Decision Rationale:

A Quantified Decision-Making Basis Using Planguage

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

Abstract:

- Decision rationale are widely discussed in the literature for design decisions [example Burge]. To a far lesser degree for requirements decisions [example Ramesh95]. And practically not at all for justification of Evolutionary project steps or iterative cycle selection [exceptions see Evo in Larman03].
- It is my contention that all software engineering, systems engineering, and management planning specifications can benefit from a variety of forms of rationale. Even specification types not mentioned above, such as source code and test plans can benefit.
- At one extreme all plan specifications, and even source code and test cases, can all be viewed as types of 'design'. So what applies to any type of design applies to them; even though they be, from another viewpoint, classified as something else.

Introduction

A major reason why there, according to the literature, seem to be problems with *design* rationale, in my view, is that:

• the requirements, particularly *quality* requirements, are not clearly and quantitatively stated [Burge is a typical example]

• the complete set of requirements, which all impact a design, are not stated clearly, completely, and altogether, for the designer (function, constraints, quality, performance, costs).

• therefore these necessary design rationale (the *requirements*) are not available for explicit use by the designer, nor are they available as reference for the selected design options.

- This paper will attempt to provide some fresh thinking about the specification rationale methods, compared to the available

literature. The current literature seems to universally ignore fundamental problems such as the quantification of quality requirements, and the quantified estimation of design impacts.
- The deeper basis for this paper is found in his book 'Competitive Engineering' ('CE') [Gilb05CE]. My viewpoint is based on extensive international consultancy practice and industrial teaching, rather than an academic point of view.

I will start with a set of principles, or, if you like position statements. 10 Principles of Rationale Specification.

- **Change Risk**: If you do not give a written rationale, then you risk violation of your specification.
- **Rationale Critique**: If you state your rationale, then others can challenge it, and help you to see if it is false or risky. [SEI97]
- **Insurance**:The cost of capturing a rationale in specifications is far less than the cost of dangers if you do not.
- **Review Basis**: Rationale specification is necessary for helping reviewers to review
- **Traceabilty**: Rationale Specification helps you realize that specification changes might be necessary when the rationale itself changes.
- **Thought Provoker**: Rationale specification makes you think twice about the specification itself.
- **Priority Info**: Rationale specification is a contribution to understanding the relative *priority* of the specification [Gilb-Maier05]
- **Risk Info**: Rationale specification is a contribution the understanding the *risks* associated with a given specification [Gilb02].
- **Validity Check**: Specifying the source of a rationale enables us to check *correctness* and to respect its *priority*.
- **Process Improvement**: Capturing rationale lays a foundation for analysis of decision-making and improvement of the decision-making process.

Planguage Introduction

- I have developed a specification language, called 'Planguage' [Gilb05CE, and Gilb88POSEM, Gilb76SM]. Planguage' contains a rich variety of direct and indirect 'rationale' specification language devices, and other method artefacts, such as rules, process definition, templates, concept glossary and principles. This paper will introduce some of the Planguage rationale artefacts, and leave others for readers to access in the more-detailed sources.
- **Classes of Rationale Artefacts in Planguage:**

  Specification Parameters

  Explicit Rationale Specification.
  - **Rationale**: is an explicit parameter that can be directly attached to any other specification, in order to explain the reason for the specification.
  - Example:
    - **PGB: Goal [UK]: 99.9% <- Annual Plan.**
    - **Authority: Board of Directors, Jan 25th.**
    - **Rationale [PGB]: Competition in UK prior to new EU Laws about competition.**
    - **Basis: Our long-range plan to be the <biggest> in all European countries.**
    -
    - *In the example above, the PGB tag is inserted to show how to tie any Rationale statement to another specific statement or statements. This format can be used irrespective of where you specify the Rationale statement. It does not have to be just below or in the immediate vicinity. The Authority and Basis statements are implied to be related, because they are just below the PGB statement.*
    -
    - *Note 'Basis' is quite different from Rationale. Rationale is a set of conditions leading to a desire to make a specification. It explains how we got to that specification. Basis is a specified set of assumptions that underlie a specification. If the basis conditions are changed, then the specification may no longer be valid.*
  -
- ***Implicit* Rationale Specification.**

- Planguage contains a number of other parameters that can be applied in connection with any type of specification, that give some information regarding the rationale – at least by implication. For example:
- **Source** (sometime written as '<-' , the source arrow).
    - A source specification can be a person, instance or document. It implies that the sourced specification is justified in or by that source, and that the authority level of the source is a degree of justification and priority.
    - Example:
        - **D1: Architecture Standard 1207 <- Our Chief Architect**
        - *The implication is that the rationale for this specification is to be found in a decision by the Chief Architect. If you want to know their rationale, ask them or refer to their specification.*
- **Authority**

  This is a specification of the authority for another specification. It is similar to 'Source', except that a source itself does not necessarily have authority or power. An 'Authority; specification is a direct reference to the power or authority level of a specification that is one form of rationale for the spec.

