only do I personally strongly prefer that view because of the smoother and more enjoyable processes, the co-operative approach also has a direction towards win–win issue solving, not a win–lose tug of war. The management arguments I use, besides the “in the end, the certifier has the last say even if he is not right, so let’s not go there”, point out the long relationships with the high switching costs.

Long relationships: high switching costs

Relationships in the Common Criteria domain are long lived and generally last many evaluations. There are many reasons for this, see in the second part of this book for the psychological reasons. In an economic view this is because of the very high switching costs.

A single evaluation already has a timeframe of 3 months to 2 years, excluding all the steps of deciding to perform the evaluation at a certain lab and scheme, negotiating the contract and actually starting the evaluation. Switching during the evaluation in practical terms means starting all over again, discarding nearly all progress gotten so far.

During the evaluations the developer, evaluator and certifier will learn and adapt to each other’s special ways of approaching aspects of the development and evaluation: typical design structures of the TOE, special situations and their solutions, what items are sensitive and where there is flexibility. During the first evaluation in that combination of parties, and in a lesser degree when a TOE with special properties is first evaluated, all parties invest time, money and energy. This results in more cost and time efficient evaluations in the future, not in the least because of highly reduced risk of unexpected issues during the evaluation itself, however is an investment for the first time (if only because more senior staff is assigned to such first evaluations).

---

5 Dune, Frank Herbert
There is also the aspect of sharing the confidential information. Just from legal and reputation reasons being active in this domain implies that every party will need to know how to hold secrets from the outside world. Breaking an NDA in a big way or even going rogue and selling information to the attackers is (luckily) unheard of as of yet, and would destroy a lab or even a certification body. So established businesses in the CC domain have a lot to lose\(^6\) and are unlikely to make such mistakes.

Yet for an evaluation to work efficiently, it is not sufficient to know that the parties involved should hold their secrets. They must also feel safe to disclose items that are not strictly required for the evaluation but very useful to do. On management level it is really useful to exchange future product evaluation plans to ensure that the evaluation is designed to adapt to the delta evaluations easily for example, yet this also shows future product features years before publication.

\(^6\) This is why I advise evaluation labs and certification bodies looking for new employees to look for new hires to have “the mind of an attacker, and the heart not use that mind for evil, ever”. 
Minimal description of the CC technicalities

The CC is designed to allow for evaluations on a broad range of functionality and chosen assurance. To many first timers this is a new concept, they are used to certification approaches such as CE electrical safety testing that combine the functional and assurance requirements in one. SFRs and SARs have huge impacts on the evaluation scope, hence explaining the concept of SFRs and SARs is always needed.

Example explanation of CC’s SFRs and SARs

If you buy a “secure” product, you want to be convinced it is “good” for you. However “secure” and “good” can mean very different things to different people. For example:

- A government wants to know their highest secrets are protected against the best attackers from the other governments, even if it means destroying the secrets (“this message will self destruct in 5 seconds”).

- A gaming end-user might want his virus scanner to protect him from known malware when he is surfing the internet, but not when he is playing games.

- A medical organisation will tolerate the chance of having patient data read by unauthorised people, but have much higher requirements on availability of the critical patient data (“what do you mean access denied, I need to know his allergies now or this patient will die”).

Formalised security product evaluation methods such as the Common Criteria define what “secure” is, by translating them to requirements on the security functionality that needs to be present in the product (the Common Criteria calls this: Security Functional Requirements or SFRs for short), and to requirements on how (deep) it is verified that the product actually does the security functionality (Security Assurance Requirements, or SARs for short).

After defining with the SFRs and SARs what “secure” is for this product and how we are going to verify it, the evaluation starts. In the very core the evaluation is the creation of a convincing argumentation (following the SARs) that the product meets its security requirements (the SFRs).

---

7 Note these examples cover the major different ideas of what a “secure” product provides: Confidentiality, Integrity, and Availability (authenticity is rare outside the cryptographic domain).
Complexity of the Common Criteria is real

Although taking care of most of the complexity of the CC and in doing so hiding it for the client is in my opinion the professional offering of a consultant and co-operative evaluator, that does not mean that complexity isn’t there. A professional should not hide that complexity to her/himself.

Convincing argument the CC is complex

I was once asked: “while I trust you when you warn for the interdependencies and the ripple effect changes may have, can you show there really is no other way to simplify the CC?”. Alas I could not provide a convincing argument there was no simpler view, all I had were examples of such interdependencies and ripple effect. Of course convincing proofs of optimality are very difficult on something as like an evaluation standard and for practical purposes my insight in how to navigate the complexity were sufficient for the task at hand. Yet this question stuck to my mind as that had been bothering me too: how to give a convincing argument that the Common Criteria is indeed complex and there is no simpler path.

A couple of years later I finally had time to research this aspect. I chose to use a for me typical evaluation of a smart card: evaluating against the smart card PP [PP-0035], at EAL4+. Such an evaluation has about 150 work units to be performed. I took a week to fully map the work units of that EAL4+ and to identify all the interdependencies there. I ended up with identifying about 450 semi-explicit dependencies and then set out to make a consistent picture out of it with the help of the graph visualisation software GraphViz.

The first results were not promising. I tried for a long time to make a clear, yet fully detailed, image, but whatever I tried, no clean structure emerged.

