The Illusion of Certainty

Grady Campbell
CMU Software Engineering Institute
4301 Wilson Blvd., Suite 200
Arlington, VA 22203
703-908-8223
ghc@sei.cmu.edu

Abstract

Acquisition policy and, even more so, acquisition practice today presumes that certainty is key to success, and that uncertainty or delays in achieving certainty regarding user needs or solution approach will necessarily impede progress. This means that when uncertainty arises during an acquisition effort, the natural response is to make decisions that resolve this uncertainty. Uncertainties arise for various reasons, such as poorly understood, conflicting, or changing needs. If under pressure to maintain progress an acquisition effort makes decisions to resolve these uncertainties without sufficient information, expertise, or deliberation, they are really only creating an illusion of certainty; in a practical sense, the uncertainty still exists. This artificial certainty then leads to flaws such as insufficient detail concerning specific needs or premature limiting of solution options. A new approach to acquisition is needed that recognizes that hiding uncertainty is detrimental to success. Systematically exposing uncertainties will be beneficial toward making acquisitions more flexible, cost-effective, and responsive to changing needs.

The Requirements Challenge in Acquisition

The purpose of the acquisition system is to acquire products that provide users in an enterprise (e.g., warfighters) with capabilities needed to perform their mission effectively and efficiently. To this end, user needs must be understood in terms of opportunities for improving the enterprise's operational systems. Based on this understanding, a product must then be conceived/identified, acquired/developed (engineered and manufactured), deployed, and sustained/evolved. As long experience has shown, many challenges arise in trying to determine actual user needs and achieve a satisfactory product that properly addresses those needs. The essence of those challenges is achieving a balance among needed capabilities, enabling technology, cost, and timeliness in providing a product.

The beginning of the acquisition life cycle is the identification of user needs and technology opportunities that suggest the potential for improved capabilities. These needs are progressively refined, elaborated, and reviewed to create specifications of requirements in increasing levels of detail, feeding into efforts that interpret those requirements to create a conformant product. Again, proper understanding of actual needs in sufficient detail to permit acquisition of a product that meets those needs is a significant challenge, but, further, needs to be met by a product are not static but continue to change. Future sustainability and improvement requires an acquirer and developer of a product to understand not only existing needs and technologies but how those needs and technologies are likely to evolve in the future.

What experience and numerous studies over the years, by the Department of Defense, Office of Management and Budget, Government Accountability Office, National Academies, and industry groups, have suggested is that properly determining needs and expressing these as requirements for product acquisition is difficult and prone to inflexibility and error. For example, quoting from [6], *"Today, 'requirements' are used to define capability needs, implying that nothing less than a specified set of criteria is sufficient. Instead, a more prudent answer is to buy the best capability affordable, in the quantity desired, and fielded in as timely a manner as possible."* However, even this opinion fails to fully address how the acquisition approach could be improved so that products would better address both current and future needs. The following discussion proposes the notion that premature decision making, leading to only an illusion of certainty, is a factor in the requirements problem and that a greater awareness and explicit accommodation of uncertainty throughout the acquisition process would be beneficial.

Understanding the Concept of Requirements

"Requirements" is a term that everyone understands on an intuitive level but it can specifically mean many different things. In acquisition policy [1], requirements is used variously to mean:

- Capabilities needed by a community of users (e.g., mission, user, or capability *requirements*)
- Rules that must be followed in performing acquisition activities (e.g., statutory and regulatory *requirements*)
- Criteria against which the acceptability of a product development effort will be evaluated (e.g, program *requirements*)
- A specification of the expected behavior of a product being acquired (e.g., product, operational, or system *requirements*) (i.e., the guidance that product developers are given to know what to build or to describe what has been built)

In a general sense, all of these uses are consistent but the practical implications of each for acquisition differ substantially. In particular, user requirements, program requirements, and system requirements are all different expressions of the single notion that a product is needed that will allow users to perform their mission more effectively. For an acquisition to be successful, all of these expressions must be consistent.

In fact, however, this presents a dilemma: acquisition rules require that an acquisition program can proceed only after its requirements have been approved. Program requirements are derived based on user requirements and are the basis for defining product requirements. With respect to an envisioned or existing product, its requirements can be thought of as a model of the observable behavior that the product must exhibit to provide the capabilities needed by users. If user requirements are inaccurate, this will undermine program and product requirements unless there is a means for modifying them during the course of product development.

What the experience of many people suggests, supported by recurring government and industry studies, is that requirements at any level of detail are often flawed: incomplete, inaccurate, misunderstood, and prone to unforeseen change. The best means we have for mitigating these flaws is systematic iteration through all aspects of product

development, allowing for the progressive refinement of a shared understanding of needs, constraints, potential solutions, and tradeoffs. If acquisition policy or practice dictates that requirements at some point early in the acquisition process must be viewed as complete and immutable, it is likely that resulting products will both embody difficult-to-correct flaws and fail to keep up with changing needs.

