NIST Interagency Report ####

National Institute of
Standards and Technology
U.S. Department of Commerce

Toward a Preliminary Framework for Assessing the Trustworthiness of Software

Tim Boland
Charline Cleraux
Elizabeth Fong

Software and Systems Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8970

September 2010

U.S. Department of Commerce
Gary Locke, Secretary

National Institute of Standards and Technology
Dr. Patrick D. Gallagher, Director

Abstract

Is trustworthiness of software measurable? The determination of trustworthiness of software is difficult. There may be different quantifiable representations of trustworthiness. This paper proposes a preliminary framework for assessing the trustworthiness of software. Such a trustworthy quantification framework will have characteristics of software systems that relate to or support trustworthiness, and seek to identify and improve metrics and measurement methods (i.e., the metrology) that enable developers and users to analyze, evaluate and assure trustworthiness in software systems and applications.

The approach currently taken involves development of a framework composed of models, with the ultimate goal being the ability to calculate a trustworthy factor for software. An example is supplied in this paper to "test out" this framework.

Keywords: Framework, measures and metrics, software assurance, trustworthy software.

Acknowledgement

The authors wish to thank Thomas Rhodes and Michael Kass, System and Software Division, NIST, for their technical contributions to this work. We also thank Dr. Paul E. Black, System and Software Division, NIST, for providing us with his PhD thesis as the example in this paper. Dr. Jeff Voas, Computer Security Division, NIST, assisted us in improving our understanding of trustworthy software factors.

Table of Contents

Toward a Preliminary Framework for Assessing the Trustworthiness of Software

Tim Boland
Charline Cleraux
Elizabeth Fong
{boland,charline.cleraux,efong@nist.gov}

1. Introduction

Is trustworthiness of software measurable? The determination of trustworthiness of software is difficult. There may be different quantifiable representations of trustworthiness. This paper proposes a preliminary framework for assessing the trustworthiness of software. Such a trustworthy quantification framework will have characteristics of software systems that relate to or support trustworthiness, and seek to identify and improve metrics and measurement methods (i.e., the metrology) that enable developers to analyze, evaluate and assure trustworthiness in software systems and applications.

The approach currently taken involves development of a framework composed of models, with the ultimate goal being the ability to calculate a trustworthy factor for software. A case study consisted of an example is supplied in this paper to "test out" this framework.

1.1 Background

This research effort is part of the National Institute of Standards and Technology (NIST) Informational Technology laboratory (ITL) under the Trustworthy Information Systems (TIS) program areas. The aim of the TIS program is to reduce the risk and uncertainty associated with information systems by improving the capability of design, build and assess trustworthy systems.

Ensuring that our nation's information systems are trustworthy is becoming increasingly important as we become more dependent on them for reliable, secure, and safe operation in nearly all sectors of our economy, national defense, homeland security, healthcare, and personal life. As systems grow in size and complexity, and become increasingly interconnected through networks and communication links, their vulnerability to attack from hostile elements, or failures due to inherent defects or exploited vulnerability, increase their risk of failure or compromise with significant impacts to businesses, services, equipment or users depending on them.

1.2 Purpose

The purpose of this project is to investigate the measurement and modeling the trustworthiness of software systems. The approach would be:

- to gather useful and objective information about the trustworthiness of software,
- to attempt to assess the trustworthiness of software, either in absolute terms or as change indication, in terms of numerical scoring.

It is important to be able to quantify trustworthiness of software in situ. Several researchers, for example, Stringini [STRINGINI], Taibi [TAIBI], Larson et. al [LARSON], Tan et. al [TAN]) have attempted to do so, but results so far are of limited scope. Other researchers Pfleeger [PALEEGER], Yang [YANG] seek to analyze/predict aspects of trustworthiness during software development. A more expansive model may be needed to quantify trustworthiness of software as a "product".

There may be different quantifiable representations of trustworthiness. One such representation (but not necessarily the only representation) is described in this paper.

1.3 Scope

Measurement and modeling of any software attribute that is considered an essential requirement of the software systems, including safety, security, dependability, quality, performance (and others) are within the scope of this project.

Measurement and modeling trustworthiness of hardware, process used to create software, and people involved in the development of software are beyond the scope of this project.

1.4 Audiences

Beneficiaries of this approach would be software suppliers, acquirers, developers, testing practitioners, and users and software managers, among others. If successful, the approach should make assessing the qualities of software by providing more detailed and objective information about the trustworthiness of software in advance of and during its use in an operational environment.

1.5 Assumptions

Trustworthiness is assumed to be measureable (quantifiable). The general premise for a trustworthiness framework is that it can be composed of many specific attributes of trustworthiness sub-models [VOAS]. These sub-models may or may not be structured in a hierarchy. In other words, trustworthiness is assumed to be decomposable into attributes that are related to trustworthiness in some way. Examples of such attributes are safety, reliability, security, correctness, usability, and possibly others. These attributes are assumed to be independent of one another (subject to caveats discussed following).

It is necessary to assume that trustworthiness is largely determined by operational context. It is also assumed that the quantifiable trustworthy framework is expressible in a structured assurance case model [KELLY].

The structure assurance case to represent the trustworthiness of software must be constructed based on sound logical principles, and is largely determined by operational context. It is assumed that trustworthy factor is product-based (not process-based). It is assumed that arguments given are primarily inductive (not deductive).

1.6 Glossary

Software Assurance – is the planned and systematic set of activities that ensures that software processes and products conform to requirements, standards and procedures [NASA]

Trustworthiness – a system that performs as intended for a specific purpose, when needed, with operational resiliency, and without unwanted side-effects, behaviors, or exploitable vulnerabilities [CNSS]

Claim – statements asserting some characteristic, property, or behavior of software that can be evaluated for truthfulness, is demonstrable, and is supported by arguments based on objective evidence [OMG-ARM]

Argument – logical proposition intended to support a claim through reasoning or logic that links evidence to the claim [OMG-ARM]

Evidence – information used to support a claim [OMG-ARM]

Risk – exposure to the chance of injury or loss (source: dictionary.com)

Framework – a structure composed of parts fitted or joined together (source: dictionary.com)

Model – a simplified representation of a system or phenomenon (source: dictionary.com)

1.7 Outline and Context of This Paper

After some essential background material, this paper presents the components of the trustworthy framework in detail. Then, an example is presented to “test out” this framework. Finally, a summary, conclusion, and references are presented. In the course of development of the trustworthy framework, there are many issues and options.

