Journal of Universal Knowledge Management, vol. 1, no. 2 (2006), 77-84
submitted: 15/5/06, accepted: 15/6/06, appeared: 28/6/06 © J.UKM

The Value of the Quality Gateway

Marc Philipp Werner
(Hewlett-Packard GmbH, Germany
maphi@hp.com)

Abstract: Over the last nine years, requirements management in the HP OpenView Operations for Windows R&D team evolved from informal documents to a documented, tool-supported process. Early quality and content checks ensure that only well-defined requirements are entered into the system; this is called the quality gateway. Clear definitions of ownership over the whole life cycle of a requirement create accountability and enable the various stakeholders in the organization to track what is being done, who is working on it, and what the status is.

Keywords: requirements management, case study
Categories: D.2.1

1 Introduction

The HP OpenView Operations for Windows R&D group operates in a highly distributed environment. The various teams are distributed in five sites, four counties, and three time zones. This makes it impossible to meet face-to-face, and even telephone conferences are hard to set up, because there is no time when all sites are in working hours.

The Operations product is standard software, not project software. This means that the environment is relatively stable, the stakeholders remain more or less the same, and revolutionary changes of the development processes are impossible. The team has the responsibility to increase market share and revenues from the product. This is accomplished by selecting those requirements that maximize customer value and revenue, and by optimizing the processes to minimize the cost for development.

2 The early years

When the project was launched originally in 1997, requirements management was not done in a very formal way. Although the development process demanded a requirements document, there was no real definition of the form or content of this document.

At that time, the Operations product was available on Unix only. To gain a foothold in the growing Microsoft Windows market, HP acquired a small company that already had a product available. The first project was defined as bringing the capabilities of the Unix product into the acquired product, based on Microsoft technologies.

Right from the start, it was obvious that the Unix product had too many features to bring all of them into the first release of the new Windows product. To decide on the priorities, a list of features, which grew and changed over time, was compiled on

whiteboards and spreadsheets. In absence of a real customer representative, a senior manager with experience in the Unix market took over that role.

About a year later than expected, the first release of Operations for Windows finally happened in 2000. The subsequent release only had two requirements: "Deliver a Japanese version of the product" and "fix the defects". Effectively, requirements management was done in the defect tracking system, by prioritizing the defects and fixing the most critical ones first.

3 First steps towards requirements management

While R&D was still working on the second release, the organization decided to invest more into requirements management. An internal expert was chartered with evaluating different requirements tools and processes. The final choice was to go with the Volere template and the Caliber tool.

In 2000, for the third release, the R&D group decided to use this tool to manage the growing list of enhancement requests that had started to arrive. At first, only high-level entries made it into the system; the typical requirement consisted of a single line of text, an importance (usually, but not always, "MUST"), and a project version ("current release" or "next release"). The team managers used these entries to assess their work. During this time, there was no engineering representation in the process, because there was no overall software architect for the whole group.

At this stage, most engineers did not even use the requirements tool. Whenever they did, they reacted with frustration: the one-line descriptions were not detailed enough to work with. So, for each item, the author (who might or might not have been available for this task) had to be consulted for more details, or the details had to be made up. Depending on the experience of the engineer, the alignment between original request and implemented result varied.

4 Train the users

To improve the situation, the whole lab took the "Mastering the Requirements Process" class from Robertson & Robertson, which introduced the idea of a clear "fit criterion" that describes measurements for requirements. Initially, engineers were happy because this described exactly what they wanted.

Unfortunately, reality quickly showed that fit criteria were not provided; the requirements submitters did not believe that it would be a good investment of their time to write a good fit criterion, and nothing really changed in the process.

At this stage, there was a lot of frustration in the team. The marketing representatives believed that they had given clear guidance to the R&D folks. The R&D managers told their engineers to base their work on the requirements. The engineers, however, could not find the information they needed. Furthermore, the tool had some performance issues for all remote users, which did not help to convince people to use it.