The acquisition system today seems to encourage, and practitioners conform to, the idea that a proper acquisition depends on achieving certainty, in requirements and in the cascade of subsequent decisions that flow through the acquisition process. What they actually achieve is the appearance of certainty but such certainty can in fact be an illusion built upon premature, inadequately reasoned decisions, inadequate understanding of needs, and failure to account for changing needs, technology, and operational context. Explicit recognition and accommodation of uncertainties is a way around this dilemma that will help programs avoid commonly experienced cost overruns, schedule delays, and product defects, while supporting concerns for proper accountability.

Causes of Uncertainty in Requirements

In thinking about the nature of requirements, we can easily identify several potential sources of uncertainty. To properly understand and specify requirements, these need to be exposed, analyzed, and documented with rationale:

- Incomplete knowledge. User needs are usually specified by people who are knowledgeable in the mission of the enterprise and how it currently works. However, they often have only limited knowledge of how those needs may be realized in solution products, how different aspects may interact in a solution, or how new solutions could change the way the enterprise works; as a result, they may express needs in ways that unintentionally seem to limit the potential solution. Furthermore, there are usually aspects of user needs about which even experts disagree. Because no one can have complete knowledge of all aspects of any endeavor, it is likely that any description of user needs will mask areas of uncertainty or disagreement. Customers may recognize that this uncertainty exists but, lacking a proper awareness of the need and means to communicate the variety of alternatives that they see, they may instead make a reasoned but ultimately arbitrary choice.
- Imprecise understanding of needs. While customers may be competent to define user needs, developers are likely to lack the same depth of understanding. Experts in a field may share assumptions, concepts, and terminology that enable them to describe needs in simpler terms that acquisition agents or developers with less of that expertise may misinterpret. It may not be apparent that developers have a different understanding until a product exists and its behavior can be observed in use. Needs are often better understood after potential solutions have been developed and comparatively evaluated, preferably with exposure to knowledgeable users, leading then to being able to define better requirements.
- Differing needs among users. Users doing similar jobs may in fact legitimately not have exactly the same needs. If requirements characterize all users doing similar jobs as having the same needs, the developer may create a product that properly

meets the needs of only some users. A product that imposes a particular viewpoint on all such users will make some of those users less effective.

- Changing needs. Needs change over time because of changes in mission, operational context, and technology. Defining needs only from the perspective of a single point in time ensures that these are inaccurate with respect to other times. Framing needs as fixed without consideration of potential change over time imposes uncertainty on what is potentially predictable change that may be better accommodated by developers if known.

Why Apparent Certainty in Requirements May Be an Illusion

By the time a product is deployed, its actual ("as-built") requirements have been effectively determined. Still, although in a practical sense there can be no actual uncertainties about the behavior of a deployed product, there may not be a complete and accurate specification of what those requirements are. In that sense, there may be no one who can be certain about all aspects of the product's behavior.

Although acceptance of a product depends on compliance with acceptance criteria usually expressed in the form of requirements, the same problem can exist for those: there may be unacknowledged uncertainties that have been improperly resolved. In a process in which encountered uncertainties are simply decided away, without proper identification and systematic analysis of factors and tradeoffs, and not documented with rationale, it is likely that some uncertainty still exists relative to actual current or future needs. This same argument applies as the reasoning behind particular requirements are traced back through acquisition decision making.

Uncertainty can exist at any level of requirements. The fundamental uncertainty is how does someone determine and communicate what they want or need. That phrase itself reveals a basic issue: how do we distinguish aspects that we must have from what we might like to have from what we would accept. The essence of engineering is identifying and weighing tradeoffs among alternatives but the nature of defining requirements is not only to make a definitive statement about what a customer needs but in doing that to also eliminate unsuitable solutions. When this is done without full knowledge of actual possibilities, potential solutions can be prematurely constrained.

In looking at how requirements are determined, there are many factors that can lead to the various kinds of uncertainty:

- No individual or collection of people will have complete knowledge of all aspects of an existing system and its operational context; requirements are inevitably only a partial description that requires particular expertise for correct understanding.
- It is not possible to communicate all that an individual or collection of people do know about a system or how it might be improved; a complete description of what is known would require years to produce and years to consume.
- Among a collection of people, even with similar spans and depths of knowledge in an area, there will be disagreements, some that can be resolved and some that are fundamental; the conventional answer is to insist on achieving agreement, even though the substance of the disagreement may itself provide more accurate insight into a correct solution than either individual or consensus viewpoint.

