Identifying Trustworthiness Deficit in Legacy Systems Using the NFR Approach

Nary Subramanian, University of Texas at Tyler
Steven Drager, Air Force Research Laboratory
William McKeever, Air Force Research Laboratory

Abstract. Trustworthiness is an important emerging requirement for software systems deployed by the U. S. Air Force. Trustworthiness, briefly stated, is the ability of a software system to be safe, secure, and reliable under a normal operating environment. However, most software systems have not been developed with trustworthiness in mind. Therefore, how do we systematically identify deficit in trustworthiness in existing systems so that they may be re-engineered with trustworthiness as a priority? The Non-Functional Requirements (NFR) Approach provides a framework for identifying gaps in trustworthiness in existing systems and recommending mechanisms to overcome this "shortfall" in re-engineered systems. In this project we applied the NFR Approach, as a case study to the middleware system called Phoenix used by the Air Force and determined an 89% shortfall in trustworthiness. The advantages of identifying this deficit include determination of trustworthiness in current systems, exploring environments in which current systems may be (re)used, and prioritizing trustworthiness requirements when these legacy systems are re-engineered.

Introduction

Trustworthiness is an important emerging requirement for software systems including those deployed by the U. S. Air Force: the National Software Strategy Report [1] has concluded that trustworthiness in software will become the most important goal by the year 2015. Trustworthiness, briefly stated, is the ability of a software system to be safe, secure, and reliable under normal operating environments [2]. However, several legacy systems in operation were not designed with trustworthiness in mind—therefore, these systems can be used in a trustworthy environment in one of two ways: employing wrappers that will improve trustworthiness of the system or re-engineering the system to be trustworthy. The second option is a long-term solution but will be expensive in terms of effort and cost required to re-engineer the several dozens of systems being currently used by the Air Force. The first option, namely, the addition of wrappers may be a more cost-effective option for many systems. However, how do we systematically identify deficit in trustworthiness in existing systems so that solutions may be developed? This is especially important when trustworthiness has different connotations for developers, users, and maintainers.

Trustworthiness has been defined differently by different sources, based on their approach to determine trust in a system. For example, in [3] trustworthiness is defined as the degree of confidence that exists that the system meets its requirements, while in [4] defines trustworthiness as a level of confidence of using software engineering techniques to reduce failure rates, enhance testing, reviews and inspections. A discussion of software trustworthiness among stakeholders often invokes numerous non-functional attributes like reliability, safety, usability, portability, or maintainability, which together ensure non-interference with the normal operation of the system.

The NFR Approach [5, 6], where NFR stands for Non-Functional Requirements, provides a framework for systematically analyzing NFRs such as trustworthiness and decomposing it further to capture other NFRs like reliability, safety, portability, etc. The NFR Approach provides the ability to accommodate alternate definitions of trustworthiness as well as provides a rationalization process that allows one to evaluate the extent to which trustworthiness is achieved by a system. More importantly, the NFR Approach helps to identify gaps in trustworthiness requirements. By understanding the extent of "shortfall" of trustworthiness, one is better prepared to identify solutions necessary to make that system trustworthy for a specified time-scale.

In this paper we apply the NFR Approach to a selected software system and identify the trustworthiness deficit in the system. For this purpose we first obtain the definition of trustworthiness for this system from its stakeholders and convert the definitions into a Softgoal Interdependency Graph, an artifact used by the NFR Approach for reasoning about NFRs, which are treated as softgoals in the system. Then the designs for the selected software system are evaluated against trustworthiness definitions using the propagation rules of the NFR Approach. This evaluation will identify deficit in trustworthiness and will permit analysis on how this deficit needs to be overcome. This analysis will help identify adaptations that are needed to make the selected software system function in a trustworthy environment. These adaptations can be stated in terms of design modifications and/or implementation mechanisms (for example, wrappers) that will help the system be used for a specific time-period in a trustworthy environment.

This problem considered is explained by Figure 1: legacy system fulfills primarily its requirements and, mostly by accident, some trustworthy requirements that represent the existing trust in the legacy system. The trustworthy system includes the requirements for trustworthiness that represent the total expected trust as well as the re-engineering requirements for the legacy system. The difference between the total expected trust and the existing trust is the trustworthiness deficit in the legacy system.

The legacy system we used as a case study is the Phoenix middleware system used by the Air Force - we identified the trustworthiness deficit in Phoenix by using the NFR Approach and developed a process for applying this approach to other software systems. Our study identified an 89% shortfall in trustworthiness in the existing Phoenix system.

This paper was presented at the Software Technology Conference held in Salt Lake City, Utah, in April 2013 [7] and was well received by the audience.

Background

The existing approaches to trustworthy analysis split into two categories: product-based and process-based. Product-based techniques [9] identify factors that impact trustworthiness and attempt to satisfy these factors in the product. Process-based techniques, like Trusted Software Methodology [3] and Trustworthy Process Management Framework [10] approach the problem with the belief that trustworthy processes will result in trustworthy products. However, the NFR Approach considers trustworthiness as a non-functional requirement for the product

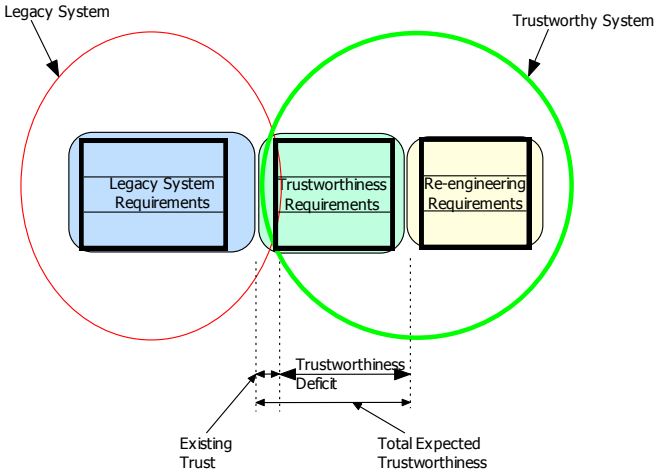

Figure 1. Context of the Trustworthiness Deficit Problem

being developed; being an NFR its constituents may interact synergistically or conflictingly and NFR Approach is fully geared to analyze these tradeoffs.

