Getting the Message Across - Creating Accessible Guides to Promote the Adoption of Systems Engineering Good Practices

Tim Armitage
Arup
The Arup Campus
Blythe Gate, Blythe Valley Park
Solihull B90 8AE
United Kingdom
+44 121 213 3311
Tim.Armitage@arup.com

Deborah Chin
New York City Transit
2 Broadway
New York, New York 10002
USA

Bruce Elliott
Arbutus Technical Consulting
120 Grange Drive
Swindon SN3 4LD
United Kingdom
+44 1793 832028
bruce.elliott@arbutus-tc.co.uk

Ed Kastenhuber
Parsons
100 Broadway
New York, New York 10005
USA

Michael E. Krueger CSEP
ASE Consulting LLC
3405 S Main St Unit # H
Santa Ana CA, 92707-4326
+1 714-708-2993
michael.krueger@ase-consult.com

Abstract. The authors believe that better deployment of Systems Engineering (SE) would benefit the rail and transit sector. They have formed an INCOSE working group to develop guidelines which advance this objective, starting with the topic of requirements engineering.

Although there is already sound guidance on requirements engineering, its influence within the sector has been limited. In hopes of changing this, the authors have created a short requirements guide, illustrated with practical examples and written in plain language. The authors' premise is that the benefits of increased accessibility outweigh any loss in comprehensiveness and precision.

The guide is being produced following an iterative process involving peer review by INCOSE groups. The current working version is attached to this paper. The authors believe that following this guide would benefit rail and transit projects. The approach described may be of value to other SE practitioner trying to convince skeptics of the value of SE.

Introduction

Background: All of the authors work within the rail and transit sector. Although this sector practices some of the principles of Systems Engineering (SE), it has not embraced the SE approach in its entirety. The authors believe that a comprehensive and appropriate deployment of SE would bring the same benefits to their sector that it had done in others. They have joined the INCOSE Intelligent Transportation and Transit Systems Working Group (ITTSWG) because they wish to contribute to the group's objective to "*promote the development and tailored application of SE best practices to ground transportation and transit systems, including public and private interests and seamless inter-modal interfaces*."

Within the ITTSWG the authors have formed a product development subgroup with a mission to, "*deliver products to provide guidance to SE practitioners in Rail and Transit which, in conjunction with existing SE guidance, helps to adopt good practice in SE in their domain*."

The long-term objective is to produce comprehensive SE guidance. However due to the anticipated length of this effort the authors have set out to produce a series of short documents that will focus on key individual topics. Because the authors see requirements as a key aspect of SE, they have started with a document providing guidance on requirements engineering.

Problem: There are plenty of sources of sound guidance on requirements engineering which are applicable to rail and transit projects. However, by striving to be all-encompassing, these comprehensive but lengthy guides may not be the best tool for the introduction of SE to the intended audience. Clearly the objective of "helping to adopt good practice in SE" is not likely to be furthered by writing more of the same; a different approach is required.

The authors observe that most guidance on requirements engineering is written by practitioners for practitioners. Comprehensiveness is a virtue in this context. But requirements engineering practitioners are convinced of the value of requirements engineering by definition. Encouraging the adoption of good requirements engineering practice requires convincing decision-makers who are not practitioners: project managers, design managers, engineering directors and so forth. The authors believe that their target audience finds the existing documents too long, uncomfortably abstract, and written in unfamiliar language. We want them to invest in new methods and then we require them to invest significant time and energy in understanding the methods in order to assess their value. We have created a serious obstacle to change.

Approach: The approach taken then is to produce an *accessible* guide: short rather than long, illustrated with practical examples and written as much as possible in plain language. The authors acknowledge that there is a price to be paid in comprehensiveness and precision but consider that it is still possible to produce sound and useful guidance.

The authors aim to complement, not replace, existing guidance and warn the reader that the content of the guide is simplified while referring them to more comprehensive sources. Indeed the chosen title of the document, the "Requirements Survival Guide" ("RSG"), carries, right from the start, the clear implication that the reader will only be told the most important points.

Process: The guide is being created as a team exercise using the following process:

1. Review selected existing sources of guidance including cross-sector guidance (INCOSE 2006, CM-SEI 2007, EIA 1997 and ISO 2002), two rail and transit documents (FHA 2007, RSSB 2000) and, for perspective, one document from another sector (NASA 1995), together with the authors' own knowledge of best practice.

