On the Need for a System Need Statement

Michael. J. Ryan
School of Engineering and Information Technology, The University of New South Wales in Canberra, Northcott Drive, Canberra, ACT, 2600, Australia.
m.ryan@adfa.edu.au

Abstract. This paper proposes that, since the system need statement is the antecedent of all subsequent system requirements, it should possess the same attributes that are required of its progeny, including taking the form of a single sentence. A process to guide the development of a system need statement is also proposed.

Introduction

Despite this universal agreement that a system need statement (or problem statement) is an essential start point for a system development, there strangely little guidance on its form, nor on any methods for its development. Based on the common English definition of a statement as an account of facts, it would seem that any form of statement (from a sentence to several pages) is acceptable and that its contents are self-evident in that no particular process is required to guide its creation. The aim of this paper is to propose suitable attributes and an appropriate form for the system need statement, and a process for its development.

Role of the System Need Statement

The need for a good problem definition is not a new notion—for example: 'A problem is half-solved if properly stated.'(John Dewey); and 'Every problem has in it the seeds of its own solution.' (Norman Vincent Peale). Or, as put in the negative by Robert Mager, '…if you're not sure where you're going, you're liable to end up someplace else' (Mager, 1975). Poor definition of the requirements for a system will invariably result in the development of a poor system. Requirements definition is therefore critical.

System requirements are not collated randomly—requirements engineers undertake requirements analysis which involves, inter alia, decomposing a statement of customer need through a systematic exposition of what the system must do in order to meet that need (Grady, 2006, p. 7). Gathering, analysing and decomposing requirements is an essential step to ensure that the system, once built, actually meets stakeholders' needs (Daniels & Bahill, 2004). High-level requirements are decomposed further into a number of lower-level functions which are then grouped, along with other derived functions, through a requirements analysis and grouping process (IEEE, 2005) into a system-level description called the System Specification. The System Specification is a functional architecture in that it describes what is required of the system, rather than how it is to be implemented. The System Specification is therefore the basis of the functional architecture of the system—it is a collection of system-level requirements in functional groupings.

The requirements in the System Specification are then further decomposed/derived into lower-level requirements which are then re-grouped from the functional groupings of the System Specification into physical groupings (called configuration items) which are documented in a number of Subsystem Specifications (Faulconbridge & Ryan, 2003). The Subsystem Specifications now define how the system will be implemented. The mapping of the functional groupings of requirements in the System Specification into the physical configuration items in the Subsystem Specifications informs the definition of project deliverables, which is documented in the work breakdown structure (WBS) (PMI, 2008).

For well-behaved systems, the requirements in both the System and Subsystem Specifications are normally grouped in a hierarchical fashion (albeit grouped functionally in the former, and physically in the latter). This hierarchy is useful because it allows requirements engineers to decompose the system into tasks of a complexity that can be more easily be understood and communicated, as well as managed within the limitations of humans and our organisations.

A hierarchical description of a system is also useful in that it supports requirements traceability, which is also an essential element of effective systems engineering as well as project management. Through *forward traceability*, design decisions can be traced from any given system-level requirement (a parent requirement) down to a detailed design decision (a child requirement). Similarly, through *backward traceability*, any individual design decision (any child) must be able to be justified by being associated with at least one higher-level requirement (a parent). This traceability is important since the customer must be assured that all requirements can be traced forward and can be accounted for in the design at any stage. Further, any aspect of the design that cannot be traced back to a higher-level requirement is likely to represent unnecessary work for which the customer is most probably paying a premium. Traceability therefore supports the project management function of project scope management.

Backwards traceability is particularly useful in that is provides confidence that the requirements database has remained within scope by ensuring that all lower-level requirements can be justified by having been decomposed or derived from at least one higher-level requirements. That is, an orphan requirement must, by definition in such a process, be out-of-scope.