The NFR Approach

The NFR Approach is a goal-oriented approach that can be applied to determine the extent to which objectives are achieved by a process or product [5, 6]. NFRs represent properties of a system such as reliability, maintainability, and flexibility, and could equally well represent functional objectives and constraints for a system (NFRs need to be contrasted with functional requirements—the latter state what the software system should do while the former states requirements that are usually observed as a characteristic of the system). In this paper we applied the NFR Approach to design a trustworthy software system by evaluating whether a specific design element satisfied trustworthy requirements for the system. The NFR Approach uses a well-defined ontology for this purpose that includes NFR softgoals, operationalizing softgoals, claim softgoals, contributions, and propagation rules; each of these elements is described briefly below (details may be seen in [5]). Furthermore, the NFR Approach uses the concept of satisficing, a term borrowed from economics, which indicates satisfaction within limits instead of absolute satisfaction, since absolute satisfaction of NFRs is usually difficult.

NFR softgoals represent NFRs and their decompositions. Elements that have physical equivalents (process or product elements) are represented by operationalizing softgoals and their decompositions. Each softgoal is named using the convention (Type [Topic1, Topic2, …]) where Type is the name of the softgoal and Topic (could be zero or more) is the context where the softgoal is used; Topic is optional for a softgoal; for a claim softgoal, which is a softgoal capturing a design decision, the name may be the justification itself.

Following decompositions of either the NFR softgoals or the operationalizing softgoals are possible:

1. AND decomposition is used when each child softgoal of the decomposition has to be satisficed for the parent soft goal to be satisficed but the denial of even one child soft goal is sufficient to deny the parent,
2. OR decomposition is used when satisficing of even one child satisfices the parent but all children need to be denied for the parent to be denied, and
3. EQUAL decomposition has only one child for a parent and propagates the satisficing or the denial of the child to the parent.

Contributions (MAKE, HELP, HURT, and BREAK) are made by operationalizing softgoals to the NFR softgoals and by claim softgoals to other contributions. Reasons for contributions are captured by claim softgoals, and claim softgoals may form a chain of evidence where one claim satisfices another which in turn satisfices another and so on. Each of the four types of contributions has a specific semantic significance: MAKE contribution refers to a strongly positive degree of satisficing of the objectives (represented by NFR softgoals) by artifacts (represented by operationalizing softgoals) under consideration[1], HELP contribution refers to a positive degree of satisficing, HURT contribution refers to a negative degree of satisficing, and BREAK contribution refers to a strongly negative degree of satisficing.

Due to these contributions, some of the softgoals acquire labels that capture the extent to which a softgoal is satisficed: satisficed, weakly satisficed, weakly denied (or weakly not satisficed), denied (or not satisficed), or unknown (indicated by an absence of any label attribute). Moreover, high priority softgoals, decompositions, and contributions may be indicated using the criticality symbol. The graph that captures the softgoals, their decompositions, and the contributions is called the Softgoal Interdependency Graph (SIG). The partial ontology of the NFR Approach is shown in Figure 2. The notations used to indicate the satisficing extent of softgoals are shown in Figure 3.

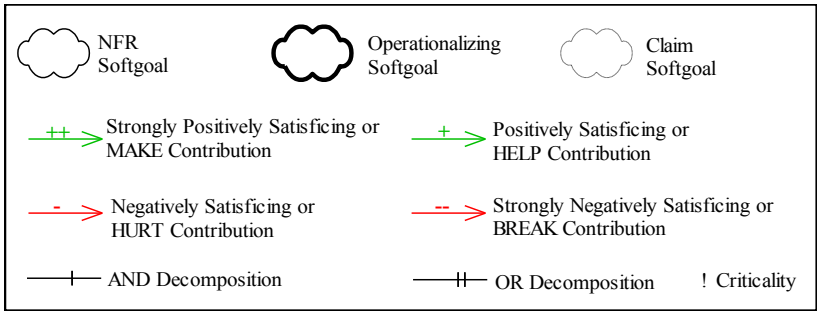

Figure 2. Partial Ontology of the NFR Approach

As shown in Figure 2, normal cloud shaped figures represent NFR softgoals, dark-bordered cloud shaped figures represent operationalizing softgoals, and dashed-bordered cloud shaped figures represent claim softgoals. A green arrow annotated with two plus symbols indicates a MAKE contribution, a green arrow annotated with one plus symbol indicates a HELP contribution, a red arrow annotated with a minus symbol indicates a HURT contribution, while a red arrow annotated with two minus

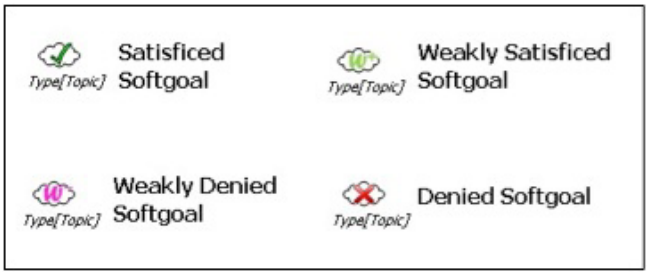

Figure 3. NFR Approach Notations for Softgoal Satisficing

symbols indicates a BREAK contribution. A line with a single cross-line represents AND-decomposition while a line with two cross-lines represents OR-decomposition. Critical elements (softgoals, decomposition, and contributions) are indicated by "!" marks. As shown in Figure 3, a softgoal with a green check mark represents a satisficed softgoal (or a softgoal with a satisficed label), a softgoal with a green W+ annotation represents a weakly satisficed softgoal, a softgoal with a pink W- annotation represents a weakly denied softgoal, and a softgoal annotated with a red X represents a denied softgoal.

