A Case for Priority Classifying Requirements

Larry Fellows
Honeywell, Inc.
MS 2H29A3
21111 N 19th Avenue
Phoenix, AZ 85027-2708

Ivy Hooks
Compliance Automation, Inc.
17629 El Camino Real, Suite 207
Houston, TX 77058

ABSTRACT

Prioritizing requirements can play a significant role in reducing requirement problems and increasing customer satisfaction. This paper describes our experiences in teaching requirement prioritization, describes how to prioritize requirements and the benefits of doing so, and describes the responses of our students.

INTRODUCTION

Early in 1997, we began the development of a two-day training program aimed at managing and writing requirements. The original intent of the course was to help Honeywell locations satisfy the Software Engineering Institute's Capability Maturity Model Level 2 Requirements Management Key Practice Area (KPA), and the requirements aspects of the Level 3 Software Product Engineering KPA. As the course developed, it was expanded to apply to systems engineers, software engineers, marketing representatives, and test engineers.

An existing requirement training class formed the basis of the Honeywell training package. Literature searches were conducted to add new data and emphasis to the requirement management portion of the class.

One area researched was that of prioritization, but very little was found on the subject. In (Down 1994) *Risk Management for Software Projects*, prioritization was mentioned as a risk management technique to ensure proper allocation of project resources. In (Davis 1995) *201 Principles of Software Development* and (Davis 1993) *Software Requirements Objects, Functions, and States* prioritization was defined as determining the relative necessity of requirements. (Pressman 1997) *Software Engineering A Practitioner's Approach* discusses establishing requirement priorities from a phased delivery aspect. Finally, (Wiegers 1996) *Creating A Software Engineering Culture* urges prioritization as a means to eliminate unnecessary requirements. Each of these books discusses some aspect of requirement prioritization, but lacks the detail concerning techniques, benefits, and problems.

Before formally presenting the first class, a dry run was conducted with managers, as well as, senior systems and software engineers from various Honeywell divisions. The Students were particularly hostile when the concept of prioritization was presented. Another literature search uncovered a new book, (Sommerville 1997) *Requirements Engineering A good practice guide*, which provided some compelling points on the subject. The following paragraphs describe the events that led to the inclusion of prioritization, student response to the subject, and recommendations for incorporating prioritization into a requirement management process.

BACKGROUND ON TEACHING PRIORITIZATION

For the past seven years, the existing class had included material on prioritizing requirements. This material was based on personal experience in managing programs.

Rarely does a customer get everything requested. Trying to make a development activity

come in at a specific cost and on a schedule will often mean that all requirements cannot be included. Analysis frequently reveals that all performance parameters cannot be met simultaneously or that maintainability, reliability, portability, and performance requirements cannot all be met with a single solution. The fact that trade studies must be done means requirements must be compromised.

Customers have found that development organizations do not, or cannot, view the requirements from their perspective. When the developers do trade studies, they invariably trade-off the wrong requirements. Developer's trade studies often indicate priorities that are the exact opposite of the customer's. The result is wasted time and effort along with the need to repeat the studies when the customer priorities are defined.

It would save time and money if the customer would indicate those requirements that have high priority versus those with low priority from the beginning. The developer would know what the customer wants as opposed to deciding what the customer wants by guessing or applying the development organization's priorities. If the priorities were known up front, the trade studies would not need to be repeated.

A similar situation exists for commercial products. Marketing, as opposed to a specific customer, may be defining the requirements, but breakdowns in communication with developers can occur. This breakdown can result in problems with delivering the new or revised product on time and on budget.

In seven years of these classes, students seemed to grasp the concept of prioritization and no objections were heard. However, there has been no feedback that anyone was actually prioritizing requirements.

The Class Dry Run – Prioritization – No Way!

Prioritization was covered in the first half-day which specifically deals with requirements management. Prioritization was covered in more depth in the last half-day while covering how to write good requirements.

Surprisingly, at the first mention of prioritization the students were visibly upset. They said, We can't prioritize a customer's requirements, the customer would have to do that. Our customers won't do it. They want all the requirements and they believe they are all equal. They will know if we want to prioritize that we are going to cut out some of their requirements and that is not acceptable.

The discussion that ensued was lengthy and frustrating for both instructors and students. Even when exposed to the arguments in the later portion of the class, the students still believed that prioritizing with a customer was nearly impossible. The students felt that the customer would assume that the primary reason to prioritize requirements was to reduce the requirement set. During the debriefing, this subject was pointed out as one of major concern.

The First Class – Prioritization – What a Good Idea!

