Twelve Requirements Basics for Project Success

Dr. Ralph R. Young
Northrop Grumman Information Technology Defense Group

The author provides a set of 12 requirements basics; these recommended approaches will contribute to your project's success. The requirements basics are based on industry experience; guidance from requirements-related books, articles, and Web sites; and the author's involvement with projects. Having an experienced requirements subject matter expert on the project staff can help the project manager and the project team guide investments that will help.

Much has been written about requirements, and surely we have experienced enough to have learned how to do things in ways that support successful project outcomes. Yet, many projects encounter requirements-related problems, most of which could have been avoided. Ample guidance is available to enable successful project management and to perform effective requirements practices [1, 2, 3, 4, 5] that are easy to apply.

Table 1: *Twelve Requirements Basics for Project Success*

1.	Make sure the project staff includes a trained and experienced requirements manager or requirements analyst.
2.	Proactively *partner* with your customer.
3.	Invest in the project's requirements process.
4.	Write a project vision and scope document.
5.	Use proven requirements elicitation techniques such as requirements workshops and prototyping to evolve the real requirements and to gain buy-in from the stakeholders.
6.	Utilize an evolutionary or incremental approach to development, deployment, and implementation of the capabilities.
7.	Use an effective mechanism to control requirements changes and new requirements.
8.	Use an effective automated requirements tool to maintain information about the requirements.
9.	Ensure that the facts concerning the requirements are accurate and that important requirements are not omitted. Ascertain the rationale for every requirement (why it is needed).
10.	Conduct inspections of all requirements-related documents. (*Inspections* are a more rigorous form of peer review.)
11.	Enlist the support and assistance of all members of the project staff in helping to perform *requirements work*.
12.	Proactively address requirements-related risks.

Table 2: *Requirements Secrets*

1.	*Half* of the features provided in most delivered software are never used once [6].
2.	More than half of the effort on most systems and software projects is wasted [6].
3.	The *stated requirements* (i.e., the list of requirements provided to the developer by the customer and users) are *never* the *real requirements*.
4.	Eighty percent of requirements errors found in system testing are a result of incorrect *facts* about the requirements or omitted requirements [7].
5.	Spending a lot of effort on testing may be misguided; it is better to invest more in an effective requirements process that results in higher quality products being provided to test.
6.	Given that a) systems and software development is complex, and b) people (with all our frailties) are involved, the probability of the development effort becoming derailed is very high – unless a proactive effort is made to *partner for success*. See Requirements Basic 2 (page 5) in this article for a description of what is envisioned.
7.	The documents we produce are fraught with errors; the earlier in the development effort those errors (defects) are discovered, the less costly it is to fix them. A peer review process should be implemented to reduce errors. All requirements-related documents should be inspected.
8.	Project managers should pay added attention to the requirements basics and allot sufficient funding for requirements-related activities.
9.	It is neither difficult nor expensive to incorporate inspections in your project's approach [8]; although this technique has been available for a long time, most projects do not use it.
10.	A *defect prevention process* is also easy to train, deploy, and implement. It can save a lot of money and time, but again, most projects do not use it.
11.	Communication on most projects is challenged; a proactive effort to foster good communications is required [1].
12.	An effective automated requirements tool is required to support requirements development and requirements management on all but the smallest project.

Table 1 provides a set of 12 requirements basics for project success with supporting information, suggestions, and recommendations provided in this article. I encourage you to consider them thoughtfully in the context of your project and to determine if some changes in its approach may be worth considering. (Of course, if you digest this article and then decide to make no changes, no improvement opportunities will result. Failure to work proactively to continuously improve is a root cause of a lack of improvement in project success rates [6].) You may relate to the Requirements Secrets provided in Table 2. The sad truth is that these are not really secrets; they are fairly well-known facts that are well documented in the requirements literature (books and articles written by academicians and practitioners concerning requirements). *So, the bottom line is not that we do not know what to do, it is that we do not do it* – a sad commentary on our willingness and commitment to discipline our efforts on projects.

Related to the importance of requirements basics is the need to be agile. How does one balance increased investment in the requirements process with the need to be agile? In my judgment, it is the project's *approach* that makes all the difference. By using an evolutionary or incremental approach, and by delivering versions, releases, and new products, we can be agile. It is not required that we learn a whole new way of doing things.