2. Select the most important points from the guidance reviewed. In doing this, the authors imagined that they had to brief a junior colleague just about to start a requirements engineering assignment on the things they most needed to know in a short elevator ride.
3. Prepare a strawman draft guide, expanding the points into full descriptions.
4. Iterate over several revisions, changing the authorship of each section between revisions to avoid any sense of personal ownership of any of the text.
5. Expose the document to review at the ITTSWG workshop in the 2007 International Symposium and at a meeting of the INCOSE UK Rail Interest Group. Colleagues at these meetings were generous in offering their time to critique the guide. The product development sub-group used the comments made to improve the guide.
6. Make the document available for trial use by rail and transit organizations and update it in the light of practical experience.
7. Submit the document to the full INCOSE publication and peer review procedure.

Steps 1 through 5 have now been completed and step 6 is about to start.

Product: The content of the current draft of the RSG is reproduced at Appendix A. The reader will notice that it employs a very limited amount of specialist vocabulary and is written in a conversational style which is designed to be easy to read. Much of the content is presented in a checklist format which is designed to allow straightforward benchmarking of existing practices. The content is illustrated with diagrams, examples and case studies.

Conclusions: It is too early to tell whether the RSG will succeed in its mission. The authors believe that they have succeeded in packaging, within the tight constraints that they set themselves, useful guidance and guidance which would have improved the fate of projects they are aware of had it been used.

Moreover, while the review has identified some variation of opinion on the best way to meet the need for short, straightforward guidance, what has never been challenged by the reviewers is the need for such guidance.

The need arises in the rail and transit sector because this sector has a tradition of delivering projects without explicit reference to SE. Other sectors have similar traditions and, even within sectors whose traditions do include SE, there are skeptics. The authors consider that their approach may be of value to other SE practitioners trying to convince skeptics of the value of SE.

Encouraged by feedback received, the authors will continue to do their best to meet this need within their own sector. They have found the business of explaining their work in concise and straightforward language difficult. However they observe that systems engineers' contribution to successful systems can only be realized in the context of a multi-disciplinary team and would encourage all systems engineers in all sectors to be prepared to take the trouble to communicate tot their colleagues in their colleagues' language as they consider that the rewards in improved teamwork will handsomely repay the investment made.

References

INCOSE, *Systems Engineering Handbook,* issue 3, 2006.

Federal Highway Administration, California Division, *Systems Engineering Guidebook For ITS*, version 2, January 2, 2007, http://www.fhwa.dot.gov/cadiv/segb.

NASA, *Systems Engineering Handbook, June 1995*
Rail Safety and Standards Board, *Engineering Safety Management Issue 3 (Yellow Book 3), 2000*, http://www.yellowbook-rail.org.uk.
Carnegie Mellon Software Engineering Institute, *Capability Maturity Model ® Integration*, http://www.sei.cm.edu/cmmi.
IEEE, *Standard for Application and Management of the Systems Engineering Process*, IEEE Standard 1220-1998
EIA, *Processes for Engineering a System*, Standard ANSI / EIA-632-1998
ISO, *Systems engineering – System life cycle processes*, Standard ISO / IEC 15288:2002.

Biographies

Tim Armitage is an experienced chartered mechanical engineer, who has worked as a consultant in the marine, automotive and railway environments. With Arup, he has worked on a number of systems engineering and systems integration programmes on behalf of Network Rail, the UK's infrastructure owner and operator. He is currently responsible for the direction and management of railway projects undertaken by Arup's Advanced Technology and Research practice.

D. Chin is a Design Manager for New York City Transit (NYCT). She has over 18 years in the transit industry, 10 of which have been in a managerial capacity leading signaling and control center projects. Ms. Chin is a key advocate in the adoption of systems engineering processes for the development of all new NYCT systems. Ms. Chin holds a Bachelors of Science in Electrical Engineering from the University of Rochester, and a Masters in Public Administration from City College of New York Baruch College. She is a Professional Engineer in the state of New York.

Bruce Elliott is an experienced system and system safety engineer in rail and air traffic services. Through his leading role in developing the Yellow Book – the UK railway's handbook of engineering safety management – he has made a significant, personal contribution to improving practice across a whole industry. He is a charted engineer in the UK.

Ed Kastenhuber has been a Parsons Corporation SE consultant in the rail and transit industry for the past 5 years. With over 30 years of electrical and systems engineering experience in both defense and commercial industries, he has led and participated in numerous projects that have been implemented using a wide range of development methodologies. He intends to leverage these diverse experiences to help develop a practical and cost-effective approach to SE project management specifically tailored to rail and transit. He is currently completing a 9-month User Requirements capture effort for a major New York City Transit control center project.