Propagation rules propagate labels from child softgoal to the parent across decompositions, from operationalizing softgoals to NFR softgoals across contributions, and from claim softgoals to contributions; propagation rules aid in the rationalization process of the NFR Approach. In a SIG represented graphically, the NFR softgoals and their decompositions are shown at the top of the figure, the operationalizing softgoals and their decompositions are shown in the bottom of the figure, while the contributions between the operationalizing softgoals and the NFR softgoals are shown in the middle. Therefore, contributions are usually received by the leaf NFR softgoals that are at the bottom of the NFR softgoal decomposition hierarchy. While detailed propagation rules may be seen in [5], a simplified list is given below:

R1. A satisficed label is propagated as satisficed by a MAKE contribution, as weakly satisficed by a HELP contribution, as weakly denied by a HURT contribution, and as denied by a BREAK contribution.

R2. A denied label is propagated as denied by a MAKE contribution, as weakly denied by a HELP contribution, as weakly satisficed by a HURT contribution, and as satisficed by a BREAK contribution.

R3. If most of the contributions propagated to a leaf NFR softgoal are satisficed then that NFR softgoal is considered satisficed.

R4. If most of the contributions propagated to a leaf NFR softgoal are denied then that NFR softgoal is considered denied.

R5. In the case of priority softgoals, or when there is a tie between positive and negative contributions, the system architect or the developer can take the decision based on or a variation of R3 and R4

R6. In the case of an AND-decomposition, if all the child softgoals are satisficed then the parent NFR softgoal is satisficed; else the parent softgoal is denied.

R7. In the case of an OR-decomposition, if at least one child softgoal is satisficed then the parent NFR softgoal is satisficed; else the parent softgoal is denied.

R8. In the case of EQUAL-decomposition (only one child) the parent is satisficed if the child is satisficed; and the parent is denied if the child is denied.

Upon applying these propagation rules, if the root (or top-level) NFR softgoals are satisficed then the goals for the domain of interest have been met to a large extent. In this paper the root NFR softgoals will be related to trustworthiness and therefore the SIG will help determine the extent to which a particular design is trustworthy.

The NFR Approach requires the following interleaving tasks, which are iterative

1. Develop NFR goals and their decompositions: in this task the trustworthiness softgoal is decomposed into its constituent NFR softgoals; this decomposition captures the trustworthiness requirements for a system as viewed by a particular group of stakeholders. These decompositions may be developed from scratch or may be extensions of existing decompositions.

2. Develop operationalizing goals and their decompositions: in this task we develop operationalizing softgoals and their decompositions. In this paper operationalizing softgoals correspond to architectural design models[2]. Each individual model may form its own operationalizing softgoal decomposition hierarchy. These models may be developed from scratch or may use existing catalogs as a starting point.

3. Develop goal tradeoffs and rationale: in this task we determine contributions between operationalizing softgoals (task 2) and the NFR softgoals (task 1) and the rationale for the contributions are captured by claim softgoals; synergies and conflicts between different NFR softgoals are captured by the contributions, and tradeoffs (manifested by changes to contributions) that take place are captured by corresponding changes to rationale. This historical record keeping also helps backtracking, if required.

4. Develop goal criticalities: in this task we assign priorities to softgoals—some softgoals (NFR softgoals, operationalizing softgoals, and claim softgoals) may be more important for the stakeholders involved and they are indicated as critical softgoals. Criticalities may also be assigned to decompositions and contributions.

5. Evaluation and analysis: in this task the propagation rules of the NFR Approach are applied to determine whether the design models satisfy the requirements (represented by NFR softgoal decomposition hierarchy) and to what extent – that is, strongly positive, positive, negative, or strongly negative; if positively satisficed then those design models satisfy the requirements and if negatively satisficed then there is scope for improvement.

Example Application of the Steps of the NFR Approach

An example SIG is shown in Figure 4 and we describe how the five steps of the NFR Approach are applied to this SIG. Step 1 involves decomposition of NFR goals for the problem of interest. The upper part of the SIG of Figure 4 captures this decomposition for the NFR trustworthiness for the Phoenix system, which is represented by the root NFR softgoal Trustworthiness [Phoenix]. Based on the definition of trustworthiness for the Phoenix system, we decomposed this NFR softgoal into Dependability [Phoenix], Reliability [Phoenix], Trustworthiness [Phoenix, Software], and Security [Phoenix], which represent, respectively, the requirements that Phoenix must be dependable, Phoenix must be reliable, software for Phoenix must be trustworthy, and Phoenix should be secure. This decomposition is an AND-decomposition, which means all child softgoals must be satisficed for the parent to be satisficed. The NFR softgoal is further AND-decomposed into NFR softgoals Security [Messages] and Timeliness [Messages], which represent, respectively, the requirements that messages be secure and timely. This completes the first step.

In the second step we decompose the design of the Phoenix system. This is shown by operationalizing softgoals at the bottom of Figure 4. We considered two views of the design: Component and Connector Logical View (represented by the operationalizing softgoal C&C View [Logical]) and Detailed Module View of the Submission Service (represented by Module View [Detailed, Submission Service]). The operationalizing softgoal C&C View [Logical] is AND-decomposed into three component softgoals representing Repository Service, Authorization Service, and Channels. The operationalizing softgoal Module View [Detailed, Submission Service] is AND-decomposed into its component softgoals Information Validator, Input Channel Manager, Policy Manager, and Forwarder. This completes the second step.

