Quantifying Security:

How to specify security requirements in a quantified way

to express the richness of security needs

and to

balance security investment with other system quality needs.

Iver Holtersvei 2, NO-1410 Kolbotn, Norway, Tom@Gilb.com, www.Gilb.com, +47 66801697

Abstract:

• The systemic advantages of quantifying the security problem
• Basic planning language for quantified security specification
• A generic model of security (Integrity, Security and Attack)
• Examples of tailored real security specifications
• Advanced security specification with information about risks, issues, dependencies and priorities
• Security design specification: how to estimate the security impacts and costs
• Evaluating security design alternatives quantitatively using impact estimation tables
• Evolutionary delivery for security capability: early security

priorities first, learn from feedback and experience.

Introduction.

- Security is a system quality. It can be dealt with in the same way that all other critical system qualities need to be dealt with – quantitatively, and systematically. We suggest that the planning language, Planguage, as defined in Competitive Engineering [Gilb05] is a strong and innovative framework for dealing with security in a systematic way. In fact even those who are not just interested in security, but in the larger set of system qualities, may be interested in this paper as an example of what one can do with other qualities.
- The theme of this paper is summarized by the following:
    - Security is a complex quality: that means it needs to be defined by a set of measures, not a single measure.
    - A single elementary measure of quality will need to be applied to a wide variety of conditions regarding when, where and for which events, in order to be made intelligible for various levels of analytical benchmarks, for constraint levels and target levels of security.
    - The security designs, to meet security requirement levels, must all be evaluated by both quantitative estimate and direct measure, to see if they help meet the security target levels. In addition they must not harm other performance or quality target levels, other constraints, and they must fit within the resource budgets.

The systemic advantages of quantifying the security problem:

- We get clarity regarding our requirements
- We get *real* agreement, not different interpretations of the requirement
- We can contract for results

- We can prioritize security as effectively as other quantified attributes like performance or reliability
- We can more logically evaluate all designs, strategies and architectures that are supposed to help us reach our security requirements.
-

- Basic planning language for quantified security specification.
  - 'Planguage' is a planning language that is designed to allow us to express and deal with quantified qualities of all kinds, including security.
  - Here is a real (doctored) example of quantifying a security requirement on a project I advised. This is not the place to explain all the parameters, but they should be reasonably self-explanatory. Mainly we give you an immediate impression of Planguage.
    - **"VERY TOP LEVEL PROJECT GOALS**
    -
    - <u>**Security Administration Compliance:**</u>
    - **Ambition**: to become compliant and to remain continuously compliant with all current officially binding security administration requirements both from CORP X and Regulatory Authorities.
    - **Scope**: Account Opening and Entitlement Reporting.
    - **Scale**: % compliant with CORP X Information Security Standards (CISS) [CORP X Information Security Office (CISO)] on a defined System or Process.
    - *Note: CISS is an officially binding security administration requirement with which we must become compliant.*
    -
    - **========= Benchmarks=================**
    - **Past** [CISS = RSA and IT DIVISION ISAG Compliance Matrix [Regional Security Administration and IT DIVISION Independent Security Administration Group, October 2003] 25% <- JC, Nov-03

- *Note: The RSA/IT DIVISION Compliance Matrix originates from Otto CXXX and is based on CISS.*
-
- **========= Targets ====================**
- **Wish** [Deadline = March 2004, Systems = High Criticality Systems] 100%
- **Wish** [Deadline = June 2004, Systems = {Medium & Low} Criticality Systems] 100%
- *Note: Wishes are stakeholder valued levels that we are not yet sure we can deliver in practice, on time, so we are not promising anything yet, just acknowledging the desire.*
-
- **Goal** [Deadline = March 2004, Systems = High Criticality Systems] 90%±5%
- **Goal** [Deadline = June 2004, Systems = {Medium & Low} Criticality Systems] 90%±5%
- **Goal** [Midline = February 2004] **50%±10%** "intermediary goal short of 100%"
- *Note: Goal levels are what we think we can really promise and focus on. These types of goals push us into thinking about possible Evolutionary result delivery steps.*
-
- **Stretch** [Deadline = March 2004, Systems = High Criticality Systems] 95%±5%
- **Stretch** [Deadline = June 2004, Systems = {Medium & Low} Criticality Systems] 95%±5%
- *Note: Stretch levels are something that we might be able to achieve if we have sufficient resources, focus and technology available, but we are not sure of that yet. We are NOT promising it now! So this is a way to hold the ideals up in case those things become available."*

• It should be obvious that we can express a wide range of things using Planguage. We can, and the above example is in no way complete in terms of the things related to a single security

• A generic model of security (Integrity, Security and Attack) in the form of a Planguage specification.

