

PROBLEMS-REQUIREMENTS-SOLUTIONS TRIANGULATION AS DESIGN NAVIGATION

Mark Bergman
Naval Postgraduate School

THE ROLE OF REQUIREMENTS IN DESIGN

Requirements are a bridge between what customer stakeholders want and what suppliers can design and build. More simply, it is they represent the link between systems analysis and design – i.e. problems and solutions. Requirements quantify and frame problems, while reducing the number of possible feasible solutions. They also define who are and are not legitimate stakeholders within a design process.

To understand the role of requirements in systems design, a few definitions need to be set. A *problem* is a gap between what the current situation and a desired improvement in the situation – i.e. the gap between an existing and a desired organizational state. In general, *stakeholders* are people who can demonstrably change the course of design process as well as determine and measure success criteria – i.e. goals. *Principal stakeholders* have resource powers can define a problem and mobilize resources to affect a solution [Bergman et al., 2002b]. Legitimate stakeholders can directly determine the success or failure of the design product. All others are not stakeholders.

A *solution* solves a given problem to the stakeholders' satisfaction. The success or failure of a solution is determined primarily by principal stakeholders; then by the other key stakeholders; followed by the rest of stakeholders in general. For example, say there exists a business owner wants an ERP system to integrate his or her sales, marketing, finance, and operations divisions. This owner must define and measure what the success criteria of the final information system. These criteria are grounded in the measuring the different between the existing information flow situation and the desired new state. Any acceptable ERP solution must meet or exceed these criteria. Furthermore, other stakeholders, like key users, IS support groups, other division managers and the like can accept or reject a solution. Furthermore, the general user community can accept or reject the solution. Therefore, the success criteria of all those that make a solution succeed or fail must be accounted for in design. These are captured and represented by requirements.

Requirements bridge problems and solutions [Bergman et al., 2002a]. They do so in two directions: 1) problem → solution and 2) solution → problem. Problems are owned, defined, and mobilized by customer stakeholders – lead by one or a group of key principal stakeholders. Solutions are designed, developed, built, deployed, and mobilized by supplier stakeholders – again, lead by one or a small group of key principals. Based on these insights, Table 1 presents an initial (non-exhaustive) list of meta-requirements – categories of requirements – based on an analysis of bridging

Table 1. Requirements as Problem-Solution Bridges

		Problem → Solution	Solution→Problem
Customers	<i>Technical</i>	<ul style="list-style-type: none"> • Determination of specific desired capabilities and expected operational constraints that together address their problem • Determination of success/failure criteria • Determination of level of acceptable risk and ROI • Determination of existing available resources that can be dedicated to problem analysis 	<ul style="list-style-type: none"> • Declaration of existing legacy systems; business rule, policies, procedures; costs, resources, timelines; and expertise that will not change regardless of the solution – i.e. high level requirements • Determination of physical operations environment • Definition of the expected user, support, and management communities • Qualification of suppliers as possible technology providers • Determination of whether or not a supplier can provide an acceptable solution within given capability, cost and time constraints
	<i>Political</i>	<ul style="list-style-type: none"> • Determination of whose problems receive organizational attention • Definition of Involved stakeholders and their levels of organizational power – i.e. authority • Framework for creating a stable, enforceable agreement – contract • Determination of which suppliers need to be involved in creating a contract 	<ul style="list-style-type: none"> • Definition of parts or the whole authority structure(s) that will govern the new system solution • Dedication of champions of the new technology • Determination of a program of organizational development and change • Support of existing contracts and service agreements • Support of ongoing relationships with suppliers
Suppliers	<i>Technical</i>	<ul style="list-style-type: none"> • Determination of which problems the supplier can adequately – technically and economically – address • Availability of expertise as well as their assistance in initial problem determination • Determination of the customer’s ability to pay for products and services rendered • Support of existing use of products and services by the customers – existing relationships; part of legacy infrastructure 	<ul style="list-style-type: none"> • Specific existing capabilities and constraints via their product lines as well as their costs and availabilities • Possible, feasible extensions to existing capabilities for more time and cost • Available and delivered profession service expertise in analysis, design, development, and deployment
	<i>Political</i>	<ul style="list-style-type: none"> • Determination of who will work with the customers • Determination of legal liability as will be specified in a contract • Determination of which customers need to be involved in creating a contract 	<ul style="list-style-type: none"> • Existing and ongoing contracts and agreements • Necessary new contracts and agreements • Support of ongoing relationships with customers

problems and solutions.

