Chapter 1

The First Principle: Stakeholder Value-Based System Definition and Evolution

This principle derives from the International Council on Systems Engineering (INCOSE) definition of systems engineering:

"An interdisciplinary approach and means to enable the realization of successful systems."

As discussed in ICSM Distilled, a system will be successful if and only if it makes winners of its success-critical stakeholders. Thus, in order to create a successful system, you need to identify which stakeholders are success-critical, to determine their value propositions or win conditions, and to define, design, develop, and evolve a mutually satisfactory or win-win system with respect to their evolving value propositions.

First, we present a surface assessment robot project that was a tremendous technical success, but failed to transition into operational use due to neglect of some potential stakeholder-unsatisfactory outcomes. We then summarize some of the key lessons learned from the project. We then present a highly successful example of the development of a commercial intravenous medical pump that employed numerous techniques to identify and concurrently satisfy its success-critical stakeholders' value propositions.

We conclude the chapter by presenting some common pitfalls that can lead to violations of the principle, and presenting some helpful practices for applying the principle. These include a Benefits Chain for identifying success-critical stakeholders; human factors techniques for identifying their value propositions; collaboration tools for negotiating mutually satisfactory system specifications and plans; techniques for operational concept formulation and analysis, and techniques for designing and contracting for ease of accommodating necessary changes

1.1 Failure Story: The Too-Good Road Surface Assessment Robot

The Carnegie Mellon University (CMU) Road Surface Assessment Robot project seemed to have done everything right. It delivered a system with radically higher accuracy, efficiency, and labor savings with respect to its manual predecessor system. However, its end product still sits in a storage room unused [Latimer, 2008].

1.1.1 Project Context

One of the key stakeholders in engineering tools for assessing the roughness of roads is the contractor for delivering a road, since any deviations to standards for the road's acceptable roughness must be fixed by the contractor. Current assessment techniques are manual, expensive, and time consuming, but much needed, as repaving of road surfaces with deviations from roughness standards is much more expensive. Any new tools for more efficiently and accurately assessing surface roughness would have a high payoff in time and cost savings.

The client for this particular project was a contractor company with a need to assess the roughness of a specially constructed road to support a dedicated set of roadways to transport standing people at speeds up to 30-40 miles per hour. The genesis for the project was the client's participation in an engineering design course at CMU that produced an attractive design for such a robot. The client's key stakeholders were its end-user division responsible for road systems; its Research and Development Office responsible for the creation or acquisition of new technology; its current group of manual road surface assessors; and its Quality Assurance organization responsible for monitoring corporate quality in its processes and products. The developer was CMU's Institute for Engineering Complex Systems.

1.1.2 Project Execution.

The project was executed over three years. The first year explored alternative operational concepts for operating a robotic vehicle that would identify and mark deviations from roughness standards on the roadway. The business case for the winning design indicated a likely 100:1 time and cost savings for inspection, which would more than repay the investment cost in its use on the first specialized roadway. The first year ended with a specification for the system's functional and performance requirements that had been validated by a combination of prototyping, analysis, and review by the corporation's relevant specialty experts. A potential feature to have the robot's logic determine whether a deviation was critical enough to be marked was rejected, as the quality assurance organization wished to be able to analyze each deviation to determine quality improvement opportunities.

The second year involved selection of outsourced components and detailed build-to specifications for the robot vehicle. The specifications also addressed operations and maintenance procedures for the robot and its components, and assurances that the robot would not cause harm to the roadways. Further prototyping indicated that earlier concerns about sensor accuracy were unlikely to be a problem. The robot vehicle was successfully developed in the third year, and passed an acceptance test on a test track that had representative deviations built into it. It also passed the test of reducing inspection time by a factor of 100.

1.1.3 Final Results.

Subsequently, the robot was tried on several miles of the first segment of specially built roadway, produced by the best-qualified subcontractor for roadways. Several high-level managers from the company and the roadway subcontractor were present to observe the initial operation. The system operated entirely according to the specifications, but not according to expectations. Besides performing 100 times faster than the manual system, it reported 100 times as many deviations. A post-analysis by the lead engineer for surface assessment determined that all of the deviations were correctly reported by the system, but over 99% of them were minor, not ride-quality threatening, and which would not have been recorded by a manual inspection.

However, this created an untenable situation, as the quality assurance organization was obliged to monitor corrective action for each deviation, and rejection of marked deviations could be considered as favoritism to specific subcontractors. The roadway subcontractor felt that its quality reputation was being unfairly degraded. And there seemed to be no degrees of freedom to repeat the robot test using comparable quality criteria to the previous manual assessment system. As a result, the best-acceptable management solution was to discontinue the robot project and to continue to use manual surface assessment methods.

