NIST Interagency Report 7755

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

November 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 describe the characteristics of software systems that relate to or support trustworthiness and will help 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

1. INTRODUCTION 5

1.1 BACKGROUND 5
1.2 ASSUMPTIONS 6
1.3 GLOSSARY 7
1.4 OUTLINE AND CONTEXT OF THIS PAPER 7

2.0 TRUSTWORTHY SOFTWARE FRAMEWORK 8

2.1 DESCRIPTION OF THE APPROACH 8
2.2 STRUCTURED ASSURANCE CASE METHODOLOGY 10

3.0 TRUSTWORTHY FACTOR COMPUTATION MODEL 10

4.0 TRUSTWORTHY ATTRIBUTE COMPUTATION EQUATIONS 14

5.0 CLAIMS/SUBCLAIMS SATISFACTION EQUATIONS 15

6.0 ARGUMENT RESULT COMPUTATION EQUATIONS 17

7.0 EVIDENCE STRENGTH COMPUTATION EQUATIONS 18

8.0 EXAMPLE AND LESSONS LEARNED 20

9.0 SUMMARY AND CONCLUSIONS 20

10.0 REFERENCES 21

APPENDIX A - EXAMPLE FOR THE TRUSTWORTHY FACTOR MODEL 23

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 some 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. This is "work-in-progress" and is presented to facilitate further discussion.

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.

The purpose of this document is to report work-in-progress in the investigation of the measurement and modeling toward the trustworthiness of software systems. The approach is:

- 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. Alternatives and issues are presented in the body of this paper to encourage future discussion and research.

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, processes used to create software, and people involved in the development of software are beyond the scope of this project.

Beneficiaries of this approach would be software researchers and testing practitioners. If successful, the approach should help assess 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.

1.2 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 and thus it is important to be able to measure that context. It is also assumed that the quantifiable trustworthy framework is expressible in a structured assurance case model [KELLY-ARG].

The structured 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 a trustworthy factor is product-based (not process-based). It is assumed that arguments given are primarily inductive (not deductive).

1.3 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)

1.4 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.

As mentioned previously and to emphasize, since the development of a trustworthy factor model is work-in-process, many issues and questions are still not determined. Therefore, this document contains many notes for future deliberations. These are issues and questions regarding modeling, representation, risks, and relationships between arguments which will form the basis for further research.

Appendix A 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 a 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.

This trustworthy software framework involves a hierarchical organization of models, with the output of one model at one "level" being the input into another model at a higher "level." However, the models themselves may be developed independently, in a modular approach. In developing these models, attention is given to feasibility of implementation. This framework is chosen for its simplicity; it is possible to have alternative frameworks of models, or to consider each model in the absence of a framework.

2.1 Description of the Approach

As mentioned previously, this 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. A "top-down" or alternative sequencing of model calculations within a framework (or even different orderings of calculations within particular models) is possible.

The approach taken allows a determination of risk, since trustworthiness is related to risk. The trustworthy factor produced has an inverse relationship to risk, in that the higher the value of the factor, the lower the risk, and vice versa.

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. Determining the sufficiency of the amount of specific explanation/explicit relationship/framework detail information provided is a subject for future research.

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 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. It is an open discussion topic as to how to calculate uncertainty or how to incorporate uncertainty calculations in a particular model or framework of models.

The use of modeling as a technique, as opposed to or augmented with simulation, is a future subject for study. It is also important to manage complexity and resultant implementation issues when employing models or a framework of models; how to best manage this complexity needs more research. For example, it may be possible to simplify the framework or remove certain elements from individual models without significant loss of veracity; this is an item for future discussion.

The models in this framework employ "linear" mathematical equations. The use of different quantifiable mathematical/statistical representations (such as nonlinear/differential equations, union/intersection, etc.) is possible.

In the models of the framework, the attempt is made to choose appropriate variable names for semantic meaning, to properly introduce the weights of the different models, to communicate definitions of the terms used in each model, to make terminology consistent as much as possible, and to provide documentation of all relationships among the equations in each model; all these efforts need further study.

2.2 Structured Assurance Case Methodology

The structured assurance case methodology [KELLY-SAFETY] 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 [NIST-JOURNAL]. Such a model consists of the following items:

- claims (denoted following by "Claim" designation) ,
- subclaims (denoted following by "SubClaim" designation),
- arguments (denoted following by "Argument" designation, and argument result denoted following by "ArgResult" designation), and
- evidence (denoted following by "E" designation, and evidence strength denoted following by "ES" designation),

These items are all related in a hierarchical fashion. Definitions are given in Section 1.3.

