A Holistic Approach to Trustworthy Software

Ian Bryant, SSDRI, UK (ian.bryant@ssdri.org.uk)

Keywords: Safety, Reliability, Availability, Resilience, Security

Abstract

Efforts in improving the overall quality of information and communications technology (ICT) systems have historically tended to cluster into a series of stovepipes, with the two major axes tending to be around Safety and Security. This paper summarises the ongoing UK public-private initiative to produce a consensus framework for trustworthy software which can be applied to all domains.

1 Introduction

Software forms a fundamental part of any information and communications technology (ICT) system, and of many other environments, such as Industrial Control Systems (ICS), which would not necessarily consider themselves to be part of the ICT arena.

The predictable and correct operation of software is therefore a fundamental need for many communities, yet few of these deployment environments have any intrinsic understanding of the need for trustworthy software, nor do they enshrine the types of good practice that will inherently produce trustworthy software.

2 Scope

A challenge to the production of trustworthy software arises from its pervasive nature and consequent difficulties in delineation.

2.1 Software Context

There is an implicit assumption that software is a bounded entity, typically sitting between a physical layer provided by hardware and a conceptual layer provided by the human(s) with which it interfaces. But this approach is overly simplistic, as illustrated by Figure 1.

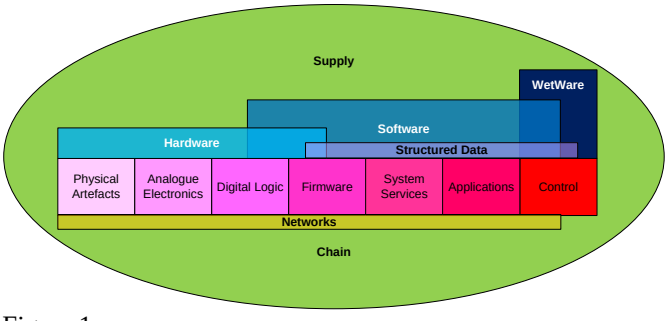

Figure 1

This shows that the software context is blurred in a number of areas, in particular:

- At the boundary with software, with firmware being software implemented in hardware, and much digital logic being hardware designed using software such as VHDL (VHSIC hardware description language),
- At the boundary with "Wetware" (the human operator), with the ever growing number of autonomic systems taking on the control function(s) that have often been assumed to be performed solely by the operator
- In the case of structured data, such as XML, which can alter the behaviour of recipient systems
- In the case of network protocols, which can be regarded as rewriteable definitions, and are in that sense akin to software

2.2 Lifecycle Context

The question of improving software is hardly new, with seminal work being done on foundation of the discipline of Software Engineering over 40 years ago [1].

Yet most discussions on the topic tend to have their context situated by the terminology used, which typically is centred around "Software Development" [2].

Yet the lifecyle of software, and the systems with which it is associated, extends from cradle to grave, as illustrated by various consensus models such as those from ISO/IEC on Software life cycle processes [3] and System life cycle processes [4].

Such lifecycles provide varying degrees of granularity and often differing terminology for the stages to be considered, but at the highest level it is important that understanding of the requirements for trustworthy software be embedded in the differing communities of interest in software and associated systems :

- Those who Specify software and/or Systems
- Those who Realise software and/or Systems, which includes the major sub-communities of Design, Development, Test, and Commissioning
- Those who Use software and/or Systems

2.3 Requirements Context

The requirements for trustworthy software may arise in two major and distinct way, both as a Functional Requirements (FR) which are explicitly evinced by the party or parties

requesting the service which is being provided by software, or as a Non-Functional Requirement (NFR) which are implicit needed but may not be directly specified by the party or parties requesting the service which is being provided by software.

Functional Requirements for trustworthy software are typically encountered in niche markets where there is a strong technical bias in the customer community, such as in the Safety-critical software industry (as exemplified by aviation flight control systems and nuclear power stations) and the Secure software industry (for example in the production of Firewalls).