Appendix A lists some issues/questions forming the basis for further research. These issues are categorized topically (and ordered within each topic according to when they arose in the preparation of this manuscript). They are identified using “shorthand” notation (example: “Doc1” for Documentation Issue 1). They are referenced in this document (using subtopic and then chronological order within that subtopic for easier document maintenance) using “NOTE:” and then a reference to the issue identifications as mentioned above.

Appendix B describes an example to illustrate the application of the trustworthy factor model, using a software program to demonstrate “proof of concept”. This example is large enough to be interesting but small enough to be manageable.

2.0 Trustworthy Software Framework

The trustworthy software framework summarized herein relies on the tenets of the structured assurance case methodology. The ultimate output of this framework would be a trustworthy factor; such as factor could primarily be used to information future development of software to make it more trustworthy, and secondarily, to take software “in situ” and evaluate its trustworthiness.

NOTE: Please see Repr4, Repr5, and Repr6 in Appendix A.

2.1 Description of the Approach

A trustworthy software framework is considered to be composed of models at different “levels” in a hierarchy. The ultimate purpose of the framework would be to produce a trustworthy factor. The approach is based on a structured assurance case methodology as described in Section 2.2. The approach taken is a “bottom-up” approach, using evidence (possibly coming from software metrics) at the lowest “level”, and then using that evidence to support the argument that the attributes that are related to software trustworthiness.

NOTE: Please see Rel16 and Solv8 in Appendix A.

The approach taken allows the user/evaluator to include context and operating environment including other information into the framework to “customize” according to the trustworthiness requirements of that particular user/evaluator. It needs to be made explicit (documented, made public) in the framework exactly how the trustworthy factor was computed.

The determination of trustworthiness of software may be made by a large number of users/evaluators for that particular software, with each user/evaluator determining a trustworthy factor for the same or different operating environments of the software (for example). If a large percentage of such users/evaluators produce high trustworthy indices for that software, then an argument can be made that this piece of software is more trustworthy than if those users/evaluators all produce low trustworthy factors for that software.

It is noted that all decisions in the framework affecting the value of the trustworthy factor should be publicly documented. All relationships in the framework need to be explicitly documented to make resultant values meaningful. The ability of users/evaluators to communicate clearly and in a structured fashion to others the precise information going into the framework for their particular calculations should cut down on any "subjectivity" of model calculations in the framework.

All measurements may involve some uncertainty; to include uncertainty, an uncertainty term "beta" with differing values may be added to terms as appropriate; "beta" may indicate "plus or minus uncertainty" measured by value of beta.

NOTE: Please see Risk5 in Appendix A.

2.2 Structured Assurance Case Methodology

The structured assurance case methodology may be used to determine the assurance of systems. It has been used in the past with success in assuring safety, and is currently being investigated as to its applicability for software [NISTIR7608]. Such a model consists of the following items:

- claims (denoted following by "Cl" designation) ,
- subclaims (denoted following by "SCl" designation),
- arguments (denoted following by "Arg" designation), and
- evidence (denoted following by "Evid" designation),

These items are all related in a hierarchical fashion. Definitions are given in Section 1.6.

For this framework, claims are made to support satisfaction of a trustworthiness attribute. These claims may be decomposed into subclaims, each of which is designed to support the referenced claim. Evidence (denoted following by "Evid" designation) is used in support of the applicable claims/subclaims. An argument (denoted following by "Arg" designation) may consist of all the applicable evidence and subclaims taken together, along with rules of inference, to support a claim.

NOTE: Please see Rel1, Doc7, Solv3 and Model13 in Appendix A.

It is important to realize that there is a skill to properly expressing all of these items in order to make resultant values in framework meaningful.

NOTE: Please see Repr9, Doc2, and Repr6 in Appendix A.

3.0 Trustworthy Factor Model

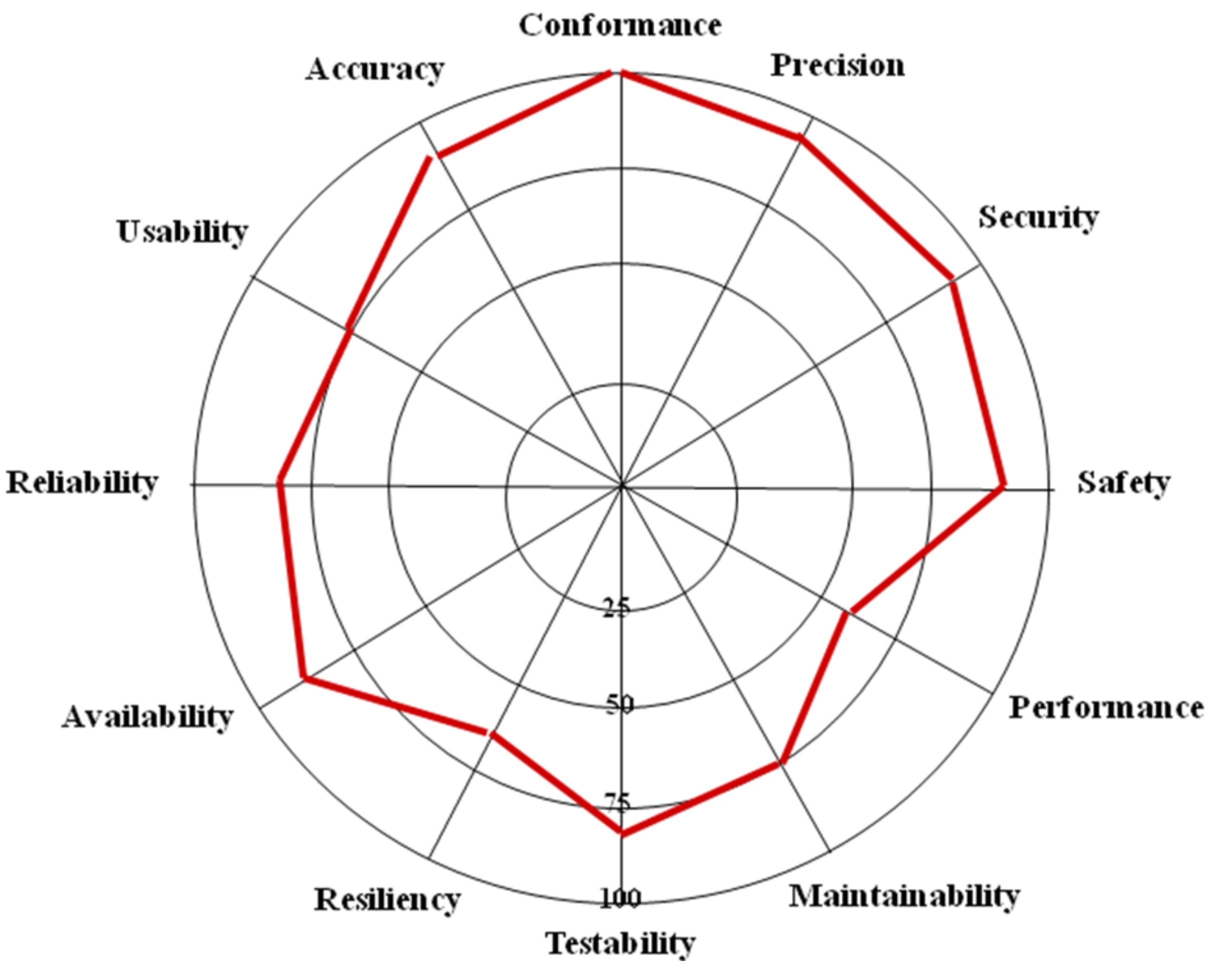
NOTE: Please see Doc1, Doc3 and Rel14 in Appendix A.