Traceability is assisted by a hierarchical numbering of requirements. As illustrated in Figure 1, Requirements 1.1.1, 1.1.2 and 1.1.3 are children requirements of parent Requirement 1.1, which is a child of Requirement 1. For Requirements 1.1.1, 1.1.2 and 1.1.3 to be children of parent Requirement 1.1, they must be able to be derived or decomposed from it—that is Requirement 1.1 is a superset of Requirements 1.1.1, 1.1.2, and 1.1.3; and Requirement 1 is a superset of Requirements 1.1 and Requirements 1.2.

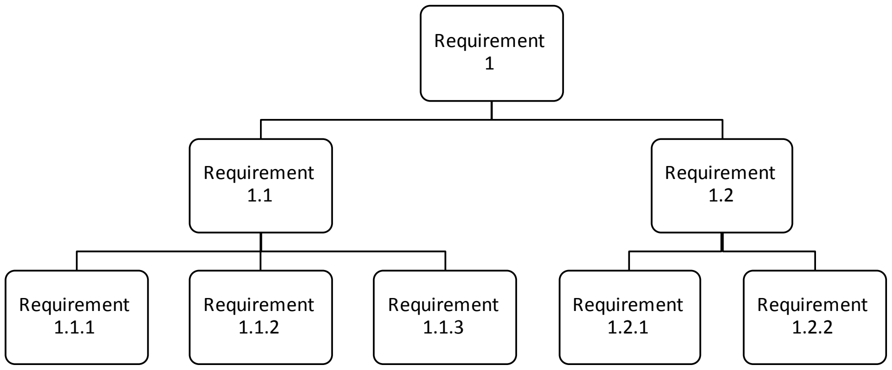

Figure 1. The hierarchical decomposition/derivation of requirements.

Backwards traceability from children to parent requirements in Figure 1 leads to a useful observation related to heredity, in that the highest-level requirements also ought to be traceable to some higher statement. As illustrated in Figure 2, it follows therefore that there is a single parent statement which serves as the highest-level requirement statement back to which all subsequent requirement statements should be able to be traced. This then is the principal role of the need statement in systems/requirements engineering—it is the parent statement that provides the seminal point for requirements engineering, and is the ultimate system requirement (Bahill and Briggs, 2001) from which all other requirements and designs flow (Grady, 2006, p. 7).

Once again it is clear that the need statement performs an essential role since its contents must be sufficient to start the system design. While it is elaborated further by the high-level requirements that are gathered subsequently, the need statement cannot be at odds with them, nor can it contain information which is not then elaborated on by at least one subsequent requirement.

That is, there is a hierarchy—the need statement provides the base statement from which the high-level requirements are drawn. The need statement therefore must be a complete description of the system and contain something pointing to every element of the system.

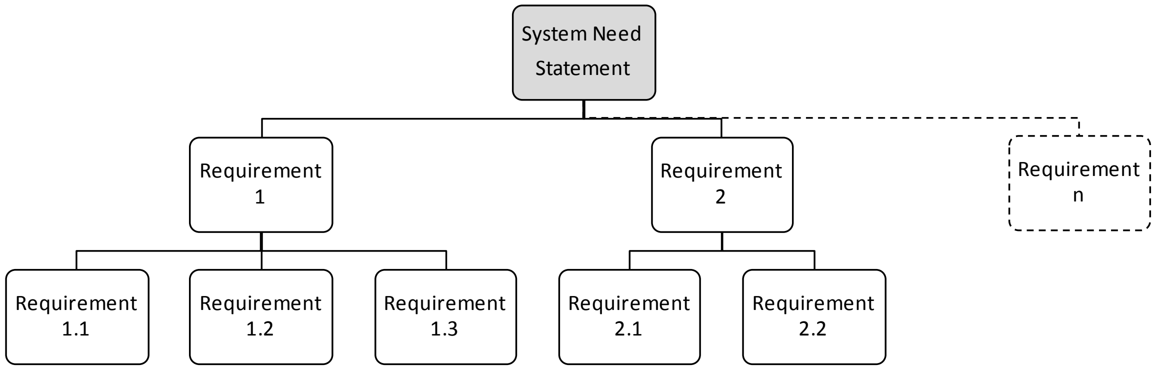

Figure 2. The system need statement as the common ancestor of all subsequent system requirements.