  Example:

  Goal [Next Year]: 60% ← Marketing Report [February, This Year].

  Authority: Marketing Director [Tim].

  Notice that the source (<-) is also given, and is not identical to the Authority for the 60% goal specification.
- **Depends On: and similar 'dependency' specifications.**
    - Planguage has a large set of parameters and devices for indicating dependency of a specification. Some of these devices are explicit parameters like 'Depends On'. Other devices are the use of a qualifier statement, where the

validity of the statement depends on the truth of a set of qualifier conditions.

- In both cases, all notion of dependencies, are a type of rationale. They help explain why that statement is there. They explain explicitly the conditions that would make the specification invalid.
- For example (explicit dependency)
    - **Goal: 90%. Depends on Market Volatility.**
    - **Goal: 90%. Assumption: Market Stability.**
    - **Goal: 90%. Impacts: Market Profitability.**
    - **Goal: 90%. Risk: Competition Increases.**
    - **Goal: 90%. Dependency: Stage 1 completed.**
- Obviously not all of these are a direct *justification* for the specification level or existence. But they all serve the same *basic purposes and intents* as rationale do, namely:
    - Traceability
    - Review-ability
    - Change Control
    - Quality Control
    - Clarity of specification purpose
    - Clarification of specification context.
- Our argument here is that direct 'rationale' alone is not sufficient to serve the often-cited purposes of 'rationale' statements. The language of 'background information' for a specification must be enriched to better serve the overall purposes such as traceability, review-ability and change control.
- Example: '**Qualifier Statements'**

Fail [<u>Europe</u>, <u>Year</u> = <u>After Ten Years</u>, <u>Peace</u>]: 60% ±20% ← <u>Annual Plan Section 6.4.5</u>.

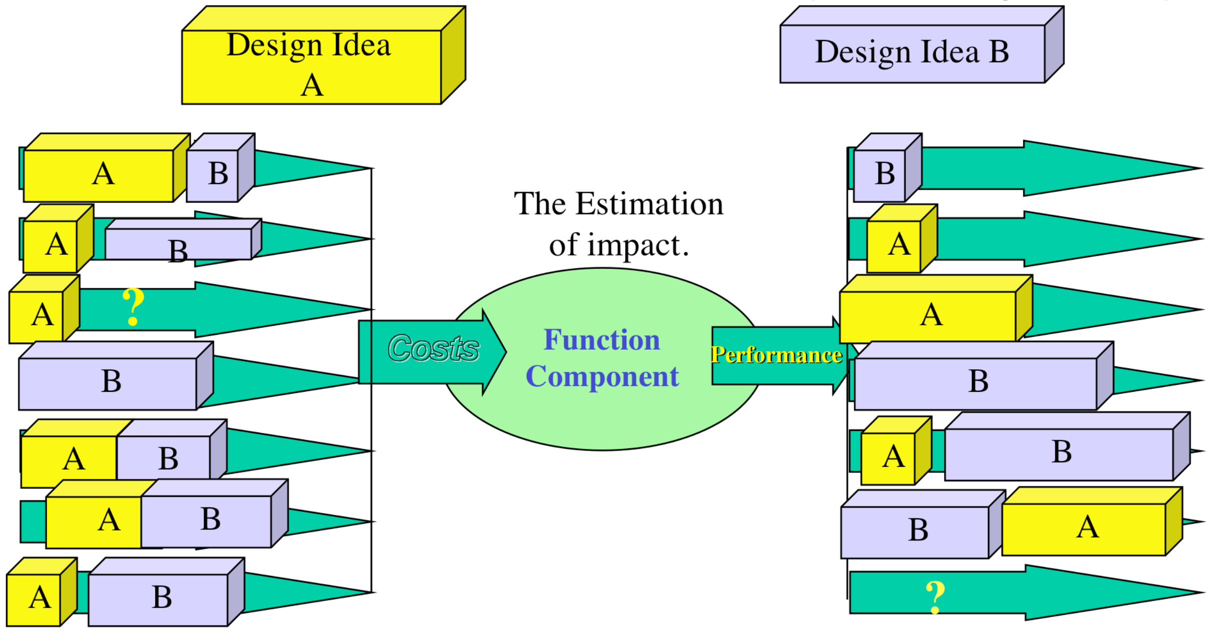
- *The three qualifier conditions must be all three be 'true' for the '60%' constraint requirement level to be a valid requirement. 'Peace' is an example of an event condition. Europe is a place condition.*