1.1.4 Primary Lessons Learned.

The operational concept analyses were good, but the operational scenarios were focused only on the technical performance of the robot vehicle, and not on the effect of off-nominal outcomes on the stakeholders involved. This happens fairly frequently on hardware and software engineering projects, which focus on the technical aspects and neglect the sociotechnical aspects. A classic example was Thomas Edison's first patent (#90,646): an electrical vote-counting device for legislative bodies. Edison learned that faster vote counting on a given bill was incompatible with various legislative strategies such as filibustering and off-line deal making across multiple special interests and bills, and resolved that he would only focus on inventions for which there was strong public demand.

Another characteristic of sociotechnical systems is that their desired features tend to emerge with use, rather than being pre-specifiable requirements. This places a value on designing for flexibility and adaptability, rather than optimizing a point solution on a single quantity and leaving no degrees of freedom to adjust the solution. As also seen in the box describing the [Weinberg-Schulman, 1974] experiment on giving developers different criteria to optimize, one should beware of the common practice of asking developers to maximize, minimize, or optimize a particular system attribute.

BOX: Be careful what you ask for. You may get it.

The Weinberg-Schulman experiment involved five teams that were given the same software assignment: to develop a program for solving simultaneous linear equations using Gaussian elimination. However, each team was given different directions about what to optimize while doing the job. One team was asked to complete the job with the least possible effort, another team was to minimize the number of statements in the program, another was to minimize the amount of memory required by the program, another was to produce the clearest possible program, and the final team was to produce the clearest possible output.

When the programs were completed and evaluated, the results were remarkable:

- Each team finished first (or, in one case, second) with respect to the objective they were asked to optimize.
- None of the teams performed consistently well on all of the objectives.

Note in particular the performance of the first team: the team asked to do the job with the least possible effort. Although this team finished first on effort to complete and second in *productivity* (lines of code produced per man-day), they finished next-to-last in the number of statements and amount of memory required to do the job, last in the clarity of their program, and third in the clarity of their output.

The results tend to confirm frequent observations that:

- *Software developers have very high achievement motivation.* If you define *good achievement* in terms of what you ask for, the developers will generally work very hard to give you just what you asked for.
- *Different software development objectives do indeed conflict with each other in practice.* In particular, as seen from the first team's performance, pure concentration on minimizing the software development effort is likely to have bad effects for software life-cycle budgets and schedules, and effectiveness, because of the penalties paid in other important software dimensions such as program size and clarity.

Table 1-1: *Results of Asking Developers to Optimize Different Objective*

END BOX

	Resulting rank of performance (1 = Best)				
Team objective: Optimize	Effort to Complete	Number of Statements	Memory Required	Program Clarity	Output Clarity
Effort to complete	1	4	4	5	3
Number of statements	2-3	1	2	3	5
Memory required	5	2	1	4	4
Program clarity	4	3	3	2	2
Output clarity	2-3	5	5	1	1

1.2 Success Story: The Hospira, Inc. Next-Generation Intravenous Medical Pump

This section summarizes the systems engineering aspects of the next-generation Symbiq™ IV (intravenous) medical pump development, developed by the Abbott Laboratories spinoff company Hospira, Inc., and documented in detail in Chapter 5 of the National Research Council book, "Human-System Integration in the System Development Process" [Pew-Mavor, 2007]. As described in [Pew-Mavor 2007], its purpose is "to deliver liquid medications, nutrients, blood and other solutions at a programmed flow rates, volumes and time intervals via intravenous and other routes to a patient, primarily for hospital use with secondary limited feature use by patients at home."

In creating a next-generation product, Hospira proposed to introduce new IV pump features: multi-channel vs. single-channel liquid delivery; ability to gang multichannel devices together; associated user-programming capabilities and programmable drug libraries for specifying parallel delivery of liquids; use of color touchscreen devices; integration with numerous types of hospital information systems; ease of use for both medical personnel and patients at home; handling of potential hardware, software, and human-user faults; compliance with U.S. and international safety standards; use of alternating-current or battery power; and the ability to be cost-competitive and attractive to traditional medical and hospital administration personnel. Many of these desiderata are highly coupled, such as multichannel hardware controls, concurrent software synchronization, distinctive displays and alarms for multichannel devices, and rigorous medical safety standards.

Views of the resulting Symbiq™ pump are shown in Figure 1-1. Its systems engineering involved a great deal of concurrent analysis and engineering of its hardware, software, human factors, operational, business, and safety aspects. It has been a commercial success, and won the 2006 Human Factors and Ergonomics Society's User-Centered Product Design Award and the 2007 Medical Design Excellence Award.

Not only were there numerous technical challenges, but also there were challenges in systems engineering of a product and a life-cycle operational concept that would produce satisfactory outcomes for a wide variety of product and operational stakeholders, whose value propositions were often in some conflict. Some customers wanted numerous features that would require a complex user interface, while others wanted a simple, easy-to-learn and easy-to-use interface. Some users wanted the most advanced color-touchscreen displays available, while others wanted a simpler, cheaper product that was harder to misuse via inadvertent screen touches. Some organizations felt that a minimal interpretation of the required safety features would be acceptable, while others advocated ultrahigh assurance levels. Some marketing personnel wanted a quick development and fielding of the basic product to capture market share, while maintainers wanted initial built-in life cycle support, maintenance, and diagnostic capabilities.