Desirable Attributes of a System Need Statement

To extend the hereditary relationship from the previous section, it follows that the parent need statement should have the same desirable attributes as those requirement statements that are its children. At the system level, requirements should be necessary, unique, singular, complete, correct, unambiguous, feasible, independent of the method of implementation, justifiable, and verifiable (IEEE, 1998). The parent statement—the system need statement—should therefore have similar attributes:

- **Necessary.** It should be obvious from the system need statement that the system is necessary—that is, for example, that it does not already exist in some form.
- **Unique.** The system need statement must describe a unique system using terms that ensure that the scope of the system is clearly identified and is sufficiently detailed so that this system is differentiated from any others.
- **Singular.** The system need statement cannot be a combination of need statements for two or more systems. That is, each system must have its own need statement, which must be unique.

- **Complete.** The system need statement is the seminal statement from which the entire system design is derived. It therefore should be complete—any subsequent aspect of the system must be able to be traced back to at least one element of the system need statement.
- **Correct.** The system need statement must contain only those elements that pertain to the system—that is, since it defines the system scope, each element must be within scope. Therefore, each of the elements in the statement must be examined carefully for correctness to confirm that it should be included.
- **Unambiguous.** The system need statement must remain within the intuition of the stakeholders and all must share a common understanding of it. It must therefore be unambiguous and convey the same meaning to each reader, regardless of their background. Ambiguity in system definition and subsequent requirements has been (Gause & Weinberg, 1989) and continues to be (Buede, 2009) a significant obstacle to project success.
- **Feasible.** The system must be considered feasible in the judgement of all stakeholders. If the system need statement is not feasible, it is not likely that the requirements derived from it will be feasible either.
- **Independent of implementation.** As is expected of all subsequent requirements that flow from it, the system need statement itself must be solution-independent.
- **Justifiable.** It is good practice that each lower-level requirement has an associated rationale recorded to describe the reason for its existence. (Faulconbridge and Ryan, 2003) It follows that the system need statement should also contain a system-level rationale.
- **Verifiable.** Each requirement must be verifiable, in that it can be proved that the system meets or possesses the requirement. Even at the high level of abstraction of the system need statement, it must be possible to be able to define the tests that will be used to demonstrate that the system is successful. While such a demonstration is verification for individual requirements, for the need statement, of course, such proof is validation.

The Format for a System Need Statement

Of these desirable attributes for a system need statement, the attributes of unique, necessary, complete, correct, feasible, and verifiable are a related to the content of the statement and must be considered carefully during its drafting (addressed in the next section). The attributes of singular, unambiguous, independent of implementation, and justifiable are similarly a matter of the content, but are also assisted by a focus on an appropriate format for the statement. The attainment of these latter attributes is greatly assisted if the following guidelines are used for the format of the system need statement:

- **A single sentence.** To ensure that the system need statement is singular, it must be able to be expressed in a single sentence. In good written English, a sentence may be defined as encapsulating a single thought—if we want to express two thoughts, a second sentence is normally required. The discipline of writing a single sentence to describe the system need therefore keeps us at the level of abstraction that defines the system as a ‘single thought’, which is the level of abstraction we are seeking at this stage in the project in order to define a unique endeavour. If we feel the need to start a second sentence, we run the risk of adding a second ‘thought’, which may well be describing a second project.