Twelve Requirements Basics for Project Success

Concerning potential improvements in your project's approach, we need to remind ourselves that each of us is a change agent on our jobs and on projects. Select an improvement opportunity from the following where you have influence and show others that you can be successful.

Basic 1: Provide training for all project participants concerning the requirements processes to be used on the pro-

ject. Also, there needs to be a technical leader who is highly skilled in requirements engineering. The training and experience required for three levels of a requirements analyst (RA) are provided in Table 3. See [5] for more information and insight, including an extensive set of references to excellent requirements books and articles by many good writers. Industry systems engineering and requirements trainer Robert Halligan believes that the number one problem in requirements engineering is that project managers (PMs) fail to require experienced RAs, and that RAs are not sufficiently trained and/or experienced to perform their roles effectively[1]. PMs can use this skills matrix to recruit and train RAs; RAs can use it to pursue an ongoing professional development program to acquire needed expertise.

It is important to distinguish between a *process* and the *skills* of project professionals. A process may be defined as a set of activities that results in the accomplishment of a task or the achievement of an outcome. Every project uses a requirements process whether it is defined and documented (written down) or not. Some of the advantages of involving a project's key leads in defining and documenting the requirements process for a project include the following:

- Because they helped create it, the people involved in defining and documenting the process acquire a better understanding of it and become more committed to its successful use.
- The leads for other areas within the project become more involved in the requirements activities and bring their expertise and experience to refine the requirements process.
- Once the process is documented, the opportunity exits to improve it based on actual project events.
- The process is likely to be more complete and useable.
- The process is more likely to be integrated into other activities and plans on the project. In other words, by using a collaborative approach, project communication is enhanced.
- Technical performers on projects become more inclined and willing to use an agreed-upon and understood requirements approach.

When addressing the *skills* of a requirements analyst in Table 3, think of these skills *supporting* the project's requirements process. The process goals are created to perform requirements work effectively and successfully; the skills applied by the requirements analyst facilitate the performance of the requirements work. For example, by ensuring that each requirement meets the criteria for a good requirement, the RA enables the project to avoid a lot of rework, thus making the requirements process much more effective. Another example is that by utilizing the *joint team* mechanism (a way to do something), responsibility for the requirements is fixed – all requirements go through a funnel so that accountability for them can be maintained. The joint team mechanism needs to include a few representatives (between two and 12 or more, depending on the size of the project) of both the customer/user and the developer who are empowered to make decisions and take responsibility and accountability for the requirements throughout the project's life cycle.

	Requirements Analyst's Skills Matrix	Junior-Level Analyst	Mid-Level Analyst	Senior-Level Analyst
Skill No.		Knowledge of = K Experience with = X		
1.	Types of requirements.	K	X	X
2.	Criteria for a good requirement.	K	X	X
3.	Customer/user involvement with requirements (joint team).	K	X	X
4.	Identifying real requirements (from the stated requirements).	K	X	X
5.	Anticipating and controlling requirements changes.	K	X	X
6.	Requirements elicitation.	K	X	X
7.	References concerning requirements (books, articles, standards).	K	X	X
8.	Requirements attributes.	K	X	X
9.	Requirements baseline.	K	X	X
10.	Training in systems engineering (e.g., life cycles, risk managment).	K	X	X
11.	Rationale for requirements.	K	X	X
12.	Requirements management tools.	K	X	X
13.	Requirements peer review/inspection.	K	X	X
14.	Requirements syntax.	K	X	X
15.	Requirements traceability.	K	X	X
16.	Requirements verification and validation.	K	X	X
17.	Requirements Review Board/Configuration Review Board/ Configuration Control Board.	K	X	X
18.	Developing and using metrics for requirements activities/processes.	K	X	X
19.	Requirements prioritization.	K	X	X
20.	Technical writing of requirements deliverables (Requirements Traceability Matrix, Software Requirements Specification, Interface Requirements Specification).	K	X	X
21.	Develop, implement, and use requirements processes.		K	X
22.	Quality assurance of requirements.		K	X
23.	Requirements allocation (to components, applications, packages).		K	X
24.	Requirements change control and change notification.		K	X
25.	Requirements repository.		K	X
26.	Requirements errors (missing, incorrect, infeasible, out-of-scope).		K	X
27.	Use-case development (with customer/user).		K	X
28.	Requirements specifications.		X	X
29.	Evaluating requirements for risks.			X
30.	Training the requirements processes.			X
31.	Requirements Impact Estimation Table.			X