---

8 “Lies-to-students” warning: There are a lot of different ways to structure such an evaluation and its documentation. I listed a work unit X as depending on work unit Y, when a change in work unit Y would in my opinion require revisiting work unit X to see if the work unit X was still true.
Somewhere I had been hoping to have been proven wrong, that there was an easier way out, even though I also trusted my insight in the complexity. So as a senior evaluator or certifier, knowledge of the dependencies and complexity will be needed to do the work with integrity.

Now it often is best not to show this full complexity to others for various reasons. In early training the full weight of this complexity is best hidden from the student so he can start with an image of the CC that is simple enough to make his own, and then to build all the intricacies from that later (a “lies to students” training approach that works well, see also my ICCC13 talk).

In my consultancy I consider taking care of the worry about that complexity to be deep down what I am bringing to the process. There are main paths through this complexity, if someone takes on the responsibility of taking care of those little but potentially nasty details.
Weight of Common Criteria

A recurring remark, generally from people not involved (yet) in the Common Criteria process, is to say that the CC is “too heavy weight”. Typically this is accompanied with saying the CC is a dead standard and/or another approach is better.

I consider looking critically at the CC and its competitors a healthy way to keep driving towards more (cost-)efficient CC evaluations. I however do not agree with the dead standard view, and my short smart-ass answer is often: “well CC is still there as the only real formalised product evaluation methodology so it must be doing something right...”.

The CC is not a lightweight evaluation methodology and that does weigh on the price of doing an evaluation. For one thing, it does not fit that well to quickly changing products (web services, software with new versions every 3 months, virus scanners, ...).

The heavier weight comes from a combination of aspects: the CC is very broad in application, can go quite deep and is internationally recognised.

Broad

The CC applies to a very broad spectrum of product types and technologies. From software to hardware, from operating systems to smart cards, from databases to data diodes, from fighter jet components to copiers. That means that the methodology must start from a more general approach and translate that to the specific product during the evaluation, whereas a product-specific methodology doesn't need to do that. You can see the difference in FIPS-140 for example, or EMVco on smart cards. Because those apply only to very specific devices, a lot of the generalised parts of CC are not needed in those evaluation methods. Yet even EMVco on smart cards for example is effectively a subset of a CC evaluation against [PP-0035] on that same smart card.

The translation of the generalised evaluation methodology to a specific domain costs something. In active domains, such as the smart card domain, this translation itself is standardised by the technical communities and hence those costs are shared and reduced.

Deep

The CC at EAL4, preferably with higher attack potentials, gives in my opinion a lot of assurance that currently no other evaluation methodology I know of delivers. That comes at a cost. It has to be clearer why a product is actually doing what it claims, which costs a lot of time, money and risk when the product isn't designed and documented for proving that it does. Evaluation labs dig deep into the product, check a lot and test even more (EAL4+ on smart cards run for a year+ and into €150.000+).
Internationally recognised

The CC is internationally recognised, so in a way we are applying a standard that is the superset of all the local views on how an evaluation should be done. Combining the views results in more work\(^9\) compared to one local view, but as there is a huge amount of re-use between the views, the additional work isn’t that high.

A typical example: in the past, smart card evaluations where done per national bank. Each of those individual evaluations was cheaper than a CC evaluation, but the total of all those evaluations was not. Hence the smart card community decided on one evaluation (CC against [PP-0035]). That one CC evaluation was more expensive than one old evaluation, but much cheaper than all the national ones (and much more scalable).

Conclusion

There are all kinds of optimisations and shortcuts in the application of the CC to a specific domain and there is much room for improvement of the efficiency. However these improvements are in my view best done the practice of translating the CC to a specific domain, not in the CC itself. Hence I do not agree with the feeling that CC is really inefficient and not vibrant as a standard.

---

\(^9\) Interestingly everyone will typically agree that “there is clearly too much overhead”, yet their views on what exactly is the overhead will be wildly different.
Security Target: The evaluation agreement

The Security Target (ST) functions as a multi-party evaluation agreement and that function changes during the evaluation process. In the beginning the ST is an agreement between the developer and his help (consultant) on the one hand, and the evaluators and the certifiers on the other hand. The ST defines what the product (Target of Evaluation: TOE) to be evaluated is (TOE scope in the ST), what security properties will be tested (the SFRs) and in what manner these properties of the product will be tested (the SARs).

For both parties this indicates the work that needs to be done: the manner of testing (the SARs) influences the depth of the documentation that needs to be made by the developer and checked by the evaluator and that checked by the certifier, and this needs to be done for each of the security properties (the SFRs).

Changing role of the ST during the process

During the evaluation the ST will be the place where the (re)negotiated agreements on the scope of the evaluation are captured. Between the three parties (the developer, the evaluators and the certifiers), the role of the ST shifts during before, during and after the evaluation.

After the evaluation the ST is officially an agreement between the developers and the end–users, but in the view of the certifiers it feels like an agreement between the certifiers and the end–users. Officially the certificate states that the certifiers are convinced that the evaluators verified by following the SARs that the product has the security properties (SFRs) under the assumption that the guidance is followed to the letter. Yet that official view is rarely really taken by the end–users.

Generally the end–users will see the certification as the stamp of approval by the certifiers, i.e. as an authoritative statement that their security problem is solved if they use this product. Certifiers are generally part of highly trusted government organisations, in most cases the national communication security agencies (NSA, GCHQ, DCSSI, BSI, NLNCSA etc), who also have the task to ensure the nation’s IT security so that expectation isn’t entirely strange.