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 "E" designation) is used in support of the applicable claims/subclaims. An argument (denoted following by "Argument" designation) may consist of all the applicable evidence and subclaims taken together, along with rules of inference, to support a claim.

It is important to realize that there is a skill to properly expressing all of these items (including use of metrics in the models) in order to make the resultant values in the framework meaningful. Appendix A provides some examples, but the best way to express these items is a subject for research.

The framework presented in this paper is based on the structured assurance case methodology. The adaptability of such a framework to other methodologies and paradigms (or the use of other frameworks) are both discussion items for the future.

3.0 Trustworthy Factor Computation Model

In this section the equations involved in computing a trustworthy factor TI are given. First the equation for TI is presented as a ratio of the actual trustworthiness (T_{actual}) to the potential maximum trustworthiness ($T_{maximum}$). Then, the equation for the actual trustworthiness is given as a function of trustworthiness attributes (for example, Att1), and associated weights (for example, wgt1). Finally, the equation for the potential maximum trustworthiness is given as a function of the weights for each attribute. Some alternatives are given, depending on the preferences of users/evaluators.

The trustworthy factor is expressed as follows:

[1] $TI = T_{actual} / T_{maximum,}$

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.

Figure 1, represented by the Kiviat chart below, indicates a possible graphical illustrative depiction of some possible terms in the above equation. The jagged line moving around the inside of the circle in the figure might represent the actual measure of trustworthiness for that term, and the outside circle might represent the potential maximum trustworthiness possible for each term.

