Evidence-Based Trustworthiness of Internet-Based Services Through Controlled Software Development

Conference Paper · April 2015

DOI: 10.1007/978-3-319-25360-2_8

3 authors, including:

Nazila Gol Mohammadi
University of Duisburg-Essen
47 PUBLICATIONS **249** CITATIONS
SEE PROFILE

Sachar Paulus
Hochschule Mannheim
61 PUBLICATIONS **517** CITATIONS
SEE PROFILE

Evidence-based Trustworthiness of Internet-based Services through Controlled Software Development

Francesco Di Cerbo[1], Nazila Gol Mohammadi[2], and Sachar Paulus*[3]

[1] SAP Product Security Research, Mougins, France
francesco.di.cerbo@sap.com
[2] paluno - The Ruhr Institute for Software Technology, University of Duisburg-Essen, Duisburg, Germany
nazila.golmohammadi@paluno.uni-due.de
[3] Mannheim University of Applied Sciences, Mannheim, Germany
s.paulus@hs-mannheim.de

Abstract. Users of Internet-based services are increasingly concerned about the trustworthiness of these services (i.e., apps, software, platforms) thus slowing down their adoption. Therefore, successful software development processes have to address trust concerns from the very early stages of development using constructive and practical methods to enable the trustworthiness of software and services. Unfortunately, even well-established development methodologies do not specifically support the realization of trustworthy Internet-based services today, and trustworthiness-oriented practices do not take objective evidences into account. We propose to use controlled software life-cycle processes for trustworthy Internet-based services. Development, deployment and operations processes, can be controlled by the collection of trustworthiness evidences at different stages. This can be achieved by e.g., measuring the degree of trustworthiness-related properties of the software, and documenting these evidences using digital trustworthiness certificates. This way, other stakeholders are able to verify the trustworthiness properties in a later stages, e.g., in the deployment of software on a marketplace, or the operation of the service at run-time.

Keywords: Trust, Trustworthiness, Software Development Methodology, Digital Trustworthiness Certificate, Metrics, Evidences

1 Introduction

The adoption and acceptance of Internet-based services by the end-users are largely dependent on whether users have trust into these services. [15]. Security and trustworthiness have become more critical because of ***a)*** ubiquity of the Internet makes it difficult to produce secure software in the first place, ***b)*** the

* Authors by alphabetical order. Corresponding author: s.paulus@hs-mannheim.de

increasing general distrust into the Internet (e.g., related to Snowden's revelations[4]). Existing software life-cycle processes and approaches were not able to successfully address this important requirement. Despite different theoretic (e.g., automated proving, data flow analysis techniques etc.) and practical (Common Criteria [10], Microsoft SDL [25] etc.) approaches, we still often see that software either is not able to be (or stay) secure, or that it is not used for potential trust reasons, or both. The authors believe that the current software development and operations approaches need some improvement in order to successfully address the complex issue of delivering trustworthy services. The major development practices towards trustworthiness used today are either too limiting and constrained (leading to a very complex and costly process) or too much based on the individual competences of the people working on the software (where the quality and trustworthiness of the software/services is difficult to manage). In case of being too constrained, it would actually impact usage, usability or simple development efforts negatively. In the other case, with a too "loose" methodology, it will be with high probability vulnerable and less aligned with the necessary goals and elements towards trustworthiness.

Proposal Overview. This paper proposes a new type of approach towards developing, deploying and operating trustworthy Internet-based services. The basic idea is to create a trade-off between the formal, *constrained* approach and the rather informal, *uncontrolled* approach. Hence, we aim at a *controlled* approach, similar to the introduction of software quality in general, but in our case adjusted to the requirements of trustworthy software and (Internet-based) services.

A new stiff development process will therefore not be successful at a large scale (one of the drawbacks of Common Criteria is that it mandates a certain development model which in practice typically leads to run the development process twice: ***1)*** for the software, and ***2)*** for the required documentation for the certification), due to the existing variety and fragmentation of development practices and methodologies. To introduce some sort of control during the software development process, we therefore need to be development process agnostic. Moreover, since the trustworthiness of software and Internet-based services can be highly impacted during deployment and operations, the approach must take the full life-cycle of the software into account.