Michael E. Krueger specializes in Systems Engineering Training and Consulting in the area for Requirements development, Configuration Management, and Project Management for Intelligent Transportation Systems (ITS). He has over 34 years of electronics experience from hardware and software design to large systems development. He serves both as an Instructor and a domain expert for the Federal Highway Administration in Systems Engineering. Mr. Krueger is a Certified Systems Engineering Professional from INCOSE and serves on the Certification Advisory Group and Reviewer. Mr. Krueger is the Chair of the ITTSWG and the past President of the Los Angeles Chapter of INCOSE.

Appendix: The Requirements Survival Guide

1 Introduction

Rail and Transit projects have become increasingly complex as demands to integrate new and existing systems have grown. Each of these projects will most likely include a wide array of custom and/or off the shelf software and hardware that will interact and exchange information. Industries such as Defense and Aerospace have long used Systems Engineering processes, Requirements Engineering specifically, to help to deal with this complexity.

If you happen to be providing management or engineering support for a project that fits the description above, and are perhaps more comfortable with a more traditional (for transit) Design/Construction approach, then this short introduction explaining the importance of requirements engineering is for you. If you are familiar with requirements development and tracking, but would like a quick "cheat sheet" reference to the issues involved, you have also come to the right place.

We haven't tried to tell you everything about requirements engineering in this Survival Guide – there are other, much more comprehensive texts that can do that. Instead we have highlighted key concepts in requirements – writing good requirements, capturing a complete set of user and system requirements, and structuring them into a manageable set that can be maintained over the life of the project – so that you can rapidly obtain an overview of this crucial aspect of Systems Engineering.

Good requirements engineering is the foundation for repeatable project success. If you do not have a satisfactory set of requirements, then you cannot be confident that you will meet the needs of the people who will be using your system. Additionally, there is a real risk that your project will cost more and take longer to complete then you expect.

Studies[1] conducted within Information Technology, Aerospace, and Defense industries have shown that problems encountered during a project's lifecycle usually have to do with requirements not being complete, clear, or achievable within the given budget. Case studies 1 and 2 below show that these same problems exist in the rail and transit industries.

Case study 1[2]

The UK West Coast Route Modernisation rail project had an initial budget of £2.5bn in 1998. By 2002 costs had risen to an estimated £14.5bn. 'Scope creep' arising from a lack of tight specification was identified as a key weakness in the program. The UK Strategic Rail Authority developed a clear measurable set of program outputs, along with more detailed infrastructure requirements. As a result of this and other measures, the estimated cost had been brought back by 2006 to £8.6bn.

[1] Standish research the Chaos report 1994, NASA report of the Comptrollers office

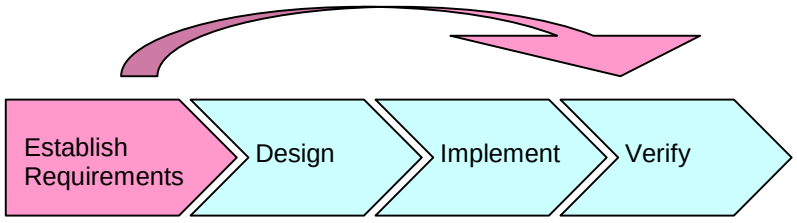
[2] Value for money report: The Modernisation of the West Coast Main Line, UK National Audit Office, November 2006

Case Study 2

A major subway control center project in the northeastern United States was undertaken over a period of a decade and a half beginning in the early 1990s. The specification development strategy for this project relied heavily on "requiring" their contractor to learn the operations and installed legacy systems environment to which their design would need to interface. It was believed at the time that this strategy would effectively substitute for identifying thorough requirements for all of the unique aspects of the subway system. Serious flaws in this approach were revealed only after the Factory Acceptance Test (FAT) effort began. The developers had not realized the extent of the exceptions to the general 'rules' and based their Automatic Routing (ARS) design on generic algorithms. FAT was planned as a four-week effort starting in April of 2003. After 21 days of testing, (80%) of the test cases had failed and 449 total variances (individual system problems reported) of all types had been taken. Most of the errors were due to location peculiarities. The railway operator assumed the job of providing the information at this point, but not before seriously impacting project cost and schedule.

We would all prefer to work on projects without surprises such as the ones described above building the system right the first time. But to achieve this we have to understand what "right" means. In other words we need clear, comprehensive, and accurate requirements.

