Ten Design Principles:

Some implications for multidimensional quantification of design impacts on requirements

Tom Gilb
Result Planning Limited (RPL)
Iver Holtersvei 2, NO-1410 Kolbotn, Norway

Abstract. Designs have multiple impacts on requirements, and can only be fully understood in terms of their impact on requirements. In other words, the contributions of designs towards the set of performance and resource requirements must be considered, when evaluating designs, in addition to their contributions to the function requirements.

This paper sets out ten principles, and outlines their various implications for design. These are basic ideas about designs, which we should explicitly acknowledge, teach and use in practice. I would be surprised to find any serious disagreement about these principles, but I would be surprised to find serious conscious practice and teaching of them today!

Introduction

1. 'Design is Means' Principle

A design is a selected means to satisfy your ends. This has a number of implications as follows:

- the justification for a selected design is the degree to which it helps you meet your requirements;
- if a design has unknown or uncertain attributes, in any of our requirements dimensions, then that design choice may turn out to be be *invalid*, or not as good as other alternatives;
- if any requirement changes, this can potentially invalidate any (or all) designs chosen up to that point in time;
- if a design is mandatory, then it can have arbitrarily bad impacts on any performance and cost requirements.

2. The 'Valid Design' Principle

A *valid* design must contribute to performance goals, within all constraints. This has the following implications:

- you must be able to prove that a design does *not* violate any defined constraints;
- you must be able to prove that a design contributes to *at least one* yet unfulfilled performance requirement;
- the design cannot be justified, if its only contribution has been made by *another* accepted design already;
- a change in the numeric performance level required for a performance requirement can invalidate a previously acceptable design, or make a previously discarded design

valid.

3. The 'Good Design' Principle

A good design contributes more value than it costs. It is a profitable design. The implications of this principle are as follows:

- it is not sufficient for a design to contribute positively to the performance requirements; it must *not* cost more than the value of the level of performance it contributes;
- a designer must be able to *estimate* the cost impacts, of a given design, on all critical resource dimensions, in order to *justify* it;
- failure to correctly estimate the *individual* design-cost impacts, can cause total project budget overrun or schedule overrun;
- if there is clarity about design *cost*, but no numeric clarity about *performance impacts*, then there is no clear overall argument for the design.

Table 1: An Impact Estimation table for evaluating the best overall design option (Gilb 2005)

Requirements	Alternative Design Ideas							**Sum for Require-ment**
	Central	Youth	Facts	London	Diploma	Events	Discounts	
Performance Requirements								
Participation	80% ±50%	60% ±70%	0% ±50%	0% ±50%	30% ±50%	20% ±50%	30% ±50%	220% ±370%
Representation	80% ±50%	80% ±50%	10% ±50%	0% ±50%	10% ±50%	20% ±50%	50% ±40%	250% ±340%
Information	0% ±50%	20% ±40%	80% ±50%	0% ±20%	20% ±50%	0% ±50%	0% ±30%	120% ±290%
Conviction	0% ±10%	20% ±50%	60% ±30%	80% ±50%	10% ±50%	80% ±50%	0% ±50%	250% ±290%
Influence	0% ±50%	40% ±40%	60% ±50%	0% ±50%	80% ±50%	80 %±5 %	0% ±50%	260% ±340%
Fun	50% ±50%	40% ±50%	10% ±50%	0% ±0%	0% ±0%	80% ±50%	0% ±0%	180% ±200%
Sum of Performance	210% ± 260%	260% ± 300%	220% ± 280%	80% ± 220%	150% ± 250%	280% ± 300%	80% ± 220%	
Resource Requirements								

Financial Cost	20% ±30%	1% ±1%	1% ±1%	1% ±1%	1% ±5%	30% ±50%	30% ±50%	111% ±135%
Performance to Cost Ratio	210/20	260/1	220/1	80/1	150/1	280/30	80/30	

4. The 'Best Design' Principle

The *best design* of a set of alternative designs, is the one that contributes *most value* in relation *to cost*. It is the most *'efficient'* design.

The implications of this principle are as follows:

- selecting alternative designs is a matter of comparing *both* performance impacts and cost impacts – of getting the best benefit for cost;
- the choice of one design over another depends on *multiple* performance and cost attributes, and cannot be evaluated in a *single* dimension;
- any design which has been chosen only on the basis of knowledge of a *single* dimension is a potential 'risk for failure' in *other* dimensions of performance and cost;
- if the *worst case* levels of a design, are not evaluated, then the selected design risks *may* be worse than you expected, and the design *may* be a totally wrong choice.
- If the *credibility level* (scale of 0.0 (none) to 1.0 (best)) of the *evidence* and *source of evidence,* about the impacts of a design, are not considered; then a presumed 'best' design might not be as good as it seems from the numeric data alone.