We introduce measurements and thus evidence collection in different phases of the software life-cycle, e.g., metric values that express the degree of fulfilment of trustworthiness properties. They allow some sort of *prediction* of the to-be-expected trustworthiness properties, and thus to steer and manage development activities and correspondingly focus and financial investments towards the trustworthiness goals. The trustworthiness evidences can be applied to any development process model, and may be used to evaluate the usefulness of trustworthiness-enabling capability patterns [19] as best practice elements (e.g., threat modelling, data flow analysis, user experience design etc.). It is, though, important that the evidences fulfil a number of conditions so that they can be successfully used. For details, see section 4. We use the digital trustworthiness

[4] See http://www.theguardian.com/us-news/the-nsa-files for an overview

certificates to express trustworthiness properties in a verifiable manner that also cater assertions on the observed evidences. This allows to propagate trustworthiness evidences to subsequent phases of the software life-cycle like in software provisioning, e.g., in a marketplace, the values of the trustworthiness metrics may serve as input for a decision support system for selecting solutions [5]. To sum up, any software life-cycle methodology can be enriched by an evidence approach (identification of trustworthiness qualities, definition of metrics and measurement procedures, evidence collection and interpretation) to make the trustworthiness of the developed software manageable and controllable. Quantitative and qualitative trustworthiness assertions expressed in digital trustworthiness certificates allow to propagate this control to other parts of the software life-cycle.

To facilitate the acceptance of proposed approach, a light-weight certification scheme is developed where, the deployment and operation of an Internet-based service can receive a certification based on the evidenced and used digital trustworthiness certificates. This process is light-weight, since it does not require any additional development-related activity if our approach is used. The only remaining additional step is a validation of the (quantitative or qualitative) assertions in the digital trustworthiness certificates by an independent body. To achieve a fast market adoption, a self-validation may be an option as well.

The remainder of this paper is organized as follows: Section 2 provides a brief overview on the fundamental concepts and related work with respect to addressing trustworthiness. In section 3, we propose different categories of development methodologies, and in section 4 we describe the elements necessary for our approach. Section 5 presents our solution in the context of different application scenarios. Section 6 concludes the paper and an gives outlook on future work, including a potential process certification for demonstrating the successful application of the overall approach to a software product or Internet-based service.

2 Background and Related work

This section summarizes the result of our analysis on existing software development methodologies in buliding trustworthiness into software. A brief overview of development methodologies can be found in our previous work [24]. Although, it does not specify how these methodologies contribute to the trustworthiness of the end-product. There, we did not aim on providing solution to extend these methodologies in enabling trustworthiness.

There are many research contributions towards constructive quality assurance of software systems, proposing guidelines, principles, and methodologies for developing high-quality software. There are also more generic and well-established development methodologies that can be tuned into more security-aware variants, and there are specialized constrained methodologies that are probably not relevant for most organizations. Sommerville [28] states that reuse-oriented, or test-driven development can, in principle, result in trustworthy systems as well, since continuous user feedback, reuse, and early testing can enhance

software quality and mitigate risks. Eliciting end-user requirement is a key aspect of User-Centered Design [29], which can be seen as potential to consider users trust concerns early in the development. Weigert [32] conclude that model-driven engineering [26] significantly facilitates the development of trustworthy software. TOGAF [13] is a comprehensive framework for developing enterprise architectures (e.g., guidelines, patterns, and techniques) based on stakeholder requirements.

Some approaches have been standardized and explicitly focus on certain software quality attributes, mostly considering security. ISO 27001:2013 [11] considers certification based on the development and operation of Information Security Management Systems, and explicitly addresses requirements for secure software development. Common Criteria (ISO 15408) [9] aims at evaluating and certifying software systems with regard to security properties, whereas SSE-CMM (ISO 21827) [10] proposes a maturity model for developing secure software, mainly covering organizational and process aspects. The Building Security In Maturity Model (BSIMM) [2, 3] initiative also aims at assessing maturity and describing related activities. Process-independent best practices for developing secure software are proposed in [14, 18, 16]. Furthermore, some projects, such as OWASP [22] or SHIELDS [27, 17], provide methods and tools for detecting, assessing and mitigating security hazards and risks.