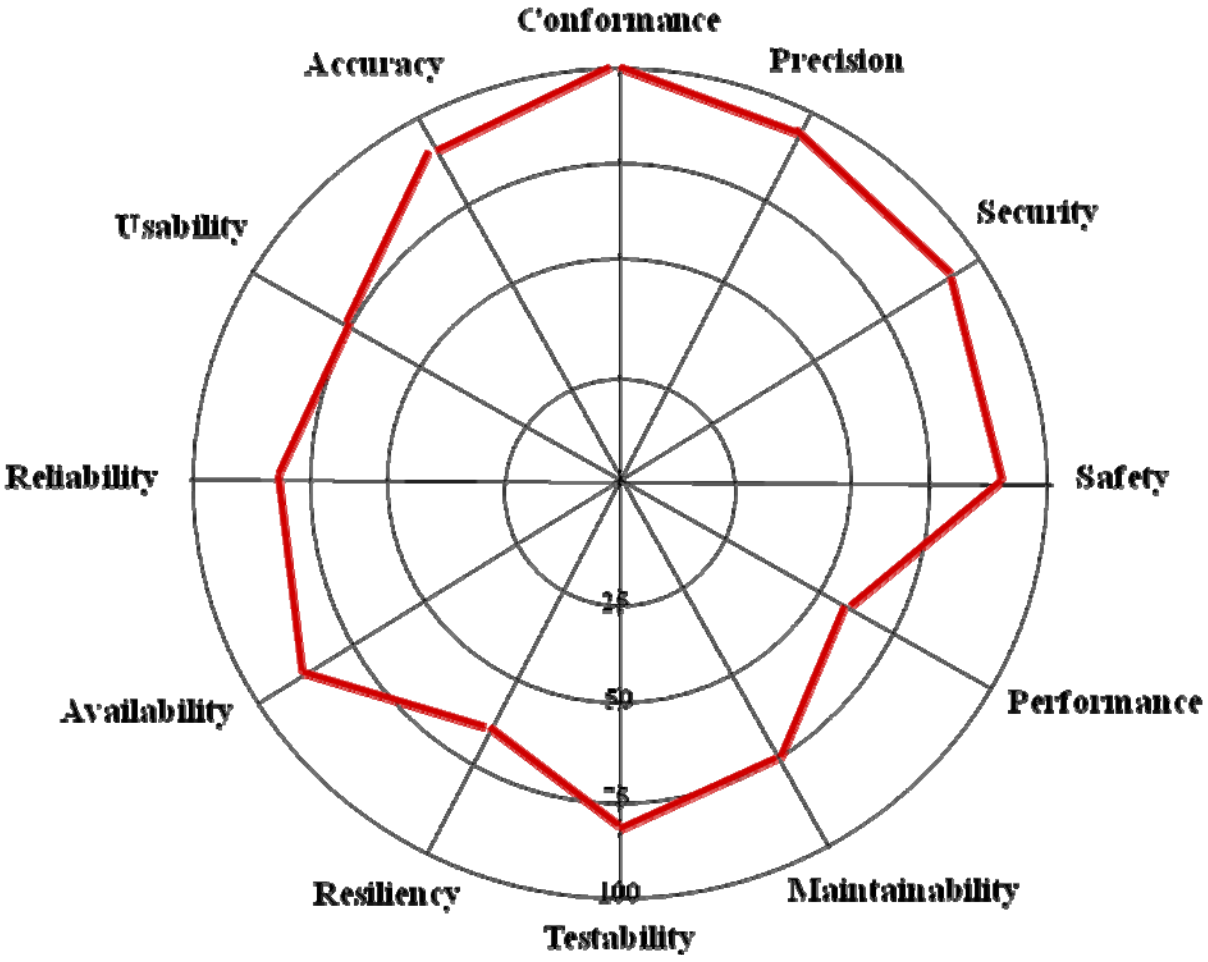

Figure 1 – Trustworthy Factors Represented by Kiviat Chart

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 is the "inverse" of trustworthiness index.

It is important to distinguish managing risk from managing trustworthiness, although the two are related. The nature of inclusion of risk analysis and measurement in the framework needs further investigation.

The equation for TI reads:

T_{actual} = [(wgt1Att1)(wgt2Att2)(wgt3Att3)…] [wgt4Att4 + wgt5Att5 + wgt6Att6 + …]

There is the possibility of a cost (in time, resources, etc.) to "engineer" software for a desired degree of trustworthiness. If such a cost is applicable, it needs to be considered; how it is considered is a subject for further study.

In the equation above, there needs to be at least one multiplicative term. If there are 0 additive terms, the additive expression is eliminated.

The use of differing notations to express number of occurrences, as well as the choice of correct default values, are subjects for further study.

In this equation, the multiplicative terms (for example wgt1Att1, etc.) represent critical (more important) trustworthy attributes (Att1 to Att6) with weights (wgt1 to wgt6). The additive terms (for example wgt4Att4, etc.) represent noncritical (desirable or less important) attributes (they may in fact be the same attributes).

Some determinants are possibly more critical than others for determining trustworthiness. A possible criterion for criticality is: if attribute fails total trustworthiness is 0. Other criteria are feasible as well.

Users/evaluators could define their own attributes, and there may be overlap among the attributes. Each attribute (for example, Att1) and each weight (for example, wgt1) has a value between "0" and "1". A weight's value represents the importance of that attribute in the evaluator/user's definition of trustworthiness and for context/other information. The values of the weights would be computed by consensus around answering questions in validated questionnaires – there is a skill to expressing the questions properly – values of weights in multiplicative part cannot be zero, but value of weights 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 weights. The values for attributes Att1 would be determined as in Section 4.0.

Currently the maximum actual trustworthiness is 1. It is possible to "give extra credit" for "over-engineering" a particular attribute so that its value would be > 1, which may potentially make TI > 1. In this way one attribute could compensate for another.

The equation for $T_{maximum}$ reads:

$T_{maximum}$ = [(wgt1) (wgt2) (wgt3)…] [wgt4 + wgt5 + …]

In this equation, there needs to be at least one multiplicative term. If there are no additive terms, then the additive expression can be eliminated.

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

If it is desired to "bound" TI there is the following equation for T_{actual}:

[2] T_{actual} = [(wgt1 Att1) (wgt2 Att2) (wgt3 Att3) (wgt4 Att4 / n + wgt5 Att5 / n +...)]

and for $T_{maximum,}$

$T_{maximum}$ = [(wgt1) (wgt2) (wgt3) (wgt4 / n + wgt5 / n + wgt6 / n + ...)]

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

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

It is possible to put time dependencies (values are functions of time t) on the terms by creating an equation:

T_{actual} = [(wgt1(t) Att1(t)) (wgt2(t) Att2(t)..) (wgt3(t) Att3(t) + wgt4(t) Att4(t) + ..)]

The use of different scoring metrics for TI (including the nature of any time dependencies and longevity of trustworthiness calculations) is an item for further study. It is also possible to calculate the change in TI (or trustworthiness) over time rather than TI or trustworthiness as an absolute quantity.