5. The 'Temporal Design' Principle (and Design 'Half-Life')

The currently best design may become invalid, at any time, due to changes in requirements, or by the addition of new designs to the total set of designs for a system. All designs are under threat of extinction at all times. Other implications are as follows:

- *final* decisions on candidate designs should be delayed until the advantage from *finalizing* the design specification, clearly outweighs the disadvantage of having to *change* the design to an improved specification; keep design flexibility open as far as it is economic to do so.
- *any* requirement change, even an apparently small change in the target or constraint levels of a requirement, or the delivery timing, can be sufficient reason to change current candidate design decisions;
- all candidate design ideas may need fresh review, for updated information about costs, timings, impacts, experiences, before final commitment to them in the next implementation stages: they may have become invalid;
- there is probably a finite half-life for a candidate design idea, depending on its rate of change in its environment culture. For any idea that would not have been our first choice 5 years ago, the half-life is probably something like 1-2 years or less. The half-life of a design idea is when it is 50% less likely to be the best idea, than when it originally was the best choice.

6. The 'Multiple Values of a Design' Principle

The value a single design contributes to meeting requirements, can usually be estimated and measured in terms of multiple quantified performance dimensions. The implications are as follows:

- You have to be able to *quantify all* critical performance dimensions, in order to understand the true value and impacts of a single design idea;
- Failure to *correctly estimate* the impact on any single critical performance attribute of a design, may cause the entire evaluation of that design idea to be invalid, or cause it to be less-optimal than assumed;
- Any attempt to evaluate a design on the basis of only one or two performance dimensions, is almost certainly missing useful information, for understanding design consequences and risks;
- Since it is unlikely, for the most part, that we have access to reliable information about design impacts

Example of a Design Specification

Tag: OPP Integration.
Type: Design Idea [Architectural].

========== Basic Information ==========
Version: Status: Quality Level: Owner:
Expert: Authority:
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).
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:
Reuse of This Design:
Design Constraints:
Sub-Designs:

========== Impacts Relationships ==========
Impacts [Functions]:
Impacts [Intended]: Interoperability.
Impacts [Side Effects]:
Impacts [Costs]:
Impacts [Other Designs]:
Interoperability: Defined As: Certified that this device can exchange information with any other device produced by this project.

========== Impact Estimation/Feedback ==========
Tag: Interoperability.
Scale:
Percentage Impact [Interoperability, Estimate]: <100% of Interoperability objective with other devices that support OPP on time is estimated to be the result>.

========== Priority and Risk Management ==========
Rationale:
Value:
Assumptions: There are some performance requirements within our certification process regarding probability of connection and transmission etc. that we do not remember <-TG.
Dependencies:
Risks: <none identified>.
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:

============== Location of Specification ========================
Location of Master Specification: <Give the intranet web location of this master specification>.

Figure 1. Here is a real (doctored!) example of a real design specification using a version of the Design Specification Template. Not all the parameters are filled out yet. Notice that even the parameters, which are not filled out (like Impacts [Side effects] and Issues), are asking important questions about the design - and hinting that responsible designers should answer such questions! (Gilb 2005)

- in many design-impact dimensions, we are initially forced to make performance and cost estimates, and take some risks when making *any* design choices; but we would be wise to avoid 100% commitment to a design until we can get some more-credible feedback on all the critical dimensions. This might only be achieved in practice once the design is evolutionarily integrated into our system, and field-trialed to some degree.

7. The 'Design Costs Eternal' Principle.

A design must also be judged on multiple *cost* dimensions, including both *short term* and *lifetime* costs. Failure to consider *all* critical cost dimensions dooms your system to risk of failure. The implications are as follows:

- A 'perfect' design for performance target levels can fail completely to acceptable because of one or more unacceptable or bad *cost* levels;
- A reasonable engineering evaluation of any design will consider not only the performance levels it contributes to, but also the multiple costs it will incur during its *lifetime*, in relation to *specified* cost *constraints*;
- A design, which was once considered acceptable or 'best', may lose 'acceptable' status because of a change in some resource availability, during its lifetime. If, for example you cannot afford or recruit costly or scarce trained human maintenance staff, you may be forced to switch to another design in mid-life – *or the entire system may abruptly have to be replaced, if it is too costly to switch to an alternative design;*