Feeling there was a major problem in this area, another search was conducted to find additional references. The recently published book, (Sommerville 1997) *Requirements Engineering A good practice guide* by Sommerville and Sawyer, contained the details we needed. This material was incorporated with the training material for the first class. The major points discussed were:

- First set of requirements often a wish list
- Too long and imprecise to be practically implemented
- Refinements and clarifications are negotiated during the course of a project
- Additional requirements or changes to existing requirements are identified in the normal course of engineering a solution
- Not all requirements are created equal

This was an attempt to gain agreement that not all requirements are equal. This should be common knowledge, but experience shows that some students do not believe their customers understood this concept.

There may actually be a case where some customers do not understand the concept. Several years ago, working with a client to put requirements into a requirement management tool, a requirement in the customer system specification said, *All requirements in this document shall be equal*. It was suspected that the use of *shall*, denoting a requirement, was a mistake, until the verification matrix was examined and showed this requirement was to be *verified by analysis*.

In this class, prioritization was not discussed until there was agreement that the first set of requirements may consist of a wish list and not all requirements are equal. Then the following points were covered:

- After you have reached agreement with the customer on the requirements to be included, ask the customer to help prioritize them
- Prioritization helps all stakeholders to identify the core requirements
- Prioritization helps designers to decide on system architecture and resolve design conflicts

A good discussion followed that helped make the idea of priorities fit within the students' perspective. The important point was getting the customer to help prioritize after agreement on the full requirement set. The class appreciated the fact they have to make trade-offs in order to deliver a system on budget and on time.

The terminology found in (Sommerville 1997) *Requirements Engineering A good practice guide* was used to address the priority categories. These categories are:

- "Essential" requirements are those that must be included in the system
- "Useful" requirements are those that would reduce system effectiveness if left out
- "Desirable" requirements are those that are not part of the core, but make the system more attractive to the users

It does not pay to have many priority classifications. Three priorities are sufficient to achieve the goals of reflecting the relative importance of requirements. The importance of having stakeholders, other than the customer, involved in the prioritization process was stressed, just as the stakeholders' importance in developing the requirements is emphasized.

More information was added to support the argument that prioritizing is beneficial to the project. Discussion points evolved around customer satisfaction. When the requirements are prioritized, the customer is happier because expectations are more realistic. It is easier to correlate requirements delivered with schedule deadlines.

Some suggestions were made to help avoid problems. One of these was to try for informal agreement on the priorities. If differences cannot be resolved informally, then they should be discussed and resolved at the negotiation table. Lengthy negotiations would certainly add effort that may be sufficient to kill the concept.

Another point made was that as requirements evolve, for whatever reason, priorities will also evolve and should be revisited and adjusted as needed. While discussing the requirement management process elements, it was pointed out that prioritization would help when adding new requirements. A new high priority requirement would have to replace a lower priority requirement if the work were to remain on the same schedule and budget. Everyone would know where the sacrifice would be made before approving the new requirement. Our belief is that the prioritization process will increase customer confidence.

What a change between the dry run and the first class! Not only was there no revolt when the subject of setting priorities was broached, but the class evaluations revealed they believed that they would gain tremendously by adding this element to their process. They felt they had arguments to aid them in meeting with their customer and convincing them of the advantage of setting priorities.

The Second Class – Prioritization – Been There/Done That.

No changes to the prioritization charts were made for the second class. When the point where prioritization was covered was reached, the students were asked if anyone had ever employed the process. Amazingly, one group answered not only did they prioritize, but the benefits were huge. The two major benefits cited were *cutting out the overtime* and *having a happy customer*. Those benefits are desirable in any organization.

The group who claimed these benefits had been working with a commercial customer over about a six-year period. The first three years were filled with frustrations. The customer was not getting what they wanted and the developers were working overtime on a regular basis trying to implement all the requirements.

After about three years, the developers changed their approach. The developers set about to do much better up-front planning. This planning effort enabled the developers to lay out the customer requirements against a schedule and say what could be done by when. The customer quickly recognized prioritizing requirements would

aid the process and did so on their own initiative. Essentially the three categories discussed earlier were used, labeled high, medium, and low.

When the developers scheduled the prioritized requirements, those with less priority went to the end of the queue. Not too surprisingly, the customer actually decided that some of their low priority requirements were not needed at all and simply deleted them.

Because delivery schedule was a major customer constraint, phased implementation would assure the highest priority requirements were delivered first with others of declining priority in following releases. As new requirements were added, they could be prioritized allowing easy revision of schedules and releases to meet the customer needs.