Nobody really took care to maintain the data, either. Although there was some structure in the tool, there was no documentation about where to put what. Duplicate

entries crept into the system; already-implemented requirements were not updated and remained in the system for years.

5 Introduction of the quality gateway

Then, in 2002, two architects were assigned to the group. Among other duties, they took control of the requirements and introduced a quality gateway, as described in [Robertson 99]. All requirements had to be reviewed and approved by the quality gateway before they could be presented to the teams at all. This principle was applied to new entries as well as to existing ones. In essence, to get into the process, a requirement had to contain the following:

- A clear name
- A description of what is requested
- A purpose, giving the reasons why this is requested
- A 'fit' criterion, detailing how the requirement can be tested from a customer point of view
- A source, stating where this requirement comes from
- A statement, from marketing, of the importance of the requirement
- A project version, also from marketing

This was a cultural shock at first. The requirements database contained over 800 open entries at that time, many of which were outdated. In addition, there were duplicate entries, and several entries where the submitter had left the group and no-one quite knew what was being asked for. The marketing representatives had an especial rude awakening, as they were being told that the majority of the items in the system needed to be reworked, augmented, or otherwise fixed.

The clean-up took the better part of a year. The submitters quickly changed from resistance to support, when they realized that they were finally getting immediate feedback on their work. The architects checked the new entries at least once a month, instead of looking at new submittals only when new releases were planned. The database started to feel like an active tool instead of a data grave. As obsolete entries were identified, over 200 requirements were deleted. The remaining items were well understood by project management, marketing, and the software architects alike. As a side effect, the architects acquired a good understanding what was already in the system, which helped to prevent further duplicates.

6 Reaping the benefits

This clean-up effort allowed the introduction of a more detailed requirements management process in the next release

First, requirements were grouped into "themes" which provide marketing and management with the high-level concepts that they could use in external communications. Because the themes of the release were decided upon earlier, the exercise of prioritizing the requirements became much easier; requirements that did not fit the themes were scratched immediately.

Second, an owner was assigned for each requirement. The owner had the responsibility of investigating the requirement and coming up with the effort estimation. These estimations were also tracked in the requirements tool. By delegating the requirements work to several people, the work load on the architects was significantly reduced. The engineers felt more connected to the project, because they were involved in the process. And finally, individual responsibility for each item also created accountability, which helped to ensure that no requirements were forgotten.

Third, requirements were mapped to project iterations. Products are developed using time-boxed cycles, referred to individually as an "integration cycle" (IC) or alternatively as a "vertical slice" (VS). A requirements report specified the integration cycle in which each item would be delivered, allowing the system test team to plan their test sequence accordingly.

The increased quality of the requirements enabled the individual engineers to work from the requirements directly. Whereas previously, they had to ask for details on 100% of the requirements, now only 10-15% needed follow-up discussions, most of which were on a much more detailed level than any of the original ones. Furthermore, the software architects could usually act on behalf of the original submitters in the discussions, because they had sufficient understanding of the intentions behind the requirement. Today, the original submitters are asked for more information in less than 5% of the requirements — once the requirement has passed the quality gateway. Of course, the effort spent on the quality gateway itself is significant; one of the architects spends about 25% of his time with requirements management.

As a result of these changes, the teams have embraced the requirements process, and more and more work is being done in the requirements tool instead of using spreadsheets and email. Project management has much more confidence in requirements planning, and now demands that each requirement must be entered in the requirements database before a schedule can be committed to. The team work items can be assessed directly from the database. Testers also work from the requirements database, creating system test cases for each requirement.

And at the end of the release, it was relatively easy to find all requirements that had been completed, so that they could be archived, making it easier to mange the data for the remaining product requirements.

7 Further refinements

Following the positive experiences with the new process, further refinements were made for the next release, which is the current one as this document is being written.