Non-Functional Requirements, also known as Qualities, Quality Attributes, Quality Goals, or Constraints, will vary between stakeholder communities and with implementation details, but a generic list can be characterized by the mnemonic "PAGICC QUESTASS":

- Performance
- ARM (Availability (incl. Resilience), Reliability (including. Robustness) & Maintainability (including. Documentation))
- Governance (Legal (including Intellectual Property Rights), Regulatory, Policy)
- ILS (Integrated Logistic Support incl. Escrow)
- Compatibility (Platforms and Dependencies)
- Cultural Fit (including Reputation and Brand)
- Quality (e.g. Faults Delivered, Fault Removal Efficacy)
- Usability
- Evolution (including. Extensibility / Scalability, PDS (Post Design Services))
- Standards
- TEA (Training, Education, Awareness)
- Accessibility
- Security (including Data Protection Act (DPA) compliance)
- Safety

Of these Non-Functional Requirements, trustworthy software is typically needed in support of Performance, ARM, Quality, Usability, Evolution, Security and Safety (PAQUESS).

To complete the consideration of requirements, it is also necessary to consider trustworthiness in respect of Non Objective Requirements (NOR) - an emergent term being used to encapsulate consumer preferences typically of form rather than function (which often give rise to usability considerations that can have deleterious effects on trustworthiness) - and Derived Requirements (DR) which are those design, development or configuration decisions not arising directly from any FR / NFR / NOR which nonetheless may impact on trustworthiness.

3 Challenges

The degree on which society is reliant on ICT – and thus software – is growing all the time.

It is difficult to conceive of any major sector of the economy in the developed world which is not dependent – often critically so – on ICT and software. Such dependence extends into private lives, with figures for the UK in October 2011 showing that over 50% of the population now has a "Smartphone" (against a backdrop of 80+ million and growing active mobile phone accounts for a population of about 62 million people).

This dependence of ICT and software can be expected to broaden and deepen in the coming years, with a number of trends already being identifiable to catalyse this dependence and complicate the problem space, including:

- The move to distributed application platforms and services (a.k.a "Cloud")
- The increasing reliance on mobile devices, which typically rely on lightweight operating systems that have less inherent controls than the operating systems on previous generation devices
- A move in business to consumerisation ("Bring-Your-Own-Device" (BYOD)), and the related issue of commoditisation in previously closed architectures, such as industrial control systems (ICS)
- The pressure for ICT consolidation for energy efficiency (the Low Carbon imperative), predominantly relying on software based virtualisation

Furthermore, there are significant changes going on in the way in which software is developed. The historic assumption was that software would be developed under engineering-style "waterfall" model, under single organisational control, but this is now far from the only approach, with factors such as Agile Development and Open Source challenging this paradigm.

It should therefore not be surprising that the impact of software problems is a high cost to the economy: figures from the US Government National Institute of Standards & Technology (NIST) indicate that software flaws and weakness costs ~$60 billion / year to US alone [5], and a 2011 University of Oxford / McKinsey report [6] confirms the trend in studies over many years [7, 8, 9] that software remains the major source of IT project failures.

From a UK perspective, a governmental risk analysis of such factors led to the identification of Cyber-attack and Cyber-deficiencies as one of the 4 top "Tier One" Risks in the 2010 UK National Security Strategy [10].

4 Risk Model

The two main communities within which Software Dependability has been a focus are Safety and Security, which approach the issue in slightly different manners.

Intrinsic to both is a concept of Risk Management, using appropriate countermeasures to reduce the Scalar quantity of "Risk" to an Acceptable Level, and to maintain that Level throughout the system Lifecycle, as illustrated at Figure 2.

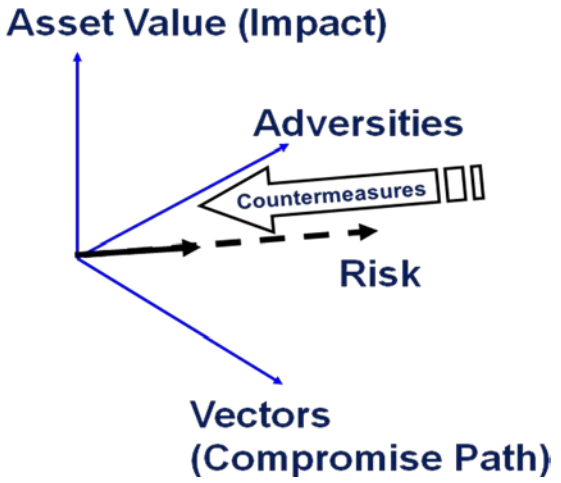

Figure 2

The focus of Security risk management is on reduction of the number of deleterious outcomes against the 3 main information properties of Confidentiality, Integrity and Availability, and it is therefore of value to examine the statistics, again from the UK, as to the number of incidents reported [11], as illustrated at Figure 3.