        - In general qualifier conditions can be inserted after most all parameter statements. Any number and type of useful qualifier conditions can be stated (like [<u>Europe</u>, <u>Year</u> = <u>After Ten Years</u>, <u>Peace</u>]). The parameter statement (like 'Fail' above) is only valid if all qualifier conditions are 'true'. The parameter specification depends on the qualifier conditions, and they largely explain what it is doing there. Additional statements can be combined to give more 'rational' information.
            - **Example**
            - **Fail [<u>Europe</u>, <u>Year</u> = <u>After Ten Years</u>, <u>Peace</u>]: 60% ±20% ← <u>Annual Plan Section 6.4.5</u>.**
              Authority: EU Commission.
              Impacts: Military Expenditure.
    - **Impacts**
        - ***The entire purpose of a 'design' is to satisfy a set of requirements.***
        - Notice, I wrote 'requirements' not 'requirement'. A design that satisfied one or more requirements, but does not satisfy *all* valid requirements, is either invalid, or in itself unsatisfactory. We can only understand the relative value of a design in the context of:
            - The *entire set of requirements* that the design must satisfy – at least partly, and at the same time, not 'violate'.
            - *All other designs* that are being considered as a set of designs to satisfy one set of requirements.
        - A given design can have the following basic impacts on **performance or quality** requirements:
            - *Partly* satisfy   (help be more reliable for example)
            - *Totally* satisfy or requirement level
            - Have a *negative* side effect on one or more requirement levels; even though it has positive effects on others.

        - It can have a *wide variety of impacts* on a wide assortment of simultaneously valid requirement levels.
        - This simple fact, which anyone can observe, is entirely ignored in the literature I have seen on design rationale. At best [Burge, is typical] this is expressed there in non-quantified forms such as:
            - "Minimizes keystrokes"
            - Or worse: "provides user guidance".[Burge]
        - In our view, to put it kindly, this type of 'rationale' does not provide a serious and useful level of rationale.
            - It does not estimate how much the design satisfies the required level of performance or quality
            - It does not consider side effect systematically and quantitatively
            - It does not give credible sources for the impact assertion in the rationale
            - It does not even reference a defined level of requirement (like Goal 90%)
            - It does not deal with constraint levels in any explicit way. Like,
                - **Fail level: 45 degrees C.**
                - **Design Impact: > 50 degrees C**
            - In short it is a set of fuzzy and undefined rationale statements meeting an equally fuzzy notion of performance and quality requirements.
    - Budgeted **Resource** Impacts
        - A given design can, in addition to the above noted impacts on performance and quality levels, simultaneously impact any number of budgeted resources (time, effort, money, space) in similar ways.

        - Understanding exactly how a design impacts resources, is clearly a critical part of the design rationale. The literature is equally poor on *costs*; [Burge, Ramesh, SEI97] are all typically silent on this critical cost aspect of a rationale.
    - Binary constraints
        - In addition to performance, quality and cost basis for design rationale, there remains the subject of binary constraints which can both dictate a design (a Design Constraint) and be violated so as to eliminate a design, whatever it other justifications.
        - Binary constraints are of the type:
            - **Must be Legal in EU.**
            - **Cannot inhibit entrance by mobility impaired customers**
    - **The Generic rationale:**
        - In general a design idea **rationale must meet** the following conditions:
            - It contributes something towards the target levels of performance and quality
            - It does not eat up budgeted resources out of proportion to its value
            - It does not violate specified binary constraints
            - It does not have unintended negative side effects that outweigh any positive value delivered to some performance and quality level targets
            - The uncertainty (risk of not delivering what the designer expects and estimates) must be acceptable and known in advance (and accepted by design review function).
- **Impact Estimation:**

- The Planguage approach to this is to apply *numeric estimation* [Gilb98IE] of the impacts of an entire set (ultimately, a *complete* set) of design ideas.
- *Here is a conceptual analogue view of Impact Estimation: horizontal size represents degree of impact.*