Realistically at best the end–users will verify that their security problem is described in the ST in the Security Problem Definition (the Threats, Organisational Security Policies and Assumptions). In the CC evaluation view, the Security Problem Definition is further mapped to the objectives layer and then the SFRs. It is the SFRs that are tested, not the Security Problem Definition. In the intermediate layer, the objectives layer, responsibility of solving the security problem is distributed over the TOE and the environment in the form of objectives for the TOE (i.e. responsibility of the product) and objectives for the environment (i.e. the responsibility of the end–user). The objectives for the environment will end up in
the guidance, i.e. the evaluation result depends on the end-user reading the manuals and implementing this.\textsuperscript{10}

An end-consumer who does not read this layer, will have a mistaken view of the promises made by the evaluation and certification process. It is unfortunately common human nature to blame such a discrepancy on the process and especially on the certification body.

Hence the certifiers will feel responsible to try and ensure that the expectations of the end-consumers match those encoded into the ST. This is an unofficial process and has a tendency to be in the unspoken underground all through the evaluation. Moments where this process can be made explicit are:

\begin{itemize}
  \item Scoping of the evaluation contract
  \item First evaluation of the ST, especially the ASE\_SPD, ASE\_OBJ and ASE\_REQ,
  \item Kick-off with the certification body
  \item Any significant change to the ST
\end{itemize}

**Changes to the ST**

As the ST is the start and end of all evaluation activities, changing the ST significantly after the ST evaluation activities (ASE) should be avoided unless absolutely needed. The following parts of the ST have big impact on the remainder of the evaluation:

\begin{itemize}
  \item TOE Identifier (especially towards ATE and AVA)
  \item TOE physical, logical and life-cycle scope (in my experience most often underestimated, this has huge implicit effects on ADV, ATE, AVA and in many cases ALC and AGD\_PRE).
  \item Objectives for the environment (these should be in AGD determining the evaluated configuration together with the above scope, hence also influencing ATE and AVA, and often ADV\_TDS/ADV\_IMP).
  \item SFRs
  \item SARs
\end{itemize}

Reducing the claimed security requirements generally is an option during the evaluation as this mostly\textsuperscript{11} means dropping parts of the work.

\textsuperscript{10} This is in my opinion one of the biggest gaps between CC evaluations and the application to the "real world".

\textsuperscript{11} “Lies to students" warning: I’m severely simplifying things here. The full complexity of the CC comes into play when doing this, including generally nicely hidden details such as EALx not automatically covering all EALx-1 work units.
As an aside, this process of dropping security requirements because the TOE doesn’t meet them, or covering such holes with the guidance, is something I explain to end-users. It is a very clear reason why all guidance is indeed mandatory: the guidance might be “covering up” something that they consider a vulnerability. I prefer the ST and the guidance to be very clear in such areas myself.
Very rough impact of SFRs/SARs on costs

Beginning developers often ask about the costs of an evaluation and don’t yet understand the complexities of a CC evaluation, let alone the impact on the work the evaluation lab will have to perform. To help them get an idea of the scoping, I’ve been using these rules of thumb as explanation with success:

A very rough rule of thumb is that the amount of evaluation work doubles per EAL level. This is a rough rule of thumb and largely depends on the quality of the evaluation evidence, structure of the TOE and the experience of all involved parties. Some SARs are more expensive than others because of the difficulty to show them, such as the module-level testing requirements (ATE_*). ALC_FLR is generally a cheap SAR to add (as it just requires showing a procedure for handling known vulnerabilities).

The amount of work scales roughly linearly with the amount of SFRs: twice the amount of SFRs means twice the amount of work. Important for the costs is that the SFRs can be clearly shown to be in a specific small area of the TOE (TSF is small). Auditing requirements (FAU_*) often need to be implemented all over the TOE, especially when using one of the pre-defined audit levels in FAU_GEN.1.1 (minimum, basic, detailed), and hence can be really expensive. If auditing absolutely needs to be included in the SFRs, try hard to use a self-defined level for auditing.

SFRs that state that the whole TOE will not do something often are difficult to show and expensive to verify (because they are not implemented in the whole TOE). For this reason for the availability (FRU_*) and privacy (FPR_*) SFRs are generally much more expensive. Be especially careful with FDP_RIP too, it might be easy or really difficult to show it is implemented and tested.

12 “Lies to students” warning: When using this, be aware and clearly communicate that this is an overly gross simplification to budget precisely on. Already at this level we’re not disclosing that EAL1 and EAL2 are generally close to each other in costs. Let alone that an EAL5 can be very cheap upgrade from an EAL4, or really expensive, depending on the TOE and developer’s situation.
Iterations: costly

Common Criteria evaluations are not one shot, but will have iterations: the developer submits evaluation evidence, the evaluator finds something unclear/wrong/.. and sends observation report, developer submits new evaluation evidence and it starts again.

The same iterations, hopefully less frequent, occur between the evaluator and the certifier. These follow similar patterns like the iteration between the developer and the evaluator, except for situations where the raised issue goes back to the developer. Escalations to the developer are much more likely to have big impact such as changes in the TOE, production procedures, the site’s security, ... . Because of the big impact on all parties, such items warrant a good communication and prevention.