Table 3: *RAs Skills Matrix*

Basic 2: Proactively partner with your customer. In our work, project practitioners like to use the word *partner*. It suggests that we are collaboratively working toward joint objectives. I am talking about *a unique type of partnering* in which an independent outside facilitator is engaged to orchestrate an approach in which a set of carefully selected stakeholders gain commitment to success through a series of planned partnering workshops[2]. The advantage of this approach is the evolution of a team that is committed to project success – it will not allow the project to become derailed in spite of the inevitable problems and interpersonal issues encountered during the days, weeks, months, and sometimes years of hard (and often conflicted and frustrating) work. Wiegers provides related ideas concerning having a product champion as a specific partnering technique in [9].

Basic 3: Understand the resources required to perform requirements processes effectively and invest in the project's requirements process. The industry average for project investment in its requirements process is 3 percent of project costs; data from NASA shows that when 8-14 percent of project costs are invested in the system life-cycle requirements process, there is a much higher probability of achieving lower costs and improved schedule [10]. The requirements process used by the project or

organization should be 1) developed by the project's stakeholders and staff; 2) documented; and 3) continuously improved based on experience on each project. Measure the requirements change activity (requirements volatility), including both new requirements and changes to requirements, to provide insight into project performance. Also, plan for change; this means estimating the amount of change and when it might occur.

Basic 4: Write a project vision and scope document. The benefit of this recommendation is to document the vision of the stakeholders concerning the goals of the project, and to achieve significant consensus on the project's scope (what is included in the project and what is excluded – for example, *product will not support users in Ireland until Vers. 3*). A template for this document is developed by Karl Wiegers [9] and is represented in [15]. In our work, we need to be aware of the need to involve all stakeholders and to gain *buy-in* (commitment to the success of the project). Allowing various stakeholder groups to review the vision and scope document, comment on it, and then revise it to address their stakeholder comments is one technique that can help gain buy-in. One project best practice is to engage stakeholders throughout the project – projects that do so are more successful than those that do not because communication is improved and expectations are more realistic [7]. Also, this provides more opportunities to build interpersonal relationships and even allows customers and users to help solve the problems that arise.

Basic 5: Use proven requirements elicitation techniques such as requirements workshops and prototypes and other forms of visual presentations to evolve the real requirements[3] and to gain stakeholder buy-in [11]. Among more than 40 requirements-gathering techniques, these two seem to be the most effective. In the case of the former, various stakeholder groups will learn more about the perspective of the other stakeholder groups and will have a better understanding of the overall needs to be addressed. Another use of requirements workshops is to review issues and make decisions that best serve all stakeholders. The latter technique is a cost-effective way to gain a better understanding of the customer's and users' real needs – prototypes may be designed, developed, implemented, and updated for a fraction of the cost of a delivered capability. Getting feedback on a visual representation is faster and easier than getting it from text. *The important thing is to evolve the real requirements before starting other technical work.* Experience has shown that using the stated requirements (the requirements provided by the customer and users at the beginning of a development effort) results in an estimated 45 percent rework [12].

Basic 6: Utilize an evolutionary or incremental project approach to development, deployment, and implementation of the needed capabilities. This is both an opportunity and a challenge – whenever one employs a new technique on a real project, additional risk is assumed. My suggestion is to ensure that there are people on the project staff who have previously utilized a new practice, technique, method, tool, or mechanism. In this way, your project can benefit from lessons learned previously (provided, of course, that you pay attention to and are disciplined to incorporate them). An evolutionary or incremental approach allows the project to build hunks of delivered code at a time, and then uses them to learn how and where to proceed. Use versions, deliveries, releases, and new products to accommodate the inevitable requirements changes.