Once they are right, the requirements will become the engine in the Systems Engineering process. Establishing the requirements is the first stage of the system development lifecycle. It directly drives the design so that the system that is implemented will meet the requirements. Later, as the simple figure below illustrates, the requirements will drive the verification process demonstrating to our end-users that we have delivered what they need. The simple figure below identifies the interactions between these development activities.



2 Requirements in a nutshell

Subsequent sections of this document will help you answer each of the following three questions:

1. Have I got all the requirements I need (and no more)?
2. Are my requirements well written?
3. Are my requirements well structured?

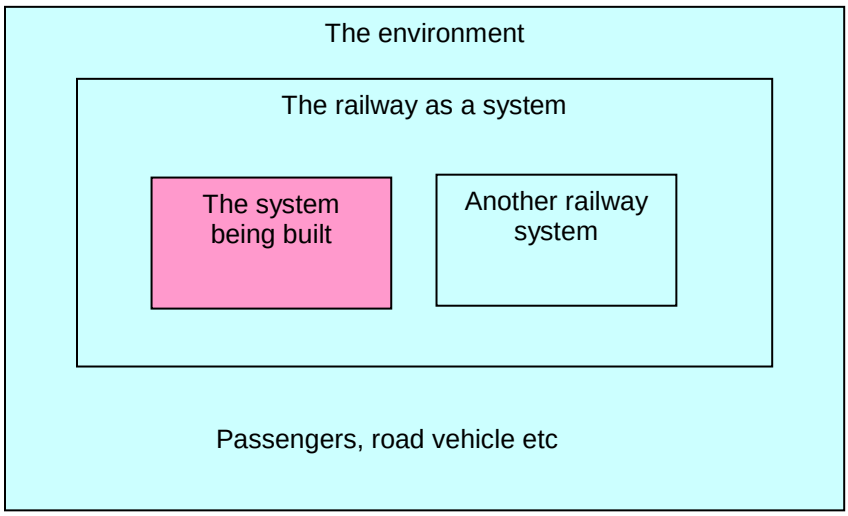
Once you are satisfied with the answers you have given yourself to these three questions, you will be well on your way towards good requirements management on your project.

Key concepts

Before we elaborate on the three important requirements questions, we will introduce some of the terminology that is used throughout the document.

A **system** is any combination of people, equipment and procedures intended to deliver some useful function. So a railway is a system and so is a train or a signaling system. A system doesn't have to have electricity running through it therefore by our definition a rule book and a bridge are both systems as well.

The **environment** or **domain** of a system is that part of the world with which the system interacts. A railway system will normally be part of a larger system. After all, the railway itself can always be regarded as a system. The environment of a railway system will normally include this enclosing system, other railway systems and things that are not part of the railway like passengers and road vehicles. The figure below illustrates the environment of a typical railway system.



Stakeholders are individuals or groups, such as operators, users, owners, maintainers, developers, trainers, and regulatory agencies that have a stake in the system that will be developed. Many stakeholders contribute directly to the definition of the system with discipline-specific specifications or standards.

A **Concept of Operations** documents the total environment and use of the system to be developed in a non-technical manner. It presents the multiple views of the system corresponding to the various stakeholders. It also includes such information as vision, goals and objectives, operational philosophies and scenarios, operational system characteristics, system constraints, institutional issues, external interfaces, and stakeholder roles and responsibilities.

Requirements for a project define what the project must deliver. The Systems Engineering standard EIA 632 defines "requirement" as "something that governs what, how well, and under what conditions a product will achieve a given purpose". Your rail or transit project will need a set of requirements defining the functions, performance, and environment of the system under development to a level that can be built.

Nearly all rail transit projects can be characterized as creating or updating some system and it is to this system that most of the project's requirements will apply. We distinguish two types of requirements:

- **User requirements** describe what the users want from the system. Typically this is phrased in terms of the beneficial effects of the system on its environment. An example might be. "The user of the metro system shall be able to monitor real-time train arrival times from the station platform" or "The user of the metro system shall be able to access real-time train schedules over the internet".

- **System requirements** describe characteristics of the system. Examples might be, “The signaling system shall provide a train separation of no more than 90 seconds” or “The metro system shall carry a minimum of 1,000 people per hour from station A to station B”.

As it happens the two user requirements examples here are qualitative while the system requirements examples are quantitative. In practice both user and system requirements will typically be a mixture of qualitative and quantitative requirements.