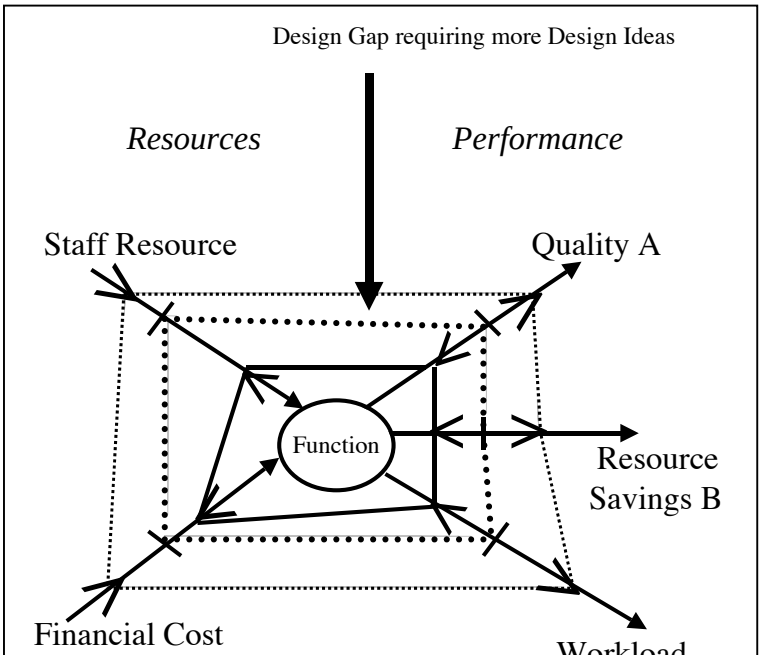

Figure 2. Diagram showing the gap between the Past and the Goal/Budget Levels and the contribution that a Design Idea makes towards filling the gap. The costs aspects are considered together with the performance impacts of the design (Gilb 2005)

- Designers should design with the concept in mind that the system should be *reasonably robust* with respect to *negative shifts* in human, time and money resources during the system lifetime – *even if this means replacement of some initial design components;*
- The last-made design decisions must be made with respect to *actual remaining resources* for both implementation and operation, at the time the design commitment is finally made. *The designer cannot assume that all design is first in the queue for implementation (some will be 'last' implemented, and may not have enough resources), nor that resources are infinite.*

8. The 'Design does not impact Function' principle

Design, by our definition (CE) only impacts performance and cost attributes. It has no impact on the *fundamental* function of a system. Functions can be designed (which *type* of bridge) but the design of a function does not change the function itself (the bridge) – only the function's performance and cost attributes.

The implications of this include:

- Design must be *appropriate* to the function of a system; meaning we cannot design in ignorance of the real functions, to reach some abstract performance levels
    - We have to know if we are in the domain of a mobile phone, a software product, or a spaceship function before deciding appropriate designs to reach a reliability level.
- Function requirements specifications must not themselves *contain* 'design' (they too often do!). That would make the design 'mandatory' when we need to retain flexibility to improve the design
- A 'design constraint' *can* be specified as a requirement, when we want to freeze a particular design, in spite of potential opportunities to optimize performance or costs by using another design. The rationale for this should be specified as well.
- To classify design as a *function requirement* is bad practice, since it robs us of

freedom to address the *real* performance requirements (which in this case may remain unspecified)

9. The Architecture Principle

Architecture is the highest level of design for a given system. It provides the framework and control over all the other more-specialized designs in the system. Nevertheless, architecture follows all the principles of 'design' itself.

The implications are as follows:

- architecture must have more *authority* and *power* than the narrower engineering and design disciplines;
- architecture needs to be reasonably settled – formally constrained, before serious engineering design can proceed;
- architecture is the same discipline as design, but controls a higher level of the system, and consequently has different abstractions of information to handle.

10. Design Evolution Principle

Design, at all levels, needs to evolve in small steps of confrontation with reality, each step followed by analysis and potential re-design. It cannot adapt to the inevitable continuous stream of information from different sources, different timings, experience feedback, problem solving insight, economic change, political change, and technical change in any other reasonable way. Design cannot be simply and correctly completed at once.

The implications of this principle are as follows:

- the design process must be taught and practiced as an iterative process;
- the design process must include the process of collecting realistic field data about design performance, and include the ability to adjust the design itself, to better meet the real needs of stakeholders;
- the design specifications must not be prematurely frozen;
- design specifications must not be changed without clear logical and profitable reason.
- The evolutionary project management process is one best practice for doing this (CE, Evo)

Conclusions

Designers need formal training and leadership in these principles

• design as practiced today is too often failing to systematically address the multiple stakeholder needs

• these principles apply all the more, the larger, and more-critical the system at stake is

• we can ignore these principles if the risks we thus incur are more tolerable than the cost of such systematic engineering as is proposed here.

Right now the partial and total failure rate of projects is so uncomfortably high that I suggest that we need to look at the option of investing more in the systems engineering intellectual processes, such as suggested here.

References

CE: Gilb, Tom, *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*, Elsevier Butterworth-Heinemann, 2005. ISBN 0750665076.

Evo: Gilb, Kai, *Evo: Evolutionary Project Management & Product Development*. Draft Book Manuscript available free at `http://www.gilb.com/`.

Biography

Tom has been an independent consultant, teacher and author since 1960. He works mainly with multinational clients; helping improve their organizations, and their systems engineering methods.

Tom's latest book is 'Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (Summer 2005).

Other books are (with Dorothy Graham) 'Software Inspection' (1993) and 'Principles of Software Engineering Management' (1988). His 'Software Metrics' book (1976, Out of Print) has been cited [Radice, 96] as the initial foundation of what is now CMMI Level 4.

Tom's key interests include business metrics, evolutionary delivery, and further development of his planning language, 'Planguage'. He is a member of INCOSE and is an active member of the Norwegian chapter NORSEC.

Email: Tom@Gilb.com
URL: `http://www.Gilb.com`

Version April 10th 2006 01:39 tg

Edited by Lindsey Brodie, Middlesex University, UK