In the third step of the NFR Approach we determine the contributions between the operationalizing softgoals and the NFR softgoals; these contributions are determined by the domain characteristics. The operationalizing softgoal Repository Service has a MAKE contribution to the NFR softgoal Reliability [Phoenix] and the justification for this contribution is captured by the claim softgoal, "C2 user: repository service provides store-and-forward capability that improves reliability" (here C2 user is one type of system user); this justification gives the rationale for the MAKE contribution. The other three contributions in Figure 4 are BREAK contributions: one between Authorization Service and Security [Messages] with claim softgoal "Limited Authorization", between Channels and Security [Messages] with claim softgoal "Channels do not encrypt messages", and between Information Validator and Security [Messages] with claim softgoal "No authentication or authorization performed". This completes step 3.

In step 4 we can define priorities for NFR softgoals, operationalizing softgoals, claim softgoals, decompositions, and contributions. These priorities depend on the domain requirements. However, for our discussion here we will assume that all elements of the SIG have the same priority.

In step 5 we apply the propagation rules of the NFR Approach to determine the extent of trustworthiness (which is the root NFR softgoal in the SIG) in the Phoenix system. For this purpose we assume[3], based on our current knowledge of the system, all claim softgoals are satisficed. Since all claim softgoals have MAKE contributions, all parent contributions (discussed as part of step 3 above) are satisficed—that is they remain unmodified by the claims. Next we assume, again based on the current knowledge of the domain, that the relevant operationalizing softgoals are satisficed, that is, Repository Service, Authorization Service, Channels, and Information Validator are all satisficed. By propagation rule R6, since all child softgoals of the operationalizing softgoal C&C View [Logical] (the children are Repository Service, Authorization Service, and Channels) are satisficed, the parent C&C View [Logical] is also satisficed. Then by propagation rule R1, four things happen: the satisficed label of Repository Service is propagated as satisficed label to the NFR softgoal Reliability [Phoenix] via the MAKE contribution between them, the satisficed label of Authorization Service is propagated as denied label to Security [Messages] via the BREAK contribution between them, the satisficed label of Channels is propagated as denied label to Security [Messages] via the BREAK contribution between them, and the satisficed label of Information Validator is propagated as denied label to Security [Messages] via the BREAK contribution between them.

Therefore, by propagation rule R4, the NFR softgoal Security [Messages] is denied since only denied labels are propagated to it. Therefore, by R6 the parent NFR softgoal Trustworthiness [Phoenix, Software] is denied since one of its children is denied. By another application of the propagation rule R6 we observe that the topmost NFR softgoal Trustworthiness [Phoenix] is also denied since one of its children is denied—this means that the current design of the Phoenix system is not trustworthy. More importantly, we know why it is untrustworthy since we have the chain of evidence in the SIG: all denied softgoals, decompositions (if any), and contributions indicate the causes for untrustworthiness. Further details of this evaluation may be seen in [11]. Another point to note is that the SIG of Figure 4 was drawn by the StarUML tool with the Softgoal Profile module plugin [8] – this tool automatically applies the propagation rules for a given SIG.

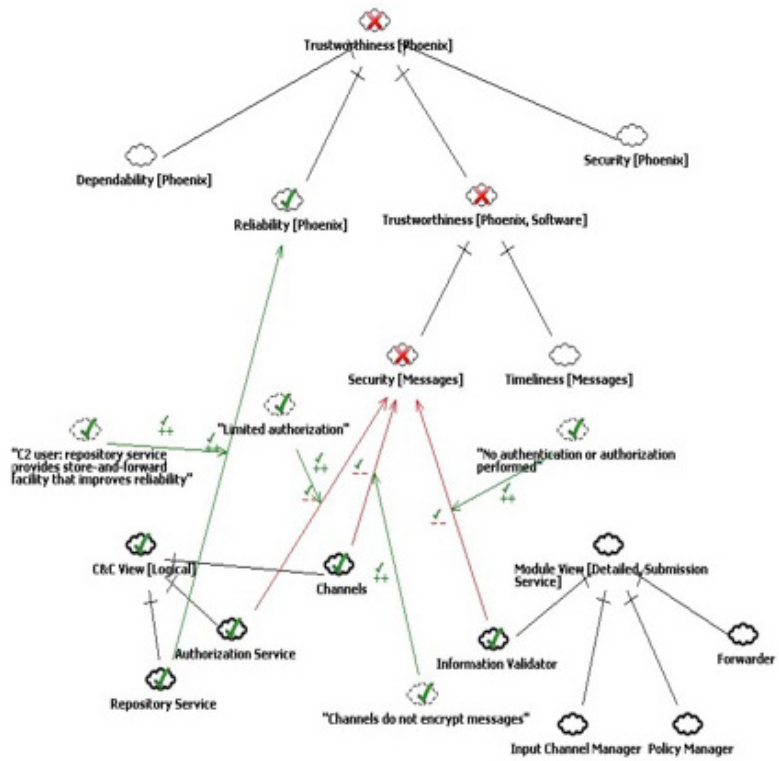

Figure 4. SIG for Evaluating the Architecture of the Phoenix System for Trustworthiness

Trustworthiness Deficit Identification Using the NFR Approach

In order to identify trustworthiness deficit we need only compare the NFR softgoal decompositions for the untrustworthy and trustworthy system. The actual NFR softgoal decomposition for the Phoenix system is shown in Figure 5. The Phoenix system has interoperable protocols, is reliable has high performance architecture, and is extensible – these are represented, respectively, by the NFR softgoals Interoperability [Protocols], Reliability [Phoenix], Performance [Architecture], and Extensible [Phoenix]. The NFR softgoal Extensible [Phoenix] is AND-decomposed into three child NFR softgoals: Scalability [Architecture], Customizability [Phoenix], and Flexibility [Services], which represent, respectively, scalability of architecture, customizability