To best of our knowledge, existing development methodologies for trustworthy systems typically focus on robustness, correctness and security functionality, while there is a need for a broader view of trustworthiness, taking for instance social and economic aspects into account. Hence, there is potential to enhance and tailor existing development methodologies so that certain aspects of a holistic view on trustworthiness are taken into account.

The Trusted Software Methodology (TSM) [1, 4] is the only comprehensive approach that describes processes and guidance for engineering and assessing trustworthy software. It covers multiple quality attributes, and focuses on processes instead of evaluating development artifacts. TSM provides a set of Trust Principles, which describe established development practices or process characteristics that enhance software trustworthiness. A development process can be assessed by means of five different levels of trustworthiness, according to the conformance to the trust principles. Though the principles constitute general best practices, however, the methodology is assumed to be applied following a military standard for software development [4]. In contrast, our focus is on enhancing a broad spectrum of general software development methodologies in order to incorporate the consideration of trustworthiness and to evaluate the practiced process targeting trustworthiness. Yang et al. [33] developed a meta-model that includes trustworthy products dependent on a trustworthy process. But their aim is to provide mechanisms to evaluate the trustworthiness of produced artifact.

An exhaustive overview of development methodologies can be found in [12]. Though Jayaswal and Patton do not specify how these methodologies contribute to the trustworthiness of the product. It documents their generic characteristics

and an overview of the historical evolution of different development strategies and life-cycle models.

3 Constrained and Controlled Development Processes

In order to narrow the scope of discussion, we introduced general categories of trustworthy software development. Based on the distinction of introduced process categories, the well-established development methodologies with respect to their suitability to enable trustworthiness-by-design can be discussed further. We may describe three main categories of development as identified, namely:
Uncontrolled: The applications are developed without any special considerations of trustworthiness. This generic approach is very risky regarding the effort (and costs) required for reconstructing, measuring and documenting the elements of trustworthiness of the development process.
Controlled: Trustworthiness is considered, measured and managed along all phases of the development process. It does not necessarily mean that the developed application is trustworthy (or not) but only that trustworthiness has thoroughly been considered with a specific attention.
Constrained: The application is developed in a special way, possibly with a specific language, to assure that the design principles result in verifiable elements of trustworthiness so that specific trustworthiness properties can be (formally) demonstrated.
Although they may in principle all lead to trustworthy systems in the end, it is more likely that this will be the case using a constrained rather than an uncontrolled methodology. Nevertheless, with an uncontrolled methodology the goal would probably not be to create the worlds most trustworthy system, but something that is trustworthy enough for its purpose. Similarly, it may not be feasible to apply a constrained methodology to all development projects. The point is that in general we need to make informed decisions on which "trustworthiness-by-design" steps we choose and apply them based on the characteristics of the methodology at hand.

A number of standards and other activities that address security in the software development process fall into the controlled methodologies category. Security is, in that sense, primarily a set of quality requirements that need to be specified in the first place, and assured along the completely remaining development and operations activities. As such, addressing trustworthiness within the development life-cycle could very much benefit from the activities that are meant to address security since they in general en- compass practices for the assurance of specific (security) properties. There has been little attention, though, in considering the measurably of security-related properties. Some works, e.g., [22, 21], have target process enhancements that target security to build a 'secure' software system, describe their measurably capabilities and identify corresponding innovation potential, specifically towards extending security to trustworthiness.

We may consider two distinct applications designed to fulfil a particular task, one application developed in an ad-hoc manner, not following any kind of development methodology, and another one developed using a carefully studied

development process and extensively tested (as in [21]). Trustworthiness metrics should allow comparing both applications and give solid evidence as to which of both is "more trustworthy", disregarding the method used to develop each one respectively. We argue that a user should justifiably distrust the first one much more than he should distrust the second one. This trustworthiness is based on the idea that, although we cannot know how trustworthy each application really is beforehand, we do know that the second one could at least have followed a better development process, which in turn is known to help avoiding vulnerabilities. Thus, we can say that the trustworthiness of applications (taking development methodology itself as evidence) is dependent on the development process and methodology and it can make the user confident to some extent.

We will extend this approach in this paper by not taking a specific development practice into account, but rather concentrate on the metrics of trustworthiness properties as used in the comparative example above so that in principle any development process can be used in a controllable way. Thus, using trustworthiness metrics, we can render an arbitrary development process into a controlled one (whereas, of course, for improving the metric values, additional practices or capability patterns may be employed).