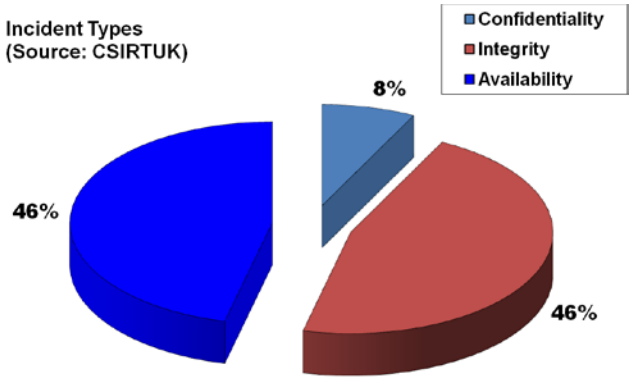

Figure 3

It is interesting to note that the majority of incidents affect Integrity and Availability, which are common concerns with the Safety world.

In fact the distinction between Security and Safety arises from their differing Adversity models, with the Security community seeking to address Threats (directed, deliberate, hostile acts) and the Safety community seeking to address Hazards (undirected events).

This means that the Security world assumes a deterministic Threat model which typically ignore Hazards, and is largely predicated upon characterization of Known classes, if not necessarily details, of Threat Actor therefore has difficulties handling the other elements of the KuU model [12], the Unknown and Unknowable (KuU).

On the other hand the Safety community typically uses Stochastic models to address Hazards, and usually ignores Threat.

If taken in the round, therefore it can be seen that the Adversity modelling techniques are fundamentally blinkered based upon the perceived Functional Requirement they address, and if the Adversity model is artificially constrained, then the Risk management countermeasures (which in the software realm will include technical approaches such as Formal Methods and Static Code Analysis) will be similarly constrained.

Furthermore, by analogy, it is assumed that the treatment of the Non-Functional Requirements of Performance, ARM, Quality, Usability, Evolution, Security and Safety (PAQUESS) will be also suffer from inadvertent blinkering of approaches.

5 Approach

The challenge therefore is to "bake in" delivery of trustworthiness in all software, recognising that implementations may vary with Audiences and Functional / Assurance Requirements. This is analogous the "Public Health" approach in the world of medicine: Prevention now avoids Treatment later.

It is suggested that the optimal approach must be on establishing Pareto ("80:20") techniques to making software better across the board, iteratively using learnings from specialists domains and interpreting them for the common good.

This implies that Specification, Realisation and Use of all software needs to take appropriate cognisance of one or more of the Facets of Trustworthiness:

- Safety - the ability of the system to operate without harmful states
- Reliability - the ability of the system to deliver services as specified
- Availability - the ability of the system to deliver services when requested
- Resilience - the ability of the system to transform, renew, and recover in timely response to events
- Security - the ability of the system to remain protected against accidental or deliberate attacks

These Facets of Trustworthiness are an extension of previous work on Dependability [13], adding Resilience to the previous set of Facets, and amending the definitions to better fit the holistic view.

6 Trustworthy Software Framework

Although a plethora of good practice that can inform the Specification, Realisation and Use trustworthy software has emerged over the 40+ years since the need for good software engineering practices was first identified, adoption thus far has generally been weak.

This weak adoption is a consequence of various factors, with two major challenges standing out.

Firstly, it can be argued [14] that the "*best has become the enemy of the good*", with adherents from such niche communities as do have specialist practices (normally driven by strong Functional Requirements for trustworthy software) often being reluctant to accept that a Pareto implementation of a subset of such practices can still produce significant benefit in other realms.

Secondly, the potential large body of knowledge is – at best – disjointed, with no easy way to either find such information, or navigate around such information to find subset appropriate to the particular need. This is often confounded by differential, and sometimes conflicting, use of terminology.