Now, the customer gets what they want when they want it. The customer is happy with the results which, of course makes the developers feel much better. In addition, there is the benefit of less overtime, especially during acceptance testing. The students said they kept waiting for things to go wrong during acceptance test that would take a large amount of overtime to resolve. They were astounded at how cleanly the acceptance testing went when everyone was aware of exactly what to expect.

These students were quick to point out; these results were not caused by prioritization alone. They believed their whole approach to planning and scheduling also aided in the improvements. While they had recorded no metrics, they knew absolutely that there were large benefits. They also discussed another customer who does not want to operate in this manner and that is declaring all requirements are of equal priority. The students, with the support of their management, are telling that customer *no* and insisting upon the approach which has provided benefits with other customers.

RECOMMENDATIONS

Our recommendations to implement prioritizing requirements in your process follow:

1. Sell prioritization benefits
2. Define the 1, 2, 3's
3. Classify requirements
4. Assess the classification and resolve issues
5. Create schedules based on priorities
6. Maintain the priorities

Step 1 – Sell prioritization benefits

To get participants to agree to prioritize, they must be convinced of the need. Point out the fact they are probably already doing prioritization, it is only the timing that needs to change. When there is not enough money or time to include everything, something must be deferred or deleted to meet budget or schedule. Then the requirements are prioritized. When a really important new requirement is added, then the requirements are prioritized and something of lower value is usually thrown out. Since this normally happens late in the project, there are fewer options.

With late prioritization, serious problems come with deferrals and deletions that are related to synchronization of effort. If one group has completed the implementation of a requirement and another hardly begun, throwing out the requirement could create considerable risk.

Present these undesirable scenarios and recommend setting the priorities early, when there are more options. This allows realistic scheduling of resources and effort. Even if it appears that everything can be done within the schedule and budget without prioritization, prioritizing the initial list allows the least important requirements to be scheduled last. Then if schedule or budget problems develop, or if a new requirement is added, the ability to defer or delete the less important requirements without impacting work already completed increases dramatically.

Point out to people that not all requirements can be priority 1, unless there are missing requirements. There are always requirements levied upon a system that are desirable or useful, or which can be deferred in phased deliveries.

Step 2 – Define 1, 2, and 3's

A simple 1-2-3 numbering system works well. The most important requirements are numbered 1 – these are essential, non-negotiable, and are needed right now. Those that are useful, negotiable, or can be deferred a little later are numbered 2. And those that are desirable, flexible, or can be done someday become the number 3 requirements. One technique is to identify those that are priority 1 and 3. The others are 2 by default. Your operations will determine how you define each of these levels.

Step 3 – Classify

Educate the stakeholders, both external and internal, about the prioritization numbering system before they classify the requirements.

It is important that all stakeholders participate in the process. It is true that some of the stakeholders have more sway than others. When developing for a particular customer, that customer has more influence on the priorities than someone does in your development or test organization. When developing a commercial product, then marketing has a large say so, but so does the development organization that is looking at the cost and complexity of a phased set of releases to a particular schedule.

Have the stakeholders sort the requirements into the three categories. This should be a rather informal process so it can be done quickly and so people do not get too stressed in performing the operation. The important aspect of this process is to get a relative sense of each requirement's importance. It is not worth it to agonize over the exactness of the categorization.

Step 4 – Assess the classification and resolve disputes

After each group defines their priorities, the next step is simple. Everything that is agreed upon gets thrown into its applicable bucket. That is, if everyone thinks requirement A is priority 1, then it is and that is that.

All of the requirements with differing opinions on priority need to have those differences resolved. So if one person thinks Requirement B is priority 1, someone else thinks it is priority 2, and others think it is a 3, now the harder work begins and the differences have to be resolved.

Again, keep this process as informal as possible. Get the groups together and simply show them how every requirement is classified. Then show them those that are in dispute and what this looks like in a tabular format. Immediate agreement to change their priority by someone will reduce the differences. Some requirements will need more discussion.

If one person feels strongly that a requirement should be priority 1, but others think it is a 2, put it in the 1 category but note that it needs to be at the end of the priority 1 implementation schedule. If there is a disagreement over another requirement ask the development expert if it is worth disputing. Some requirements are going to be done early anyway because they will be implemented with other similar items to save development effort. If the disputed requirement is one of these or if it is very small in terms of time and money then put it in the higher priority list and stop the debate. Again, the key is to keep it simple and quick. There is no exact answer or solution.

Step 5 – Create schedules based on priorities

When agreement on the priorities has been reached, development schedules can be laid out to determine if any priority 3 items are going to have to be deferred to a later release. Everyone can see where work is beginning and ending on each requirement. Work in different areas on the same requirement can be synchronized to avoid potential schedule problems.