The trustworthy factor is expressed as follows:

$$TI = T_{actual} / T_{maximum},$$

With values of TI between “0”and “1” (“0” being totally untrustworthy, and “1” being completely trustworthy, so TI is “normalized”). T_{actual} is the actual measure of trustworthiness, and $T_{maximum,}$ is the potential maximum trustworthiness possible.

The figure, as represented by the Kiviant chart below, indicates a possible graphical illustrative depiction of the terms in the above equation.



A risk index can be expressed as follows:

Risk Index = (Risk) – 1, where Risk = 1 / TI, so that as TI approaches 0, risk index approaches infinity, and as TI approaches 1, risk index approaches 0. So risk index would be the "inverse" of trustworthiness index.

NOTE: Please see Rel6, Risk4, Risk6 and Repr6 in Appendix A.

The equation for TI would read:

T_{actual} = [(c1Att1)(c2Att2)(c3Att3)(..]* x [(c4Att4 + c5Att5 + c6Att6 + ..)]*

NOTE: Please see Rel2, Risk2, Repr3, and Model3 in Appendix A.

The character "*" is the Backus-Naur form notation for "zero or more occurrences". If 0 occurrences of one bracket only, value defaults to 1. If 0 occurrences of both brackets simultaneously, value defaults to 0.

NOTE: Please see Rel4 and Rel15 in Appendix A.

In this equation, the multiplicative terms (for example c1Att1, etc.) represent any (0 or more) critical (more important) trustworthy attributes (Att1 – Att6) with coefficients (c1 – c6). The additive terms (for example c4Att4, etc.) represent any (0 or more) noncritical (desirable – less important) attributes (they may in fact be the same attributes?).

NOTE: Please see Doc4, Doc5, and Model14 in Appendix A.

A possible criterion for criticality is: if attribute fails, is total trustworthiness 0? Users/evaluators could define their own attributes, and there may be overlap among the attributes. Each attribute (a "goal" from GQM [GQM] (for example, Att1) and each coefficient (for example, c1) has a value between "0" and "1". A coefficient's value may represent the importance of that attribute in the evaluator/user's definition of trustworthiness and for context/other information? The values of the coefficients would be computed by consensus around answering questions in validated questionnaires – there is a skill to expressing the questions properly – values of coefficients in multiplicative part cannot be zero, but value of coefficients in additive parts could be.

Some of the list items in the Adelard Safety Case documentation [ADELARD] might make excellent questions in this regard, for these and following coefficients/weights. The values for attributes Att1 would be determined as in Section 3.0.

NOTE: Please see Model8 in Appendix A.

Currently the maximum actual trustworthiness is 1. Do we want to "give extra credit" for "over-engineering" a particular attribute so that its value would be > 1? (Doing so may potentially make TI > 1? A larger question is: can one attribute compensate for another?)

NOTE: Please see Rel15 and Risk7 in Appendix A.

The equation for $T_{maximum,}$ would read:
$T_{maximum}$ = [(c1) (c2) (c3)] * [(c4 + c5 + ..) ...]*

In other words, Att1 = Att2 = Att3...= 1 (implicit in equation above).
It is assumed that $T_{maximum,}$ would have the same number of attributes as would $T_{actual.}$

If one wanted to "bound" TI one could make equation for T_{actual}:
T_{actual} = [(c1 Att1) (c2 Att2) (c3 Att3) (c4 Att4 / n + c5 Att5 / n + ...)] *

And for $T_{maximum,}$
$T_{maximum}$ = [(c1) (c2) (c3) (c4 / n + c5 / n + c6 / n + ...)] *

Where n is the total number of “desirable” attributes considered (the multiplicative terms are already bounded). So in this case:

$0 =< T_{actual} =< T_{maximum} =< 1.$

NOTE: Please see Solv1 and Rel10 in Appendix A.

If one wanted to put time dependencies (values are functions of time t) on the terms, one could say, for example,

T_{actual} = [(c1 (t) Att1 (t)) (c2 (t) Att2 (t)..) (c4 (t) Att4 (t) + c5 t) Att5 (t) + ..)] *

NOTE: Please see Rel7, Model11, Risk3, Repr7, Solv7 and Rel12 in Appendix A.

4.0 Trustworthy Attribute Equations

NOTE: Please see Rel8 and Model15 in Appendix A.