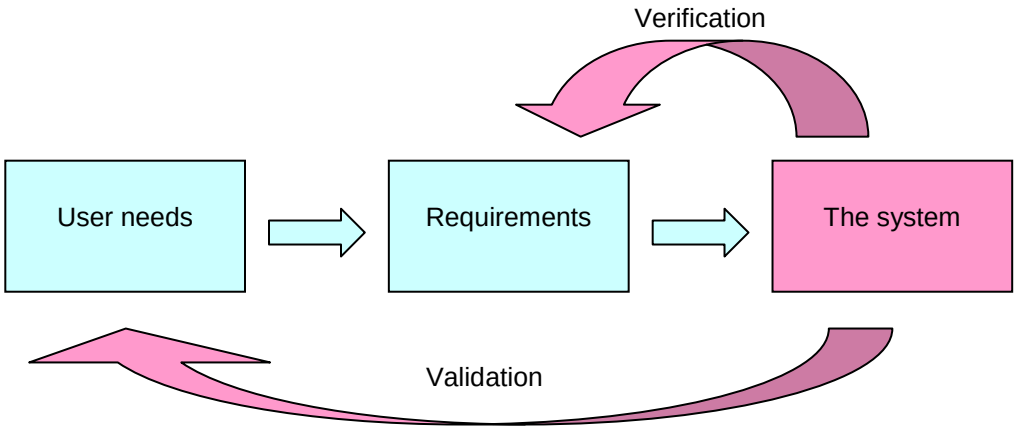
You will need the users to confirm that the user requirements correctly express their needs and desires, which will only be reliable if the users can clearly understand the user requirements. The system requirements documents may end up quite technical but, bear in mind, when considering the guidance in this note that keeping the user requirements intelligible to the users is paramount.

Typically we collect the user requirements first and then specify system requirements that will deliver the capabilities that had been defined by the user requirements. To do this reliably we need to understand and document the system’s domain**. Domain knowledge** comprises relevant facts about the environment of the system. These facts are not requirements but they are closely related to your requirements. For instance, a relevant piece of domain knowledge might be that, because station A is at an airport, the passengers will have more luggage than usual and the trains will need to stop longer at the platform than usual. If you overlooked this fact then the omission might be enough to mean that the system would not meet its user requirements even if it has met its system requirements.

It is generally a good idea to get the requirements straight before starting to design the system but it may be necessary to choose between radically different possible solutions in order to derive useful system requirements. If there is a user requirement to increase capacity on a line, for instance, it will almost certainly be necessary to decide whether this is to be delivered through larger trains or more frequent trains as the two solutions will have very different system requirements.

Of course, you or your customer will want to check that the system meets its requirements, typically by carrying out demonstration, tests, inspections, and analyses. We call this **verification** and distinguish it from **validation**, which is the process of checking to ensure that the system (product) and requirements meets the needs of the user.

- **Verification** answers the question - Was the system (product) **built "right" –** does it meet the requirements?
- **Validation** answers the question - Was the **"right" system** (product) built. - does it meet the user needs?



5. Have I got all the requirements I need (and no more)?

The complexity of the system will determine how many requirements to write and to what level of detail. The requirements will usually be structured as a hierarchy, with top-level requirements that fulfill the particular stakeholder needs and are fully supported by lower-level requirements.

Requirements can be pulled from a number of sources including the Concept of Operations, stakeholder interviews and railway operator policies. Although the process of capturing requirements is primarily done at the beginning of the system development cycle, it should be considered an iterative activity where requirements are continually refined and validated with the stakeholders.

Tip: Many rail projects modify existing railway systems. The user requirements may be quite straightforward but the system requirements may still be complex because of the need to work with the rest of the railway places constraints on the modified system. Many of these constraints may be embedded within the standards catalogue for the railway. Applicable requirements from these standards should be regarded as system requirements. There may be constraints however which are not embedded within standards but nonetheless should also be identified and regarded as system requirements.

☑ **Have *all* stakeholders been identified?**

Before you can start requirements development, ensure that you have identified all the potential stakeholders of the system. Stakeholders are individuals, groups or agencies who are affected by the system, such as end-users, maintainers, trainers, and even regulatory agencies.

☑ **Do I *understand* the domain in which the system will operate?**

With complex systems and with simple systems in un-familiar domains it is critical that the requirements are produced with a full understanding of the environment in which the system will be designed, built, operated, maintained and disposed. Collating of domain knowledge from domain experts within the stakeholder group is a key aspect of this process.

☑ **Are *all* functions and needs described?**

Ensure that the set of requirements defines all the system functions that are needed to satisfy the stakeholder needs.