-
- *Here is a real example, courtesy Stuart Woodward, London*

Proposed Design Ideas →	*Sum of Estimates*	CAP Foundation	Upgraded Data Model	API	Risk Monitoring	CAP Groups	Counterparty Hierarchies
PERFORMANCE REQUIREMENTS							
Credit Information Response 60 mins. <-> 2 mins. [2003]	***105%***	15%	30%		25%		35%
Credit Request Cycle 60 hours <-> 48 hours [2003]	***95%***	40%	15%	25%	15%		
Credit Request Capacity 5 <-> 100 [2003]	***85%***	40%	5%	25%		15%	
RESOURCE REQUIREMENTS							
Project Duration 0 <-> 300	***86%***	12%	10%	8%	16%	10%	30%
Manpower Cost 0 <-> 750	***77%***	10%	12%	15%	5%	10%	25%
OVERALL IMPACT							
Total Performance Level Increase	***285%***	95%	50%	50%	40%	15%	35%
Total Cost	***163%***	22%	22%	23%	21%	20%	55%
PERFORMANCE / COST RATIO		4.32	2.27	2.17	1.90	0.75	0.64

-
- The numeric estimation will apply to all target levels required (performance, quality, budgeted resource levels) **at once.**
- The designs evaluated must not, even before impact estimation, fail to satisfy all binary constraints.
- The impact estimation will help us to
    - Understand if the *entire* proposed *set* of design ideas has sufficient rationale to be adopted
    - Understand if any *single* design idea has such poor rationale (low impacts, negative effects, high costs, bad value to cost ratio, high uncertainty) that we should not include it in the set of acceptable designs, even for review purposes.
- The Rationale of the Impact Estimation.

- Without going into extensive detail here [Gilb98IE, Gilb05CE suffice] it is important to at least list the 'rationale' for the impact estimation.
- Each estimate, of impact of one design on one performance/quality/cost requirement has the following structure:
  - An estimate is made,
  - assuming a given set of designs already in place before next design implementation,
  - and for a defined design implementation *environment* (example education and culture of people using the system) that will be hosting the design –
  - of the expected impact increment (or final level achieved) of that design,
  - on a defined scale of measure (like Scale: Mean Time Between Bug experienced by User).
  - For example: 60 seconds
  - An estimate of the upper and lower bounds of the impact (best case/worst case level): For example **(60 sec) ±20 seconds.**
  - This 'real scale' estimate is converted into a % of target (target = 100%, baseline = 0%) so that we can more clearly and immediately see if the design will satisfy our target levels on time.
  - The estimates will have specific evidence cited (on what basis, if any did you make the estimate? Is it based on experience?). For example: ***Evidence: The distinct software always gave more than 60 seconds MTBF."***
  - The evidence will have a specific source of evidence cited: example ***Source: IEEE***

Software June 2003, NASA Case, page 23, use of Distinct Software.

- *The combination of evidence and source will be rated on a credibility level scale and the Credibility Level (from 0.0 to 1.0) will be attached as part of the set of data for each estimate listed just above here.*
- *Here is the Credibility table we use at present:*

Credibility Rating	**Meaning**
0.0	**Wild guess, no credibility**
0.1	**We know it has been done somewhere**
0.2	**We have one measurement somewhere**
0.3	**There are several measurements in the estimated range**
0.4	**The measurements are relevant to our case**
0.5	**The method of measurement is considered reliable**
0.6	**We have used the method in-house**
0.7	**We have reliable measurements in-house**
0.8	**Reliable in-house measurements correlate to independent external measurements**
0.9	**We have used the idea on this project and measured it**
1.0	**Perfect credibility, we have rock solid, contract-guaranteed, long-term, credible experience with this idea on this project and, the results are unlikely to disappear**

- Here is an example of building up the Impact Estimation ‘Rationale’

Strategy->	**A**	**B[A]**	**B [NOT A]**	**C**
LEARNING PAST=10, PLAN=1				MUST=5 [end this year]
1a. Impact (SCALE)	4.5 min.	1 min.	8 min.	4.0 min
1b.Goal %increment	50%	100%	22% (2/9)	120% (6/5)
2. ± Uncertainty	±40%	±50%	±80%?	±20%
3. Evidence	Project Ajax, 1996	Competitor Beta **EVID-B**	Guess	Contract Guarantee
4. Source	Ajax report, pg.6	World Report p.17	John B.	Supplier Delta
5. Credibility	0.8	0.6	0.2	0.6
6. Comments	**A [NOT B]**	Assumes **A**	**B** alone	high cost