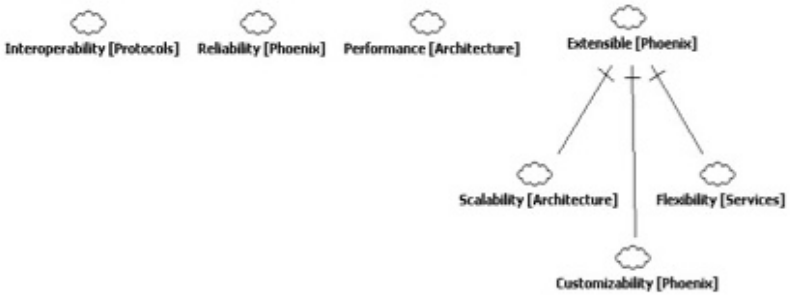

Figure 5. SIG for Non-Functional Requirements for the Legacy Phoenix System

of Phoenix, and flexibility of services. If we draw a SIG similar to Figure 4, we will find that all of these NFR softgoals are satisficed by the current design of the Phoenix system.

We also obtained the trustworthy requirements for the Phoenix system from the stakeholders—these requirements, from the point of view of one stakeholder, are captured by the SIG of Figure 6. As may be observed, this SIG is the same NFR softgoal decomposition as shown in the upper part of Figure 4, and as we know from the discussion in Section 3, the current design of the Phoenix system does not satisfice these NFR softgoals and is therefore untrustworthy as far as this stakeholder is concerned. In this section we will determine the extent of untrustworthiness by developing the Deficit Equation based on the NFR Approach.

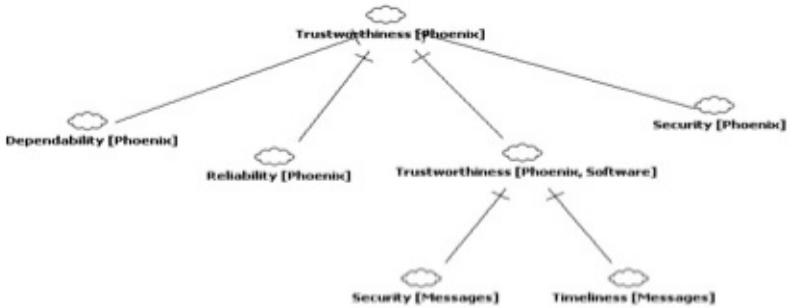

Figure 6. SIG with Trustworthiness Requirements for the Phoenix System from a Stakeholder

These two SIGs (of Figures 5 and 6) may align themselves in three different ways:

1. **Case A: No commonality between the SIGs because they are totally different**
2. **Case B: Some overlap between the SIGs**
3. **Case C: Complete overlap between the SIGs.**

The implications of each of the three possibilities are now discussed. When there is no commonality between the two sets of SIGs as in Case A, this means that the legacy system does not satisfy any trustworthy requirements at all, and has a very high trustworthiness deficit. In Case B, when there is some overlap between the SIGs this means that the legacy system already satisfies some of the trustworthiness requirements, that is, the legacy system is trustworthy to some extent already. Therefore, the trustworthiness deficit in this case is medium, certainly lesser than in Case A. Finally in Case C, when there is a complete overlap between the SIGs, the legacy system already satisfies all trustworthy requirements and the trustworthiness deficit does not exist.

Figure 7 shows the two sets of SIGs for Case A, Figure 8 shows the situation with the SIGs for Case B, and Figure 9 shows the juxtaposition of the SIGs for two scenarios of Case C. The SIGs in Figures 7, 8, and 9, are hypothetical SIGs. SIGs may overlap on individual softgoals or softgoal decompositions. In Figure 8, there is an overlap on two softgoals—that is, these softgoals are common to the legacy system requirements as well as to the trustworthiness requirements. In Figure 9, there is an overlap on softgoal decomposition: in Scenario 1, the overlap is at the root of the SIG, while in Scenario 2, the overlap is at the middle of the SIG. Therefore, the extent of overlap helps identify trustworthiness deficit.

We can quantify this trustworthiness deficit using the following steps:

1. **If no goal overlap occurs, deficit is 100%**
2. **If there is goal overlap, deficit is given by the Deficit Equation, where TS stands for trustworthiness SIG:**

$$deficit = \left(1 - \frac{overlapping\ goals + overlapping\ decompositions}{total\ number\ of\ goals\ in\ TS + total\ number\ of\ decompositions\ in\ TS}\right) x100\%$$

Equation 1. Deficit Equation

3. **If there is complete overlap, deficit is 0.**

Therefore, in Figure 7, there are no overlapping goals and no overlapping decompositions while there are four goals and one decomposition in the trustworthy system; therefore, by the deficit equation,

*deficit = (1 − 0/(4+1))*100 = 100%. (for Figure 7 representing case A)*

Therefore, the deficit is 100% in Figure 7. In Figure 8, there are two overlapping softgoals, no overlapping decompositions, four softgoals and one decomposition in the trustworthy system. Therefore,

*deficit = (1 − 2/5)*100 = 60%. (for Figure 8 representing case B)*

In Figure 9, for scenario 1, there are four overlapping softgoals, one overlapping decomposition; therefore,

*deficit = (1-5/5)*100 = 0%. (for Figure 9, scenario 1, representing case C)*

In Figure 9, for scenario 2, the same situation like scenario 1 is obtained and the deficit is again 0%. That the deficit is 0% for both scenarios of Figure 9 should not be surprising since the original system satisfies all trustworthiness requirements.

Likewise, in the SIG of Figure 6, there are seven softgoals and two decompositions in the trustworthiness SIG. Also, comparing the SIGs of Figure 5 and Figure 6, we find that there is only one softgoal in common, namely, Reliability [Phoenix]. Therefore, the trustworthiness deficit is shown in Equation 2.

$$deficit = \left(1 - \frac{1}{9}\right) * 100 = 89\%$$

Equation 2.