☑ **Do I have any *unnecessary* requirements?**

Avoid adding requirements because they provide a nice feature. All requirements should be traceable to a stakeholder need. Unnecessary requirements will add cost and complexity to the system.

There are a number of reasons any given requirement may not be necessary. For one, it may not be unique. Frequently, multiple requirement authors cover similar ground or sometimes even the same author tries to reinforce an idea by stating it more than once. Another reason a requirement may not be necessary is that it just doesn't add any value to the system. You have to ask yourself if any system capability would suffer if this requirement was eliminated. One type of valueless requirements is the "negative" requirement that identifies something the system is not required to do. (These are considered disposable since there are virtually an unlimited number of them.) "The system shall not be required to control traction power" is one example of this.

☑ **Are my requirements *complete*?**

Completeness of requirements ensures that all aspects of stakeholder needs are completely defined by the set of requirements.

☑ **Are my stakeholders *involved*?**

Stakeholder involvement is the only way to validate that the right requirements are being documented. But bare in mind that stakeholders' needs are likely to conflict and you will have to negotiate resolutions to these conflicts. For example one stakeholder may desire features which the budget holder does not want to pay for. The budget holder for a system is always a stakeholder and therefore needs to be involved in the resolution of conflicts.

Tip: Do not fall into the trap of looking at functions in isolation to other functions. This may cause problems when the functions are integrated together. Participate in stakeholder walkthroughs to ensure the correct requirements are being developed and validate the requirements by tracing the requirements to an associated need. Obtain stakeholder approval and support during this process to generate a baseline set of requirements.

There are different types of requirements and the following checklist will help you to check that you have every type that you need:

☑ **What is the system expected to do that will satisfy the needs of the operator or agency?**

Part of this process is gaining an understanding of the domain in which the system will operate. These are **functional requirements**.

☑ **How well does the system need to perform its required functions?**

Identify operation of the system under designated conditions. These are **performance requirements.**

☑ **Under what conditions does the system have to operate and meet its performance goals?**

These are **environmental and non-functional requirements,** and include reliability, availability, safety and environmental criteria.

☑ **What other external systems act as stimuli to the system or expected outputs from the system?**

These are **interface requirements.**

☑ **What are the data elements of the system?**

These are **data requirements**.

☑ **What are the interfaces of the system (internal and external to the system) and how are these interfaces managed?**

These are **interface requirements**

☑ **What requirements pertain to the production, development, testing, training, support, deployment, and disposal of the system?**

These are aspects of systems development that are needed but do not show up as part of the system and are often referred to as **enabling requirements**. These requirements can

be found in the various project plans, such as the Systems Engineering Management Plan (SEMP), statements of work for the contractor, and memorandums of understanding among participating stakeholders.

☑ **What constraints are imposed on the system due to technology, design, tools and standards of the railway operator?**

Some people would say that these are not requirements at all but "how to" statements. However, if they are real constraints on the design of the system that must be respected then they should probably be included in the requirements – customers generally do not like being told that something that they require is not a "requirement".

Tip: Do not approve the requirements too early. Give ample time to develop a set of requirements that are complete and well written. At the same time, for practical reasons of cost, schedule and political pressures within an operator or agency, avoid going through an endless cycle of requirements analysis. At some point in your project, you will have to baseline your requirements and progress the project. Following the steps as outlined above should ensure the development of a solid baseline set of requirements.

6. Are my requirements well written?

Now let's focus a little more closely on the content of each requirement. The primary method of communicating between individuals or groups on a project is through requirements. Well written requirements are seldom noticed. A "bad" requirement will eventually get your attention however, and frequently at a most inopportune time. Poorly written requirements may haunt engineers for the entire development life of a project and cause trouble for the end users even longer.

The way to check to see if your requirements have been well written and are communicated properly is to ask the following series of questions for each and every requirement. If the answer is "no" to any one of these questions there may be more work to be done.

However, bear in mind that, for user requirements, the single most important question to ask is, "Can the user clearly understand it?"

☑ **Is the requirement *Accurate*?**

Simply put: is what this requirement is saying correct? Does it faithfully respond to a higher level requirement, system goal, or user need? The number of requirements can grow dramatically as each new level of decomposition adds more detail. Requirements are "traced" downward and upward from each level. An accurate requirement must address some aspect of the parent requirement from which it has been derived.

☑ **Is it *Feasible*?**

Can this requirement be met through design? You have to make sure that you don't require a particular system component to do something if is not achievable such as the proper information has not been provided to it. For example, requiring an arrival time lateness calculation to the second is not possible if the schedule data is in half minutes.