- Even if people are able to correctly characterize needs at a point in time, needs as well as enabling technologies change over time; requirements that only describe what is needed currently will be incorrect at other times in the future.
- Both natural language and graphical notations are frequently understood differently by different people, particularly when lacking similar expertise; two examples of gaps in communications are between users and developers and between systems engineers and software engineers.

There are other factors, in the nature of requirements, that can also lead to uncertainties about the needed product:

- Users typically view their needs in terms of being able to accomplish their job and new or improved capabilities that would enhance this; this view is often constrained by inaccurate assumptions about what is possible and what can and cannot be changed.
- When a product built to address users' needs is deployed, it often changes the users' perceptions not only of what their needs are but also of what is possible, leading to their needs "changing" yet again.
- Requirements being only approximate descriptions of needs and constraints on potential solutions may in fact omit information that the customer knows and assumes but that the product developer does not.
- Requirements are frequently not limited to what is absolutely needed but also reflects the customer's perception of what is desirable without distinguishing between these; this in fact often precludes options that would allow the developer to make better tradeoffs in creating a product that is a best fit to purpose within given cost and schedule constraints.

Using Uncertainty in Writing Better Requirements

Parnas and Clements [2] argue for the ideal of a rational process for the design and building of (software) products. As part of this, they characterize requirements as a definition of the expected observable behavior of a needed product, sufficient to answer developer questions about what is to be built. A way to understand this is to recognize that requirements constitute a *model* of a needed product. From this perspective, requirements should define the capabilities that a product needs to enable for its users.

However, it is not enough to describe a product at a fixed point in time and with uncertainties hidden. In fact, Parnas and Clements argued that a rational process is not usual practice because of incomplete and inaccurate information in documentation caused by underlying issues in how documents are written (e.g., poorly organized, stream of consciousness exposition, poorly and inconsistently written by multiple authors, dispersed repetition of related or conflicting information, confusing and inconsistent terminology, narrowly conceived). In fact, these problems still exist and are often symptoms of false certainty. Authors must produce requirements that appear certain even if uncertainty has not been properly resolved. No means is given to indicate areas of uncertainty, doubt, or likely change that are left to be resolved.

A common effect of resolving uncertainty with arbitrary decisions is to prematurely limit solution options. Not having a means of specifying requirements so as to permit alternative solutions, the customer may instead resort to describing a particular solution

that has worked for similar problems in the past. By hiding the uncertainty and precluding further analysis, the developer may be forced to adopt the described solution rather than having the option of developing and evaluating other potentially more appropriate solutions.

A factor in being able to evolve a product as user needs change is the ability to recover the rationale for why requirements are what they are. Some products today include obsolete capabilities and are difficult to change simply because no one is sure why the product does all that it does, why it is built the way it is, and whether any aspects of what it does are no longer needed by users. A natural by-product of focusing on and explicitly analyzing uncertainties is that the rationale for the resolution actually chosen will be documented. By capturing this rationale, we have at least a partial characterization for the product not only as it finally exists but also as it might have been differently done, giving a basis from which to revise the product when needs change.

An Existing Approach for Limited Accommodation of Uncertainty

Some experience already exists with building products with a focus on uncertainty. When an organization has a need to build multiple products, in support of customers having similar but not identical needs, a product line approach [7, 8] provides a process for building a set of similar products from a common base of reusable software, documentation, and test assets. This approach depends on identifying precisely the ways in which the products needed will be alike (commonalities) and the ways in which they will differ (variabilities). The techniques for identifying how products will differ and in resolving those differences to create a specific product is similar to the model proposed for identifying and resolving requirements uncertainties in general.

With a product line approach, not all types of uncertainty are addressed but only those related to customers' changing or diverse needs. Uncertainties related to potential changes in customer needs or to needs that differ among customers are systematically identified and formulated as decisions that will be resolved late in the production of each specific product in consultation with the individual customer for that product. The ability to resolve these uncertainties in different ways is systematically engineered as production options. This provides the means to deliver a customized product to each customer and to deliver a revised product to each customer as their needs change. Identifying decisions that encapsulate the implications of diverse and changing customer needs on product requirements for a set of customized products is integral to the concept of a product line. This approach also provides the means to rapidly build alternative solutions to particular customer needs as a means to helping the customer find the best fit to their needs.

A Strategy for Comprehensively Accommodating Uncertainty

A recent National Academies study [4] has recommended that system requirements ("big R") for IT systems be defined strictly and fixed at the mission capabilities level and that more detailed requirements ("little r") continue to be developed and refined throughout the acquisition process. Consistent with the challenges of uncertainty, this study advised that requirements evolve progressively through ongoing interactions with end users and assessments of available technologies. The study alluded to a recent Joint Capabilities Integration Development System policy that prescribed a similar approach.