4 From Trustworthiness Qualities to Evidences and Certificates

Assessing a software development process to identify its intrinsic qualities generally is a non-trivial task. An assessment methodology must take into account multiple aspects and factors at the same time, from human interactions to the artefacts produced. Therefore, analysing a software development process requires to answer to two important questions: ***a)*** which aspects must be considered and ***b)*** with which criteria (e.g., metrics) such aspects must be analysed. Starting from the complexity of the problem, established and recognised software measurement approaches and among them notably GQM [30] propose to address both questions at the same time, in order to produce meaningful analysis. Synthesising roughly, the GQM approach is composed by two processes: a top-down refinement of Goals into Questions and then into Metrics, and a bottom-up interpretation of the collected data. The definition of metrics is particularly important: choosing scientifically sound and acknowledged metrics permits to obtain results that are easier to understand and compare with other similar initiatives. In particular, when these conditions are met, the results of the measurement process become objective elements whose interpretation in the bottom-up phase of GQM may lead to meaningful interpretations. Moreover, such objective elements demonstrate the soundness of claims made on the qualities of a software process; when this happens, these elements become *evidences* i.e., elements able to provide assurance about a specific quality. This concept is used in certification, where it is the cornerstone for supporting certificate claims.

Trustworthiness Metrics for Controlled Software Development. Software metrics have a long tradition in software quality, first practical applications date back to the 1980s (e.g. [8]). There are basically two approaches dealing with

metrics: *bottom-up* (use what you can measure) and *top-down* (try to measure what you think is valuable). One probably can say that the more mature a software development area is, the better the two converge. With respect to security and trust, unfortunately, there are a number of bottom-up metrics, that often do not match corresponding bottom-down approaches. A typical example is the area of hacker-proofness: whereas one would like to measure observations that express the security against hacks, most available bottom-up metrics give information about attack vectors, attack surfaces, successful attacks etc.

Therefore we decided to follow a top-down approach as proposed by [30] with the GQM methodology to develop metrics. Since subsequently we want to meaningfully aggregate metric values, a number of additional requirements are necessary.

First of all, we assume that all values will be percentage numbers, or alternatively numbers between 0 and 1, similarly to probabilities or assurance factors. The higher the number, the more the aspect/attribute in question should contribute tot he trustworthiness. This way, we will be able to compare values and to perform a number of useful computations. This is a design principle for all top-down metrics, but also needs to apply for all bottom-up metrics that may be considered later on. If we want to use different qualitative levels like "low", "medium" and "high", then there must be a mapping of the qualitative levels to percentage values (reference value = 100, real value = 0, 25, 50, 75 or 100 say for a 4-level qualitative metric (which we think should be avoided as far as possible).

The aggregation of metrics to attribute values is done by computing a weighted sum of the metrics that are considered to belong the the attribute. The weight may be interpreted as an importance of the aspect measured in view of the trustworthiness attribute in question, for example when measuring the percentage of function calls that are protected by authorization checks, how much this measurement should be considered to contribute to the "security" attribute.
We have developed significant number of metrics (more than 100 available in a online tool[5]), that may serve as a basis for future use in the context of the controlled software development approach. The online tool on the metrics discusses how they could be used programmatically as well as for a theoretical discussion [20].

Evidences and Assertions. An *Evidence* can be expressed as a proof of a claim that a specific quality can be associated to an *Asset* (i.e. a software or one of its parts). For instance, evidences can be derived from the calculation of specific metrics like software metrics computable on a software artefact (e.g. lines of code, McCabes cyclomatic complexity or others [7]), process or behavioural metrics. However, this definition can also comprise evidences coming from existing certification schemes: in Common Criteria, for example, evidences that evaluators gather are defined as "anything" that can prove the compliance with a mandatory CC requirement or the respect of a criterion: they generally con-

[5] http://optet.atc.gr/metrictoolwiki/en/StartingPoints

sist of documents, interviews as well as statements made by evaluators on their assessment.

Digital Trustworthiness Certificates. Certificates are means to cater assurance to third-parties that trust the certification body. However, certificates are not always descriptive [31], i.e., they not disclose full information about collected evidences and their collection process. The Digital Trustworthiness Certificate concept (DTWC) [6] permits to describe precisely evidences and their interpretation to support claims on trustworthiness qualities of a software. Such evidences may be product or process metrics, anyway compliant with the definition previously proposed.