It transpired that effort estimation tracking does not really work in the tool. For most requirements, more than one person needs to be involved in tracking the requirement. For example, on our products, we need input from people in the server, GUI, testing, and documentation teams. This made it hard to track the data in the single field that the tool provides. Therefore, effort estimation has been moved back to the investigation reports, which describe what needs to be done to realize a requirement.

For the same reasons, a single owner for each requirement did not work out too well either. However, the tool actually provided the means to assign multiple responsible persons in various roles. Using this capability, each person on the team can now search all requirements on which he or she needs to contribute.

Another pain point in the last release was the discussion of the requirements, which usually happened by email. People not on the distribution list lacked the background to understand changes to the requirements. Therefore, the discussion on the items was moved into the tool as well, which provides a small forum-like capability for this.

Finally, the life cycle of the requirement has been defined more closely, and all the people involved are asked to keep the status up-to-date. This makes it easier for management to track the project progress.

8 The process in detail

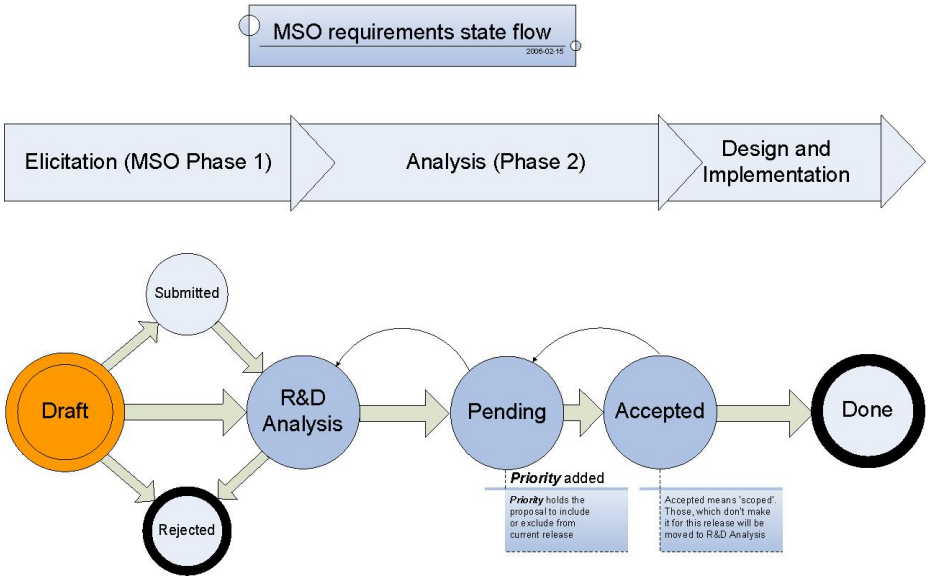

Figure 1: Overview of the requirements management process

All new requirements are entered into a special folder called “Incoming”. Anybody in the organization is allowed to enter requirements here, although most requirements come from marketing.

About once a month, the architects review this folder. All requirements are checked for completeness (see above) and quality. Requirements that do not pass this quality gateway are rejected; the author is notified and is given 60 days to improve the requirement, or it will be deleted. Requirements that pass the quality gateway are given the status “R&D analysis”.

Then, the marketing representative checks the new entry, and decides which planned project version (current release, next release, and so on) should satisfy the

requirement. The marketing representative also decides on the importance of the requirement such as “will delay release for this”, “has to be investigated in detail”, or “opportunistic requirement”.

When both reviews have been completed, the requirement is moved from the “Incoming” folder to the final destination in the system.

All requirements that belong to the current release are assigned to a group of responsible engineers who have to investigate the requirement and come up with the effort estimation. One engineer has the special role of owner, which means that he or she is asked to coordinate the group. The owner is also responsible for updating the requirement status. When the investigation is completed, the requirement is given an R&D proposed priority (in-plan, out-plan, opportunistic) which reflects the confidence that the requirement can be implemented in that release, given the manpower and time available. The requirement status changes to “Pending”.