Therefore, the trust deficit in the legacy Phoenix system is relatively high. The missing trustworthiness requirements are given in Table 1. As can be seen in Table 1, six requirements come from softgoals and two from softgoal decompositions. These missing requirements will allow us to identify the environments the legacy software system may be safely used in.

Therefore, the process (or checklist) for identifying trustworthiness deficit using the NFR Approach is as follows:

1. Obtain legacy system requirements; create the SIG
2. Obtain trustworthiness requirements; create the SIG
3. Identify extent of overlap between legacy system requirements SIG and trustworthiness SIG
4. Apply the Deficit Equation to evaluate trustworthiness deficit
5. Identify missing trustworthiness requirements – both from softgoals and decompositions in the trustworthiness SIG.

In the first step obtain the requirements for the legacy system either by reverse engineering or from system documentation, then create the SIG with, if needed, stakeholder involvement. Then obtain the trustworthiness requirements for the re-engineered system and create the SIG, again, if needed, with stakeholder involvement. Then identify the extent of overlap between the two SIGs. Apply the Deficit Equation to identify the extent of the deficit. Then identify the missing trustworthiness requirements in the legacy system from softgoals and decompositions in the trustworthiness SIG.

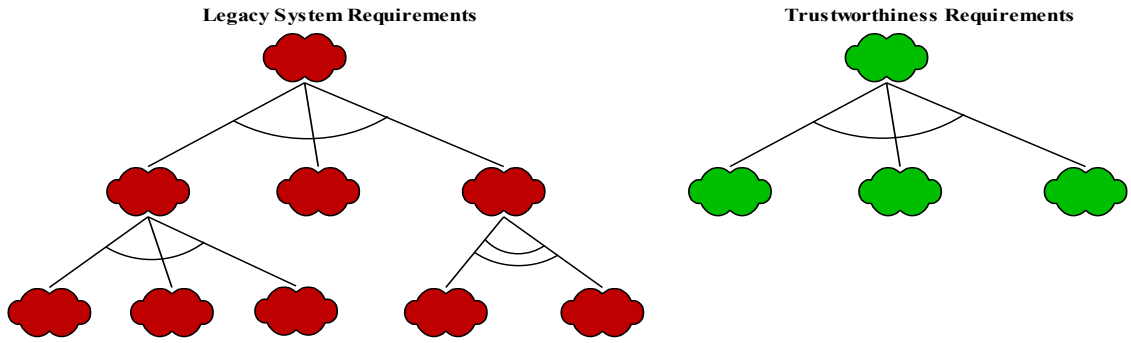

Figure 7. SIGs for Case A: No Commonality, High Deficit

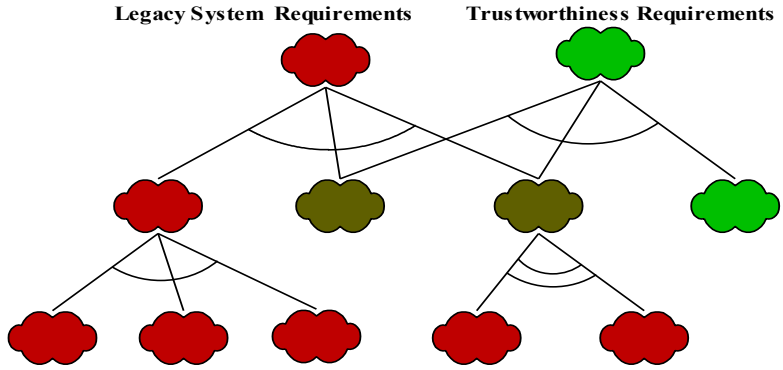

Figure 8. SIGs for Case B: Some Commonality, Medium Deficit

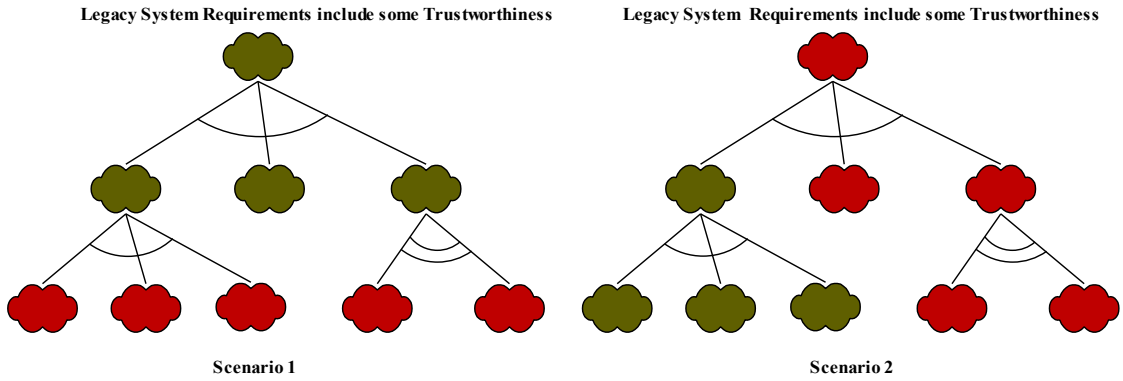

Figure 9. SIGs for Case C: Complete Overlap, Zero Deficit, Two Scenarios

No.	Missing Trustworthiness Requirements	Source
1	Phoenix system should be trustworthy.	Softgoal: Trustworthiness [Phoenix]
2	Phoenix system should be dependable.	Softgoal: Dependability [Phoenix]
3	Phoenix system should be secure.	Softgoal: Security [Phoenix]
4	Phoenix system software should be trustworthy.	Softgoal: Trustworthiness [Phoenix, Software]
5	Phoenix system should send messages securely.	Softgoal: Security [Messages]
6	Phoenix system should send messages in a timely manner.	Softgoal: Timeliness [Messages]
7	Trustworthy Phoenix system should be dependable, reliable, have trustworthy software, and be secure.	Decomposition: Trustworthiness [Phoenix] is AND-decomposed into Dependability [Phoenix], Reliability [Phoenix], Trustworthiness [Phoenix, Software], and Security [Phoenix].
8	Trustworthy Phoenix system software should send messages securely as well as in a timely manner.	Decomposition: Trustworthiness [Phoenix, Software] is AND-decomposed into Security [Messages] and Timeliness [Messages].