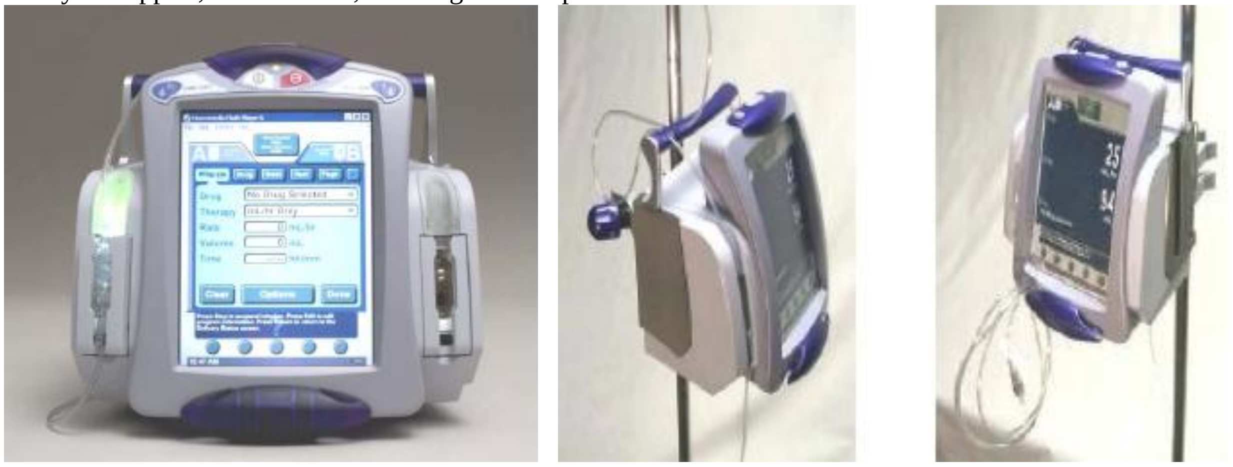

Figure 1-1. *Symbiq™ Pump Industrial Design*

In such situations, many organizations focus on making quick requirements decisions and rapidly proceeding into development or outsourcing. However, Hospira's understanding of the uncertainties and risks caused them to pursue a risk-driven, incremental-commitment course of buying information to reduce risk. As described in [Pew-Mavor 2007], they used a version of the Incremental Commitment Spiral Model (ICSM). The following sections describe the project's incremental system definition progress through the ICSM Exploration, Valuation, Foundations, and Development phases. Some evolution of terminology has occurred: the [Pew-Mavor 2007] version uses ICM instead of ICSM, and Architecting phase instead of Foundations phase. A description of the project has been included as a case study of successful systems engineering in Part 7 of the Systems Engineering Research Center-INCOSE-IEEE Systems Engineering Body of Knowledge [Pyster et al. 2012; http://sebokwiki.org]. A summary of the ICSM is also included in the section of Part 3 of the SEBoK on Life Cycle Models.

1.2.1 Symbiq^{TM} Exploration Phase Summary

In the Exploration phase, the project carried out numerous stakeholder needs, technical opportunities, and business competition analyses, and determined ranges of preferred options. Stakeholder needs analyses included contextual inquiry via shadowing of nurses using IV pumps and followup interviews; and creating task flow diagrams, use environment analyses, and user profiles analyses. Technical opportunity analyses included initial conceptual designs of multichannel pump configurations, evaluation of commercially available single-color and multicolor display devices and touchscreen capabilities, and software approaches for specifying multichannel delivery options and synchronizing concurrent processes.

Business competition analyses included hiring a management and marketing planning firm to perform next-generation pump major competitor strengths and weaknesses analyses with respect to such capabilities as number of pump channels, therapies, programming options, air-in-line management, battery and alternating current capabilities, biomedical domain expertise, and alarms. Several key competitive advantages of a next-generation pump were identified, such as bar-code reading capability, small size and light weight, stand-alone functional channels, an extensive drug library, a high level of reliability, and clear mapping of screen displays and pumping channels.

Market research also identified market windows, market segment analyses, pricing alternatives, hospital purchasing decision analyses, and safety aspects. These were iterated with respect to focus groups of key thought leaders in critical care. The results were factored into a product concept plan, cost analysis, and business case analysis. These were independently reviewed by experts as part of the Valuation Commitment Review process, which resulted in a go-ahead decision with an identification of several risks to be managed.