- **Integrity**: '*The ability of the system to survive attack*'
    - Gist: Integrity is a measure of the confidence that the system has suffered no harm: its security has not been breached and, its use has resulted in no 'corruption' or impairment to it.
    - *Note: An attack on the Integrity of a system can be accidental or intentional.*
    - *Note: The Integrity of a system depends on the frequency of threat to it and the effectiveness of its security.*
    - Type: Elementary Quality Requirement.
    - Scale: Probability for a defined [System] to achieve defined [Coping Action] when confronted with a defined [Attack] using defined [Security] measures, under defined [Conditions].
        - Coping Action: defined as: {Detect, Prevent, Capture, Thwart, Recover}.
    - *Note: here is an example of specifying a requirement using the defined scale above.*
    - Goal [System = Our Product, Coping Action = Detect Attack, Attack = In House Amateur Hacker, Security = Microsoft Package, Conditions = Firewall Breached] 99%.
    -
- In this model the term security is the *means* to achieve a resulting system integrity, after being subject to a defined attack.
- There is a numeric relationship between these 3 aspects of security, as I have specified in earlier books [Software Metrics 1976, Principles of Software Engineering Management, 1988):
- *Integrity is arithmetically related to threat frequency and security effectiveness*
-
- The numeric relationship between these three ideas; integrity is a function of the threat potential and the security strength.

Knowing two of these factors allows you to calculate the third. But to be realistic you must work it out on a threat (or attack) type by type basis. The combined set of type calculations gives you total integrity, security or threat potential for the system.

-
- A simple formula is:
- INTEGRITY = Sum of all instances of [1 - Threat x (1 - Security)].
- Or more simply:
  - The Integrity level of a system depends on the degree of threat and the security design's ability to cope with that class of threat.
- This is similar in principle to Ohm's Law and to the well known relationship that Availability is a function of Reliability and Maintainability. In all 3 cases (Integrity, Ohm's Law, Availability relationship), knowing or assuming 2 of the 3 factors allows us to calculate the third.
- So, for example, if planned Integrity is maximum one failure per time period, and there are 100 expected or assumed attacks on the system in a given timeframe, then the effectiveness of the security device must be at least 99%.
- This relationship is in principle the same as the Availability (read Integrity as a class of availability), Reliability (read Attack as a class of things that cause the system to fail) and Maintainability (which is like the Security design that prevents to attack and leads to a level of integrity.

<u>Here are some more detailed security quantification aspects.</u>

- • Examples of tailored real security specifications.
  - Here is an example of how to use the above model to express a security requirement:
  - Integrity Example
    - **<u>Integrity</u>**:
      - Type: Elementary Quality Requirement.

        - Scale: Probability for a defined [System] to achieve defined [Coping Action] when confronted with a defined [Attack] using defined [Security] measures, under defined [Conditions].
        - Meter: test one or more Security measure designs for all defined Coping Actions, and all defined Attack(s), under all defined Conditions.
        - Goal [System = Survey Database using Conformit software,
            - Coping Action = Detect,
            - Attack = Professional Top Class Hacker, Security = Complete Security Architecture [Version 1.0],
            - Conditions = {No Advance Warning, Inside Mainframe Building, All Electronic Specs Available to Hacker}] 50%
    - Direct Security Statement using Parameterized scale of measure
        - As I was writing this paper I helped a course participant by working out this example:
        - **Security:**
        - Stakeholders: NSM
        -
        - Scale: % probability the a defined [Assailant] does NOT succeed in a defined [Compromise] for defined [Data] under defined [Conditions].
        -
        - Meter [for Supplier of Security System payment] Use a professional Norwegian hacker. Give them up to 100 break-in attempts.
            - *Note [Meter] If 1 or more of these is successful, then payment is not due the security suppliers, since the assumption is that it cannot be a better than 99.00% system. If great accuracy is desired*

increase number of hacks, and make sure they are representative of the best, by using at least 10 per 1000 attempts by professional hackers.