There needs to be at least one (critical) attribute on the right-hand side of an equation. If there is only one attribute given, and the associated weights in numerator and denominator are the same, then the weights are not applicable, and the given attribute value is the computed value for TI.

4.0 Trustworthy Attribute Computation Equations

In this section the equations for computing the trustworthy attributes mentioned previously are given. These attributes are computed as a function of claims (for example, Claim1) and associated weights (for example, wgt1), against associated requirements (for example, Req1) and associated weights (for example, wgt1). Some alternatives are given, depending on the preferences of users/evaluators.

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

[3] Att1 = [wgt1 Claim1 + wgt2 Claim2 +…] / [wgt1 Req1 + wgt2 Req2 + …]
Att2 = [wgt3 Claim3 + wgt4 Claim4 +…] / [wgt3 Req3 + wgt4 Req4 + …]

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.

The Claim's (Claim1, Claim2, 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 weights wgt1, wgt2, etc. represent the importance or consequence (validity) of that requirement or claim in determining that attribute (Att1, for example); values of these weights are determined as mentioned above for TI coefficients in Section 3.0. In the above equation the weights are the same for each claim-requirement pair, but it is possible to have different weights for each of those. Values for claims are determined as in Section 5.0. Values for requirements 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 just claims will be used. It is also possible that requirements (and attributes) may "overlap" among themselves or have interdependencies among themselves; how to best define these interdependencies if present is an open issue.

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). How to manage the relationship of requirements to attributes (and even claims to requirements) – for example, using a "matrix" approach or an alternative – needs more discussion.

To separate claims and requirements into "critical" and "desirable" similar to what was done for TI above, the equation for Att2 (for example) is:

Att2 = [(wgt3 Claim3) (wgt4 Claim4)...] / [(wgt3-1 Req3) (wgt4-1 Req4)...]
[wgt5 Claim5 + wgt6 Claim6 + ...] / [wgt5 Req5 + wgt6 Req6 + ...]

In the above wgt3-1 needs to be different from wgt3 (for example) or the weights would cancel out.

It is important to verify and subsequently document how exactly a claim or requirement specifically relates to an attribute in terms of trustworthiness.

An alternative to the equation for Att1 involves setting a threshold value $Att1_{min}$, so that if Att1 >= Att_{min} , then Att1 = 1; if Att1 < Att_{min} , then Att1 = 0.

There needs to be at least one claim on the right-hand side of an equation. If there is only one claim in the equations above, and the weights for claims and associated requirements are the same, then the weights are not applicable and the claim value is the computed value for the trustworthy attribute.

5.0 Claims/Subclaims Satisfaction Equations

In this section equations for claims or subclaims satisfaction are given. Equations for claims satisfaction are presented as a function of argument results (for example, ArgResult1), and associated weights (for example, wgt1) giving the relative importance of that argument result to the associated claim satisfaction. If subclaims are used, an equation for a claim as a function of subclaims (for example, SubClaim1) for that claim along with associated weights (for example, wgt1), is presented. Finally, if subclaims are used, equations for subclaims are given as functions of argument results and associated weights. Some alternatives are given, depending on the preferences of users/evaluators.

It is assumed that each claim would either be satisfied or not satisfied (binary result). So a claim would be 0 (not satisfied) or 1(satisfied), depending on the values of one or more argument result terms ArgResult1, ArgResult2, etc. (explained following). So if a threshold is exceeded in the claims equation below, the claim value is 1; if not, the claim value is 0.

It is possible to allow partial satisfaction of claims (perhaps using "fuzzy logic," "Bayesian probability," or other probability considerations), in which case the threshold approach would not be used. Developing additional guidance on what constitutes sufficiency for a claim (or requirement) having been met satisfactorily (or how support is specifically measured for a claim) is a necessary activity for the future.

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 (for example)

[4] Claim1 = [wgt1 ArgResult1 + wgt2 ArgResult2 +…]/[wgt1 + wgt2 + ...]

In this equation wgt1, wgt2, etc. are weights 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. It is possible to remove the denominator term in the equation above to make the weights possibly have even more importance; it would then be necessary to multiply each expression by 1/n, where n is the number of arguments, in order to bound C1aim1 at 1.

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

Values of ArgResult1, Arg1Result2, etc., are determined as described in Section 6.0.