Step 6 – Maintain the priorities

Priorities, like other requirement information, need to be maintained throughout the project lifecycle. As analysis is performed and design issues are resolved, the priorities need to be revisited. This enables the maintenance of realistic, detailed implementation schedules. Likewise, when new requirements are introduced a reassessment of at least some of the priorities is required. Maintaining the priorities and the resulting implementation schedules increase the probability of functionality being delivered when needed.

QFD

Prioritization discussions would be incomplete if Quality Functional Deployment (QFD) were ignored. This particular method of setting priorities is well documented and courses in how to use the method are available from a number of sources. Accordingly, we have chosen not to address this methodology in this paper.

Using QFD is considerably more complex than the method discussed in this paper. Most projects do not need to undertake the large investment in time and effort required for QFD. Other programs, those involving large numbers of diverse stakeholders, with very different and often contradictory viewpoints, may find QFD is the solution to their prioritization problem.

SUMMARY

We were convinced, before the students brought us a real-life example, that prioritization is essential for managing requirements. We just had not conceived of all of the possible benefits.

Some marketing is required to convince both customer and developer that the effort to do prioritization is worthwhile. Our limited experience seems to indicate that this is not a widely accepted practice. To get the commitment to prioritize the requirements, the benefits of this activity need to be demonstrated to the customer. The development organization needs to understand these benefits and gain the confidence to approach the customer for assistance with prioritization.

A fairly simple process can be used to establish priorities that will help to dramatically reduce rework and schedule slips in a typical project. The resulting reduction in effort will be considerably more than the effort expended establishing priorities.

Working through the prioritization with the customer will allow the developers to better understand the customer, another essential ingredient of the development process. It creates dialog, which reduces misinterpretations by the developers. It will help the customer understand the development schedule problems and feel like a part of the solution.

The customer has many tasks competing for attention and requirements are just one of them. If the requirement definition process is lengthy, the customer gets confused by all the different versions and really does not know what is in the set. Prioritizing the requirements helps clarify the requirements.

The need to prioritize increases with the number of requirements. Those programs with large numbers of requirements need to prioritize early and maintain prioritization with changes to the requirements and their attributes.

Although we do not recommend maintaining the priorities as a baseline item, it is clear that they should be documented. To avoid misunderstandings and conflicting expectations, both customer and developer should agree to the documented priorities.

REFERENCES

Davis, Alan M. *Software Requirements Objects, Functions, and States*. PTR Prentice Hall, Englewood Cliffs, NJ: 1993

Davis, Alan M. *201 Principles of Software Development*. McGraw- Hill, Inc., New York, NY: 1995

Down, Alex et al,. *Risk Management for Software Projects*. McGraw-Hill, London: 1994

Pressman, Roger S. *Software Engineering A Practitioner's Approach Fourth Edition*. McGraw-Hill, Inc., New York, NY: 1997

Sommerville, Ian; Sawyer, Pete. *Requirements Engineering A Good Practice Guide*. John Wiley and Sons, West Sussex, England: 1997

Wiegers, Karl E. *Creating A Software Engineering Culture*. Dorset House Publishing, New York, NY: 1996

BIOGRAPHIES

Larry Fellows

Larry is a staff engineer in the Honeywell Software Initiative. The mission of the Software Initiative is to enable business growth by championing software competency improvement throughout the corporation. This mission is accomplished through partnerships, liaison activities, joint projects, and assessments of Honeywell locations worldwide.

Larry comes to Honeywell from Wilcox Electric, Inc. were he worked as the Software Test Lead and Systems Test Manager. Most recently, Larry was the Software Engineering Process Group Chair. Prior to Wilcox, Larry was a Software Test Manager for General Electric Aerospace. He has also worked as a systems engineer, software developer, and software development manager for Vitro Corporation. He has sixteen years experience in the US Navy with advanced submarine navigation, communications, and ASW systems. Larry holds a BS in Computer Science from the University of Nebraska.

Ivy Hooks

Ivy is president and CEO of Compliance Automation, Inc. (CAI). CAI is the developer of Vital Link, requirement management software, and provides training and consulting in requirements management and writing. Ivy has provided requirements training for the past seven and one-half years and written numerous papers on different aspects of requirements engineering.

Prior to the creation of CAI, Ivy was President of BGJ&A and prior to that, with Barrios Technology Inc. For twenty years, she had a distinguished NASA career at the Johnson Space Center. She is the recipient of many awards and is a charter member of INCOSE, a Fellow of the Society of Women Engineers, and a member of IEEE.