        -
        - Goal [Assailant = Professional Norwegian Hacker, Compromise = Detailed Knowledge, Data = Norwegian Government Budget, Conditions = Before Secrecy Lifted] 99.90 %
        -

- • Extended background security specification with information about risks, issues, dependencies and priorities, using the Planguage specification language.
    - **<u>Integrity</u>**:
        - Type: Elementary Quality Requirement.
        - Scale: Probability for a …. *as above examples in detail*
        - *Goal […. as above examples in detail*] 50% <- TG
        - *Source: NASA Security Procedures 2004*
        - Rationale: Deterrence of Professional Hackers
        - Authority: Congressional Budget for NASA
        - Issues:
            - I1: will the guideline level change in this years unpublished budget?
            - I2: does this impact NASA business outside the USA?
        - Dependencies
            - D1: Federal Penalties for Hacking.
        - Risks
            - R1: the proposed security technology does not work at the levels estimated
            - R2: improved hacking paradigms, beyond currently know state of the art.
            -
    - The point is that a wide variety of considerations that are related to the quantification can be noted in an integrated

    way with the quantified requirements. This increases the probability that as risks, dependencies, issues and the like are resolved or evaluated in reviews and risk analysis, then we can better see potential consequences on numeric specifications.

- The specification and planning language (Planguage) allows a variety of additional statements of interesting quantified levels on the defined requirement quantification scale. Hopefully this is relatively self-explanatory.
    - <u>**Integrity**</u>:
        - Type: Elementary Quality Requirement.
        - Scale: Probability for a *…. as above example in detail*
        - Meter: test one or *…. as above example in detail*
        - Benchmarks -------------------------- reference levels
        - Past [2004, …..] : 15%
        - Record [Lab Tests]: 99%
        - Trend [Next Year]: 60% +
        - Constraints --------------------------- minimum levels
        - Fail 30%
        - Survival   20%
        - Targets ------------------------------ levels to aim at
        - Wish   80% +
        - *Goal […. as above example in detail*] 50%
        - *Stretch   55%*
        - Impacts ----(if we reach the Goal level, what happens?)
        - Primary Impact: Legal Certification
        - Secondary Impact: Insurance Costs
    - Details of the meaning of these statements will be found in Competitive Engineering [Gilb05].
- ●
- Security design specification: how to estimate the security impacts and costs.
    - **Example of a Real Design Specification**
    -

- **Tag**: OPP Integration.
- **Type**: Design Idea [Architectural].
- ============ Basic Information =========================
- **Version:** 0.1
- **Status:** Draft
- **Quality Level:** not measured
- **Spec Owner:** Andy
- **Domain Expert:** Tom
- **Requirement Authority:** none
- **Source**: System Specification Volume 1 Version 1.1, SIG, February 4 - Precise reference <to be supplied by Andy>.
- **Gist**: The X-999 would integrate both 'Push Server' and 'Push Client' roles of the Object Push Profile (OPP).
- **Description**: Defined X-999 software acts in accordance with the <specification> defined for both the Push Server and Push Client roles of the Object Push Profile (OPP).
- Only when official certification is actually and correctly granted; has the {developer or supplier or any real integrator, whoever it really is doing the integration} completed their task correctly.
- This includes correct proven interface to any other related modules specified in the specification.
- **Stakeholders**: Phonebook, Scheduler, Testers, <Product Architect>, Product Planner, Software Engineers, User Interface Designer, Project Team Leader, Company engineers, Developers from other Company product departments which we interface with, the supplier of the TTT, CC. "Other than Owner and Expert. The people we are writing this particular requirement for."
- ============= Design Relationships =========
- **Reuse of Other Design: none**
- **Reuse of This Design:**
- **Design Constraints:**
- **Sub-Designs:**

- == Impacts Relationships ==(*note: <u>most</u> impacts have not yet been estimated*)
- **Impacts [Functions]:**
- **Impacts [Intended]:** <u>Interoperability</u>.
    - Interoperability: **Scale:** Probability that this device can exchange information with any other defined [Device] produced by this project. **Estimate**: <100% .
- **Impacts [Side Effects]:**
- **Impacts [Costs]:**
- **Impacts [Other Designs]:**
- ============== Priority and Risk Management =============
- **Rationale:**
- **Value:**
- **Assumptions**:
    - A1: There are some performance requirements within our certification process regarding probability of connection and transmission etc. that we do not remember <-TG.
- **Dependencies:**
- **Risks**:
    - R1: We do not 'understand' fully (because we don't have information to hand here) our certification requirements, so we risk that our design will fail certification <-TG.
- **Priority:**
- **Issues:**
-
- Above is a real (doctored!) example of a design specification using a version of the Design Specification Template (CE 7.9, Gilb05). Not all parameters are filled out yet. Notice that even the parameters, which are <u>not</u> filled out (like Impacts [Side effects] and Issues), are asking important questions about the design - and hinting that responsible designers should answer such questions! Detailed explanation of the parameters above will be found in Gilb05. The main point is

to give an impression of how to structure a specification of a design, to discuss quantification of the design impacts.