Table 1. Missing Trustworthiness Requirements in the Legacy Phoenix System

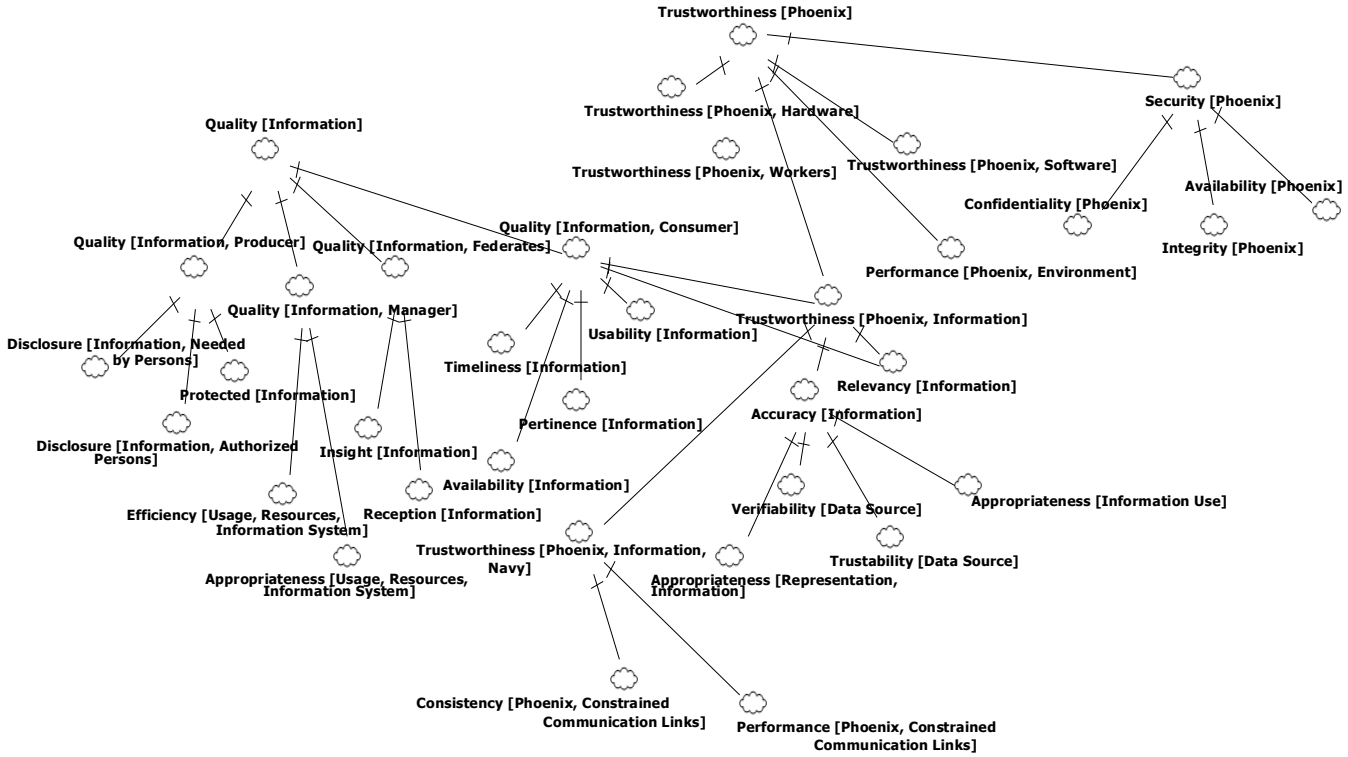

Figure 10. Another Definition of Trustworthiness for the Phoenix System

We mentioned earlier that NFR Approach helps us analyze reasons for poor trustworthiness as well. This analysis proceeds from the SIG of Figure 4 where we see that the main reason for poor trustworthiness is the denial of the NFR softgoal Security [Messages]; this is contributed by three design elements (as discussed in Section 3), which are Authorization Service and Channels of the Component and Connector View and the Information Validator in the Submission Service. Therefore, any improvement in securing messages in all of the three design elements will significantly improve trustworthiness of the Phoenix system. Further details of this analysis may be seen in [11].

It should be noted that the definition of trustworthiness shown in Figure 6 is the view of one stakeholder. Another stakeholder gave the definition of trustworthiness shown in Figure 10, which as can be seen is more complicated. However, the checklist given above can be applied to this definition as well and the trustworthiness deficit can be identified. However, we did not find one single set of attributes that defined trustworthiness acceptable to all stakeholders. As such, NFR Approach provides a process for identifying trustworthiness deficit given any definition of trustworthiness.

Conclusion

Trustworthiness is expected to be an important requirement for software systems in the future. However, not all legacy systems were developed with trustworthiness in mind. It will be helpful if we could systematically identify gaps in trustworthiness in a software system so that the suitability of the software system for use in trustworthy environments may be determined. This is also important to understand the environments where the software system may be used or re-used as well as to determine the requirements that need prioritizing when the software system is being re-engineered. We applied the NFR Approach [5, 6] for this trustworthiness deficit identification since the NFR Approach is useful in dealing with non-functional requirements (NFRs) such as trustworthiness. The NFR Approach considers trustworthiness as a goal to be achieved by the software system and identifies the deficit by determining the extent to which the system falls short of the goal.