From examining these definitions, the role of requirements in design becomes clear. They must address the technical and political realities of connecting problems and solutions – as per Table 1. Requirements represent mandatory technical specifications, frameworks, and legacy infrastructure as well as the political alignments and agreements between stakeholders. A common form of these agreements is legally binding contracts. In the past, requirements have been defined as mandatory technical specifications [Kotonya and Sommerville, 1998, Loucopoulos and Karakostas, 1995, Wieringa, 1996]. Yet, from the examination of bridging customer stakeholders' problems and supplier stakeholders' solutions, this technical role is necessary, but not sufficient. Instead, the overall role of requirements in design is to provide a stable technical and political framework that enables and encapsulates the bridging of customers' problems and suppliers' solutions.

UTILIZING REQUIREMENTS FOR TRIANGULATING DESIGN

For any one problem, there are a number of possible, feasible solutions. Conversely, any one solution can address multiple problems. Therefore, there is a many-to-many relationship between problems and solutions. Requirements act like an “associative entity” between problems and solutions¹. Requirements provide a conduit from solutions to problems with reduces the number of possible problems by associating them with feasible combinations of technologies and political agreements. Moreover, the choice of possible solutions is framed by the selected problems and their specific requirements.

Therefore, a combination of problems-requirements-solutions is a triangulation model for design. Still, requirements are strict in their definition and application. For any combination of problem and solution, the bridging requirements must be satisfied. Interestingly, there are fewer restrictions on problems and solutions – i.e. they have some flexibility to change to better fit the needs of the customers and abilities of the suppliers. Arguably, this makes the problems and solutions highly dependent upon requirements. Still, requirements must be rediscovered, created, or modified due to changes in problems or solutions. Therefore, they are codependent.

Together, problems-requirements-solutions become a model for discovering feasible and satisficing designs. Adjustments to any one part of the model require corresponding adjustments in the other two parts. As each part of this design model is reduced and becomes more specific, the same happens to the other two parts – i.e. as problems become more specifically defined, more specific requirements and solutions can be determined or created. This leads to the ability to produce a set of measures or metrics that indicate the quality the technical and political balance for any particular design. Based upon this feedback, further adjustments to any part of this design model can be made to discover improved balances².

In design, this leads to three possible combination states: 1) infeasible, 2) feasible, but suboptimal, and 3) satisficing. Infeasible are combinations of problems and

solutions that cannot rectify the technical and/or political requirements. All other combinations of problems-requirements-solutions are feasible – i.e. class 2. Of these possibilities, theoretically, only one combination has the optimum balance of technical and political requirements. Still, as Simon pointed out, this is likely too complex to determine [Simon, 1996]. The best one can do is satisfice in this situation. Hence, class 3 design is classified as satisficing. In other words, it is the best choice amongst the combinations based on finding the best balance of feasible technical (architectural) combinations, political arrangements, and satisfaction of customers' and suppliers' requirements within a given amount of time and resources.

Altogether, requirements can be utilized to inform the feasibility and satisfiability of possible designs. Since there are many possible problems-solutions combinations, there is no single set of “correct” requirements. Instead, there exist separate, yet likely overlapping sets of requirements that correspond to problems-solutions variants. As previously mentioned, these requirements can be adjusted in response to the nuances that are discovered or created while refining problems and solutions. This argument breaks with the overly positivistic view of a single true set of requirements that tends to dominate their definition and use in the design process³. Instead, requirements can be determined and modified in conjunction with changes in problems-solutions combinations. They are able to be utilized to find satisficing designs, not simply frame them.

AN AGENDA FOR THE NEXT 10 YEARS

Requirements specifications are boundary objects [Star and Griesemer, 1989] that connect systems design stakeholders together – esp. customers and suppliers [Bergman et al., 2007, Carlile, 2002]. In general, current requirements are concisely written statements that explicitly and logically describe what functional capabilities a solution must provide and under what nonfunctional conditions and constraint it must operate. Modern methods tend to extract these statements from scenario analysis and use cases. The requirements specification is often a backbone of customer-supplier contracts that stipulate what is to be further designed and developed – esp. for large scale government or commercial systems.