The key to unlocking these barriers to adoption is felt to be the development and maintenance of a Trustworthy Software Framework (TSF), which aims to:

- Provide a "meta-ontology" as a neutral linkage between various domain specific terminologies, Citations, Methodologies and Information sharing techniques (CMI)
- Provide multiple levels of abstraction to align with the needs of different audiences and outputs - for instance the view of the information required for general awareness will be much less granular than that for post-graduate researchers
- Provide ways to access differing sets of information in a manner appropriate to economic sector, community of interest or risk profile

This TSF is visualised in Figure 4, and is being validated against, a number of domains, including but not limited to:

- Safety
- Security
- Dependability
- Resilience

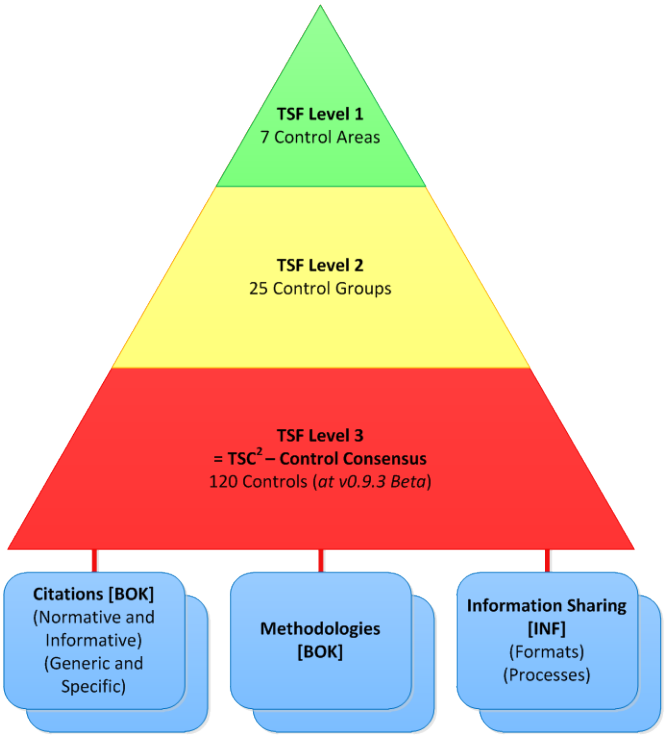

Figure 4

7 SSDRI

It is posited that there would be a significant benefit if the overall software community could be persuaded to take a Pareto approach to improving software trustworthiness across the board, and therefore that some cross-cutting coordination, support and innovation activity is required to achieve such a result.

Such an activity needs to be applicable across all sectors of economic activity, both public and private, and to be effective should recognize the challenges implicit in globalization, although there should be measurable benefits from national level initiatives.

In the UK this recognition led to the creation of the Software Security, Dependability and Resilience Initiative (SSDRI) in July 2011 as a public-private partnership, established to:

> *"enhance the overall software and systems culture, with the objective that all software should become designed, implemented and maintained in a secure, dependable and resilient manner".*

SSDRI is genuinely cross-sectoral, being governed by a Steering Committee drawn from the Demand-side (in both public and private sectors), the Supply-side, and those producing the Corpus of knowledge, and is operated by the new Cyber Security Centre (CSC) at De Montfort University (DMU)

7.1. Environmental Shaping

The primary challenge in what is seemingly a technical discipline – software trustworthiness – is actually a non-technical issue: to make Stakeholders, in particular senior decision makers, realize the potential risks that are being exposed by the currently poor overall state of software, and the attractions of improving the baseline of trustworthy software across the board, in addition to the sort of niche activities (such as Formal Methods and Static Code Analysis) required for specialist communities like Safety and Security.

7.2. Conceptual Evolution

Although many of the concepts required for software trustworthiness have long been established, there is still a need for Conceptual:

Composability and Traceability represents a major challenge, as most software is an assemblage of subordinate component. This produces a layer model, as illustrated at Figure 5.

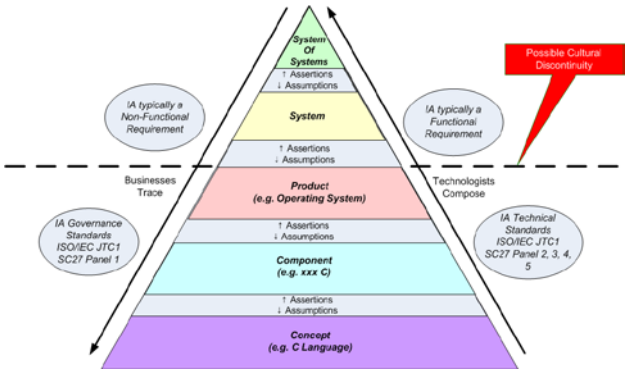

Figure 5

Between each of these layers there are implicating transitional flows of information about Assertions (↑) and Assumptions (↓), which can be Positive (+ve) and/or Negative (-ve), yet no good modelling is currently available for this composition and tracing challenge.