☑ **Is it *Clear and Unambiguous*?**

Minimize chances for the requirement to be misunderstood. Avoid using terminology that may be unfamiliar to the intended audience. Expect that the requirement will have to survive on its own without the help of the author having to explain it. Remove all non-essential language. "The simulator shall allow the user of the system to…." is one example of too many unnecessary words. The subject of the requirement must always be

the system to be designed. In this exercise we don't care what the user can or cannot do. Another ambiguous example, "The xyz application shall only display a message_not_permitted error when it is authorized" leaves the reader wondering if action should be taken based on the authorization of the application or the message.

☑ **Is it *Technology Independent*?**

Many of us who have chosen this line of work have done so because they enjoy solving problems. Unfortunately, this tendency can occasionally get in the way when defining a set of requirements. In most instances, you should stick to *what* is required and not *how* to accomplish it. Remember, there must have been a reason why the person or organization that was selected for the design effort got the job in the first place. It is important that you don't inhibit their expertise needlessly by handcuffing them to your solution.

On the other hand, if you desire a certain product or solution for a legitimate reason, such as compatibility with an existing system, you still have to say so. Many requirements authors won't do this for fear that someone will accuse them of crossing over some (determined by them) requirements/design boundary. Instead these fearful authors will skirt the issue by writing many 'peripheral' requirements in the hope that the designers will correctly guess at what is really needed. If you feel that this may apply to some of your efforts you may need to ask the previous "clear and unambiguous" question one more time.

☑ **Is it *Atomic*?**

If you are able to break a requirement apart into two or more separate requirements without much effort you probably should. Utilize simple sentence structure with one and only one "shall". It helps to more accurately characterize what is working and what is not when you get to the verification phase. Looking for the word "and" in a requirement is one way to check for this. Also, be careful with lists of items in a requirement, particularly if they are in any way unrelated to each other.

☑ **Is it *Verifiable*?**

A requirement is of little value if you can't check if it can be met during the verification phase. At the time a requirement is written one, and only one, of the four following methods of verification should be identified:

- **Demonstration** – A capability that can be shown to test witnesses by following an approved step-by-step procedure.
- **Test** - An actual measurement by a calibrated piece of test equipment
- **Inspection** – The component attribute is examined for compliance.
- **Analysis** – requirements that cannot be met directly by any of the previous three categories but can be inferred indirectly. An example could be that a control center "shall be able to track 200 trains." If there are only 50 trains available, system performance measurements as trains are added in the real environment could be monitored and compared with a simulated environment where 200 trains can be inserted.

The wording of a requirement is often the primary culprit for an unverifiable requirement. "…shall be optimized", "to the maximum extent possible", "to the best of <the provider's> ability", are all examples that will turn a real requirement into an ignorable pseudo-requirement. Also avoid utilizing words such as "allow, capable,

flexible or adaptable" since it will be an impossible task to verify whether these requirements have been met.

If it appears that more than one verification method must be performed, this may indicate that you have a compound requirement on your hands. In this case: DO NOT PASS GO! Reverse direction and revisit the Atomic question again.

7. Are my requirements well structured?

One other major topic for our discussion is how to make these requirements all fit together. Well written requirements are important, but it is equally important that the requirements associated with a particular project or product are well-structured. Requirements always change but it is easier to manage this change if they are well-structured.

The following questions should be asked as early and as often as possible during a project's lifecycle:

☑ **Are the requirements organized into a hierarchy?**

Requirements should clearly be documented in an ordered manner. From user to system level, requirements may then be decomposed into increasingly detailed refinement of the original top level set of requirements into one or more lower levels. Along with decomposition, requirements must then be allocated to a sub-system or product component at the next lower level. Although this decomposition and allocation exercise typically occurs along clear functional boundaries, the resulting divisions may also reflect the structure of the development team. If, for example, there is a voice communications capability that will most likely be provided by one developer and an off-line reporting function by another, the next layer below system requirements may take two separate paths along these lines.

Tip: "Requirements Leveling", or keeping a consistent and appropriate level of detail at each requirements layer, should be maintained during the decomposition and allocation process.

☑ **Can you follow the traceability between the requirements as well as to your sources, design, and test cases?**

Tracing between requirements

No matter how these onion-like layers of requirements are structured, however, they must continue to be "linked" together in such a way that one does not lose track of the relationships between parent and child. This linkage is called **Requirements Tracing** and the documented output of this process is called a **Requirements Traceability Matrix** or **RTM**.