- **No conjunctions.** In addition to being expressed in a single sentence, the system need statement should not contain any conjunctions if it is to be singular. The statement cannot state, for example, that the purpose of the system is to ‘do this… and … do that…’. The presence of the ‘and’ in the statement implies that there are two purposes, and therefore two systems. Having said that, we can of course have conjunctions in phrases and adjectival clauses. For example, we can say that a clause in a need statement proposing a system that is ‘… comfortable and secure …’; but we cannot have a system that is ‘a medical centre and riding school…’ because two (very different) systems are implied. Note that we could have a system need statement that is for a doctor surgery and a dental surgery and a physiotherapy practice …’, but only because we can (and therefore should) logically group those functions into a need statement for a ‘medical centre’. As a general rule therefore, we should not have any conjunctions because, if elements of the same type are joined by ‘and’, there is a group noun or adjective that can replace them.
- **Contain no more than 5- 7 concepts.** The system need statement cannot contain any more than five-to-seven key elements if it is to remain within the intuition of the reader (Miller, 1956) and to remain unambiguous.
- **Include an ‘in order to’ clause.** The system justification, or rationale, can be made explicit by the inclusion of an “… in order to…” clause at the end of the system need statement. This clause then ties the system need statement back to the business case for the system.
- **Avoid physical terms.** To ensure that the subsequent system need statement is independent of the physical implementation means, the elements of the need should always be couched in functional, not physical, terms. The system need statement must not imply any particular physical solution.

A Process to Guide the Development of a System Need Statement

The development of the system need statement has three main activities (described in more detail in the remainder of this section):

- Identify candidate elements of the system need statement by simply listing them.
- Iterate the set of system need statement elements by a process of review and test (determine whether every element should be included, and what other elements might be included).
- Form the selected elements into a single concise system need statement.
- Confirm the system need statement with stakeholders.

Identify Candidate System Need Statement Elements

Crafting a complete balanced need statement in one (top-down) pass is very difficult for most stakeholders, particularly if the system is large, complicated, has a number of interfaces, and/or is a new system (that is, it contains elements with which they have no direct experience). It is therefore often productive to ask stakeholders to simply list (in a bottom-up manner) the key words or phrases that are candidates to be considered for inclusion in the need statement. Once each of these candidate need elements has been assessed for suitability, these are grouped and re-grouped into five-to-seven elements at a consistent level of abstraction.

The identification of candidate need elements can be obtained through a workshop with key stakeholders, through individual interviews with those same parties, or through any other appropriate requirements engineering technique (see for example, Alexander and Stevens, 2002; Goguen and Linde, (1993); Hull, (2002); Katonya and Sommerville, (2000); Somerville and Sawyer, (1997); Young, (2001); Young, (2004); and IEEE-STD-1223 (1998)). Care must be taken to ensure that each of the elements of the system need statement has the desirable attributes of being unique, necessary, complete, correct, feasible, and verifiable.

Note that when designers and stakeholders are considering these high-level elements, they are in fact articulating the major requirements of the system (at least as they currently understand them). The inclusion or exclusion of an element here will significantly change the design, but that effect can be considered at a useful level of abstraction that is within the intuition of the stakeholders and their current level of understanding.

For example, when developing a need statement elements for a domestic burglar alarm, attendees at a stakeholder workshop might list such candidate elements as:

- Alarm properties: flexible, reliable, sustainable, easy to use, and affordable.
- Alarm functions: deterrence, detection, classification, and reporting of unauthorised entry.
- Rationale: to alert the resident that the security of the residence has been compromised.

Iterate (Review and Test) Need Statement Elements

Since the need is to be developed iteratively, any reasonably comprehensive set of need elements represents an appropriate place to start the iteration. Having made a start, stakeholders iterate their understanding and converge to an agreed set of elements through a process of review and test. Briefly, there are two main tests—whether every need element identified should be included; and whether other important elements have not been included. When conducting those tests, stakeholders consider the question: “Would the resultant system be significantly different if this need element was included or omitted?”.

For example, the need statement for our domestic burglar alarm example will have a need element that refers to ‘residents’. We might consider whether the definition of resident to any lower level of detail would change the nature of the system. If the system is to be used by users of all ages, ethnic backgrounds, languages, abilities, and so on, then the need element ‘resident’ should not be defined any further so that lower-level designers infer from the need that all the system should be able to accommodate every nature of user. If however, support is only required for adult, English-speaking users, then the need element should be reworded to make the nature of resident explicit, because the resultant system will be identifiably different as a result.