1.2.1 Symbiq^{TM} Valuation Phase Summary

The Valuation phase focused on the major risks highlighted in the Valuation Commitment Review. These included the multi-channel pump options, the types of programmable therapies, the need for tailorable medication libraries, the display screen and user interface options, and the safety considerations. The Valuation phase also elaborated the product concept plan for the most attractive general set of options, including a development plan and operations plan, along with an associated cost analysis, risk analysis, and business case for review at the Foundations Commitment Review.

The multi-channel pump options were explored via several hardware industrial design mockups and early usability test of the mockups. These included evaluation of such desired capabilities as semi-automatic cassette loading, special pole-mounting hardware, stacking of channels and total number of channels, and tubing management features. The evaluations led to overall choices of a semi-automatic cassette loading capability with a red-yellow-green LED display to indicate concerns with the loading mechanism and with the pump in general. Field exercises with prototypes of the pole mountings indicated the need for quick release/activation mechanisms, which were subsequently implemented. Risk analyses of alternative stacking mechanisms and total number of channels established a preference for side-by-side stacking, a decision to develop 1-channel and 2-channel units, and to support a maximum of 4 channels in a stacked configuration.

The types of programmable therapies considered included continuous delivery for a specified time period; patient weight-based dosing; piggyback or alternating delivery between the two channels; tapered or rampedrate delivery, intermittent-interval delivery, variable-time delivery, and multistep delivery. These were evaluated via prototyping of the software on a simulated version of the pump complexes, and iterated until satisfactory versions were found.

Evaluation of the tailorable medication libraries addressed the issue that different hard and soft safety limits were needed for dosages in different parts of a hospital (emergency room, intensive care, oncology, pediatric care, etc.), and thus that a need for hospitals to be able to program their own soft limits (overridable by nurses with permission codes) and hard limits (no overrides permitted). Stakeholder satisfaction with the tailoring features was achieved via prototype exercises and iteration with representative hospital personnel.

A literature review was conducted to determine the relative advantages and disadvantages of leading input and display technologies, including cost and reliability data. After downselecting to three leading vendors of touch screen color LCD displays and further investigation of their costs and capabilities, a business risk analysis focused on the tradeoff between larger displays and customer interest in small-footprint IV pumps. The larger display was selected, based on better readability features and the reduced risk of wrong user entries with larger screen buttons. Extensive usability prototyping and operational concept formulation was done with hardware mockups and embedded software that delivered simulated animated graphic user interface (GUI) displays to a touchscreen interface that was integrated into the hardware case.

The safety risk analysis in the Valuation phase followed ISO 14971:2000 standards for medical device design, focusing on Failure Modes and Effects Analyses (FMEAs) based on the early high-level design, such as entry of excessive drug doses or misuse of soft safety limit overrides. Subsequent-phase FMEAs would elaborate these analyses, based on the more detailed designs and implementations.

As in the Exploration phase, the results of the Validation phase analyses, plans and budgets for the succeeding phases, the resulting revised business case, evidence of solution feasibility, and remaining risks with their risk management plans were reviewed by independent experts, and the Foundations Commitment Review was passed subject to a few risk level and risk management adjustments.

1.2.2 Symbiq^{TM} Foundations Phase Summary

During the Foundations phase, considerable effort was focused on addressing the identified risks, such as the need for prototyping of the full range of GUI usage by the full range of targeted users, such as doctors and home patients, for interoperability of the Symbiq software with the wide variety of available hospital information systems, and for fully detailed FMEAs and other safety analyses. Comparable added effort went into detailed planning for development, production, operations, and support, providing more accurate inputs for business case analyses.

GUI prototyping was done to a set of usability objectives, such as:

- 90 percent of experienced nurses will be able to insert the cassette the first time while receiving minimal training. 99 percent will be able to correct any insertion errors.
- 90 percent of first time users with no training would be able to power the pump off when directed.
- 80 percent of patient users would rate the overall ease of use of the IV pump 3 or higher on a 5-point scale of satisfaction with 5 being the highest value.

Similar extensive evaluations were done on the efficacy and acceptability of the audio alarms, including use of a patient and intensive care unit simulator that included other medical devices that produced noises and other distractions such as ringing telephones. This helped to adjust the alarms and the understandability of the visual displays.

Software interoperability risk management involved extensive testing of representative interaction scenarios between the Symbiq^{TM} software and a representative set of hospital information systems. These resulted in several adjustments to the software interoperability architecture. Also, as the product was being developed as a platform for the next generation of infusion pump products, the software design was analyzed for overspecialization to the initial product, resulting in several revisions. Similar analyses and revisions were performed for the hardware design.

As the design was refined into complete build-to specifications for the hardware and the operational software, the safety analyses were elaborated into complete Failure Modes and Effects Analyses of the detailed designs. These picked up several potential safety issues, particularly involving the off-nominal usage scenarios, but overall confirmed a high assurance level for the safety of the product design. However, the safety risk assessment recommended a risk management plan for the Development phase to include continued FMEAs, thorough off-nominal testing of the developing product's hardware and software, and extensive beta-testing of the product at representative hospitals prior to full release.