The DTWC is a digital artefact that is machine-readable and -understandable, in contrast with traditional certificates that are documents often written in legal terms. It aims at representing claims on the trustworthiness qualities of a software, where trustworthiness is defined as "the objective performance evaluation of the relevant set of quality attributes, based on the evidence from observed system behaviour" [23]. It is based on a Linked Data vocabulary and it is composed by four main parts: system description, trustworthiness problem definition, trustworthiness property specification and evidences (Fig. 1). The system description is scalable in granularity (from methods, libraries and modules up to the whole architecture) and allows for including references to other DTWCs in order to depict exhaustively a software system; the problem definition captures the threats for software qualities that have been considering for a software during its design, development process and runtime execution, similarly to what ISO 27000 proposes for threat modelling. The property specification expresses claims about the qualities of the software (or its specific parts), while the evidence section supports such claims with objective elements. The DTWC is descriptive as it permits a full disclosure of the elements of confidence for a software product or its development process. More details on DTWC can be found at [6]. It

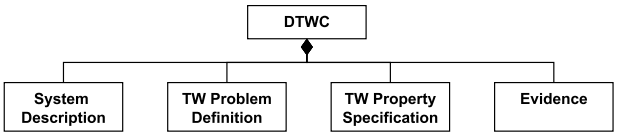

Fig. 1. A schematic representation of DTWC main components

is possible, therefore, to create a conceptual link among the qualities that are observable in controlled development methodologies and the possibility to represent them in details through descriptive certificates and in particular through DTWC. Metrics identification and measures are captured as DTWC evidences, that have an explicit link to claims on software/process qualities (the trustworthiness properties) associated to a software or its parts (assets).

5 Use Cases for Lightweight Certificate based on Trustworthiness Metrics

It is possible to use DTWC, exploiting their machine-intelligibility, in different use cases: be them automatic or human-oriented. For the former case, one can think to DTWC as meta-data providers in software/service discovery or for Linked Data operations. The latter case requires to present DTWC content in a manner that is: ***i)*** easily recognizable (ideally visually), ***ii)*** simple enough to address properties or objectives that can be grasped intuitively, ***iii)*** allows a drill down to look up detailed information that specifies that the certificate actually means in detail and ***iv)*** that is used in an intelligent way to support users in selecting applications based on their requirements.

The characteristics of DTWC, as seen in Section 4, permits fulfiling all the mentioned requirements. The hierarchical structure of DTWC and its system description permits to describe a software system and its components, their claims and evidences at different granularities, so that for example different software can be analysed selecting a comparable and objective basis. It also permits a hierarchical visualization of the certificate with "zoom-in" to lower levels of detail about the evidences.
The "zoom-in" functionality is helping in the following manner: users can easily perceive the trustworthiness property or objective that is "guaranteed" by the certificate, and optionally, e.g. for experts, to dive into the more detailed description levels. Manufacturers can target one specific, or, if necessary, a set of attributes and/or objectives to get certified. Developers can stay with their own development model, they may need to provide some transparency on their practices to improve measurement results over time if demanded by the market. Finally, regulators can observe the market evolution, and mandate specific properties if considered necessary, without changing the technical means.

A key success factor is the technical separation of the visualization of the certificate on one side, and of the information complexity that is contained in a certificate. must be different levels of information related to the certified property/objective in question. We currently envisage three levels of information: ***i)*** a visual component that can be displayed using graphical means (like the "TV sticker"), ***ii)*** a high level descriptive format that uses the metric(s) in question, showing the names and values of the metric and ***iii)*** a low level descriptive format, that explains the computation, meaning and possible interpretations of the metric in view of the property/properties in question. This information is NOT specific to a software component, and could be retrieved via URI(s) from e.g., the metrics online service.

6 Conclusion and Future Work

In This paper, a new way of demonstrating trustworthiness of software and Internet-based services is described based on individual evidences, either quantitatively using trustworthiness metrics or other types of qualitative evidences. We have applied this research in limited use cases and we can demonstrate their

usefulness. The major work that still needs to be done is to perform feasibility studies by applying trustworthiness metrics to larger real life examples and correspondingly improving the metrics over time.