These iterations are the biggest variable costs factor in the evaluations for both the developer and the evaluator, in terms of money and especially also in calendar time consumed. The extra costs of fixing the evaluation evidence and re-doing part of the evaluation task are quite clear (although due to all the dependencies, this might be much more than an inexperienced person will expect).

The extra calendar time such an iteration takes is however often underestimated. Such an iteration is likely to take 1-2 months of calendar time: the evaluator is performing an evaluation task (typically a complete class or family, like ASE or ADV_TDS) and will gather most of the questions / unclarities / fails to deliver them in one go. After all, some of the questions might resolve themselves further in this evaluation task due to new insights, some might be part of a greater question, and sending each issue individually creates just too much overhead.

A full iteration’s costs breakdown

An evaluation task will take between 1 week to 1 month of work and has some administrative overhead (version and quality management, often requiring review and approval of senior evaluators or managers).

The developer receives the comments. The developer will need time to understand the issues that need to be fixed (often this involves calling / emailing the evaluators) and assign the people to perform these changes. Unless the developer has people waiting for the feedback (inefficiently idling), just this process step can take the better part of a week or two.

Assuming the people immediately start working on it, the changes will not be trivial (otherwise the evaluators would have labelled them as understandable typos and not raised a major issue for them), and will take days to weeks (and if it is a big item in the 1-3 months range).

After the content changes, the proper version management procedures must be followed. This commonly includes review, optional improvement and approval steps, and then secure sending to the evaluation lab.
In total the calendar time taken for such an iteration will rarely be less than a week and generally will be more in the order of 1–3 months (even if the actual amount of work spent is just a fraction of this, scheduling and availability inevitably introduces delays).

Hence from management point of view, controlling the amount of iterations is by far the best approach to control the cost (internal and external) and especially the calendar time (with the all important time-to-certified product).

Next best is to plan the delivery time of the observation report, keep to this time and allot sufficient time to fix it. This will allow some reduction in the turn around times because people will be ready. However the improvement here really is not that big as we’re talking only about the 1–2 weeks of waiting for scheduling per iteration and besides: somewhere in the process there will be a time buffer to accommodate unforeseen delays in that process.

**Reduction of amount and impact of iterations**

Not iterating is by far the most efficient manner to reduce this variable cost factor. Some iteration will always occur, yet high quality evidence, designed for efficient evaluation/certification, will significantly reduce the risk on the need for heavy observation reports simply because the difficulties of getting the convincing argumentation clear have already been conquered (typically by a consultant experienced in the evaluation practice, or an experienced evaluator). Such evaluation evidence can also be designed to be easier to maintain in the face of changes by making the evidence modular and avoiding duplication of information where possible.

**What can the evaluator do?**

The evaluator must be convinced himself and build a convincing argument for the certifier, hence the evaluator cannot (or at least: should not) accept items not meeting the criteria. The evaluator does however have the room to accept minor typos and inconsistencies that do not impact his proper understanding, as well as asking the developer additional questions to clarify items.

The most effective way to reduce the iterations however is to clearly and exhaustively describe what is missing to complete the convincing argument. Not by writing what should be written (that would be a consultant’s task), but by writing what information or aspects are missing for the evaluator to complete his convincing argument.

A good goal is to have the observation report as the complete to-do-list for the developer: if all items on the observation report have been addressed, the criteria should be met (i.e. this is the last iteration). Due to the complexity of the Common Criteria creating unhoped for ripple effects, or due to prior misunderstanding resolving and changing the evaluator’s view on the issue, this might not actually be the last iteration.
Another, if somewhat counter-intuitive, method of reducing the iterations, is to be extremely reluctant to start a new iteration if the set of evaluation evidence is clearly not sufficient to reach the criteria. For new evaluation tasks, it can be very cost-effective to quickly scan through the evaluation evidence and get an impression whether there are major showstoppers blocking a pass-verdict. For updated evaluation evidence, a quick scan to see if it seems to address the previously identified missing items properly. Both types of quick scans can and should be integrated in the acceptance of new evaluation evidence. A good method to have the developer support you in this process, is to give the issues in a clear list/table format and have the developer quickly point to where in the new updated evaluation evidence this is addressed. This has two advantages: the evaluator has a much easier task in finding the argued changes, and the developer is guided towards ensuring all issues are addressed.
Costs and contracts

Evaluation contracts can be in time and materials, or as fixed price. Generally fixed price contracts are strongly preferred by developers for easier budgeting. For the evaluation lab, the quality of the input (the evaluation evidence, TOE, and especially the speed with which the developer learns to repair defects in these two areas) is a great unknown and crucially influences the risk and impact of these iterations. A fixed price evaluation contract ending in successful evaluation of the TOE is from the evaluation lab’s perspective economically very risky as the lab has no control over the input (and also is highly dubious for the independence of the evaluation lab, as it now has a hard incentive to just approve the TOE even if they know there are flaws in it).

A common meet-in-the-middle solution is to offer a fixed price contract with escape clauses such as an upper bound on the amount of iterations (common: 1, sometimes 2, full evaluation rounds and 1–3 rounds for “small” fixes), on the amount of site visits (common: 1) and on the amount of testing needed (essentially an a–priori estimate on how convincing the design information will show that attacks will or will not be needed).