Functional grouping should also be considered at this stage. Since the highest level of abstraction is sought in the need statement, stakeholders should identify when need elements can be combined. This aggregation will also assist in reducing the length of the sentence. The content of the functional groups should not be discarded, however, as those terms will no doubt be explicit in the subsequent decomposition. For example, the list of burglar alarm properties (flexible, reliable, sustainable, easy to use and affordable) could be grouped into ‘market leading’, if that term was useful and meaningful in the system context. On the other hand, it may be considered that the grouping of ‘deterrence, detection, classification, and reporting of unauthorised entry’ into ‘preventing unauthorised entry’ is not appropriate, since ‘preventing’ does not capture the meaning contained in the individual elements of ‘deterrence, detection, classification, and reporting’.

Note that the iteration of this task is very productive. In a facilitated workshop, for example, participants could be asked to diverge and suggest candidate elements of the need by listing them, and then be guided to convergence on an agreed set of elements through the positive and negative tests outlined above. Van Gundy (1992) suggests that this divergence/convergence is part of a creative way to address problems in a structured manner.

Although understanding of the elements of the need is now sufficiently mature to proceed to decomposition/derivation, further iteration is required. In fact, stakeholders should not at all be confident that they have captured every element of the need since they cannot assume that the original statement of a problem is necessarily the best, or even the right one (Maier & Rechtin, 2000).

Form Need Statement

Once the major elements of the need have been identified, stakeholders pause with that level of understanding and form the elements into a single concise sentence—the system need statement. With regard to purpose, the need statement should always include a short “… in order to …” clause at the end of the need statement describing why the system is to exist—this is an important step because this portion of the need statement is in fact a system-level rationale, which begins the good requirements-engineering practice of recording a rationale for each functional requirement. For example, a draft need statement for our domestic burglar alarm might be” ‘To provide a market-leading domestic alarm system that can deter, detect, classify, and report unauthorised entry to a residence in order that the resident is aware of the residence’s state of security’.

There is often tension at this point between the need to be concise and the desire to include in the need statement every aspect of the design that is important (at the appropriate level of abstraction)—the statement should be as specific as possible so that the detail is not missed. One solution to this dichotomy is to provide for selected key words a footnote in which the detail is described. Alternatively, and probably most usefully, a project glossary can be developed, in which the term in question can be explicitly defined in the context of this particular project. For example, if there are to be a number of classes of user of the system, the need statement can simply refer to ‘users’ and the project glossary can be used to define the various classes of user in the required detail. In our burglar alarm example, there is great utility in defining the terms ‘resident’, ‘unauthorised entry’, and ‘residence’ in a glossary (as well as perhaps ‘deter’, ‘detect’, ‘classify’ and ‘report’).

Once stakeholders are satisfied that the need statement is suitable to use as a start point for the design (recognising that they will invariably return later and modify the need statement as they understand the problem better), they can now begin the process of need analysis whereby they identify the tasks that are required by the need statement and those that are inferred (derived) from it. Most of the required functions are explicitly stated by the need statement. From the need, however, other required functions can be inferred that have not been explicitly stated.

While requirements engineers could move on from here with explicit and derived need elements, it is probably useful to revisit the need and re-write it so that all need elements are explicitly contained. This assists subsequent traceability and also results in a more-complete need statement to be presented in subsequent project documentation.

Confirm Need Statement

The need can be confirmed within the group of stakeholders that developed it or, more usefully, it can be communicated to a wider group and used to elicit comment that can inform subsequent revision. Because the need is effectively a single-sentence description of project scope, early communication facilitates a shared understanding of the system among all stakeholders.

Conclusion

The system need statement is an essential start point for system development and should therefore be developed with more than a passing interest. To be useful as the antecedent for all subsequent system requirements , the system need statement should share the attributes of system requirements: in particular, it should be couched in a single sentence and be necessary, unique, singular, complete, correct, unambiguous, feasible, independent of the method of implementation, justifiable, and verifiable (testable).