Furthermore, an evaluation and certification process shall be developed to actually implement the independent validation of the certificates. To support an open market development, that process shall be as open as possible. Therefore, the concept of the certification process should allow different approaches in the following dimensions:
- Using different certification authorities (these being, in the language of certification processes according to ISO 17021, the certification bodies, as well as the owners of the technical "CA" in PKI terms), allowing from self-signed certificates, industrial certification bodies up to nation-level certification authorities/-bodies if deemed necessary.
- Using different evaluation laboratories, that perform the actual assessment / verification of the metrics values, allowing from the manufacturers' own quality assurance department, industrially driven standards organizations up to specialized accredited evaluation labs.
- Using different scopes and context conditions, as well as choice of metrics (and so evidences), allowing to adapt to different business and maybe also consumer scenarios, in different verticals with different success factors and requirements for trustworthy software or services.

One important question that needs to be specified in a first application of this scenario is the reliability of the overall process. In a first implementation, we would recommend to stick to ISO 17021 and to apply software quality measurement techniques as described in ISO 25021ff., thereby hoping to benefit from the existing infrastructure for accreditation and certification that has been established and has proven to be successful in the market, e.g. for quality management systems, information security management systems, and even Common Criteria certification schemes.

Some success factors still apply, and therefore the results of this analysis rely on a number of assumptions. Further work is needed to define a common taxonomy (specifically, relating properties, attributes, the high-level content of certificates, the description and interpretation of metrics, and so on). To support the adoption of this approach, there is a need to come up with simple scenarios and implementations of the certification process and even the certificates. Correspondingly, further work is needed to specify templates for certificates for some exemplary scenarios that are easy to grasp (e.g. ''hacker-proof" by providing a 100% metric value on the input validation and output sanitization of all interfaces). It is furthermore important to avoid different interpretations of the different information detail ''levels" within a certificate between different stakeholders and/or verticals or more generically groups. Therefore, in contrast to the flexibility of the model in terms of objectives measured, or certification reliability, the different ''levels of taxonomy" (we currently suggest three levels, as described above) must be fixed from the start.

Acknowledgements. This work is supported by the EU-funded project OPTET (grant no. 317631).

References

1. E. Amoroso, C. Taylor, J. Watson, and J. Weiss. A process-oriented methodology for assessing and improving software trustworthiness. In *Proceedings of the 2Nd ACM Conference on Computer and Communications Security*, CCS '94, pages 39–50, New York, NY, USA, 1994. ACM.
2. BSIMM-V. The Building Security In Maturity Model. `http://www.bsimm.com/`.
3. B. Chess and B. Arkin. Software security in practice. *Security Privacy, IEEE*, 9(2):89–92, March 2011.
4. G. Chisholm, J. Gannon, R. Kemmerer, and J. McHugh. Peer review of the trusted software methodology. Technical report, Argonne National Lab., IL (United States), Feb 1994.
5. F. Di Cerbo, M. Bezzi, S. P. Kaluvuri, A. Sabetta, S. Trabelsi, and V. Lotz. Towards a trustworthy service marketplace for the future internet. In *The Future Internet*, pages 105–116. Springer, 2012.
6. F. Di Cerbo, S. P. Kaluvuri, F. Motte, B. Nasser, W. Chen, and S. Short. Towards a linked data vocabulary for the certification of software properties. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2014 International Conference on*, pages 721–727. IEEE, 2014.
7. N. E. Fenton and S. L. Pfleeger. *Software metrics: a rigorous and practical approach*. PWS Publishing Co., 1998.
8. R. B. Grady and D. L. Caswell. *Software metrics: establishing a company-wide program*. Prentice Hall, 1987.
9. International Organization for Standardization. ISO/IEC 15408-1:2009 - Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model (SSE-CMM). `http://www.iso.org`.
10. International Organization for Standardization. ISO/IEC 21827 - Information technology – Security techniques – Systems Security Engineering – Capability Maturity Model (SSE-CMM). `http://www.iso.org`.
11. International Organization for Standardization. Iso/iec 27001:2013- information technology – security techniques – information security management systems – requirements. `http://www.iso.org`.
12. B. K. Jayaswal and P. C. Patton. *Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software*. Pearson Education, 2006.
13. A. Josey. TOGAF Version 9.1 Enterprise Edition. An Introduction. Technical report, The Open Group, 2011.
14. S. Lipner. The trustworthy computing security development lifecycle. In *Proceedings of the 20th Annual Computer Security Applications Conference*, ACSAC '04, pages 2–13, Washington, DC, USA, 2004. IEEE Computer Society.
15. V. Lotz, S. P. Kaluvuri, F. Di Cerbo, and A. Sabetta. Towards security certification schemas for the internet of services. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–5. IEEE, 2012.
16. G. McGraw. *Software Security: Building Security In*. Addison-Wesley Professional, 2006.
17. P. Meland, S. Ardi, J. Jensen, E. Rios, T. Sanchez, N. Shahmehri, and I. Tondel. An architectural foundation for security model sharing and reuse. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 823–828, March 2009.
18. Microsoft. Security Development Lifecycle. `http://www.microsoft.com/security/sdl/default.aspx`.