    - The main drift, of the many parameters above, is to try to get a basis for estimations of the impacts of a design, on a set of requirements. In this case there is clear documentation that estimates have *not* been made, and the design is on shaky ground at this stage. We do *not,* in other words, know what we are doing. One step, in such a case, might be to implement the design in practice, as a short evolutionary step, and measure the effects. This could be thought of as a 'feasibility study' method.
    -
    - The Impact Estimation table, discussed below, is an extension of the impact specification above. But the table can be applied to many designs at once. We can get an overview of alternative designs, and an overview of a set of complementary designs using the table.

-
- • Evaluating security design alternatives quantitatively using impact estimation tables.
    - Here is a real example (related to some of the specifications given earlier in this paper) of an analysis of how *all* the designs are expected to impact *all* the requirements, quantitatively.
        - **"IMPACT ESTIMATION TABLE**
        - *Notes:*
        - *The table below shows the estimated impacts of each of our top level strategies on our top level goals*
        - *The % estimated impact of a strategy is on a scale where <u>100% means</u> the strategy brings us to the stated goal level on time and <u>0% means</u> there is no impact. The estimated impact ought to be based on a credible benchmark, such as a previous system state, or the view of a qualified commentator*

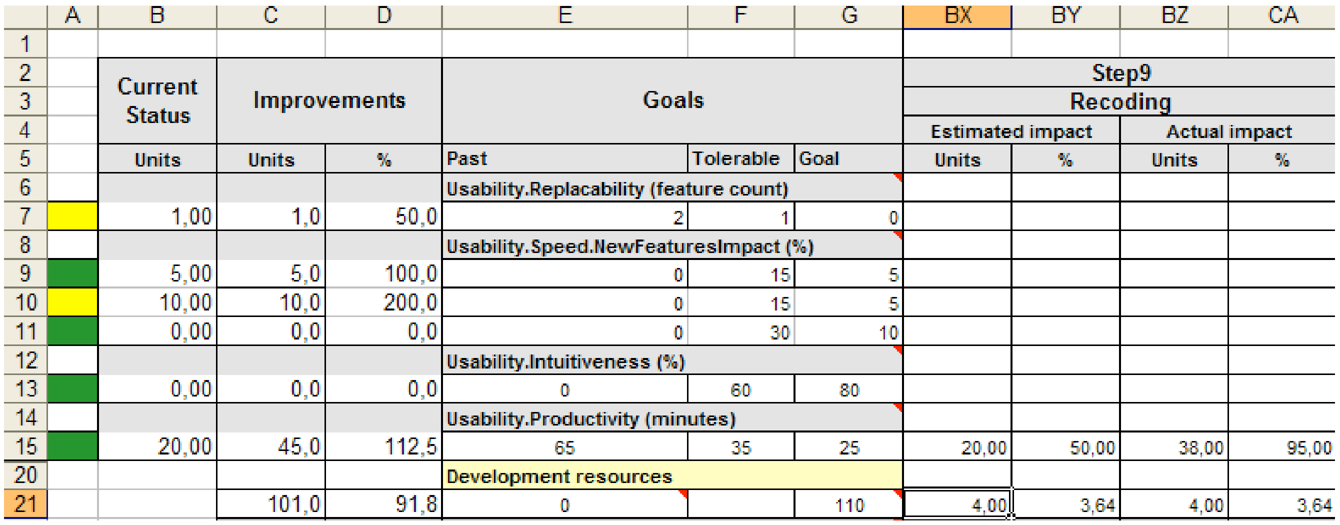
- *Total % impact indicates which of our strategies brings us most benefit, in terms of achieving all of our defined goals*
- *Evidence is 'the source of the facts' used to calculate the design impact estimate. This source can be a person of authority in the matter, or a document for example*
- *Cost is the money amount that is known, or estimated, for implementation of the strategy. The degree to which the cost estimate is certain, is reflected in the impact estimate credibility rating*
- *Credibility is a rating between 0.0 and 1.0, of the quality of the basis for the estimate, where credibility = 1.0 means that the basis of the estimate is regarded to be 100% reliable and credibility = 0.0 means the basis of the estimate is completely unreliable. The credibility rating is used as a multiplier – to modify estimates in the direction of a more-probable number.*
-