-
- Once you have developed these basic estimates, you can use them to compare design ideas, with respect to uncertainty (the ± estimate) and the credibility level, like this:

○

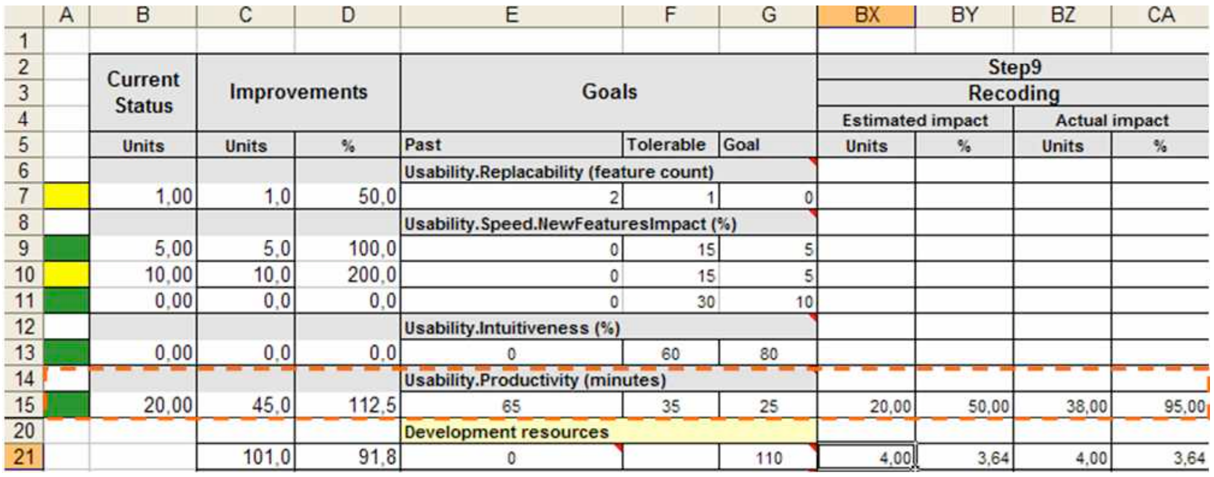
	On-line Support	**On-line Help**	**Picture Handbook**	**On-line Help + Access Index**
Learning Past: 60minutes <-> Goal: 10minutes				
Scale Impact	5 min.	10 min.	30 min.	8 min.
Scale Uncertainty	±3min.	±5 min.	±10min.	±5 min.
Percentage Impact	110%	100%	60%	104%
Percentage Uncertainty	±6% (3 of 50 minutes)	±10%	±20%?	±10%
Evidence	Project Ajax: 7 minutes	Other Systems	Guess	Other Systems + Guess
Source	Ajax Report, p.6	World Report, p.17	John B	World Report, p.17 + John B
Credibility	0.7	0.8	0.2	0.6
Development Cost	120K	25K	10K	26K
Performance to Cost Ratio	110/120 = 0.92	100/25 = 4.0	60/10 = 6.0	104/26 = 4.0
Credibility-adjusted Performance to Cost Ratio (to 1 decimal place)	0.92*0.7 = 0.6	4.0*0.8 = 3.2	6.0*0.2 = 1.2	4.0*0.6 = 2.4
Notes: Time Period is two years.	Longer timescale to develop			

•

- Finally this information can be combined with feedback from incremental deliveries to control progress and to learn from experience, as my client FIRM AS [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

- A real impact estimation table used to give the rationale for specific implementation steps, then compare reality, weekly, with the estimates.
-
- Here are the initial product (web surveys software) improvement results obtained from this method: Source: [Johansen04]. The second line below refers to the results on the chart above.
-

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000

Table 1: Improvements to product qualities

- This should illustrate the following points:
  - rationales should be given *numerically*
  - they should probably only make assertions on a *small scale and short term* (like a weeks work)

    - then *actual confirmation* of the *rationale reality*; *measurement* with *real stakeholders*, and the incremented system should be undertaken.
    - This feedback at frequent intervals will help keep the designers and project managers completely *realistic*

Summary

- The conventional ideas of how to deal with software and systems specifications with 'rationale' are not nearly powerful enough to serve their intended purposes well.
- The author suggests that the rationale behind any design idea (including 'means objectives' (defined as those which support higher level requirements <- Ralph Keeney), and any other specification type, should be specified in *numeric terms*, and be related primarily to *numeric* requirements levels.
- In addition the 'numeric rationales' should be constantly tested in the short term using Evolutionary feedback from real stakeholders and real environments.