19. N. G. Mohammadi, T. Bandyszak, S. Paulus, P. H. Meland, T. Weyer, and K. Pohl. Extending development methodologies with trustworthiness-by-design for socio-technical systems - (extended abstract). In *Trust and Trustworthy Computing - 7th International Conference, TRUST 2014, Heraklion, Crete, Greece, June 30 - July 2, 2014. Proceedings*, pages 206–207, 2014.
20. N. G. Mohammadi, S. Paulus, M. Bishr, A. Metzger, H. Knnecke, S. Hartenstein, T. Weyer, and K. Pohl. Trustworthiness attributes and metrics for engineering trusted internet-based software systems. In *Cloud Computing and Services Science - Third International Conference, CLOSER 2013, Aachen, Germany, May 8-10, 2013, Revised Selected Papers*, pages 19–35, 2013.
21. A. A. Neto and M. Vieira. Untrustworthiness: A trust-based security metric. In *Risks and Security of Internet and Systems (CRiSIS), 2009 Fourth International Conference on*, pages 123–126. IEEE, 2009.
22. Open Web Application Security Project (OWASP). CLASP Project (Comprehensive, Light-weight Application Security Process). `https://www.owasp.org/index.php/Category:OWASP_CLASP_Project`.
23. OPTET Consortium. Initial concepts and abstractions to model trustworthiness. Project Deliverable D3.1, OPTET Consortium, 2013. available on http://www.optet.eu.
24. S. Paulus, N. G. Mohammadi, and T. Weyer. Trustworthy software development. In *Communications and Multimedia Security - 14th IFIP TC 6/TC 11 International Conference, CMS 2013, Magdeburg, Germany, September 25-26, 2013. Proceedings*, pages 233–247, 2013.
25. B. Potter. Microsoft sdl threat modelling tool. *Network Security*, 2009(1):15–18, 2009.
26. D. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, Feb 2006.
27. SHIELDS. Detecting known security vulnerabilities from within design and development tools. `http://www.shields-project.eu/`.
28. I. Sommerville. *Software Engineering*. Addison-Wesley, Harlow, England, 9 edition, 2010.
29. A. Sutcliffe. Convergence or competition between software engineering and human computer interaction. In A. Seffah, J. Gulliksen, and M. Desmarais, editors, *Human-Centered Software Engineering Integrating Usability in the Software Development Lifecycle*, volume 8 of *Human-Computer Interaction Series*, pages 71–84. Springer Netherlands, 2005.
30. R. Van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. Goal question metric (gqm) approach. *Encyclopedia of Software Engineering*, 2002.
31. K. Wallnau. *Software component certification: 10 useful distinctions*. Technical note. Carnegie Mellon University, Software Engineering Institute, 2004.
32. T. Weigert. Practical experiences in using model-driven engineering to develop trustworthy computing systems. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2006), 5-7 June 2006, Taichung, Taiwan*, pages 208–217, 2006.
33. Y. Yang, Q. Wang, and M. Li. Process trustworthiness as a capability indicator for measuring and improving software trustworthiness. In Q. Wang, V. Garousi, R. Madachy, and D. Pfahl, editors, *Trustworthy Software Development Processes*, volume 5543 of *Lecture Notes in Computer Science*, pages 389–401. Springer Berlin Heidelberg, 2009.