The problem is that requirements in a written, logic-based form and format fail to be a useful design boundary object. Put simply, this form and format is not useful for modeling, triangulation, and feedback, esp. for the vast majority of customers. As per Bergman, Lyytinen, and Mark (2007 – Forthcoming) [Bergman et al., 2007], a design boundary object must meet four essential features to be useful by its target stakeholders. These are: 1) promote shared representation, 2) transform design knowledge, 3) mobilize for design action, and 4) legitimization of design knowledge. These features are defined in Table 2. The current form and format of requirements tends to be a compendium of formal and pseudo-logical statements. A logical format of written text is generally not understandable by most customers. Moreover, it is not an operational language – i.e. it does not well describe work functions, flow, and procedures. Hence, a requirements specification does not promote a shared representation of requirements between customers and suppliers. This makes it very

difficult to transform design knowledge across involved stakeholder groups.

Table 2. Design Boundary Object Features⁴

Feature	Definition	Domain	Description
Promote Shared Representation	Encapsulates understandings based on a common syntax and semantics, which are shared across social worlds	Technical	<ul style="list-style-type: none"> Form shared functional representations, i.e. data and technical models, prototypes, architectures, specifications, etc. [Sutcliffe and Gault, 2004, Whitten and Bentley, 2006] Transform knowledge at the boundary between social worlds [Carlile, 2002, Karsten et al., 2001]
		Political	<ul style="list-style-type: none"> Form shared political representations, i.e. agreements, contracts, "sign-offs," memorandums of understanding, etc. Perspective sharing, i.e. making sense of other social world's perspective. [Karsten et al., 2001]
Transform Design Knowledge	Manipulate and converse representations that will propel movement between design routines as to facilitate finding a feasible functional solution and stabilize the political ecology	Technical	<ul style="list-style-type: none"> Move knowledge from ambiguous to specific; objective/goal to a problem; instable to stable; idea to solution [Henderson, 1991, Pohl, 1996] Realign operational structure to stabilize functional ecology Enable design traceability [Ramesh and Jarke, 2001]
		Political	<ul style="list-style-type: none"> Hand-off of power and control from provider to recipient world(s) [Carlile, 2002] Realign power to stabilize political ecology [Fairholm, 1993, Markus, 1983] Enable agreements traceability
Mobilize for Design Action	Source and wield resources and power to propel progress along a design path.	Technical	<ul style="list-style-type: none"> Participate in SAD routines as to invite functional expertise, review solutions, etc.) [Henderson, 1991] Reduce problem ambiguity for solution discovery in a design path Conscribe expertise relevant for problem identification and solution [Henderson, 1991]
		Political	<ul style="list-style-type: none"> Participate in decision making, mobilization of resources and allocation of design tasks [Hirscheim et al., 1995] Mobilize bias towards preferred resolution [Hirscheim et al., 1995]
Legitimization of Design Knowledge	Grant a legitimate status to a boundary object through validation of its content as to align with the stakeholders' intent.	Technical	<ul style="list-style-type: none"> Certify, verify and validate the truthfulness and correctness of design knowledge [Henderson, 1991]
		Political	<ul style="list-style-type: none"> Demonstrate acceptability of goal(s), problem(s) and solution(s) in the given institutional order as to authorize the movement of design knowledge across social worlds [Bergman et al., 2002b]

Furthermore, written requirements describe specific capabilities and constraints, but they do so out of context. There is no grounding model or equivalent to bring these requirements together into a coherent whole. Without this context, the specification can no longer mobilize design direction or motivate any rational actions. Altogether, it cannot legitimately represent customers' desires since the customers cannot directly verify and validate them.

To counter this weakness in requirements specifications, an improved design boundary object needs to be created to support problems-requirements-solutions triangulation as well as meets the four design boundary object features. Therefore, my call for research for the next 10 years is determining thus:

- *What design boundary objects should be created to enable problems-requirements-solutions triangulation?*
- What is their form and format?
- How well do they represent problems-requirements-solutions combinations?
- How easy are they understood and manipulated, esp. by principal and key stakeholders?
- How can they be developed and tested in the field?
- What are the methodologies to produce them?

Arguably, there are no great (or even good) existing models that currently meet these criteria. UML models are too technically detailed to fulfill this function. Business process and workflow models miss the social and technical requirements details to be suitable for suppliers. Use case and scenarios continue to be useful in informing the formation of a possible requirements boundary object, but they are not a substitute for them. Ethnographic field work can provide rich, deep descriptions of problems and requirements, leading to possible solutions. But, a rich text suffers from the same problems as written logical requirements. A grounded model based on these insights may be a viable direction, but the resulting form and format must conform to the four features of a viable design boundary object.