The equations for Att1, Att2, etc. are as follows:

Att1 = [(c7 Cl1) (c8 Cl2) (c9 Cl4)..]* / [(c7 Req1) (c8 Req2) (c9 Req4) ..]*
Att2 = [(c10 Cl1) (c11 Cl5) (c12 Cl6)..]* / [(c10 Req1) (c11 Req5) (c12 Req6)..]*

In the above * means 0 or more occurrences. If 0 occurrences of Cl’s, default value is 0. If 0 occurrences of Reqs, default value is 1. In above examples there is one claim per requirement so same number of items in numerator as in denominator, but it may be possible to have more than one claim per requirement.

NOTE: Please see Model1, Rel5 and Repr2 in Appendix A.

The Cl’s (Cl1, Cl2, etc.) are claims (or test assertions, or based on probability, for example) and the Req’s (Req1, Req2, etc.) are requirements to satisfy particular attributes Att1, Att2, etc. The coefficients c7, c8, etc. represent the importance or consequence pertaining to that requirement or claim in determining that attribute (Att1, for example); values of coefficients are determined as mentioned above for TI coefficients in Section 2.0. Values for claims Cl are determined as in Section 4.0. Values for requirements Req are always 1. An example of a claim might be “Code X does not have any buffer overflows”.

There is a skill to properly expressing requirements, but it is possible that requirements will be deleted and claims will just be used in this model.

NOTE: Please see Model10 and Model12 in Appendix A.

Requirements may or may not be “shared”. For example, in the equation above Att1 and Att2 equations share a common claim Cl1 and requirement Req1. This means that this requirement Req1 is common to both Att1 and Att2 (shared requirement). However, the usage of Req1 may be different in Att1 than in Att2, and this is reflected in the possibly differing values of c7 and c13. The other requirements are different in the two equations. Att1 and Att2 values are “normalized” to be between 0 and 1, independent of the number of claims and requirements (defined by the user/evaluator).

NOTE: Please see Repr1 and Rel5 in Appendix A.

One could also separate claims and requirements into “critical” and “desirable” similar to what was done for TI above, so in that case, one could have, for example,

Att2 = [(c10 Cl1) (c11 Cl5) (c11 Cl6)..]* / [(c10 Req1) (c11 Req5)(c12 Req6)..]* x
[(c13 Cl7 + c14 cl8 + c15 cl9 + ..)]* / [(c13 R17 + c14 R18 + c15 R19 + ..)]*

NOTE: Please see Model9 in Appendix A.

As an alternative to the equation for Att1 (for example), one could also set a threshold value $\mathrm{Att1}_{\mathrm{min}}$, and if Att1 >= $\mathrm{Att}_{\mathrm{min}}$, then Att1 = 1; if Att1 < $\mathrm{Att}_{\mathrm{min}}$, then Att1 = 0.

NOTE: Please see Rel9 and Solv6 in Appendix A.

5.0 Claims/Subclaims Equations

NOTE: Please see Solv2 in Appendix A.

It is assumed that each claim would either be satisfied or not satisfied (binary result). So Cl_i would be 0 (not satisfied) or 1(satisfied), depending on the values of one or more argument result terms Arg1Result, Arg2Result, etc. (explained following).

It may be possible to allow partial satisfaction of claims (perhaps using “fuzzy logic” or “Bayesian probability”?).

NOTE: Please see Model4, Model6, Solv5 and Rel15 in Appendix A.

Some examples of claims might be best practices/checklists/templates/bullets from Adelard Safety Case [ADELARD]. All claims should be at same level of granularity if possible.
So C11 = [(c1 x Arg1 Result) (c2 x Arg2 Result). .]*

In the above * means 0 or more occurrences. If 0 occurrences, default value is 0, but default value assigned needs further investigation? Also c1, c2, etc. are coefficients designed to measure the degree of relevance of the argument result to the referenced claim. These are included because it is possible to have a valid and/or sound argument result with little or no relevance to the associated claim.

An alternative approach might be to just make claims function of evidence strength and bypass arguments altogether.

Values of Arg1Result, Arg2Result, etc., are determined as described in Section 5.0.

NOTE: Please see Model4 in Appendix A.

Claims may be broken down into subclaims (with associated argument); if there is a claim Cl1, and subclaims SCl1-1, SCl1-2, SCl1-3, etc. of Cl1, then the equation for Cl1 is:
Cl1 = [(c3 x SCll-1) (c4 x SCl1-2) (c5 x SCl1-3)..] +

In the above + means “1 or more occurrences” in Backus-Naur form, so it is assumed that each claim will have at least one sub-claim – maybe have weights on the subclaims as well.

NOTE: Please see Solv4 in Appendix A.

In other words, if the SCl1’s have binary values (0-satisfied or 1-not satisfied), then all SCl1’s need to be 1 for Cl1 to be 1. In this case the value of each SCl1 would be determined by the exceeding of a threshold for each argument equation (discussed following) for each SCl1.., so that:

SC11-1 = [(c6 x Arg11-11Result) (c7 x Arg11-12Result)..]*
SC l1- 2= [(c8 x Arg11-21Result) (c9 x Arg11-22Result)..]*

etc., where values of Arg1-1Result, etc. would be determined as described in Section 5.0, and c6, etc. are coefficients of relevance. If thresholds are exceeded, values are 1; if not, values are 0.

In the above * means 0 or more occurrences. If 0 occurrences, default value is 0, but specific default value needs further investigation?

An open question is how many subclaims are enough – may need to add factor for “subclaim gap” – does satisfaction of all subclaims equal satisfaction of the resultant claim (perfect decomposition? – are subclaims all independent of one another?

NOTE: Please see Rel17 in Appendix A.

6.0 Argument Result Equations

NOTE: Please see Rel11 and Model16 in Appendix A.

The value of the argument Arg1Result for claim Cl1 is determined by:

ArgResult1 = [(w1 ES1-1 + w2 ES1-2 + w 3ES1-3 + ..)]* / [w1+ w2 + w3 + ..]*

In the above * means 0 or more occurrences. If 0 occurrences, default value is 0 in numerator, 1 in denominator, but default value needs further investigation? There would be the same number of items in the numerator as in the denominator.