Basic 7: Use an effective mechanism to control requirements changes and new requirements. Controlling changes may mean generating new releases. Most projects utilize Change Control Boards (CCBs), but they usually address detailed project activities such as changes in the code. Use of a higher-level CCB (for example, the joint team I recommend to be responsible for the requirements) to maintain control of the requirements is a good example of applying added discipline on a project. It is logical (though most often not done) that requirements should be prioritized, and that the highest priority and most difficult requirements be addressed first. One reason to do this is that some requirements are *unknowable* at the beginning of a development effort; it requires some development work and related research effort to discover them. It is critical to understand that changes to requirements and new requirements can cause a project to go out of control, thus creating the need for a mechanism to control them. I recommend limiting changes to a maximum of .5 percent per month (6 percent per year) in the capability currently being developed in order to help keep the project under control [13][4].

Basic 8: Use an effective automated requirements tool to maintain information about the requirements. Although many projects do not follow this practice, in my judgment, an automated requirements tool is required for any project except tiny ones. This is because it is necessary to have a lot of information concerning each requirement – its *attributes*. For example, we need to know the following:

- The source of the requirement (who nominated it).
- A unique identifying number for it.
- Its priority (on a scale of one to three).

Table 4: *The Criteria of a Good Requirement*

Each Individual Requirement Should Be:	
Necessary	If the system can meet prioritized real needs without the requirement, it is not necessary.
Feasible	The requirement is *doable* and can be accomplished within available cost and schedule.
Correct	The facts related to the requirement are accurate and it is technically and legally possible.
Concise	The requirement is stated simply.
Unambiguous	The requirement can be interpreted in only one way.
Complete	All conditions under which the requirement applies are stated, and the requirement expresses a whole idea or statement.
Consistent	The requirement is not in conflict with other requirements.
Verifiable	Implementation of the requirement in the system can be proved.
Traceable	The requirement can be traced to its source, and it can be tracked throughout the system (e.g., to the design, code, test, and documentation).
Allocated	The requirement is assigned to a component of the designed system.
Design independent	The requirement does not pose a specific implementation solution.
Non-redundant	The requirement is not a duplicate requirement.
Stated using a standard construct	The requirement is stated as an imperative using the word shall.
Associated with a unique identifier	Each requirement has a unique identifying number.
Devoid of escape clauses	Requirements do not use *if*, *when*, *but*, *except*, *unless*, and *although* and do not include speculative or general terms such as *usually*, *generally*, *often*, *normally*, and *typically*.

- Its relative cost to implement (low, medium, high).
- Its relative difficulty to implement (low, medium, high).
- Each requirement meets the criteria of a good requirement (the criteria shown in Table 4 should be met for each requirement. If they are not, it is likely that the requirement statement is not really correct).
- The *rationale* for the requirement (why is the requirement needed?) [15][5].
- Change history (how has the statement of the requirement changed over the system life?).
- Traceability[6] (of each requirement to its source, as well as the design, code, test, documentation, and training materials).
- Status (draft, final, approved, pending approval, disapproved).
- Assigned to (component of the system).

Two related needs should be considered. The first is that the requirements are used by different people (including customers, users, and project team members) with different *viewpoints* [16]. We need to be able to organize the requirements to provide different perspectives, for example, for customers, users, or testers. The order and the actual requirements selected for these activities are dependent on the viewpoint. The second need is that at any given time, one may need to see all requirements or only those changed ones; an automated requirements tool provides the capability to obtain various reports that are required to perform good requirements work.

Basic 9: Avoid requirements errors. A requirements error is a defect that is discovered in delivered code that is a result of a requirement statement. Data from NASA provided by requirements consultant and trainer Ivy Hooks show that 80 percent of the requirements errors that remain in delivered software are a result of incorrect facts (49 percent) and omitted requirements (31 percent) [10]. This is truly amazing when you think about it – clearly a key opportunity to improve requirements work. Making a concerted effort (investing more in the requirements process) to avoid these two types of requirements errors can save money and time and also improve the quality of delivered capabilities.

Some of the things that can be done to address incorrect facts include the following:

- Provide stakeholder reviews of requirements work products.
- Require that an authoritative source document be specified.
- Require verification of each requirement as part of the planning for testing.