Therefore, this is a call for the formation of new models or the like that can enable problem-requirements-solutions triangulation. Following Simon's lead [Simon, 1996], design methodologies should focus on creating ever improved components and subsystems in order to master the complexity of ever more complex systems. This indicates a new model based style of design for large systems. In general, I propose that problems-requirements-solutions triangulation could be enabled by operational work proto-architectural models – i.e. system models of how an organization operates. These proto-architectures show high-level, feasible combinations of technologies, social functions, and procedures that together, conform to and enforce known requirements. These can be reviewed and tested for acceptability by the involved design stakeholders – esp. customers and suppliers. These models must be clearly verified and validated via analysis, simulation, or demonstration to show that they actually address the problem(s). Such models should include socio-technical components and subsystems for completeness. They need to have to be clear enough for involved stakeholders to promote the shared representation and malleable enough to manipulate them – i.e. enable problems-requirements-solutions triangulation.

Operational work proto-architectural models should clearly transform the needs of the customers and the abilities of the suppliers into shared representations of feasible designs. Furthermore, well formed work operational proto-architectural models should mobilize action by the involved stakeholders to refine, agree to, and drop specific problems-requirements-solutions alignments. Eventually, legitimization occurs when an

operational work model is well enough defined to be show as technically (and economically) feasible as well as producing a stable political agreement across the key stakeholders.

As an alternative, agile programming work has begun to show that storyboarding and user interfaces may also be a viable requirements boundary object for problem-requirements-solutions triangulation. There is new evidence in the literature that understanding a system from a “control-panel point-of-view” may be enough to quickly facilitate determining good or satisficing designs via improved communications between customer and suppliers [Armitage, 2004, Hwong et al., 2004, McInerney and Maurer, 2005]. Still, it is unclear if this approach provides enough detail to adequately cover the technical and political issues at an appropriate depth in order to produce feasible, legitimate designs. Addressing this weakness could make the discovery and creation of agile programming based requirements boundary objects another viable branch of research.

There should be many more possible research directions beyond those described in this essay. I expect that research into problems-requirements-solutions triangulation will yield new models and methodologies that will result in improving the production of good designs. Still, any model produced from this work must minimally meet the four features of design boundary objects – Table 2. Further, as these models (or equivalent) become further refined, it should become more apparent to determine those early designs are merely good versus those that are great – i.e. satisficing. This would be a vast improvement over the current general determination the success or failure of system design *ex post* – i.e. acceptance or rejection after deployment. Correspondingly, any improvements in early designs produce a geometric or exponential savings in resources and effort [Boehm, 1981, Sommerville, 2006]. In turn, this will free up resources to solve evermore problems, produce new systems, and further the art of design.

REFERENCES

- Armitage, J. (2004) "Are agile methods good for design?," *Interactions* (11) 1, pp. 14-23.
- Bergman, M., J. L. King, and K. Lyytinen (2002a) Large Scale Requirements Analysis as Heterogeneous Engineering, in C. Floyd and R. Klischewski (Eds.) *Social Thinking - Software Practice*, Cambridge, MA: MIT Press, pp. 357-386.
- Bergman, M., J. L. King, and K. Lyytinen (2002b) "Large Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering," *Requirements Engineering Journal* (7) 3, pp. 152-171.
- Bergman, M., K. Lyytinen, and G. Mark (2007) "Boundary Objects in Design: An Ecological View of Design Artifacts," *Journal of the AIS* (Forthcoming)
- Boehm, B. W. (1981) *Software engineering economics*. Englewood Cliffs, N.J.: Prentice-Hall.
- Carlile, P. R. (2002) "A Pragmatic view of Knowledge and Boundaries: Boundary Objects in New Product Development," *Organizational Science* (13) 4, pp. 442-455.
- Elmasri, R. and S. B. Navathe (2006) *Fundamentals of Database Systems*, 5th edition. Reading, MA: Addison Wesley.