In order to develop a process by which NFR Approach may be systematically applied to any software system, we applied it, as a case study, to the Phoenix system. The Phoenix system is a middleware system used by the Air Force with about 100,000 lines of code. We first obtained the current requirements (or legacy requirements) satisfied by the Phoenix system. We then obtained the trustworthiness requirements for the Phoenix system from the stakeholders. Then applying the NFR Approach we determined the trustworthiness deficit in the Phoenix system to be 89% - that is, the system is highly untrustworthy. Based on this case study we believe that the process of the NFR Approach can be applied to any software system to identify its trustworthiness deficit.

For the future we plan to extend the deficit equation to include both hardgoals and softgoals—that is, consider both functional and non-functional requirements [12]. We also plan to apply the NFR Approach to larger systems than Phoenix and confirm that the NFR Approach is scalable to larger systems. We also plan to quantitatively assess trustworthiness in a software system [13] so that changes to design may be motivated by quantitative considerations.

Disclaimer:

Approved for Public Release [88ABW-2013-4662] 07Nov13, Distribution unlimited.

ABOUT THE AUTHORS

Nary (Narayanan) Subramanian is currently an Associate Professor of Computer Science at The University of Texas at Tyler, Tyler, Texas. Dr. Subramanian received his Ph.D. in Computer Science from The University of Texas at Dallas. His specialization is software engineering with particular focus on software architectures and requirements engineering. He co-founded the International Workshop on System/Software Architectures (IWSSA) and served as a co-chair for seven years between 2002 and 2011. He established and directed the Center for Petroleum Security Research at UT Tyler. He has over fifteen years' experience in industry in engineering, sales, and management. He is a member of the IEEE. His research interests include software engineering, system engineering, and security engineering.

Department of Computer Science
University of Texas at Tyler
Tyler, Texas, USA
E-mail: nsubramanian@uttyler.edu
Phone: 903-566-7309

Steven Drager is a principal electronics engineer with the Air Force Research Laboratory Advanced Computing and Communications Division leading research in trusted computing, high performance systems and emerging models and technologies for computation. Mr. Drager has over 20 years at AFRL beginning in reliability physics working wafer-level testing for oxide breakdown, hot carrier degradation and electromigration and then on the development and standardization of the analog and mixed-signal extensions to the VHSIC Hardware Description Language (VHDL-AMS). He has spent the last 10 years leading research in high performance embedded computing architectures, quantum computing architectures and algorithms, and software-intensive systems producibility.

Information Directorate
Air Force Research Lab
Rome, New York, USA
E-mail: Steven.Drager@us.af.mi
Phone: 315-330-2735

William McKeever has been with the Air Force Research Laboratory's Information Directorate, since 2003. He is co-lead of the Trusted Software-intensive Systems research which seeks to develop techniques, methodologies and tools to guarantee trust (as measured by correctness, security, reliability, predictability, and survivability)and migrate the analysis from execution (testing and monitoring) to design (correct and formal/security specifications) and development (composition and auto-generation) to meet DoD System needs. Mr. McKeever received a BS in Computer Science from Plattsburgh State University of New York, and a MS in Computer and Information Science from State University of New York Institute of Technology.

Information Directorate
Air Force Research Lab
Rome, New York, USA
E-mail: William.Mckeever.1@us.af.mil
Phone: 315-330-2897

REFERENCES

1. <http://www.cnsoftware.org/NSS2Report>
2. NIST Trustworthy Information Systems program available at <http://www.nist.gov/itl/tis/>.
3. E. Amoroso, et al "A Process-oriented Methodology for assessing and improving software trustworthiness", Proceedings of the 2nd ACM Conference On Computer and Communication Security, 1994, pp. 39-50.
4. D. L. Parnas et al "Evaluation of safety-critical software", Communications of ACM, Volume 33, Issue 6, 1990, pp. 636-648.
5. L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, Boston, 2000.
6. N. Subramanian and L. Chung, "Software Architecture Adaptability - An NFR Approach", Proceedings of the International Workshop on Principles of Software Evolution (IWPSE 2001), ACM Press, Vienna, September, 2001, ACM Press, pp. 52-61.
7. N. Subramanian, S. Drager, and W. McKeever, "Identifying Trustworthiness Deficit in Legacy Systems Using the NFR Approach", Software Technology Conference, Salt Lake City, Utah, April, 2013.
8. <http://staruml.sourceforge.net/en/modules.php accessed on August 10, 2013>.
9. D. Taibi, "Defining a Open Source Software Trustworthiness Model", Proceedings of 3rd International Doctoral Symposium on Emperical Software Engineering, 2008.
10. Y. Yang, Q. Wang, M. Li, "Process Trustworthiness as a capability indicator for measuring and improving software trustworthiness", ICSP '09 Proceedings of the International Conference on Software Process: Trustworthy Software Development Processes, pp 389-401.
11. N. Subramanian, S. Drager, W. McKeever, "Designing Trustworthy Software Systems using the NFR Approach", Chapter Paper, to appear in Emerging Trends in ICT Security, Edited by Babak Akhgar and Hamid Arabnia, Elsevier Publication, 2014.
12. L. Chung et al., "Goal-Oriented Software Architecting", Relating Software Requirements and Architectures, (Eds.) P. Avgeriou et al., Springer, 2011, pp. 91-109.
13. N. Subramanian, S. Drager, and W. McKeever, "Evaluating Trustworthiness Using the NFR Approach" poster presented at the Cyber and Information Challenges Conference in June 2012, organized by Armed Forces Communications and Electronics Association, in Utica, NY.

NOTES

1. When contributions (MAKE, HELP, HURT, or BREAK) are between claim softgoals and other contributions, then the "objectives" are these other contributions and the "artifacts" are the justifications captured by claim softgoals.
2. The NFR Approach supports any level of realization: strategic level, conceptual level, system level, requirements level, architectural design level, detailed design level, code level, and so on; however, in this paper we considered architectural design models.
3. If this assumption changes at any time we update the SIG to reflect the changes and re-evaluate by applying the propagation rules.