This plan and the other Development and Operations phase plans, product feasibility evidence, and business case analysis updates were reviewed at a Development Commitment Review, which resulted in a commitment to proceed into the Development phase.

1.2.3 Symbiq^{TM} Development Phase Systems Engineering Summary

The Development phase was primarily concerned with developing and testing the hardware and software to the build-to specifications, but continued to have an active systems engineering function to support change management, operations, production, and support planning and preparation, and further safety assurance activities as recommended in the risk management plan for the phase.

For hospital beta-testing, thoroughly bench-tested and working beta versions of the IV pump were deployed in two hospital settings. The hospitals programmed drug libraries for at least 2 Clinical Care Areas. The devices were used for about 4 weeks. Surveys and interviews were conducted with the users to capture their "real world" experiences with the pump. Data from the pump usage and interaction memory was also analyzed and compared to original doctors' orders. The beta tests revealed a number of opportunities to make improvements including revision of the more annoying alarm melodies, and revising the data entry methods for entering units of medication delivery time in hours or minutes.

Usability testing was also conducted on one of the sets of abbreviated instructions called TIPS cards. These cards serve as reminders for how to complete the most critical tasks. Numerous suggestions for improvement in the TIPS cards themselves as well as the user interface came from this work including: how to reset the "Air-in-Line" alarm and how to address the alarm and check all on screen Help Text for accuracy.

The abovementioned usability objectives were used as acceptance criteria for the validation usability tests. These objectives were met. For example, the calculated task completion accuracy was 99.66 percent for all tasks for first time nurse users with minimal training. There were a few minor usability problems uncovered that were subsequently fixed without major changes to the UI or that affected critical safety related tasks.

The risk analysis was iterated and revised as the product development matured. FMEAs were updated for safety critical risks associated with three product areas: the user interface, the mechanical and electrical subsystems and the product manufacturing process. Some detailed-implementation problems were found and fixed, but overall the risk of continuing into full-scale production, operations, and support was minimal. Systems engineering continued into the Operations phase, primarily to address customer change requests and problem reports, and to participate in planning for a broader product line of IV pumps.

Overall, customer satisfaction, sales, and profits from the Symbiq^{TM} IV pump have been strong, and satisfaction levels from the management, financial, customer, end user, developer, maintainer, regulatory, and medical-community stakeholders have been quite high. This is largely because of the extensive stakeholder involvement in the system's formulation, analysis, and process commitment decision milestones.

1.3 The Fundamental System Success Theorem and Its Implications

In ICSM Distilled, we presented overviews of the Fundamental System Success Theorem and the System Success Realization Theorem (Boehm-Ross, 1989; Boehm-Jain, 2006). The Fundamental System Success Theorem states that:

A system will be successful if and only if it makes winners of its success-critical stakeholders.

The proof of "if" was summarized as:

1. Everyone significant is a winner.
2. Nobody significant is left to complain.

Some external critics may complain that the system did not use their favorite technology, but if they are not success-critical, they cannot affect the success of the project for the success-critical stakeholders.

The proof of "only if" was summarized as:

3. Nobody wants to lose.
4. Prospective losers will refuse to participate, or will counterattack.
5. The usual result is lose-lose.

The proof is elaborated by example in three frequently-occurring instances of the primary stakeholders in an enterprise involving a customer contracting with a developer for a software system that will benefit a community of users, as shown in

Table 1-2. *Win-lose Generally Becomes Lose-Lose*

Proposed Solution	"Winner"	Loser
Quick, Cheap, Sloppy Product	Developer & Customer	User
Lots of "bells and whistles"	Developer & User	Customer
Driving too hard a bargain	Customer & User	Developer

In Case 1, the customer and developer attempt to win at the expense of the user by skimping on effort and quality. When presented with the product, the user refuses to use it, leaving everyone a loser with respect to their expectations.

In Case 2, the developer and user attempt to win at the expense of the customer (usually on a cost-plus contract) by adding numerous low-value "bells and whistles" to the product. When the customer's budget is exhausted without a resulting value-adding product, again everyone is a loser with respect to their expectations.

In Case 3, the user and customer compile an ambitious set of features to be developed and pressure competing developers to bid low or lose the competition. Once on contract, the surviving bidder will usually counterattack by colluding with the user or customer to convert the project into Case 2 (adding user bells and whistles with funded Engineering Change Proposals) or Case 1 (by minimally interpreting vague contract terms for user help, maintenance diagnostics, etc.). Again, everyone is a loser with respect to their expectations.

The Failure Story and Success Story above also illustrate the theorem. The Too-Good Road Surface Assessment Robot shows that it only takes one losing success-critical stakeholder to turn a technically outstanding product into an unsuccessful one. The Next-Generation Intravenous Medical Pump success story shows how much effort Hospira went to in identifying and satisfying their financial, doctor, nurse, administrator, and safety assurance stakeholders via business case analyses, interviews, surveys, prototypes, and safety analyses in the process of defining and developing their product.