Developers and evaluators both should be aware that for the developer the switching costs of going to another evaluation lab when these limits are reached are huge. At the same time the developer will be under high pressure internally and externally to finish this evaluation. This seems to put the evaluation lab in a very strong renegotiation position, at the cost of severely straining the relationship with the developer. At the same time, when reaching this point, evaluation has been running quite strongly beyond the expectations it started with, so the evaluation lab can also at that point have found out that their expectations of the input quality have been wildly optimistic (and quite possibly will have run over budget themselves too). Emotions of frustration and feelings of pressure will put strains on the relationship.

In this emotionally charged situation, any hint of misuse of either party’s situation will damage the long term relationship. Hence it is a good idea to already have a clause in the contract that determines the financing approach in that case (often this is a time and materials construct with a fixed hourly rate). In any case at least an agreement on how to handle this unfortunate situation should be reached before the start of the evaluation. Discussions on price when everyone just “wants the <insert favourite cursing words> evaluation to be done” are going to be emotionally heated and will strain the relationship a lot.

As a developer it increases my trust in an evaluation lab if they take the effort to explicitly address this during the contract discussions, or at least react openly when the developer raises this issue.

As a lab I consider addressing this issue up front a serious risk reduction step, assuring that even during that somewhat likely event of frustration the long term relationship to the client isn’t damaged.
Interpersonal sides of the evaluation

Common criteria evaluation work is also work between humans and subject to interpersonal processes. In the end we are all human, including the “money centred” consultant, the “way too difficult” evaluator, and the “annoyingly headstrong” certifier.13

In my view being able to navigate what happens between the people in the roles and how that influences the process, is important for any professional in the Common Criteria evaluation process.

Even though we are working in a highly technical field, this requires skills in interpersonal relating: to be trustworthy, to handle the frustrations between the parties, to play the games between the parties, to keep your integrity and to pass something that is good even though it is not perfect.

Trust

For a CC evaluation process to flow smoothly, trust between the parties and especially between the people is important. This trust has many aspects, from keeping secrets, to not punishing showing uncertainty or even irregularities in the evaluation, to keeping the other’s dignity and standing. In the end trust is all about feeling the other will not take advantage of you showing vulnerability and in doing so hurt you.

Trust is really fragile, it grows slowly and can break very abruptly (and just like a vase that is broken, broken trust can sometimes be repaired but will never be as good). The main way to create trust, is to allow it to grow and consistently not break it.

Trust grows fastest when you are consistent and predictable in your behaviour, transparent and honest about your reasons, and have room for the other to have different views without making that wrong.14

---

13 I have fulfilled all these roles and have been called all these things. I fully expect that it won’t be the last time either.

14 This does not mean you have to accept their point of view.
Frustrations

Frustrations in the process occur due to valid conflicts of interest (a certifier not going along with a too big diversion from the standard) but also due to miscommunication and simply stress in the process.

Especially near the end of the evaluation, all parties will have had enough of seeing a specific issue return and not be resolved again. The frustration there will be much higher as the impact on the certification time will be seen as higher, and because everyone will have a feeling of not being understood by the other. This feeling of frustration tends to reach a critical level about 75–85% into the evaluation: just before or around the functional and vulnerability testing.

Personal contact

An effective strategy is to ensure that everyone knows the human behind the role a bit. Having some social time together, drinking a thee or a beer together, prior to the crunch time will help a lot to smooth the unnecessary friction. It simply becomes easier to pick up the phone or visit and work the issue out. At the very least ensure that key players meet at the kick-off of the project.

Even in this quite technical domain, non-verbal signals matter a lot in the communication. In-person meetings are best to defuse tense situations\textsuperscript{15} for this reason, however tend to require quite a bit of planning and travel, and is often seen as inefficient. Video conferencing is still uncommon but could be a good option.

Phone calls work well to discuss more sensitive and/or subtle items, especially for sounding out the reaction to uncovered issues, changed plans and such.

Give the other room

Take care to give the other side enough time to consider the issue and form an opinion, otherwise they will feel either pressured or feel not held to their “off the cuff” remark. Send an email with describing the issue and the requested result well before the call to allow the other side the room to decide their stance. Splitting the description of the issue and the request from the actual decision in the call works well too: Start the call by saying that you want to explain the situation and get a feel of what their view is, and explicitly state that it is not the intention to make the decision right there. Follow up on the item with a written request for that decision (with all the technical details that may be necessary).

\textsuperscript{15} Excluding situations where personalities clash. I have in consultancy work advised to not meet because say that particular programmer at the developer and that specific evaluator assigned to that task for some reason always ended up fighting (and there wasn’t time to coach either of them on this).
Games inspectees and inspectors play

The relationship between developer and evaluator, and similarly of evaluator and certifier, have both cooperative and competitive aspects on several levels. These aspects are not unique to the Common Criteria, they happen between inspectors and inspectees in the nuclear industry, pollution control and education quality [Handhaving].

In the CC evaluation process the information needs to flow from developer to evaluator to certifier. The higher the EAL, the more detailed the information becomes, leading to what in the research on enforcement of rules is called “inspectee intimacy”: the evaluators (the inspectors) know intimate details of the developer (inspectee). Similarly the certifiers will know details on the evaluators’ processes and approaches. This makes the inspectee dependent on the inspector, but also the other way around: the inspector needs the honest information to make the right decision. Together this creates two-way dependencies (for the inspector to get the information, for the inspectee to get the approval) and competition (the inspectee might like to get items approved the inspector does not want to approve).