In the above equation "w1, w2, etc." are weights (with values from 0 to 1) representing relative importance/significance/consequence of evidence strength in argument result calculation (for example, considering "inductive gap" or "contextualization of evidence" from [KELLY]). Weights are computed by responses to questions in validated questionnaires. In other words, in the denominator, all ES's are assumed to be 1.

NOTE: Please see Repr10 in Appendix A.

ES's are evidence strength (see Section 6.0); values of ES are determined as described in Section 6.0. So ArgResult1 is normalized between 0 and 1.

In the equation for Arg1Result the higher the values of ES1-1, etc., the higher the value of Arg1Result.

To determine the value of Arg1Result, the evaluator/user would set a valid argument threshold value Arg1limit (determined from consensus around answers to questions in validated questionnaires – maybe "as confident as reasonably practical"? – considering "assurance deficit" issues?).

NOTE: Please see Rel6 in Appendix A.

If Arg1Result >= Arg1Limit, then Arg1Result = 1 in the equation for Cl1 above;

conversely if Arg1Result < <Arg1Limit, then Arg1Result = 0 in the equation for Cl1 above.

If one wanted to allow partial satisfaction of claims, then we wouldn't use the argument threshold approach and just allow Arg1Result to be a fraction between 0 and 1 in the equations for claim values.

NOTE: Please see Repr8 in Appendix A.

7.0 Evidence Equations

The ES's represent evidence strength indicators for each piece of evidence used to support the claim. Evidence must be measured in some fashion, and ES's attempt to do this. Values of ES's may be between 0 and 1. The ES's point to (are explicitly associated with) the actual evidence. The equation for evidence strength score ES1-1 (normalized from 0 to 1) would be:

ES1-1 = [(wght1 S1 + wght2 S2 + wght3 S3 + wght4 S4 + wght5 S5 +..)]* / [(wght1 + wght2 + wght3 + wght4 + wght5 + ..)]*

NOTE: Please see Model2, Rel3, and Risk1 in Appendix A.

In the above * means 0 or more occurrences. If 0 occurrences in numerator, default value of numerator is 0, and if 0 occurrences in denominator, default value of denominator is 1, but default values assigned need further investigation? It is assumed that there is same number of items in numerator as in denominator.

NOTE: Please see Model5 in Appendix A.

In above equation S1, S2, S3, S4, S5,… are actual evidence strength factors (with values from 0 to 1) related to "axes" of evidence evaluation explained as follows. Examples of evidence item might be "test result for code lines x to y", or "this code has failed 9 out of 10 times in the past (probability-based)" or "fault tree analysis". An evidence item must be reproducible (related to S1), objectively measurable (if possible) (related to S2), relevant to its claim/degree of support for claim (related to S3), not subject to compromise/tampering (related to S4), and accurate/precise/minimal uncertainty or error (related to S5).

NOTE: Please see Rel13 in Appendix A.

Other axes may be added by the user/evaluator as needed (for example, one for "validity" if not included in S5 - independence from other evidence, role of humans, consideration of assumptions/scope/justification/consequence, visibility, other factors, etc.). Thus the numerator of the equation for ES1-1 is the actual evidence strength evaluated according to the factors above, and the denominator of the equation is the maximum possible evidence strength (assuming S1 = S2 =...= 1).

In the equation for ES1-1 wght1, wght2, etc. are the relative weights (values from 0 to 1) that may be assigned for the importance/other information related to each piece of evidence or evidence strength factor. There is exactly one piece of evidence associated with each evidence strength factor.

NOTE: Please see Rel8 in Appendix A.

There may also be counter-evidence items (evidence against a particular claim). If we want to subtract counter-evidence items in the evidence strength equation, then the modified evidence strength score ESm1-1 might be:

ESm1-1 = [(wght1 (S1p – S1n) + wght2 (S2p – S2n) + wght3 (S3p – S3n) + ..)]* / [(wght1 + wght2 + wght3 + wght4 + ..)]*

In above equation S1p, S2p, etc. are “positive” supportive evidence strength factors related to axes as mentioned above, and S1n, S2n, are “negative” non-supportive evidence strength factors related to axes as mentioned above (maybe considering “assurance deficit” issues?).

It is possible that previously-computed trustworthiness determinations may be fed back into the evidence model as evidence at a future time, for incorporation into the framework. Thus, the models in the framework may be run continuously, with previous determinations of trustworthiness serving as input to future determinations of trustworthiness.

NOTE: Please see Model11 and Model17 in Appendix A.

8.0 Example and Lessons Learned

Appendix B illustrates an example to assess that the trustworthy factor model described here is feasible. The example used a relatively small web server program written in C with 150 line of code.

NOTE: Please see Model7 and Doc6 in Appendix A.

Some items learned so far from doing the example indicates:

- The experiments can get very complicated very quickly, so the scalability issue needs to be addressed.
- The bottom-up approach taken in this example may present special challenges to argument construction.
- The structured arguments comprise argument elements that are being asserted by the author of the argument. The evaluation and acceptance of an argument by a separate party may not be the same.
- Investigators may have different perspectives coming in, and it is necessary to coordinate/resolve those perspectives early on.

9.0 Summary and Conclusions

An approach has been given which attempts to provide some quantification of software trustworthiness. Such an approach seems consistent with some earlier and current approaches (for example, [VOAS] and [MYERS]). Context is built in to this approach, and the model parameterization is fully and specifically documented, and able to be communicated. This approach shows some promise when validated against the case study data described herein.

There are still issues/questions requiring further exploration. Some of these are listed in Appendix A.

10.0 References

[ADELARD] Adelard LLP Resources, Robin Bloomfield, "Safety Cases for PES" 2002, available at http://adelard.com/web/hnav/resources/iee_pn/index.html

[BLACK] Black, P. E., Axiomatic Semantics Verification of a Secure Web Server", Ph.D. dissertation, Brigham Young University, Utah, USA (February 1998).