Claims may be broken down into subclaims (with associated arguments); if there is a claim Claim1, and SubClaim1, SubClaim2, SubClaim3, are subclaims of Claim1, then the equation for Claim1 is:

Claim1 = [(wgt1 SubClaiml) (wgt2 SubClaim2) (wgt3 SubClaim3)…]

In the above it is assumed that each claim will have at least one sub-claim – "wgt" terms are weights on the subclaims, indicating their importance to the overall claim.

In other words, if the subclaims have binary values (0-satisfied or 1-not satisfied), then all subclaims need to be 1 for the associated claim to be 1. In this case the value of each subclaim is determined by the exceeding of a threshold for each argument equation (discussed following) for each subclaim, so that, for example:

SubClaim1 = [(1/n) wgt1-1 ArgResult1-1) + (1/n) wgt1-2 ArgResult1-2...]
SubClaim2 = [(1/n) wgt2-1 ArgResult2-1) + (1/n) wgt2-2 ArgResult2-2…]

etc., where values of ArgResult1-1, etc. would be determined as described in Section 6.0, and wgt1-1, etc. are weights of relevance as mentioned previously. If thresholds are exceeded in the subclaim calculations given above, values are 1; if not, values are 0. These equations are using the alternate form for claims equation discussed above, to give more importance to the wgt values. It is also possible to allow partial satisfaction of subclaims (as for claims), in which case thresholds would not be used.

A subject for discussion is how many subclaims are sufficient to satisfy a claim (identification of a factor for "subclaim gap"), in terms of satisfaction of all subclaims equaling satisfaction of the resultant claim (perfect decomposition or independence of subclaims). It is important to define subclaims correctly to achieve "sufficiency of coverage," and to define any interdependencies among subclaims.

There needs to be at least one argument result in the right-hand side of an equation. If there is only one argument result in the above equations for claims or subclaims, and the corresponding weights in numerator and denominator are the same, then the weights are not applicable and the argument result value is the computed value for a claim or subclaim.

6.0 Argument Result Computation Equations

In this section equations for determining argument result are presented. Argument results are computed as a function of evidence strength values (for example, ES-1), along with associated weights (for example, wgt1). Some alternatives are given, depending on the preferences of users/evaluators.

For example, the value of the argument ArgResult1 for claim Claim1 is determined by:

[5] ArgResult1 = [wgt1 ES-1 + wgt2 ES-2 + wgt3 ES-3 + ...] / [wgt1+ wgt2 + wgt3 +…]

There is the same number of items in the numerator as in the denominator. If a denominator is not included (to give even more importance to the "wgt" values), how to effectively "normalize" ArgResult1 needs to be studied.

In the above equation "wgt1, wgt2, 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-ARG]. Weights are computed by responses to questions in validated questionnaires. In other words, in the denominator, all evidence strength values are assumed to be 1.

"ES" variables indicate evidence strength (see Section 7.0); values of evidence strength are determined as described in Section 7.0. So ArgResult1 (for example) is normalized between 0 and 1.

In the equation for ArgResult1 the higher the values of ES-1, etc., the higher the value of ArgResult1.

To determine the value of ArgResult1 (for example), the evaluator/user sets a valid argument threshold value ArgLimit1 (determined from consensus around answers to questions in validated questionnaires – for example, "as confident as reasonably practical" or considering "assurance deficit" issues).

If ArgResult1 >= ArgLimit1, then ArgResult1 = 1 in the equation for Claim1 above in Section 5.0 ;

conversely if ArgResult1 < ArgLimit1, then ArgResult1 = 0 in the equation for Claim1 above in Section 5.0.

To allow partial satisfaction of claims, then the argument threshold approach is not used, and the approach taken is to just allow ArgResult1 to be a fraction between 0 and 1 in the equations for claim values.

It is possible to develop argument templates, patterns, or other guidance arguments to use in calculating argument results; this is particularly important when argument results must be calculated repeatedly, since such calculation may be a "labor-intensive" process.

There needs to be at least one evidence strength value on the right-hand side of an equation. If there is just one evidence strength value provided, and the corresponding weights in numerator and denominator are the same, then the weights are not applicable and the evidence strength value is the computed value for the argument result.

7.0 Evidence Strength Computation Equations

In this section, equations for computing evidence strength are presented. Evidence strength is given as a function of evidence strength factors, along with associated weights of importance to the overall evidence strength. Evidence strength is based on an evaluation of the actual evidence, but is distinguished from the actual evidence. Some alternatives are given, depending on the preferences of users/evaluators.