At another level, the cooperation / competition split can also be seen in the way the inspector operates to the inspectee. In cooperative mode, the inspector tries to teach / negotiate / adapt the inspectee towards complying to the rules by explaining the goals and levels needed (this is in the literature often called the “educating inspector”), and by adapting to the specific situation of the inspectee. The underlying assumption here is that the inspectee in principle wants to comply, but does not know what he needs to comply to, or how. The relationship is considered to be important here and both forms the basis of this process and is strengthened by it.

In competition mode, the inspectee seeks to avoid complying to the requirements. A whole range of behaviour is available to avoid compliance, from hiding the non-compliance, to challenging the requirements, to challenging the competence of the inspector. The inspector in this model compensates by going to the “sanctioning inspector” mode.

As inspector (evaluator to the developer, certifier to the evaluator) one can use this cooperation / competition split as a “good cop, bad cop”-approach. Explanation of what as inspector you are looking for, what is missing to meet the criteria, and possibly examples of approaches that would meet the criteria, help the inspectee and seduce him to follow this path (the “educating inspector” as “good cop”).

Backstop of the approach is the limits to which adaptations can be done. An external “bad cop” can be introduced to personify this. For evaluators this is the internal reviewer but more often the certifier, for certifiers this is the head of the scheme or colleague certifiers of other schemes.

One thing I have seen happening in myself and others as an inspector, is that the more the limits of what can be accepted are sought by the inspectee, the more your role as inspector changes from “good cop” to “bad cop”. The more the inspectee in
the view of the inspector pushes to the limits, the more the inspector starts to enforce the limits personally. In the beginning this is stated at “they will not accept this”, beyond a certain threshold it becomes “I will not accept this”.

There is a memory effect to this process: Repeated and stressed pushes to the limits makes the inspector start more in the “bad cop” sanctioning mindset. Another way of formulating this is that the worse the relationship is strained, the more strict both parties will become and will end up in a trench–warfare stalemate. In the CC stalemates are in the end, for that battle, won by the inspector: the certificate is not issued. In the long term, this causes the inspectee to avoid the inspector, with loss of work in the lab or scheme as a result.

Dilemma sharing/client intimacy

In this approach, the inspectee shows the inspector the dilemma the inspectee is faced with and invites the inspector to think with him in solving the dilemma. This approach is often fruitful in regaining the more cooperative view of the inspector and in that way is a positive strategy. A sufficient remainder of interest of the inspector to keep/regain the relationship is needed for this. By choosing the dilemma such that either solution is acceptable to the inspectee, the inspectee can try to manipulate the inspector. This “false dilemma” approach is hard to spot for the inspector. For one it is a well known rhetoric approach to focus the listener’s (inspector’s) mind on only these two options, putting the listener in the box so to say. The inspectors also generally like to feel cooperative, to be part of a solution, not just the problem.

A warning on the use of this false dilemma approach as inspectee: Even the suspicion of manipulative use of this approach can severely damage the relationship. Such suspicion is more likely to arise when the relationship is already weakened, leading to a downwards spiral that is very hard to break. Once the inspector start mistrusting every single thing you say, convincing him will be hard and ultimately much more expensive.

As inspector, the risk of going too far along in this approach should be considered. In general, my advice is to keep away from making creating the solution. A very fruitful approach is to show only the advantages / disadvantages for the options (but let the inspectee generate them), and if possible give a verdict on whether such an option would be acceptable or not.

It is best to be pessimistic and honest in these processes: a provisional “pass” verdict will be heard as a definitive one, making a change to a “fail” verdict hard to understand and accept by the inspectee. I have learned to only give “pass” estimates when there is no serious doubt, i.e. almost as a promise to pass it. If it is likely to succeed but has some doubt, explain the doubt clearly. This allows the inspectee to address this doubt, for example by generating another option right there or avoiding the failing behaviour when executing the option. Conversely,

16 As always with a cop-out possibility of more complete insight to fail it.
options that are almost certainly to fail should be labelled on the spot as such, preferably with explanation of what makes it fail. All other options should be labelled as “unknown” or maybe “could work, I don’t know”.

Risk of leakage

As inspector one has often seen the solutions of other inspectees to solve the problem at hand. It can be very tempting to show such solutions to the inspectee, especially if the solutions do not seem particularly innovative/proprietary, in a desire to seem cooperative or to co–create something (in contrast to “always failing something”). This is quickly becomes a violation of the common secrecy requirements (of the CCRA Article 5, the scheme and any NDAs between the parties) and undermines the required intimacy for effective interactions.

Especially beginning inspectees, who feel they have to catch up to the more experienced inspectees, often seek such help. I find that it is useful to explain my refusal to leak that knowledge by reflecting it back: “Let’s say I explain to you today how competitors are doing it. Tomorrow I am at your competitor <insert their nemesis>. You don’t want me to explain to them what you do, do you?”.

Another aspect of this, is that it is human nature not to be able to see the flaws in one’s own work, but easily spot them in others’ work. By co–creating the solution, as inspector you will no longer be able to adequately see the weaknesses in it. Especially with higher attack potentials this is a serious danger to the assurance.

There is a highly positive result to this approach too: I have seen time and time again that denying the knowledge of how other inspectees are solving the problem at hand, stimulates the innovation resulting in wondrous and completely unexpected out–of–the–box new solutions. Explaining what the properties are that the solution has to have to solve the problem is the key in facilitating this.