[CNSS] Committee on National Security Systems (CNSS) "National Information Assurance (IA) Glossary", Instruction 4009, Revised June 2006. Available at http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf

[GQM] Basili, V. R., Caldiera, G., Rombach, H. ., "The Goal Question Metric Approach", Computer Science Technical Report Series, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992. http://www.cs.toronto.edu/~sme/CSC444F/handouts/GQM-paper.pdf

[KELLY] Kelly, T. "An Introduction to Structured Argumentation Within Assurance Cases", presentation, August 25, 2009

[MYERS] Myers, A. "Securely Taking on New Executable Stuff of Uncertain Provenance (STONESOUP), IARPA Project at Cornell University, http://www.cs.cornell.edu/andru/stonesoup/

[LARSON]: Larson, D. and Miller, K. "Silver Bullets for Little Monsters: Making Software More Trustworthy", IT Professional, April 2005

[NASA] Nasa-GB-A201, "NASA Software Guidebook," available at http://sate.gsfc.nasa/assure/agb.txt

[NIST] Rhodes, T., Boland F., Fong, E., Kass, M., "Software Assurance Using Structured Case Models," Journal of Research of the National Institute of Standards and Technology Volume 115, Number 3, May-June 2010.

[OMG-ARM] OMG Systems Assurance Taskforce, "Systems Assurance Task Force: Argument Metamodel (ARM)", Version 1, OMG Document Number Sysa/10-03-15. available at http://www.omg.org/cgi-bin/doc?sysa2010-3-15.

[PFLEEGER] Pfleeger, S., "Measuring Software Reliability", IEEE Spectrum, August 1992, pp. 56-60.

[SOAR] Goertzel, K. M. Winograd, T., McKinley, H. L., Oh, L., Colon, M., McGibbon, T., Fedchak, E., Vienneau, R., "Software Security Assurance: State-of-the-Art Report (SOAR) July 31, 2007

[STRIGINI] Strigini, L., "Thoughts for Panel on Measuring Trustworthiness," March 2009

[TAIBI] Taibi, "Ph.D. Program – Defining an Open Source Software Trustworthiness Model", 5 October 2007

[TAN] Tan, T., He., M., Yang, Y., Wang, Q., Li., M., "An Analysis to Understand Software Trustworthiness", 9th International Conference for Young Computer Scientists, 2008

[YANG] Yang, Ye: "Process Trustworthiness as a Capability Indicator for Measuring and Improving Software Trustworthiness", Lecture Notes in Computer Science, 2009

[VOAS] Voas, J. "Trusted Software's Holy Grail", Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), 2002

Appendix A – Issues/Questions

Documentation

- Issue Doc1: give variables different names with some semantic meaning
- Issue Doc2: give more explanation and explicit relationships in framework details
- Issue Doc3: need documentation or equations of the relationships between elements in the hierarchy
- Issue Doc4: communicate definitions of the important terms in model (context, trustworthiness…)
- Issue Doc5: verify the consistency of used terminology in the paper
- Issue Doc6: add riche examples to show exactly what is expecting
- Issue Doc7: express properly the requirements, claims, and assertions of the case study

Relationship between arguments

- Issue Rel1: relate to structured assurance case methodology
- Issue Rel2: introduce and explain coefficients in equations of the different levels
- Issue Rel3: use direct evidence instead of evidence strength
- Issue Rel4: use union or intersection of groups to make other relationships easily
- Issue Rel5: define dependencies between attributes
- Issue Rel6: compare trustworthiness with risk which is 1/T
- Issue Rel7: study the time dependency and the longevity of trustworthiness calculations
- Issue Rel8: include uncertainty calculations
- Issue Rel9: define thresholds for trustworthiness contribution of each attribute
- Issue Rel10: normalize the equations for argument not bounded
- Issue Rel11: include probability considerations
- Issue Rel12: compare the measure of change in trustworthiness with trustworthiness as an absolute quantity
- Issue Rel13: chose the order of calculations in models: as in the hierarchy or else
- Issue Rel14: express properly the requirements, claims, and assertions of the case study
- Issue Rel15: chose to use * or + to express number of occurrences and its default value
- Issue Rel16: think about making simpler the hierarchy and get rid of certain elements
- Issue Rel17: define the right number of subclaims, and their relationships

Modeling

- Issue Model1: take care of the overlap of requirements or conditions
- Issue Model2: define how evaluate and compute the weights depending on the context
- Issue Model3: compute the cost of trustworthiness
- Issue Model4: give a quality or level guidance for claims and requirements
- Issue Model5: use metrics in model
- Issue Model6: measure degree of support or validity of claim
- Issue Model7: give a case study to verify the credibility of the framework
- Issue Model8: include an Adelard safety case in model
- Issue Model9: verify the application of trustworthiness to an attribute
- Issue Model10: manage complexity and implementation
- Issue Model11: think about circularity in models
- Issue Model12: think about the necessity of having requirements
- Issue Model13: make the framework arbitrary if chosen
- Issue Model14: determine the criticality of attributes
- Issue Model15: take care about the over-engineering of attributes
- Issue Model16: considerate argument patterns
- Issue Model17: considerate the circularity in the decomposition

Representation

- Issue Repr1: considerate the relation to different representations with the model
- Issue Repr2: define dependencies between attributes
- Issue Repr3: chose quantifiable representation like stable/unstable, linear/nonlinear
- Issue Repr4: think about framework for models versus models themselves
- Issue Repr5: representation of feasibility of implementation
- Issue Repr6: make the framework arbitrary if chosen
- Issue Repr7: compare/represent the measure of change in trustworthiness with trustworthiness as an absolute quantity
- Issue Repr8: considerate argument patterns
- Issue Repr9: determine to create a model or a simulation
- Issue Repr10: think about the most adapted programming language implementation

Risk – other complements

- Issue Risk1: include threat attack methodology and context
- Issue Risk2: compare trustworthiness with risk which is 1/T

- Issue Risk3: study the time dependency and the longevity of trustworthiness calculations
- Issue Risk4: compare about managing risk and managing trustworthiness
- Issue Risk5: include uncertainty calculations
- Issue Risk6: include risk analysis and measurement
- Issue Risk7: take care about the over-engineering of attributes

Solving

- Issue Solv1: bound the Trustworthy factor (TI) by 1 or use different scoring metrics
- Issue Solv2: give a quality or level guidance for claims and requirements
- Issue Solv3: use metrics in model
- Issue Solv4: allow partial satisfaction of claims and subclaims
- Issue Solv5: measure degree of support or validity of claim
- Issue Solv6: compute threshold values for trustworthiness contribution of each attribute
- Issue Solv7: compute the measure of change in trustworthiness with trustworthiness as an absolute quantity
- Issue Solv8: chose the order of calculations in models: as in the hierarchy or else

Appendix B - Example for the Trustworthy Factor Model

Introduction

To illustrate our study about quantifying trustworthiness in software, we provide an example parametric assurance model against a simple software application. The example is intentionally kept very simple so that it is easy to understand.