Strategies / Goals	Identify Binding Compliance Requirements Strategy	System Control Strategy	System Implementation Strategy	Find Services That Meet Our Goals Strategy	Use The Lowest Cost Provider Strategy
Security Administration Compliance 25% ➔ 90%	100%	100%	100%	50%	0%
Security Administration Performance 24 hrs ➔ 4 hrs	75%	100%	100%	100%	0%
Security Administration Availability 10 hrs -> 24 hrs	0%	0%	0%	100%	0%
Security Administration Cost 100% ➔ 60%	50%	100%	100%	100%	100%
Total Percentage Impact	225%	300%	300%	350%	100%
Evidence	ISAG Gap Analysis Oct-03	John Cxxx	John Cxxx	John Cxxx	John Cxxx
Cost to Implement Strategy	15 effort days (US$ 5,550)	15 effort days (US$ 5,550)	15 effort days (US$ 5,550)	15 effort days (US$ 5,550)	1 effort day (US$ 1,110)
Credibility	0.9	0.6	0.6	0.75	0.9
Cost-Adjusted Percentage Impact	202.5%	180%	180%	262.5%	90%

- • Evolutionary delivery for security capability: early security priorities first, learn from feedback and experience.
    - One practical approach to getting some quantitative data about the proposed security designs is to implement them as a sequence of evolutionary deliveries on real systems, perhaps initially in limited scope field trail pilot modes.
    - This would give an opportunity of testing the security designs in a fairly realistic environment, and at least simulating qualified expert attacks on the system, to measure how effective the security designs actually were in practice. If the

security designs were not as effective as estimated they can then be adjusted or replaced until effective enough security designs are found and measured; before scaling up to more users of the system. The impact estimation table above has been used effectively to estimate expected quantified impacts, then to capture the measured feedback of a series of evolutionary steps and thus control and track progress towards numeric objectives.

- The structure of such an evolutionary project tracking mechanism looks like this, although this one was tracking usability characteristics, it can equally well be used to track any quantified quality such as any type of security. [Johansen04]



	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7	🟨	1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9	🟩	5,00	5,0	100,0	0	15	5				
10	🟨	10,00	10,0	200,0	0	15	5				
11	🟩	0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13	🟩	0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15	🟩	20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

- *Figure: the real use of an impact estimation table to track quantified quality progress. In this case 4 different quality objectives are being tracked for this industrial product (Confirmit web survey software). In this weekly evo step (9th increment), a set of proposed designs called 'Recoding' are implemented to try to move 'Usability.Productivity' towards its goal of 25 minutes, from a starting value of 65 minutes. It was estimated that this step would move us 50% of the way to the goal; in other words 20 minute of the 40 minute distance. As good luck would have it, the actual progress based on measurement (at Microsoft Labs in Redmond, a trial* *customer) was 38 minutes reduction (or 95% of the way to the goal. The development resources were time boxed and were as estimated 4 days of effort.*

Summary – Quantifying Security

- There is a basic quantified security model that relates Integrity, Attacks and Security.
- All these concepts can be specified, as assumptions or requirements using defined scales of measure and defined levels on the scales of measure
- All of these concepts can be measured in practice, at least in a test environment, so that we can get feedback on the effectiveness of any give security design.
- The expected effect of a security design can be estimated based on previous experience, or on current experience in an evolutionary step pilot environment. This will allow us to tune our security design so that it at least performs at the levels required for the Integrity of the system, under assumed loads of attack.
- The effectiveness of security design can be seen in relation to the development costs and operational costs, so that cost effective security designs can be selected. An impact estimation table can be used for this purpose.
- Any presentation of security technology can and should at least contain our best estimate of effectiveness against various relevant attack types, and costs, to aid our initial engineering judgement about which security technology to initially select and pilot (to verify the expectations set by the security technology specification).
- The use of quantified thinking about security must allow us to do better engineering reasoning about security than the non-quantified situation.