	Risk	Approach	Suggested Risk Response Strategy
1.	Changing requirements	Mitigation	Implement a high-level CCB (Joint Team); conduct formal requirements reviews such as a System Requirements Review.
2.	Incomplete or missing requirements	Mitigation	Elicit requirements from a variety of stakeholders; conduct requirements workshops; conduct formal requirements reviews; and ensure that all high-level requirements are addressed and met in the requirements work products.
3.	Unclear requirements	Mitigation	Perform requirements analysis and rework existing requirements; provide stakeholder reviews of requirements work products; require that an authoritative source be specified; and require verification of each requirement as part of the planning for the test program.
4.	Invalid requirements (requirements that may not specify what the customer really wants)	Mitigation	Revisit customer; re-establish needs; redevelop requirements; and delete requirements that do not have a good rationale or do not meet the criteria of a good requirement.
5.	Infeasible requirements (requirements technically difficult to implement)	Mitigation	Revisit customer, re-establish needs, redevelop requirements; and delete requirements that do not have a good rationale or do not meet the criteria of a good requirement.
6.	State-of-the-art requirements (something never done before, or your company has never done)	Mitigation	Obtain input from similar projects/companies; perform detailed requirements analysis and find someone who has done it before and obtain advice concerning lessons that were learned from doing something new or in a new way.
7.	Inadequate interface definition	Mitigation	Establish interface management methods (Interface Requirements Documents/Interface Control Documents) and implement responsibility assignment matrices.
8.	Non-verifiable requirements	Mitigation	Involve testers in early requirements development activities; and reword requirements to establish feasible compliance methods.
9.	Incorrectly allocated requirements	Mitigation	Conduct formal requirements reviews and implement a formal requirements management tool.
10.	Non-traceable requirements	Mitigation	Understand traceability more thoroughly and perform detailed requirements analysis to determine potential sources or to eliminate non-traceable requirements.

Table 5: *Suggested Requirements-Related Risk Response Strategies*

Some of the things that can be done to address omitted requirements include the following:

- Elicit requirements from a variety of stakeholders.
- Conduct requirements workshops and review the requirements collaboratively.
- Conduct formal requirements reviews.
- Ensure that all high-level requirements are addressed and met in the requirements work products.

Basic 10: Utilize inspections of requirements-related documents. Inspections are not difficult to perform, and their use is cost-effective; a concise explanation of how to perform both peer reviews and inspections and how to install a peer review process on a project can be found in [8].

Basic 11: Enlist the support and assistance of all members of the project staff in performing requirements work. It is not just the requirements manager and/or the RA who need to be involved in requirements-related work on a project. Most members of the project staff can help; however, they need to be made aware of *how* they can help and that the PM's expectation and request is that they *do* help. A technique I have used successfully to accomplish this is an *early project requirements briefing* that is made

to the entire project staff. Participants in the briefing can be invited to suggest how they can help with the requirements work with the objective of optimizing the efforts of all project members. A related need is to develop a strong sense of teamwork throughout the project. In my experience, an empowered and committed team can accomplish most anything it sets out to do.

Basic 12: Work proactively to address requirements-related risks. As is well known to practitioners, most projects do not address the risks they face with the discipline that they should. There are many excellent books concerning how to do this. A good starting point is Chapter 11 of [15]. Another source is [9], which provides a chapter on software requirements and risk management. Table 5 (see page 7) describes typical requirements-related risk response strategies that should be helpful.

Summary

From my experience in working with projects of all sizes, there is a lot of benefit to investing in and continuously improving the project's requirements process. There really is no good excuse for the extent of rework that is typically required on projects, or for the failure of projects, or even delivery of reduced functionality and/or over budget or beyond schedule issues. It is within our power to do better. Use of one or more of the requirements basics discussed here is likely to be helpful. Consider discussing the contents of this article at a project staff meeting with the objective of identifying the *top three* ideas, suggestions, or recommendations to improve the requirements practices of your project.◆