Uncertainty in requirements is not a “problem” to be eliminated. Recognizing and properly exposing uncertainty is an aid to communicating more effectively about needs and potential solutions. Some uncertainties, once recognized as such, can be resolved through an analysis of alternatives and tradeoffs during the development process; others are inherent aspects of the problem being addressed and require different resolutions over time or for different customers. From experience with product lines, there are good techniques for expressing uncertainty which can guide developers in providing needed flexibility with mechanisms for tailoring and product customization to better accommodate the needs of customers over time.

A strategy comprising three elements will give the means to better expose and resolve or accommodate uncertainties in requirements:

- View product development as a process whose ancillary purpose is the elaboration, refinement, and correction of requirements as initially defined.
- Document all differences and their implications when domain experts have differing views on any aspect of requirements.
- For any aspect of requirements for which there are alternatives, that is the subject of tradeoffs, or that may change in the future, document the rationale for its current realization in comparison with identified alternatives.

These elements together are meant both to improve the requirements as a description of the product being acquired and also to provide the basis for revising those requirements as understanding of needs improve or when needs or technology change in the future. When uncertainties exist and are resolved, capturing the rationale provides valuable insight to future developers. When needs change, this rationale provides a basis for understanding the implications of having to differently resolve those previous uncertainties in order to revise the product. To build a product, all requirements uncertainties have to be resolved in some way to finally build a product but the goal is to not resolve an uncertainty prematurely or for all time.

A key focus suggested in a proposed roadmap for improving software producibility [5] was bridging the conceptual gap between customers and product developers, based on recognizing that there are alternative ways of expressing any problem and many potential solutions that can result. With this perspective, no specific expression of requirements is the “right” one; rather different expressions may be suitable for different purposes. However, underlying all valid expressions are a set of assumptions about certainty, which aspects of needs and associated potential solutions are intrinsic and fixed and which are tradable and changeable. In product lines, these assumptions, of commonality and variability, provide a framework in which true certainties provide a framework within which uncertainties are identified as choices that customers and developers must make through an ongoing process of evaluation and refinement over the life of a needed product.

As is true for product lines, a shift to uncertainty-based acquisition does not require major changes in acquisition policy [3] but rather only a change in the level at which programs are required to establish binding requirements. These should be at the level of observable mission-enabling capabilities to be achieved through an iterative process of learning and refinement rather than with a premature over-constrained specification of a specific top-down solution to narrowly conceived needs. This will require changes in

the practice of acquisition, replacing the illusion of certainty with a process that, by exposing uncertainties, builds a stronger foundation for the efficient, predictable delivery of correct and effective capabilities needed by customers.

References

1. DoDI 5000.02, "Operation of the Defense Acquisition System", USD(AT&L), 8 December 2008.
2. D.L. Parnas and P.C. Clements, "A Rational Design Process: How and Why to Fake It", IEEE Trans. on Software Engineering 12 (2), February 1986 , 251-257.
3. G.H. Campbell, "A Software Product Line Vision for Defense Acquisition" (CMU/SEI-2002-TN-002), CMU Software Engineering Institute, June 2002.
4. *Achieving Effective Acquisition of Information Technology in the Department of Defense*, The National Academies Press, Washington, D.C., 2009, pp. S4-S5.
5. G.H. Campbell, “Advancing Producibility for Software-Intensive Systems”, Software Tech News 11 (4), December 2008, 13-17.
6. *Creating a DoD Strategic Acquisition Platform*, Defense Science Board Report, April 2009.
7. G.H. Campbell, S.R. Faulk, and D.M. Weiss, *Introduction to Synthesis*, Software Productivity Consortium, June 1990. <http://www.domain-specific.com/PDFfiles/IntroSyn.pdf>
8. P.C. Clements and L.M. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2002.

Acknowledgements

The views expressed here are solely those of the author and do not represent positions of the Software Engineering Institute, Carnegie Mellon University, or the U.S. Department of Defense. Comments by John Robert, Dr. Ken Nidiffer, Patricia Oberndorf, and Terry Dailey were very helpful to me in improving this paper.

Biography

At CMU’s Software Engineering Institute (SEI), Grady Campbell identifies, develops, and transitions improvements in the process and practices of software acquisition and engineering. In the early 1990’s, Mr. Campbell was responsible at the Software Productivity Consortium for conceiving and developing the first comprehensive software product line methodology. Subsequently, he was an independent consultant in software product lines and a Visiting Scientist to SEI’s Product Line Practices Initiative. Other highlights in his 40 years of experience include a Naval Research Laboratory Software Cost Reduction project and a project to build an application generation environment based on adaptable software.