An example of a domain where I am very active: A major field of attacks in the smart card domain is side channel analysis. There is a huge set of specific attack techniques, as well as defence techniques. The attacks all depend on the combination of two aspects: an operation is performed on secret data, and via some side channel that operation leaks information sufficiently and often enough to reconstruct the secret data. Defence mechanisms seek to destroy either, or preferably both, of these aspects.

Note that this description in itself does not describe how to protect, but gives sufficient direction for designing solutions. Of course more technical knowledge such as the type of side channels available, amount and type of leakage of those side channels, etc all feeds into this, but with this information I have seen developers creating the most surprising and effective solutions.

17 As I write this book and polish this to my idea of perfection, I fully expect you as a reader to find all manner of mistakes and missed insights in this document I can no longer see. Please be so kind as to give me that as constructive feedback (see Contact info at the end).
False suggestion of knowledge and authority

One of the inspector strategies is “the suggestion of knowledge and authority”. Here the inspector suggests more knowledge than he actually has. I have seen this behaviour in junior and otherwise uncertain inspectors. I think this is a poor strategy within the context of the Common Criteria. It shifts away from the core idea that the inspector has to be convinced by the inspectee to give the approval (pass evaluation report/certificate). When challenged on the faked knowledge or authority, the inspector is in the situation where he can only admit not knowing (and lose a serious amount of authority and respect) or avoid that challenge by even stronger claims on his authority (further increasing the stakes). I’ve seen this process explode and been called in to repair it, it never is pretty.

In my experience it is much more productive as an inspector to use the strategy “I need to make a convincing argument and I am currently missing … as information. Can you <inspectee> help me?”.

For the inspector the trick to this strategy is not to feel inferior, unintelligent or as a result less authoritative for not knowing the information in question. For me this rests on knowing I did all I could in keeping up with the technology developments and attack technology, leaving what I do not know as specialised and interesting knowledge to learn¹⁸. Asking the inspectee to explain it clearly to you as inspector gives the inspectee the room to show his expertise and the smart inventions. In the many times I have done this, I have never once been rebuffed or encountered difficulties in this approach and always learned something new. Simply starting the question with “I might be asking something silly as someone not coming from your domain, but <insert question>?” goes a very long way. As a side effect of being honest on not knowing, this knowledge can’t be used against you as leverage and the downwards spiral of bluffing doesn’t start.

¹⁸ Having genuine curiosity on that new technology/trick/method/… will help a lot in convincing the inspectee, and in a greater picture, to enjoy this work.
Saying “pass”: the difficulty of approval

In myself in the various roles and in others I have seen over and over that there is a significant learning hurdle before strict but fair approval can be given. Clearing this hurdle as a person makes the difference between evaluator/certifier that requires supervision and one that can perform this task efficiently and effectively by himself. Clearing this hurdle as an organisation (evaluation lab, certification body) makes that organisation much more efficient and powerful. Clearing this hurdle as a scheme, means the scheme will soar in productivity and standing (and then hit practical growth limits like the speed of hiring and training).

As evaluator and as certifier, one must be strict enough to ensure sufficient assurance has been reached, but not too strict so that the whole process becomes unnecessarily heavy or even get stuck completely. Balancing the two sides is difficult and requires a good understanding of the rules surrounding approval and the self-confidence to say something is good enough (even though it is not perfect).

The rules of approval are in this dilemma the easiest part. The CC, CEM, domain guidance and scheme rules, mixed with some knowledge and experience, define what is sufficient. In the strictest reading, CC part 1 and part 3 define what needs to be shown (CC Part 1, Section 7.2 “Evaluation”, “The TOE evaluation consists of applying the SARs (from the Security Target) to the evaluation evidence.”), and the CEM work units help explain what tasks are expected to show this. The text under the CEM work units is officially just guidance, a guideline to help the evaluators in their task (CC Part 1, Section 7.2 “Evaluation”, “How the results of applying the SARs are documented, and what reports need to be generated and in what detail, is determined by both the evaluation methodology that is used and the evaluation scheme under which the evaluation is carried out.”).

In many schemes however, the CEM guidance is taken as strong guidance, and any deviation from the work units or even a paragraph of the CEM guidance may require explanation. This can be in response to certifier questions on a case by case basis, a balanced approach (the UK scheme for example describes the usually expected content per work unit in their scheme guidance), and often this grows to the size that every CEM paragraph is addressed up front to avoid such costly discussions.

This determination of what needs to be addressed was the easier part. A harder part is in the judgment whether some requirement is met or not, especially in those cases where this is a subjective call. And the CC contains these too, as any evaluation method that is not overly restrictive will require some human judgement, i.e. as much as some people would like to avoid this, there is a measure of human judgement involved.\(^\text{19}\)

\(^{19}\) I personally am more and more of the opinion that it is the subjective judgements that are the ones that add the value in any evaluation, as those are the ones where the human evaluator is thinking about the issue and putting her/his reputation on the line.
Especially requirements asking the evaluator to assess that something is “clear enough”, “sufficient”, “secure”, “consistent” etc in the end are a subjective judgement call. For example in APE_INT.1–3, CEM155, the italic items in that work-unit will require some form of judgement: “The evaluator determines that the overview is clear enough for TOE developers and consumers, and sufficient to give them a general understanding”.