The "ES" variables represent evidence strength indicators for each piece of evidence used to support the claim. Evidence strength must be measured in some fashion, and the evidence strength calculations 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. For example, the equation for evidence strength score ES-1 (normalized from 0 to 1) would be:

ES-1 = [wgt1 S1 + wgt2 S2 + wgt3 S3 + wgt4 S4 + wgt5 S5 +…] / [wgt1 + wgt2 + wgt3 + wgt4 + wgt5 + ...]

It is assumed that there is same number of items in numerator as in denominator.

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 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).

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 ES-1 wgt1, wgt2, 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. Computation of weights is as described previously for other weights in other models, but the methodology of weight calculation considering various factors (for example, context) needs further discussion. There is exactly one piece of evidence associated with each evidence strength factor.

An alternative form for ES-1 is to use:

[6] ES-1 = [(1/n) wgt1 S1 + (1/n) wgt2 S2 + (1/n) wgt3 S3 + (1/n) wgt4 S4 + (1/n) wgt5 S5 +...]

where n represents the number of S factors (discussed previously).

There is the possibility of counter-evidence items (evidence against a particular claim). For example, it is possible for certain selected S factors to be negative as appropriate in the equation above (in which case there is the possibility for ES1-1 to be negative). Alternatively, to subtract counter-evidence items in the evidence strength equation, the modified evidence strength score ESm1-1 is:

ES-mod-1 = [wgt1 (S1p – S1n) + wgt2 (S2p – S2n) + wgt3 (S3p – S3n) + ...] / [(wgt1 + wgt2 + wgt3 + wgt4 + …]

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 (considering “assurance deficit” issues, for example).

An alternative to calculating evidence strength is to use direct evidence; in this case, how the evidence itself is employed in argument construction needs to be investigated.

It is possible that previously-computed trustworthiness determinations are fed back into the evidence model as evidence at a future time, for incorporation into the framework. Thus, the models in the framework are run continuously, with previous determinations of trustworthiness serving as input to future determinations of trustworthiness. The issue of “circularity” in model calculations (and in model decomposition) needs to be studied further.

There needs to be at least one evidence strength value on the right-hand side of an equation. If there is just one evidence strength value provided, and the corresponding weights in numerator and denominator are the same, then the weights are not applicable and the evidence strength value is the computed value for the overall evidence strength.

8.0 Example and Lessons Learned

Appendix A illustrates an example to assess that the trustworthy factor model and framework described here is feasible and credible. The example used a relatively small web server program written in C with 160 lines of code.

Some items learned so far from doing the example include:

- 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 into 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; future research should address (among other factors) the possible quantification of operational context, as well as any possible automation (including programming language support) of these calculations. Other approaches (for example, threat attack methodology) should be considered in relation to this approach. The subject of what metrics/measurement methods to use in calculations is an item for further research. Larger and more detailed examples or case studies should yield additional information as to the ultimate utility of this approach in informing software trustworthiness.

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

[KELLY-SAFETY] Kelly, T. "Arguing Safety – A Systematic Approach to Safety Case Management", DPhil Thesis, York University, Department of Computer Science Report YCST, May 1999

[KELLY-ARG] 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-JOURNAL] 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 - 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 constructed an assurance case by collecting evidence to substantiate the claims of trustworthiness. Our goal was 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 in the C programming language.

We treated this code as a program of unknown pedigree, as we wished to find out how "trustworthy" the application is. We executed the web server application, varying the input to verify that the program functions as intended. We performed static source code analysis on the web server code using two automated static source code analysis tools. In addition, a human analyst verified the report of the analysis tools, and identified 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 used 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 selected 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 proposed two arguments:

- Argument-1 (ARG-1): Targeted fuzz testing of input is effective in revealing potential vulnerabilities in web applications.
- Argument-2 (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.

The "Argument (ARG)" designations are specifically distinguished from the argument result terms computed following, although they are related.