- Fairholm, G. W. (1993) *Organizational power politics: tactics in organizational leadership*. Westport, Conn.: Praeger.
- Grady, J. O. (2006) *Systems Requirements Analysis*. London: Academic Press.
- Henderson, K. (1991) "Flexible sketches and inflexible data-bases: Visual communication, conscription devices and boundary objects in design engineering," *Science Technology and Human Values* (16) 4, pp. 448-473.
- Hirscheim, R. A., H. Klein, and K. Lyytinen (1995) *Information Systems Development and Data Modeling, Conceptual and Philosophical Foundations*. Cambridge: Cambridge University Press,.
- Hwong, B., D. Laurance, A. Rudorfer, and X. Song (2004) "User-Centered Design and Agile Software Development Processes," *CHI, April* pp. 25-29.
- Karsten, H., K. Lyytinen, M. Hurskainen, and T. Koskelainen (2001) "Crossing boundaries and conscripting participation: representing and integrating knowledge in a paper machinery project," *European Journal of Information Systems* (10pp. 89-98.
- Kotonya, G. and I. Sommerville (1998) *Requirements engineering: processes and techniques*. Chichester ; New York: J. Wiley.
- Loucopoulos, P. and V. Karakostas (1995) *System Requirements Engineering*. London, UK: McGraw-Hill Book Co.
- Markus, M. L. (1983) "Power, Politics and MIS implementation," *Communications of the ACM* (26pp. 430 - 444.
- McInerney, P. and F. Maurer (2005) "UCD in agile projects: dream team or odd couple?," *Interactions* (12) 6, pp. 19-23.
- Okasha, S. (2002) *Philosophy of Science*: Oxford University Press, USA.
- Pohl, K. (1996) *Process-centered requirements engineering*. New York, NY: Wiley.
- Ramesh, B. and M. Jarke (2001) "Toward reference models for requirements traceability," *IEEE Transactions on Software Engineering* (27) 1, pp. 58-93.
- Robertson, S. and J. Robertson (2006) *Mastering the Requirements Process*, 2nd edition. Reading, MA: Addison-Wesley Professional.
- Simon, H. A. (1996) *The Sciences of the Artificial*, 3rd edition. Cambridge, Mass.: MIT Press.
- Sommerville, I. (2006) *Software Engineering*, 8th edition. Reading, MA: Addison Wesley.
- Star, S. L. and J. R. Griesemer (1989) "Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39," *Social Studies of Science* (19pp. 387-420.
- Sutcliffe, A. and B. Gault (2004) "The ISRE Method for Analyzing System Requirements with Virtual Prototypes," *Systems Engineering* (7) 2, pp. 123-143.
- Teorey, T. J., S. S. Lightstone, and T. Nadeau (2005) *Database Modeling and Design: Logical Design*, 4th edition. San Francisco: Morgan Kaufmann.
- Thayer, R. H. and M. Dorfman (2000) *Software Requirements Engineering*, 2nd edition. Piscataway: Wiley-IEEE Computer Society.
- Whitten, J. L. and L. D. Bentley (2006) *Systems Analysis & Design Methods*, 7th edition. New York: McGraw-Hill/Irwin.
- Wieringa, R. (1996) *Requirements engineering: frameworks for understanding*. New York, NY: Wiley.

¹ An associative entity is created to normalize many-to-many relationships between two database entities, Elmasri, R. and S. B. Navathe (2006) *Fundamentals of Database Systems*, 5th edition. Reading, MA: Addison Wesley, Teorey, T. J., S. S. Lightstone, and T. Nadeau (2005) *Database Modeling and Design: Logical Design*, 4th edition. San Francisco: Morgan Kaufmann. It enables 1-to-many relationships by focusing on the unique instance that the combination of entities creates. Examples of this are invoices and sales orders associative entities.

² A danger here is “analysis paralysis.” It is possible to keep trying out different combinations while never settling on any one choice. This can be made worse if there is any creeping requirements or featurism. That can explode the possible number of design combinations. Therefore limits must be placed on how long and how far the search for the best design can continue.

³ Positivism is the philosophical belief that there exist one objective truth and the goal of science is to uncover it, Okasha, S. (2002) *Philosophy of Science*: Oxford University Press, USA. Arguably, there is a widely espoused belief in the existence of the one true set of objective requirements that are the results of appropriately performed systems analysis, Grady, J. O. (2006) *Systems Requirements Analysis*. London: Academic Press, Robertson, S. and J. Robertson (2006) *Mastering the Requirements Process*, 2nd edition. Reading, MA: Addison-Wesley Professional, Thayer, R. H. and M. Dorfman (2000) *Software Requirements Engineering*, 2nd edition. Piscataway: Wiley-IEEE Computer Society. The requirements absolutely frame and ground any possible design. This essay challenges this belief and replaces it with one that is more flexible based on the nuances of differing combinations of problems and solutions.

⁴ This table is taken from Bergman, M., K. Lyytinen, and G. Mark (2007) "Boundary Objects in Design: An Ecological View of Design Artifacts," *Journal of the AIS* (Forthcoming).