Objective of the Example

The example consists of taking a software program and computing its trustworthy attributes against claims of "reliability". We construct an assurance case by collecting evidence to substantiate the claims of trustworthiness. Our goal is to attempt to quantify the trustworthiness of that code.

Description of the Example – Web Server code

The example consists of source code for a simple web server. This code was created by Fred Cohen and used by Paul Black for his dissertation [BLACK]. The 160 lines of code are written in C programming language.

We treat this code as a program of unknown pedigree, as we wish to find out how "trustworthy" the application is. We execute the web server application, varying the input to verify that the program functions as intended. We perform static source code analysis on the web server code using two automated static source code analysis tools. In addition, a human analyst verifies the report of the analysis tools, and identifies any additional weaknesses that they find.

We constructed a simple "claims, arguments and evidence" assurance case model, with the top claim of "The web server is reliable".

We use the dynamic testing report and source code analysis reports as the evidence substantiating the claim that the web server application is reliable.

Trustworthy Factor Assignment

We select one attribute of trustworthiness, namely "reliability" as the main claim of our argumentation. The definition of reliability (for our purposes) is the ability to deliver continuing service without failure.

Structured Assurance Case Models for Web Server code

The top-level claim for the web server code is “reliable”. We propose two arguments:

- ARG-1: Targeted fuzz testing of input is effective in revealing potential vulnerabilities in web applications.
- ARG-2: Static source code analysis combined with human analysis provides a high degree of confidence that weaknesses that could lead to a loss of reliability, are discovered.

Figure 2 is the structured assurance case for web server code.

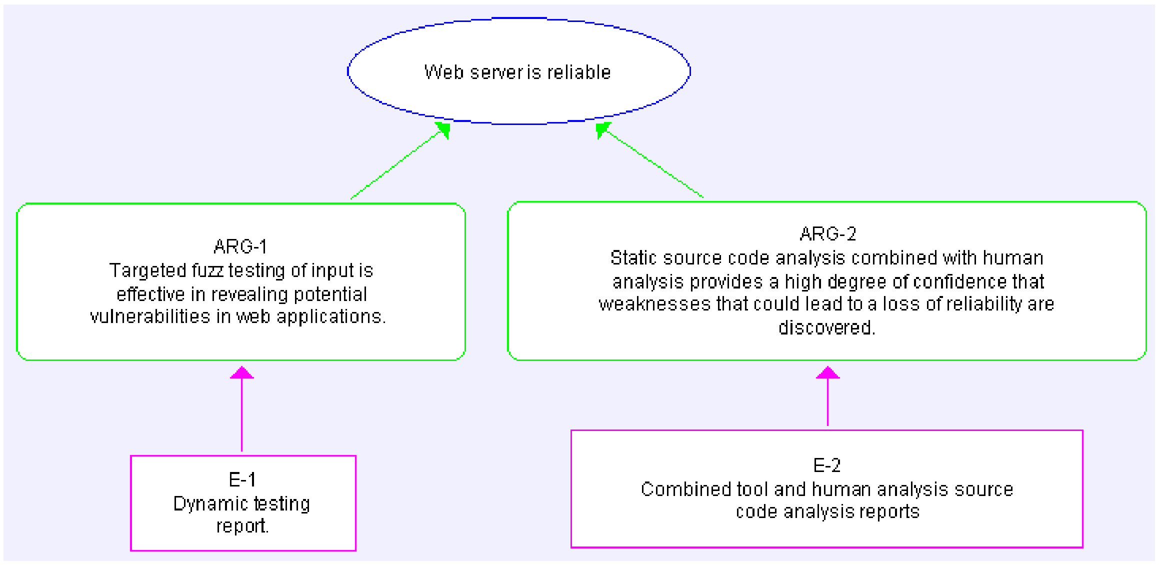

Web server is reliable
ARG-1
Targeted fuzz testing of input is effective in revealing potential vulnerabilities in web applications.
ARG-2
Static source code analysis combined with human analysis provides a high degree of confidence that weaknesses that could lead to a loss of reliability are discovered.
E-1
Dynamic testing report.
E-2
Combined tool and human analysis source code analysis reports

Figure 2 – Structured Assurance case for web server code

The following describes the steps to calculate a trustworthy factor as formulated in this report. We start at the lowest level with the evaluation of evidence strengths at the bottom node. We then work upwards on each link evaluate the argument strengths up to the top claim level.

Evidence level: **Evaluate the evidence strengths ES-1 and ES-2**

The evidence strength is evaluated by a human with series of statements or reasons intended to establish a position or a process of reasoning. The following scales were used to evaluate the evidence strength:

Scales:	0 = strongly disagree,	0.25 = partially disagree,	0.5 = neutral,
	0.75 = partially agree,	1 = strongly agree	

Evidence Evaluation	Evidence E-1: Dynamic testing report.	Evidence E-2: Combined tool and human source analysis report.
This evidence is *measurable*.	Evaluator-C = 1	Evaluator-C = 1
	Comments =”Evidence is measurable in its coverage of types and bounds of input used to test the application.”	Comments = “Evidence is measurable in the number of true vulnerabilities discovered.”
This evidence is *reproducible*.	Evaluator-C = 1	Evaluator-C = 0.75
	Comments = ”If the same input tests are performed on the same code executed in the same environment, one will get the same results.”	Comments = “Human analysis results could vary depending upon expertise of different reviewers.”
This evidence is *not subject to tampering*.	Evaluator-C = 1	Evaluator-C = 1
	Comments = “The assumption is that the dynamic analysis report is not altered in any way.”	Comments = ”The assumption is that the human/tool analysis report is not altered in any way.”
This evidence is *accurate*.	Evaluator-C = 0.75	Evaluator-C = 1
	Comments = ”Coverage of input is robust, and faults or failures have been documented against that input. Coverage however is not exhaustive.”	Comments = ”Because the application is small (160 lines), confidence in the accuracy of tool/human analysis is high.”