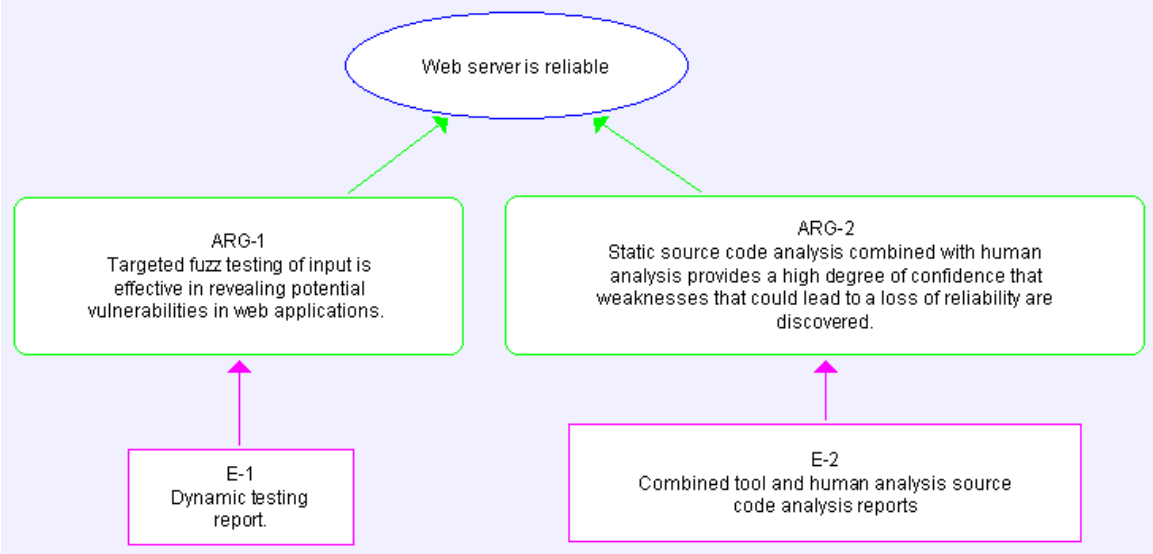

Figure 2 – Structured Assurance case for web server code

Figure 2 illustrates a summary of the structured assurance case developed for this web server code. Note that in Figure 2 E-1 and E-2 are actual pieces of evidence, and are distinguished from evidence strength calculations ES-1 and ES-2 following. ARG-1 and ARG-2 represent developed arguments; based on these arguments, argument results were calculated as ArgResult1 and ArgResult2 following. The top oval is the top level claim represented as "Claim1" in calculations following.

The following describes the steps to calculate a trustworthy factor as formulated in this report. We started at the lowest level with the evaluation of evidence strengths at the bottom node. We then worked upwards on each link evaluate the argument strengths up to the top claim level. In our example, we used three human evaluators, denoted as Evaluator-1, Evaluator-2, and Evaluator-3, to give more credence to the results of the evaluation.

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

The evidence strengths ES-1 for evidence E-1 and ES-2 for evidence E-2 (from Section 7.0) is evaluated by humans with a 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 Strength Evaluation	Evidence E-1: Dynamic testing report.	Evidence E-2: Combined tool and human analysis source code analysis reports.
This evidence is *measurable*.	Evaluator-1 = 1	Evaluator-1 = 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-1 = 1	Evaluator-1 = 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-1 = 1	Evaluator-1 = 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-1 = 0.75	Evaluator-1 = 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

This evidence should be important; therefore, it is assumed that the weight of E-1 is 0.8 and weight of E-2 is 0.9. The four evidence strength factors S (see Section 7.0) from the evaluations above are given as 1, 1, 1, and 0.75 for Evidence E-1, and 1, 0.75, 1, and 1 (respectively) for Evidence E-2.

Given the above, the overall evidence strengths ES-1 and ES-2 can be calculated (using Equation [6] in Section 7.0) 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 Result level: **Evaluate the pertinence of the evidence for the arguments**

The Argument Result evaluation (from Section 6.0) is given by humans – albeit informally – to communicate and persuade stakeholders that sufficient confidence can be had in a particular system. The "Argument" designations given following are developed as in Figure 2, and are used as an aid in evaluating argument results. The following scale was used.

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

---

Argument Result Evaluation	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.
The *validity* of this argument is based upon empirical evidence that supports the success of this methodology in software engineering.	Evaluator-2 = 0.75 Comments ="Fuzz testing is not exhaustive, so there remains a possibility that a weakness may still exist."	Evaluator-2 = 0.75 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-2 = 1 Comments =“The premise that targeted fuzz testing is effective is true.”	Evaluator-2 = 1 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-2 = 1 Comments =“Fuzzing (black box) analysis is considered independent of source code analysis for this exercise.”	Evaluator-2 = 1 Comments =“Source code (white box) analysis is considered independent of fuzzing analysis for this exercise.”