References

1. Whitten, Neal. Neal Whitten's No-Nonsense Advice for Successful Projects. Vienna, VA: Management Concepts, 2005.
2. Young, Ralph R. Effective Requirements Practices. Boston: Addison-Wesley, 2001.
3. Alexander, Ian F., and Richard Stevens. Writing Better Requirements. London: Addison-Wesley, 2002.
4. Gottesdiener, Ellen. Requirements By Collaboration. Reading, MA: Addison-Wesley, 2002.
5. Young, Ralph R. The Requirements Engineering Handbook. Boston: Artech House, 2004.
6. The Standish Group. "What Are Your Requirements?" West Yarmouth, MA: The Standish Group International, Inc., 2003.
7. Stevens, Richard, Peter Brook, Ken Jackson, and Stuart Arnold. Systems Engineering: Coping with Complexity. London: Pearson Education Limited, 1998.
8. Wiegers, Karl E. Peer Reviews in Software: A Practical Guide. Boston: Addison-Wesley, 2002.
9. Wiegers, Karl E. Software Requirements. 2nd ed. Redmond, WA: Microsoft Press, 2003 <www.processimpact.com/goodies.shtml>.
10. Hooks, Ivy F., and Kristin A. Farry. Customer-Centered Products: Creating Successful Products through Smart Requirements Management. New York: The American Management Association, 2001.
11. Young, Ralph R. "Recommended Requirements Gathering Practices." CROSSTALK Apr. 2002 <www.stsc.hill.af.mil/crosstalk/2002/04>.
12. Leffingwell, Dean. "Calculating Your Return on Investment from More Effective Requirements Management." Rational Corporation, 1997 <www.128.ibm.com/developerworks/rational/library/347.html>.
13. Jones, Capers. Estimating Software Costs. New York: McGraw Hill, 1998.
14. Whitten, Neal. "Meet Minimum Requirements: Anything More Is Too Much." PM Network. Sept. 1998.
15. Young, Ralph R. "Project Requirements: A Guide to Best Practices." Vienna, VA: Management Concepts, 2006.
16. Sommerville, I., P. Sawyer, and S. Viller. "Viewpoints for Requirements Elicitation: A Practical Approach." Proc. of the 1998 International Conference on Requirements Engineering (ICRE '98), 6-10 Apr. 1998, Colorado Springs. IEEE Computer Society (1998): 74-81 <http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/icre/1998/8356/00/8356toc.xml>.

Notes

1. Personal e-mail to the author, September 2, 2006.
2. Request a briefing available on this topic from facilitator Charles Markert <markert@facilitationcenter.com>. For additional information, see <www.mid-atlanticfacilitators.net/>.
3. The term *real requirements* may be new to you. I refer to the requirements that are provided to developers by customers and users as the *stated requirements*. Industry experience has shown us that the stated requirements are *never* the real requirements for an application. Customers and users need help from the development team to *evolve* the real requirements from the stated requirements. This is one reason project managers should invest more time and money in the requirements process. It is also an area where projects can leverage the experience and expertise of *qualified* requirements analysts.
4. Capers Jones has reported that the U.S. average for requirements changes is about 2 percent per month during the design and coding phases, but can be much higher [12, p. 429]. Two percent per month equates to 24 percent per year; this is too much change in my experience to enable the project to remain in control.
5. See Neal Whitten's insightful and helpful article, "Meet Minimum Requirements: Anything More Is Too Much" [14] to gain an understanding of why it is in everybody's best interests to figure out the *minimum* set of requirements that will meet real needs. This article is reprinted with the author's permission as an appendix in [145].
6. The best guidance available on the important topic of traceability is James D. Palmer's article, "Traceability," originally published in *Software Requirements Engineering*, R.H. Thayer and M. Dorfman Eds. 1997, pp. 364-374. This article is reprinted with the author's permission as an appendix in [15].

About the Author

Ralph R. Young, Ph.D., teaches courses concerning requirements and process improvement and facilitates workshops to strengthen the use of practices and techniques on projects. He is the author of *Effective Requirements Practices*, *The Requirements Engineering Handbook*, *Project Requirements: A Guide to Best Practices*, and co-authored *Performance Based Earned Value*.

Northrop Grumman Information Technology Defense Group
7575 Colshire DR
McLean, VA 22102
Phone: (703) 556-1030
Fax: (703) 556-2802
E-mail: ralph.young@ngc.com