Evidence Strength Calculation

It is assumed that the weight of E-1 is 0.8 and weight of E-2 is 0.9.
The evidence strengths ES-1 and ES-2 can be calculated as:

ES-1 = 0.25 * 0.8 * 1 + 0.25 * 0.8 * 1 + .25 * 0.8 * 1 + 0.25 * 0.8* 0.75 = **0.75**
ES-2 = 0.25 * 0.9 * 1 + 0.25 * 0.9 * 0.75 + 0.25 * 0.9 * 1 + -.25 * 0.9 * 1 = **0.84**

Argument level: **Evaluate the pertinence of the evidences for arguments**

The Argument Result evaluation is given by human – albeit informally – to communicate and persuade stakeholders that sufficient confidence can be had in a particular system. The following scale was used.

Scales: -1 = strongly disagree, 0.5 = partially disagree, 0 = neutral, 0.5 = partially agree, 1 = strongly agree

Argument Result Evaluation	Argument A-1: Targeted fuzz testing of input is effective in revealing potential vulnerabilities in web applications.	Argument A-2: Static source code analysis combined with human analysis provides a high degree of confidence that weaknesses that could lead to a loss of reliability, are discovered.
The *validity* of this argument is based upon empirical evidence that supports the success of this methodology in software engineering.	Evaluator-A = 0.75	Evaluator-A = 0.75
	Comments = ”Fuzz testing is not exhaustive, so there remains a possibility that a weakness may still exist. ”	Comments = “Because source code analysis is a ‘white box’ method, likelihood of discovering weaknesses is greater than fuzz testing. However, even white-box tools and human analysis may not identify all weaknesses.”
This truth of the premise contributes to the *soundness* of this argument.	Evaluator-A = 1	Evaluator-A = 1
	Comments = ”The premise that targeted fuzz testing is effective is true.”	Comments = “The premise that combined tool and human analysis is effective is true ”
This evidence strength is *independent* of other evidence strengths for this argument.	Evaluator-A = 1	Evaluator-A = 1
	Comments = “Fuzzing (black box) analysis is considered independent of source code analysis for this exercise.”	Comments = ”Source code (white box) analysis is considered independent of fuzzing analysis for this exercise.”

Calculation of the evidences pertinence

The weights to qualify the pertinence of the evidences for the arguments are the average of the evaluations from the different adjectives. So the weights w1 and w2 are calculated to be the average of all the evaluations by Evaluator-A, and the equations of w1 and w2 are calculated as follows:

w1 = (0.75 + 1 + 1)/3 = **0.917**
w2 = (0.75 + 1 + 1)/3 = **0.917**

The Argument Result can be calculated as:

Argument1Result = (ES-1 * w1) /w1 **= 0.75**
Argument2Result = (ES-2 * w2) /w2 **= 0.84**

Argument level: **evaluate the pertinence of the arguments for the claim**

The pertinence of the arguments is evaluated by human with the following scale.

Scales:	0 = strongly disagree,	0.25 = partially disagree,	0.5 = neutral,
	0.75 = partially agree,	1 = strongly agree	

Reliability Attribute Equation

Reliability Requirement (claim): “The web server code must be reliable at all times in all environments”.

Argument 1: Targeted fuzz testing of input can effectively reveal vulnerabilities to tainted input attacks.

Argument 2: Automated static source code analysis combined with human analysis provides a high degree of confidence that weaknesses that could lead to a loss of reliability are discovered.

Attribute Evaluation	Argument A1: Targeted fuzz testing of input is effective in revealing potential vulnerabilities in web applications.	Argument A2: Static source code analysis combined with human analysis provides a high degree of confidence that weaknesses that could lead to a loss of reliability, are discovered.
Satisfaction of the argument is *directly related* to satisfaction of this reliability requirement.	Evaluator-B = 1	Evaluator-B = 1
	Comments = ”Fuzz testing results directly impact claim of reliability”	Comments = “Static analysis results directly impact claim of reliability”
This argument is *independent* of other arguments for this reliability requirement.	Evaluator-B = 1	Evaluator-B = 1
	Comments = “Treating the web server as a ‘black box’ makes this testing independent from source code analysis.”	Comments = “The source code analysis was not informed by the dynamic input testing.”

Calculation of the reliability requirement satisfaction

The weights to qualify the requirement satisfaction are the average of the evaluations from the different adjectives. So the weight wt1 for Argument1 and wt2 for Argument2 are calculated as follows:

wt1 = (1 + 1)/2 = 1
wt2 = (1 + 1)/2 = 1

Attribute Satisfaction:
Att = (wt1 * arg1 + wt2 * arg2) / (wt1 + wt2)
= (1 * 0.75 + 1 * 0.84) / (1 + 1) = **0.80**

Top claim level: **evaluate the pertinence of the top claim for the trustworthiness**

The relation of the claim vs. trustworthiness is evaluated by human with the following scale.

Scales:	0 = strongly disagree,	0.25 = partially disagree,	0.5 = neutral,
	0.75 = partially agree,	1 = strongly agree	

Satisfaction of this reliability requirement is *important for determining trustworthiness* of this black-box code in a given context.

Evaluation of evaluator-C = 1, comments = “if the code is not reliable, the software should not be trustworthy”.

Claim satisfaction: C = 1

<u>Trustworthiness Index Calculation</u>

Calculation of Trustworthy Factor:
TI = 1 * C * Att = 1 * 1 * 0.80 = **0.80**

Alternative calculation of TI:
Tactual = 1 * 0.80 = 0.80
Tmaximum = 1 * 1 = 1
TI = Tactual / Tmaximum = 0.80 / 1 = **0.80**

Based upon the assurance case, we can conclude that the trustworthy factor for the web server code, between ranges from 0 to 1 is **0.80**. This means we partially agree that this web server code is reliable.