In the 80–20 rule terms, this is the 20% that generates 80% of the thinking and doubt for the evaluator / certifier.

It is much easier to find a reason to fail something. There are always many reasons why something might not be sufficient, or might be labelled as such. Not all terms will be clear 100% (is the configuration of an FPGA: software, firmware, or hardware?), there is always a way to misread it. Some terms that are well understood by the target audience in an organisation specific market might be unclear or have different meaning for the evaluators/certifiers (DFA, FSM, GUI, platform, ...). There will always be some remaining typing and language mistakes in the text. The checks done on say consistency or source code can be done in ever more detail. All 100 interfaces have been tested but not in all $9,33e^{157}$ (100!) calling orders. And the list goes on and on...

So it is quite easy for an evaluator/certifier to keep finding potential improvement points (sometimes even the developer himself starts doing this). However enough is enough. Assurance is just that: we have done the SARs and are reasonably convinced the TOE does the SFRs. The goal is not to be absolutely 100% convinced, but to be reasonably convinced.

Making this step takes courage and being convinced it is probably not perfect, but it is good enough.

The rest is just delaying this decision.

From game-theory, this makes sense in emotional cost/benefit analysis: there is little reward in approving something except the feeling of doing the right thing, very little cost in delaying except for management and client pressure, and only potential punishment occurs in the case one approves something that with hindsight isn't quite good enough (or seems to be).

Still it is delaying the decision that it is good enough.

For beginning evaluators and certifiers, the best guidance to where this limit lies, is of course gotten from your experienced colleagues. A good way to guide the discussions, is to force yourself to describe exactly what proof obligation in what

---

20 Differential Fault Analysis, Deterministic finite automaton, Dual Factor Authentication, ...

21 Finite State Model, Force Scanning Microscope, Flying Spaghetti Monster, ...

22 Note that I’m not offering a fee per mistake found in this book, even as I will reward you sending me errata or engaging in discussion with me to improve this book.
CEM paragraph you cannot verify, and where you looked to find that proof. For example: “APE_REQ.1–3, CEM267: The evaluator has read the ST and has found in SFR FXY_ABC.1 the labelling ‘top_secret’ and ‘secret’. The evaluator has searched in [ref1], [ref2], and [ref3] for these words and has not found them. The evaluator does not consider ‘top_secret’ and ‘secret’ common knowledge. It is not clear to the evaluator whether ‘top_secret’ is higher than ‘secret’, which fails the work unit.” As you can see, this shows the path taken in the attempt to verify the claim, and gives good points to discuss whether and where the verdict is too strict.

Describing it in terms of a path towards approval and identifying the obstacles in that path has three benefits:

1. It forces the evaluator/certifier to make explicit what is missing or incorrect, forces to take a clear stance on the issue.

2. The description at same time conveys to the writer of the original document what is missing, what task could not be performed, allowing the writer to understand and fix this exact problem.

3. When the problem is fixed, changing the evaluation/certification analysis is simple. In most cases it is a mere matter of removing a “not” in the sentence and changing the status of the work unit.

Writing for approval does not mean: accepting what is wrong. It means you should write the analysis with the intention of showing that the requirements are met and identifying all the obstacles that keep you from achieving this goal.

---

23 For the record this is too strict in my opinion: ordering of “top secret” as more secret then “secret” is common knowledge in the security field. The example does show that writing the reasoning for not approving it will clarify the issue.
Careful with previews

During the evaluation process, the developer will often ask you to give your impression. This is very understandable: The developer has been stressed during the process. This especially goes for site visits, design reviews and vulnerability analysis, after all he also knows the weight these are carrying. Every time you looked thoughtful or wrote down a note to yourself, there is a feeling of dread for the developer.

So your statement will have a heavy weight. A thoughtful “hmmmm” quickly sounds like a disapproval, and a “this looks good” will be heard as “you will pass”. You should be truthful to the final judgment, changes in the verdict (especially from pass to fail) will be hard for the developer.

If you are not (yet) extremely certain about your verdict, resist the temptation to say it will be ok. It is better to be honest in that you do not know yet and have learned that predicting it is too often wrong. In vulnerability testing this is especially true: a TOE can seem to be invincible until that last “what if I do this? Heeee that is strange…” moment where the TOE suddenly breaks.

If you are not sure on how to construct a convincing argument with the things you saw and you feel comfortable in sharing this with the developer, by all means do so. Saying to a developer that you are trying to prove he meets the criteria but you miss this one step to make the argumentation, shows him you are working with, not against, him and often results in the developer suggesting alternative arguments you did not consider.

Just don’t promise him everything will be ok when you’re not yet sure it will be.

---

24 This process also applies to evaluators asking early results of the certifier, although generally evaluators will understand the concept that only a final pass verdict counts.
Contact

Obviously I welcome any questions, remarks, errata, discussion or just a thank you. By far the best way to reach me is by email to wouter@yourcreativesolutions.nl.

Catching me at a conference such as the annual International Common Criteria Conference is of course also possible. Look for me at my talk or in a lively discussion.

Thanks, errata and further information related to this book can be found on http://www.yourcreativesolutions.nl/books/.

Already thank you for the feedback and reading this book until the end!

With warm regards,

Wouter
Bibliography

The Common Criteria can as usual be found on the Common Criteria Portal <http://commoncriteriaportal.org>/.