Of these desirable attributes for a system need statement, the attributes of unique, necessary, complete, correct, feasible, and verifiable are a related to the content of the statement and must be considered carefully during its drafting (addressed in the next section). The attributes of singular, unambiguous, independent of implementation, and justifiable are similarly a matter of the content, but are also assisted by a focus on an appropriate format for the statement. The attainment of these latter attributes is greatly assisted if the system need statement is confined to a single sentence containing five-to-seven concepts (couched in functional terms with no conjunctions) and justified by an ‘in order to’ clause.

References

Alexander, I.F. and R. Stevens. 2002. Writing better requirements, London: Addison-Wesley.

Bahill, A.T. and C. Briggs. 2001. The systems engineering started in the middle process: A consensus of systems engineers and project managers, *Systems Engineering*, 4(2).

Buede, D. 2009. The engineering design of systems: Models and methods, Hoboken, N.J.: John Wiley & Sons.

Daniels, J. and T. Bahill. 2004. The hybrid process that combines traditional requirements and use cases, *Systems Engineering*, 7(4).

Faulconbridge, R.I. and M.J. Ryan. 2003. Managing complex technical projects: A systems engineering approach, Boston, M.A.: Artech House.

Gause, D.C. and G.M. Weinberg. 1989. Exploring requirements: Quality before design, New York: Dorset House Publishing.

Goguen, J. and C. Linde. 1993. Techniques for requirements elicitation, *International Symposium on Requirements Engineering*, San Diego, C.A.: IEEE Computer Society Press, 152–164.

Grady, J.O. 2006. System requirements analysis, Burlington M.A.: Academic Press.

Hayes, D.S. 2000. Evaluation and application of a project charter template to improve the project planning process, *Project Management Journal*, 31(1), 14−23.

Hull, M.E.C., Jackson, K. and Dick, A.J.J., et al. 2002. Requirements engineering, London: Springer.

IEEE Computer Society (IEEE). 2005. IEEE-STD-1220-2005—IEEE standard for application and management of the systems engineering process, New York: IEEE Computer Society.

The Institute of Electrical and Electronic Engineers (IEEE). 1998. IEEE-STD-1233-1998, IEEE guide for developing system requirements specifications, New York: The Institute of Electrical and Electronic Engineers.

Katonya, G. and I. Sommerville. 2000. Requirements engineering: processes and techniques, Chichester: John Wiley and Sons.

Kerzner, H. 2009. Project management: A systems approach to planning, scheduling, and controlling, Hoboken, N.J.: John Wiley & Sons.

Mager, R.F. 1975. Preparing instructional objectives, Belmont CA: Pitman Learning.

Maier, M.W. and E. Rechtin. 2000. The art of systems architecting, Boca Raton: CRC Press.

Miller, G.A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information, *The Psychological Review*: 81−97.

Project Management Institute (PMI). 2008. Guide to the project management body of knowledge, Newtown Square, PA: Project Management Institute (PMI).

Pyzdek, T. 2003. The six sigma project planner, New York: McGraw-Hill.

Sommerville, I. and P. Sawyer. 1997. Requirements engineering: A good practice guide, Chichester: John Wiley and Sons.

Van Gundy, A.B. 1992. Idea power: Techniques and resources to unleash creativity in your organisation, New York: American Management Association.

Young, R.R. 2001. Effective requirements practices, London: Addison-Wesley.

Young, R.R. 2004. The requirements engineering handbook, Norwood, MA: Artech House.

BIOGRAPHY

Dr Mike Ryan is a Senior Lecturer with the School of Engineering and Information Technology, University of New South Wales, at the Australian Defence Force Academy (UNSW@ADFA) in Canberra. He holds Bachelor, Masters and Doctor of Philosophy degrees in electrical engineering as well as a Graduate Diploma in Management Studies. His research and consulting interests are in the fields of communications and information systems architectures, project management, systems engineering, and requirements engineering. He is the author or co-author of nine books, three book chapters, and over fifty technical papers and reports.