Linked to the subject of composition is that of understanding the – potentially globalised- Supply Chain, with Cloud Computing presenting a Disruptive Challenge

Finally, to aid the composition of software a catalogue of generic Design and Effects Patterns (i.e. to cover both Functional and Non-Functional concerns) would significantly aid the Training of the current workforce.

7.3. Practice Improvement

In "mature" industries (e.g. Aviation Engineering), all practitioners intrinsically accept responsibility for producing quality output.

The challenge is therefore to embed software trustworthiness practices at all levels, so it becomes "part of the Culture":

- Training of current workforce
- Education of future workforce
- Awareness of all specifiers, producers and consumers

This TEA activity (Training, Education and Awareness) needs to take a Pareto approach to improving the baseline of software trustworthiness across the board, with any community specific needs (e.g. Safety and Security) addressed as extensions to this baseline.

7.4. Independent Verification

For market segments where a degree of assurance as to software trustworthiness is desirable, independent validation is a preferred technique, yet this is only currently adopted in niche communities such as Safety and Security, and is typically targeted as High Assurance needs.

An aspiration has been identified for:

- A widely applicable independent Black Box testing approach for "Due Diligence" needs to address Mass Market software either with or without specific Functional Requirements for software trustworthiness
- "Maturity Model(s)" for assurance of software trustworthiness in the Supply Chain

7.5. International Collaboration

Although there should be measurable benefits from National level initiatives, to genuinely software trustworthiness this needs to recognize the challenges implicit in globalization of the Supply Chain.

7.6. Standards Contribution

Noting Henry Ford's maxim that "*Standardization can be thought of as the best that you know today, but which is to be improved tomorrow*", it is highly desirable for all learnings from software trustworthiness to be formalised through a widely recognized Standards Development Organisations (SDO), such as ISO/IEC, ITU-T and ETSI.

8 Conclusions

The historic focus on trustworthy software has typically been held within niche communities such as Safety and Security, yet software is so pervasive across all sectors of economic activity that such a stovepiped approach can no longer be regarded as acceptable.

The Trustworthy Software Framework (TSF), and the Software Security, Dependability and Resilience Initiative (SSDRI) which aims to support and evangelise TSF, is therefore a vital approach to rectifying the weak adoption of good software engineering practices, recognising that despite 40+ years having elapsed since a need was first identified, adoption thus far has generally been weak.

Acknowledgements

The support of the UK National Cyber Security Programme (NCSP) for underwriting the funding of SSDRI and in the development of TSF is gratefully acknowledge, as is the support of numerous UK stakeholders across all sectors of economic activity.

References

[1] P Naur, B Randell. “Report on a conference on Software Engineering”, NATO Science Committee, January 1969

[2] J R Harrison, W Whyte. “Secure Software Development Failures: who should correct them and how”, TSB Cybersecurity KTN, June 2008

[3] ISO/IEC 12207:2008 Systems and software engineering -- Software life cycle processes

[4] ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes

[5] "The Economic Impacts of Inadequate Infrastructure for Software Testing", NIST, May 2002

[6] B Flyvbjerg, A Budzier. "Why Your IT Project May Be Riskier Than You Think", Harvard Business Review, September 2011, pp. 601-603

[7] B. Boehm, “Software Engineering Economics”, Prentice Hall, 1981

[8] T. DeMarco, “Controlling Software Projects”, Prentice Hall, 1982.

[9] “Chaos” Technical Report, Standish Group International, 1994 et seq.

[10] "A Strong Britain in an Age of Uncertainty: The National Security Strategy", Cabinet Office Cmd7953, October 2010

[11] UK Computer Security Incident Response Team (CSIRT-UK), retrieved 2010-05-20

[12] R E Gomory,”The known, the unknown and the unknowable”, Sci. Amer. 272 (1995), 120.

[13] I Sommerville, “Software Engineering”, 9th Edition, Addison-Wesley 2011

[14] Lt Gen Sir E F G Burton KBE, address to SSDRI Stakeholder Forum, May 2012