One implication of the theorem is that it is risky to use terms such as "optimize, minimize, and maximize" in project guidance. As Nobel Prize winner Herbert Simon showed in his (Simon, 1957) book *Models of Man,* success in multi-stakeholder situations is achieved not by optimizing, minimizing, or maximizing of individual criteria, but by satisficing with respect to the stakeholders' multiple criteria, in which the stakeholders do not get everything they want, but get an outcome that is better than their previous situation. Another implication is that there are several practices necessary to achieve a stakeholder win-win situation, which is discussed next.

1.4 The System Success Realization Theorem and Its Implications

As discussed above, the Fundamental System Success Theorem does not tell us how to realize and maintain a win-win state. This requires the

System Success Realization Theorem: Making winners of your success-critical stakeholders requires:

6. Identifying all of the success-critical stakeholders (SCSs).
7. Understanding how the SCSs want to win.
8. Having the SCSs negotiate a win-win set of product and process plans.
9. Controlling progress toward SCS win-win realization and adaptation to change.

The next four sections elaborate on key aspects of these four success-realization elements.

1.4.1 Identifying All of the Success-Critical Stakeholders (SCSs)

A good way to identify your system's SCSs is via the Concurrency View of the ICSM shown in Figure 5 in the Introduction. The first three rows in the Figure cover the concurrent activities most prominent at the beginning of the Exploration Phase: Envisioning Opportunities, System Scoping, and Understanding Needs. They involve exploring what is unsatisfactory with the current system-of-interest (which may be an individual system, a product line of similar systems, or an enterprise that includes a number of product lines and individual systems) and identifying alternative candidates for improving it. The Understanding Needs activity generally involves using existing feedback from the system's current SCSs (end users, operators, administrators, maintenance and support personnel, suppliers, distributors, customers, system-level managers, higher–level managers, and perhaps others) to determine what needs fixing to make it better serve their needs. This activity will produce a baseline set of SCSs.

The Envisioning Opportunities activity may involve ways to improve or extend the current system by modernizing its aging infrastructure, extending its capabilities, capitalizing on emerging technology, or forming strategic partnerships with organizations having complementary capabilities that would enable the current system to better serve the existing customers and users, or to serve a broader base of customers and users. Or it may involve setting up a new organization to pursue a new technology or business opportunity. In either case, such activities are likely to extend the baseline set of SCSs, or perhaps to reduce the baseline by phasing out or divesting existing product lines or services.

The System Scoping activity during the Exploration Phase will perform an initial winnowing of the candidate initiatives, based on top-level feasibility evidence and business-case analysis. These will also identify further SCSs, such as finance specialists, venture capitalists, or other research and development sponsors.

The later parts of the Exploration Phase and the Valuation Phase will involve the concurrent engineering of system goals/objectives/requirements, the concurrent top-level architecting and designing of the human, hardware, and software aspects of alternative system solutions and their top-level life cycle plans, and a top-level analysis of the relative feasibility of the candidate solutions. These activities will also involve iteration of the earlier needs, opportunities, and system scoping analyses. They will also identify further candidate SCSs, such as systems engineers, systems developers, system assurance personnel, and as needed, contract management personnel, business process engineering personnel, business management personnel, and specialists in needed technical and human factors disciplines.

The Foundations phase may identify further SCSs as the project works out its success-critical preparations for development, transition, and operations. These may include representatives of organizations concerned with safety, security, or other general-public concerns; representatives of critical supply chain management functions; and representatives of diverse customers receiving specially-developed versions of the developed system.

All of these add up to a rather formidable set of SCSs. The good news is that most projects will involve a relatively minimal subset of the potential SCSs, frequently just representatives of the four major life cycle roles of end user, acquirer, developer, and maintainer, with others brought in on an as-needed basis. And for smaller, less-critical projects, the roles of end user, acquirer, and maintainer can often be merged into a single SCS, as with agile methods.

1.4.2 Understanding How the SCSs Want to Win

The activities discussed above will not only identify SCSs, but also identify many of their win conditions or value propositions with respect to the system being defined. A good example is the information on what is unsatisfactory with the current system-of-interest. Other examples include information on opportunities to improve or extend the current system; and desired capabilities identified in exploring improved concepts of operation. Beyond these, references such as the National Research Council report, *Human-System Integration in the System Development Process* (Pew and Mavor, 2007) identify further methods for eliciting stakeholders' desired capabilities, such as field observation, task and workflow analysis, participatory analysis, scenario development, prototyping, executable models and simulations, stories and use cases, and concept mapping.