Calculation of the evidence strength pertinence in determining argument result

The weights to qualify the pertinence of the evidence strength for calculation of the argument result are the average of the evaluations from the different assertions given above. So the weights wgt1 and wgt2 are calculated to be the average of all the evaluations by Evaluator-2, and the equations of wgt1 and wgt2 are calculated as follows:

wgt1 = (0.75 + 1 + 1)/3 = **0.917**
wgt2 = (0.75 + 1 + 1)/3 = **0.917**

The Argument Result can be calculated (using Equation [5] from Section 6.0) as:

ArgResult1 = (ES-1 * wgt1) /wgt1 **= 0.75**
ArgResult2 = (ES-2 * wgt2) /wgt2 **= 0.84**

As mentioned in the preceding pages, since there is only one evidence strength term for each argument result, and the weights for numerator and denominator are the same for each equation, the weights in this case are not applicable and (for example) the value of ES-1 is the value of ArgResult1, and the value of ES-2 is the value of ArgResult2. Therefore the above equations could be shortened to:

ArgResult1 = ES-1 = **0.75**
ArgResult2 = ES-2 = **0.84**

Claim Satisfaction level: **evaluate the pertinence of the argument results for the claim**

The pertinence of the argument results for the claim (Claim1 – see Section 5.0) is evaluated by humans with the following scale.

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

Reliability Requirement (Claim1): "The web server code must be reliable at all times in all environments."

The "Argument (ARG)" designations given following are illustrations from Figure 2, and are correlated with argument result calculations done previously.

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

ARG-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.

Requirement (Claim1) Evaluation	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.
Satisfaction of the argument result is *directly related* to satisfaction of this reliability requirement (claim).	Evaluator-3 = 1	Evaluator-3 = 1
	Comments ="Fuzz testing results directly impact claim of reliability"	Comments ="Static analysis results directly impact claim of reliability"
This argument result is *independent* of other argument results for this reliability requirement (claim).	Evaluator-3 = 1	Evaluator-3 = 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 (claim) satisfaction

The weights to quantify the reliability requirement (claim) satisfaction are the average of the evaluations of the pertinence of the argument results to the claim from the different assertions given above. So the weight wgt1 for ArgResult1 and wgt2 for ArgResult2 are calculated as follows:

wgt1 = (1 + 1)/2 = 1
wgt2 = (1 + 1)/2 = 1

Requirement (Claim) Satisfaction (from Equation [4] in Section 5.0):

Claim1 = (wgt1 * ArgResult1 + wgt2 * ArgResult2) / (wgt1 + wgt2)
= (1 * 0.75 + 1 * 0.84) / (1 + 1) = **0.80**

For the reliability attribute equation (see Section 4.0), there is one requirement (claim) for this attribute in this example, so the value of Claim1 is the value of Att1, since there is only one term on the right-hand side as mentioned previously, the weights wgt1 cancel out in numerator and denominator, and Req1 = 1. Therefore, from Equation [3] in Section 4.0 (shortened version):

Att1 = Claim1 = 0.80

Reliability Attribute level: **evaluate the pertinence of the reliability attribute for trustworthiness**

The relation of the reliability attribute to trustworthiness (see Section 3.0) is evaluated by humans 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 attribute is *important (critical) for determining trustworthiness* of this black-box code in a given context (if this attribute is not present the trustworthiness should be 0 from Section 3.0), so this is a critical attribute for trustworthiness.

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

Reliability pertinence to trustworthiness weight: wgt1 = 1

Trustworthiness Index Calculation (from Section 3.0)

Calculation of Trustworthy Factor (from Equation [2] in Section 3.0):
TI = wgt1 * Att1 = 1 * 0.80 = **0.80**

Alternative calculation of TI (from Equation [1] in Section 3.0):
$\mathbf{T_{actual}}$ = 1 * Att1 = 1 * 0.80 = **0.80**
$\mathbf{T_{maximum}}$ = 1 * 1 = **1**
TI = $\mathbf{T_{actual}}$ **/** $\mathbf{T_{maximum}}$ **=** 0.80 / 1 = **0.80**

In the above alternative, maximum trustworthiness is 1, and wgt1 = 1 from evaluation above. As mentioned in Section 3.0, in this alternative calculation, since there is only one term on the right hand side, and wgt1 for numerator and denominator is the same, we can conclude that the value of Att1 is in fact the computed value of TI for this example, so in the shortened version of Equation [1]:

TI = Att1 = **0.80**

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