Linking source information to the requirements

Besides linking requirements to each other, they should also be traced to source information. The rationale used in the derivation of the system's requirements should be documented in addition to the requirements themselves. User requirements could be linked to user scenarios that describe the possible range of users, environments and influences. System requirements should be linked to applicable legislation, standards and procedures from other system stakeholders. Both user and system requirements should include traceability to user needs, mission goals, design constraints, and assumptions.

Creating and maintaining the above linkages are particularly important for large, multi-year projects as is frequently the case in the transit industry. Long-term projects may

undergo a complete turn-over of the personnel that had been initially involved in the derivation of the original requirements. New staff will need to know the rationale behind the generation of the requirements so that they will be able to make informed decisions and permit the project to proceed seamlessly.

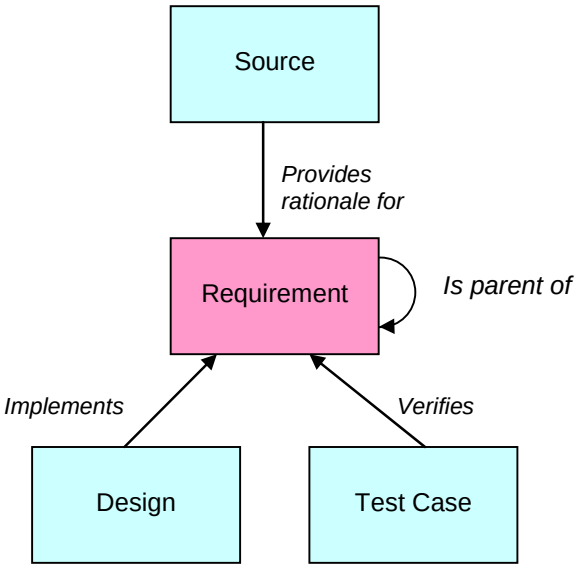
Linking requirements to design

As the life-cycle is progressed, and a design is evolved, it is important to maintain traceability back to the system and user requirements. This provides an essential mechanism to enable checks to be made to ensure that all requirements are being addressed and that no un-required functionality is adding cost and complexity to the project.

Linking requirements to test cases

The testing effort will begin at almost the same time as the implementation or construction phase does. Testing reverses the requirements decomposition process by methodically integrating and testing system components until eventually system requirement are verified and user requirements are validated. Linking test cases to requirements will confirm that all requirements are being fulfilled.

The diagram below illustrates the different sorts of linkages that we have just described



☑ **Are you using appropriate tools?**

Most complex projects will need a formal requirements database that will help you accomplish this task. Each requirement should be identifiable, by means of a unique permanent identifier. The identifier will tell the reader something about the level that a particular requirement is at and its association with requirements at higher and lower levels.

There are many commercial products available that help with this interconnectivity. Requirements management tools exist to help in structuring requirements and maintaining the relationship between them (e.g. DOORS, RTM, CORE, RequisitePro). Most projects with more than 100 requirements (subjective estimate by the authors, of course) will benefit from using this specialized database software and consideration should be given to document and maintain the needed information. Additionally, these tools will help in the execution of a formal change control and logging process that

should be adopted to track changes and their rationale.

8. Final thoughts

This short guide is only a basic introduction to managing requirements on rail and transit projects. There is a lot more to be said on the topic and you may find the reading list below useful.

And, remember that, when you have finished writing down your requirements, you have set the starting point for design and verification. So make sure that:

- The requirements are kept up-to-date as things change, and
- The requirements are actually used!

9. Further reading

EIA – 'Processes for Engineering a System', EIA Standard ANSI / EIA-632-1998

INCOSE – 'Systems Engineering Handbook'

CALTRANS – 'Systems Engineering Guidebook for ITS' (available from www.fhwa.dot.gov/cadiv/segb)

NASA – 'Systems Engineering Handbook'

10. Acknowledgements

This note was prepared by the Product Development Sub Group of the INCOSE Intelligent Transportation and Transit Systems Working Group (ITTSWG). The members of this group were Tim Armitage (Arup), Deborah Chin (New York City Transit), Bruce Elliott (Atkins), Ed Kastenhuber (Parsons), Kenneth Kepchar (FAA) and Michael Krueger (ASE Consulting). The members provided their time as professionals committed to advancing the state of the art and practice in Systems Engineering; their views do not necessarily represent those of their employers.

This note has also benefited from careful and insightful review from participants in public meetings of the ITTSWG in San Diego and of the UK Rail Interest Group in London.

The images in this note are reprinted with the kind permission of Atkins and New York City Transit.