Table 1-2. Top-Level Stakeholder Value Dependencies: Quality Attributes

Stakeholder Classes / **Dependability Attributes**	Information Suppliers	System Dependents	Information Brokers	Information Consumers		System Controllers	Developers	Maintainers	Administrators	Acquirers
				Mission -						
Protection				critical	uncrit.					
Safety		**		**		**				
Security	*	**	**	**		**				
Privacy	**		**	*	*					
Robustness										
Reliability		*	*	**		**		*	*	
Availability		*	*	**		**		*	*	
Survivability		*	*	**		**		*	*	
Quality of Service										
Performance			**	**	*	**		*	*	
Accuracy, Consistency	**		**	**	*	**			*	
Usability	*		*	**	**	**			*	
Evolvability			*	**	*	*		**	*	**
Interoperability			**						*	**
Correctness							*			**
Cost							*			**
Schedule			*	**	*	*	**			**
Reusability							**	*		*
**** Critical * Significant () Insignificant or indirect**										

These methods are good for identifying capabilities, but not as effective for identifying the system's desired quality attributes. Fortunately, some research results such as Table 1-2 (Boehm et al. 2004) provide a top-level baseline for how the major quality attributes vary by system stakeholder. Most of the terms in Table 1-2 are reasonably self-explanatory, but two may require further explanations. System dependents are people that are not involved in the system's development or operation, but are dependent on some of its attributes (e.g., safety for airline passengers and medical patients). System controllers perform real-time control of a system (e.g., airline pilots or electric power distribution controllers). Stakeholders may belong to multiple classes: an airline pilot is both a system controller and a system-dependent passenger.

The dependency ratings refer only to direct dependencies. For example, system developers, acquirers, and administrators are concerned with safety or security only to the extent that a system's information suppliers, users, and dependents are concerned with them. And information suppliers and system dependents are only concerned with reliability and availability to the extent that these help provide their direct concerns with security and safety. Further information on how the priorities of system functional capabilities and quality attributes vary by stakeholder role will be discussed in Chapter 2.

1.4.3 Having the SCSs Negotiate a Win-Win Set of Product and Process Plans

As seen in Table 1-2, the relative priorities for different quality attributes vary by stakeholder role. This is also the case for functional capabilities, which means that the SCSs will often have to negotiate mutually satisfactory or win-win combinations of system capabilities and quality attributes. This generally means that each stakeholder will not get everything they want, but will get an outcome that is better than their current situation.

For over 15 years, we have been developing, applying, and evolving methods, processes, and tools for diverse stakeholders to express their win conditions, to identify conflicts among the win conditions, and to negotiate win-win agreements (Boehm et al. 1995; Boehm-Gruenbacher-Briggs 2001; Kukreja-Boehm 2012). The six generations of WinWin toolsets have been used on over 200 projects, with increasing success rates in speed and stakeholder satisfaction in using the approach and tools. The latest version has used a Facebook style that has made it much easier for non-technical stakeholders to use, and to remain involved in renegotiating changes as a project progresses.

Figure 1-3 depicts the relationship between the various elements of a WinWin equilibrium. It is defined as having all of the stakeholders' Win Conditions covered by Agreements among all of the stakeholders, with no outstanding Issues. This is true by default at the beginning of the process. When a stakeholder enters a Win Condition, the other stakeholders can discuss it and decide whether or not to agree with (a modification of) it as part of the system definition. If all of the stakeholders agree, the Win Condition is covered by an Agreement, and the WinWin equilibrium is preserved.

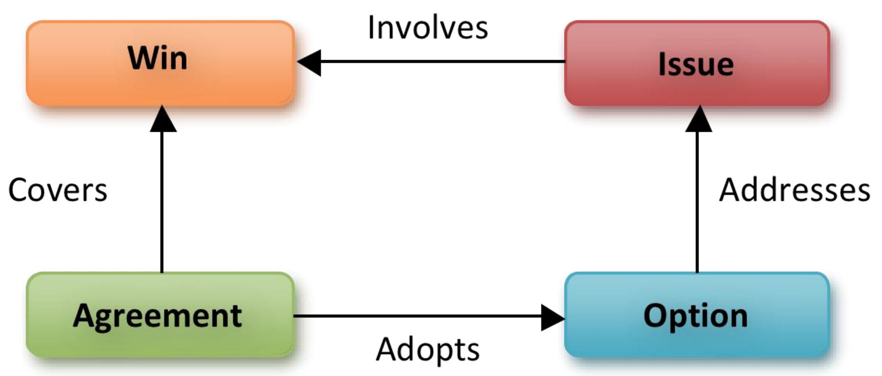

Figure 1-3. The WinWin Equilibrium Model

If stakeholders disagree, they should submit Issues saying why they disagree, and ideally what needs to be done to enable them to agree. The resulting loss of the WinWin equilibrium due to an outstanding Issue needs to be addressed by stakeholders submitting Options to resolve the Issue. Almost always, the resulting discussions and evolution of the Options results in a resolution of the Issue and restoration of the WinWin equilibrium. If not, it is best to know this early and either rescope the project or terminate it, as a win-lose situation almost always leads to a lose-lose result, as was seen in Figure 1-2.