In parallel, the testers perform another review of the requirement quality from their perspective. If the requirement does not meet the testability criteria, the author is contacted to add the missing details. Although this should be part of the quality gateway already, organizational reality forces us to postpone this step to the investigation phase, unfortunately.

As part of the “end-of-investigation” milestone, marketing and R&D management review the list of requirements again and discuss the proposed priorities. Trade-offs are made, sometimes the schedule is extended to allow for some more opportunistic items. At the end of this scoping session, all priorities are updated to reflect the agreement, and the requirement statuses change to “Accepted”. Also at this time, requirements that are accepted as out-of-plan are usually moved to the next release. After this milestone, new requirements for the current release are accepted only with a formal change request that makes visible the effort involved and potential project delay.

Next, R&D comes up with an implementation plan that shows which project iteration will deliver which requirements. This is used as a basis for the cross-functional teams (testing, documentation) to plan their schedules.

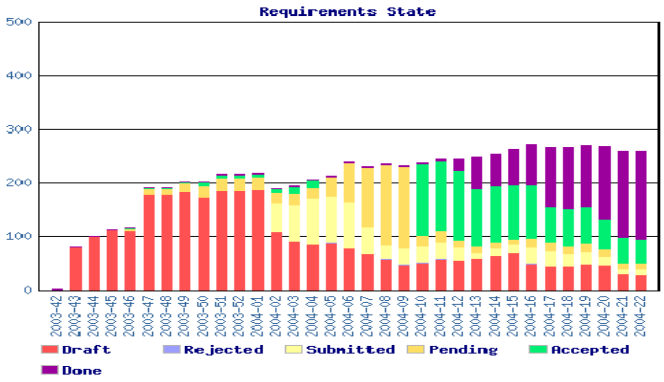

Figure 2: Example of a weekly requirements status report

As soon as a requirement is implemented, its status is changed to "Done" by the requirement owner. Weekly reports show the progress of the team to management.

9 Conclusions

In summary, the value we gain from a solid quality gateway is as follows:

- Stakeholders trust requirement data that is alive, and reflects current thinking.
- R&D can work directly from requirements instead of querying the submitter.
- At least two people have a good overview of all the requirements, both planned and future ones.
- Project schedules can be created on a per-requirement basis.
- Testing can be planned for each requirement before R&D knows the implementation specifics.
- R&D avoids spending time investigating fuzzy requirements
- Stakeholders can track the progress of all requirements, for example, marketing can always check the implementation status for each requirement.

10 Future Plans

No process is ever perfect. Our engineers have already asked for the next improvements, which will be incorporated in future releases.

One request is for finer granularity for requirements in the development process. Today, component teams may face unpleasant surprises if they had not expected to contribute to a requirement. Each team would prefer to have individual requirements for each component, (for example, the server component or the GUI component) instead of sharing a single item. This would allow for a better model of dependencies, for example, where the server team has to deliver before the GUI team. As the component teams are not necessarily in the same location, this would help to decouple their activities. It would also allow tracking the effort estimations in the requirements management tool again. In the end, the complete investigation report document would be a derivative of information stored in the requirements tool.

Another challenge is the necessary integration with changing R&D development process paradigms. Our current approach is still based on the traditional waterfall model which places the investigations and specifications at the beginning of the project. Current theories suggest the superiority of agile processes that investigate, design, and implement in each iteration. Although our current requirements management process is not mutually exclusive with such agile approaches, a change of thought process is required from management and marketing that might be a similar cultural shock as the original introduction of the quality gateway.

Acknowledgements

Gerald Heller provided guidance and consulting both for the requirements management process described inhere, as well as for this paper itself.
Keith Hodson helped with language and grammar.

References

[Caliber] Borland CaliberRM, http://www.borland.com/us/products/caliber/index.html

[Robertson 99] Robertson, Suzanne & James: Mastering the requirements process. Addison-Wesley, Harlow, Great Britain, 1999

[Volere] Volere template, http://www.volere.co.uk/template.htm