An excellent source for creating options for mutual gain is *Getting to Yes* (Fisher-Ury, 1981). At the enterprise level, an excellent set of patterns of success and examples for developing a success-oriented culture is the Mitre Report, Patterns of Success in Systems Engineering (Rebovich and DeRosa, 2011). Based on extensive project analyses, it includes several social patterns of success, such as establishing a Circle of Trust among stakeholders, and organizing around networks vs. nodes. And it includes several technical patterns of success, such as Seeing is Believing via prototyping and multiple-stakeholder views, and planning to re-plan vs. planning the work and blindly working the plan. From a stakeholder value standpoint, some good techniques for expediting win-win convergence are expectations management (Karten, 1994) and matching people's tasks to their win conditions.

The key principles in matching people's tasks to their win conditions involve *searching out win- win situations* and *expanding the option space to create win-win situations.* Some effective techniques available to the project manager for searching out win-win situations include:

- Breaking options into parts (functions, activities, increments, phases), and configuring combinations of sub-options into win packages for each participant. For example, under some conditions, establishing a separate leader for successive system increments has worked well, particularly if the increments are large, with different technical and/or organizational centers of gravity.
- Realigning options along win-win axes. For example, some projects have successfully shifted the authority and responsibility for quality assurance from the developer (who may consider it a bore) to the maintainer, who has considered it a major win-leverage opportunity.

Some effective techniques available to the project manager for expanding the option space to create win-win situations are:

- Linking tasks to future options and career paths ("Quality assurance may be a bore, but it's a ticket to a fast-track career path").
- Expanding the scope of a task ("Quality Assurance should not be a bore. I think you could lead the way in helping us make quality assurance a more proactive function in getting us quality products. That would be a real achievement").
- Linking tasks to extra rewards ("Rescuing this integration and test mess will be a killer, but I'll make sure you get a good bonus and lots of kudos if you succeed").
- Providing extra support ("This schedule is very ambitious, but I'll provide your team with the first-class tools and facilities you'll need to meet it").
- Surfacing new options ("We can't develop all the functions in 12 months, but if we do an incremental development, we can satisfy your top-priority needs in 12 months'').

1.4.4 Controlling Progress Toward SCS Win-Win Realization and Adaptation to Change.

As a project progresses through the life cycle phases and stages, it will increasingly find that its initial assumptions about the nature of its product plans and process plans will undergo change. Some of the changes will preserve or strengthen the win-win equilibrium, but others will cause aspects of the project to change into a win-lose relationship among the stakeholders. Clearly, such changes need to be monitored, analyzed, and handled to keep the project in a win-win equilibrium.

Some of the most frequent sources of needed corrective action are that there have been no destabilizing changes in:

- The value propositions of the SCSs
- The organizational relationships among the SCSs
- The need to accommodate new SCSs
- The competitive marketplace and emerging technology
- The project's feasibility evidence and business case
- The infrastructure upon which the system is built
- The interfaces between the system and its co-dependent external systems
- The key personnel involve in the project
- The financial or other resources required for the project
- The regulations that the project needs to satisfy

When such changes are destabilizing, the relevant SCSs need to revisit their previous negotiated win-win solution and determine the most cost-effective way to recover a win-win equilibrium. In outsourcing situations, it is best to identify the relevant destabilizers as project stability assumptions that will need to be addressed if they become invalid.

References

Barry Boehm, Rony Ross, "Theory-W Software Project Management Principles and Examples," IEEE Transactions on Software Engineering, Volume 15, Issue 7, July 1989, pp. 902-916

Barry Boehm, LiGuo Huang, Apurva Jain, Ray Madachy, "The Nature of Information System Dependability: A Stakeholder/Value Approach"

Boehm, B., Lee, M.J, Bose, P., and Horowitz, E., "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach," 17th International Conference on Software Engineering (ICSE'95), ACM-IEEE, May 1995, p. 243.

Boehm, B., Grunbacher, P., and Briggs, R., "Developing Groupware for Requirements Negotiation: Lessons Learned," IEEE Software, May/June 2001, pp. 46-55.

Barry Boehm, Apurva Jain, "A Value-Based Theory of Systems Engineering," Proceedings, INCOSE 2006.

Kukreja, N. and Boehm, B., "Process Implications of Social Networking-Based Requirements Negotiation Tools," Proceedings, ICSSP 2012, June 2012.

Latimer, D.T. IV, "Effectiveness of Engineering Practices for the Acquisition and Employment of Robotic Systems," PhD Dissertation, Department of Computer Science, University of Southern California, May 2008.

Pew, R., and Mavor, A., Human-System Integration in the System Development Process, NAS Press, 2007.

Simon, H., *Models of Man*, Wiley, 1957.

Weinberg, G., and Schulman, E., "Goals and Performance in Computer Programming," Human Factors 16 (1), February 1974, pp. 70-77.