

Maybe We Shouldn't "Write" Requirements

By David Gelperin

Summary

In this column, David Gelperin presents a problem familiar to many of us—what is the best way to record requirements? Given the limitations of static templates, how can we best manage high-volume, multidimensional requirements information?

Included at the end of the text are reader comments as posted during original publication. This column first appeared 7 Oct 2002 on StickyMinds.com at <http://www.stickyminds.com/r.asp?F=W5936>

We're planning a new project and I've been thinking about our past struggles with requirements. We have always created a requirements document, but that may not be our best approach this time.

Documents have worked fine on our smaller projects, where requirements could be specified in twenty or thirty pages. We even expect to "write requirements" and our language reflects this expectation. In addition, multiple templates are available within our organization and on the Web. The templates provide various information layouts for requirements documents; however, the effectiveness of documents does not scale up.

Large documents are hard to check and hard to use. Because a document's organization is fixed, authors must choose the single best way to arrange information. Unfortunately, for large aggregates, there is no single best way or even one pretty good way. The same requirements information needs to be organized in different ways to satisfy different needs including the need to check accuracy and completeness. As page count and number of stakeholders increase, the need increases to develop information with a fluid, rather than fixed, organization; i.e., a database.

For example, my wife Sharon and I recently created our guest list for our son's wedding. We could have used a word processor, which would have worked well for twenty or thirty people. Since we were inviting about 130 people, we used a spreadsheet program. Along with names, addresses, and relationships, we entered response and meal preference information. The spreadsheet's sort capability enabled us to easily determine (1) the number of people choosing vegetarian, fish, or meat dinners, and (2) the list of out-of-towners and exactly where they were from. The spreadsheet's computational capability provided an important derived value, the total number of acceptances.

While sorting by relationship, we noticed that some cousins who we planned to invite were not on the list. While the missing cousins might have been detected from an alphabetical list, aggregating by relationship made their absence much easier to spot.

Requirements information, too, needs to be aggregated in various ways. Examples include (1) all critical requirements, (2) all requirements from a single stakeholder, (3) all unfinished requirements, and (4) all interface requirements. Higher volume, multidimensional information is best managed by a processor that can calculate, query, and reorganize.

Computational capability supports the use of derived (virtual) information. For example, explicit links from a test to its source requirements, permit the derivation of links from the source requirements to the tests. This significantly increases the manageability of multiple aggregates of linked information that change frequently.

Since larger scale requirements should be developed using either a spreadsheet or a database, I must now decide whether a document or database will best serve our new project.

I think of a spreadsheet as a limited function database (with extra computational capabilities). The choice is again a matter of scale. With dozens of requirement properties, a spreadsheet becomes too hard to use—as a document did. So we have a usage spectrum with documents and databases at the endpoints and spreadsheets in the middle. What do you think? I'd like to hear what you use and how you decided to use it.

Reader Comments

David, while you have now mentioned agile software development, it is clear that many of the system requirements discussed here are orders of magnitude beyond the “write-it-on-an-index-card” and Alistair Cockburn’s “two-people-at-a-white-board” agile approach. If you were to undertake these projects on an agile basis, the little-step-by-little-step approach would very possibly be completely inadequate to handle the innate complexities of these larger systems. Also no one seems to have mentioned that market pressures and other forces can now mean that software may need to be delivered much earlier than intended, so the concept of testing to satisfy the absolutes of must-have/nice-to-have requirements disappears. The state/health of requirements needs to be known at the current point in time. Risks involved with outstanding requirements need to be reported to give an indication of the upside/downside/rescoping needed to release in the (shortened) timeframe. As far as I know, no tools address this issue yet. [Paul Gerrard has done a lot of work on this].....(10/13/02) Erik Petersen

David, As you know (from RESG, University College London, Oct-2002), over the last few years I have been developing hierarchical event driven use case modelling techniques for capturing requirements. Each use case having a clear goal with a scenario detailing how that goal must be met, avoiding solutions if at all possible (as that bits called design). I have found that using a hierarchy, does make the task of managing a large amount of requirements much more manageable. The decision on the use of databases or documents is from my point of view very simple, the database wins every time. If you want a hardcopy of all or some of your requirements, you simply run a query and produce a report. When you design the structure of your database you can add any attributes necessary for supporting queries that you wish to run. I have come across a number of requirement databases including Rational products mentioned by many other posted replies, however I mainly use Telelogic’s DOORS and highly recommend it. However what ever you decide to use to manage your requirements, a document or a database, the most important factor to a successful development is to write good requirements. A very useful book on this subject is “Mastering the Requirements Process” by Suzanne and James Robertson (ISBN 0-201-36046-2), a superb book. Good Luck.(10/11/02) Malcolm Stenning

David, I think the volume of response and the desire for some give and take discussion suggest that this should be turned into a Roundtable discussion. The mechanics of the weekly column actively prevent a person from responding more than once to the article, so John Daugherty can’t respond to your response with clarifications. The lifecycle and media of requirements seems particularly fruitful these days. Thanks! Bob Lee....(10/11/02) Robert E. Lee

I spent years developing 1000s of requirements for the space program. We had volumes of data and many requirements documents. All were created and managed manually - including traceability. That was all I knew until I left the space program. What I found in the business world horrified me. Forget requirements documents; forget requirements! I now work exclusively with Rational's RequisitePro and I love it. It allows me to create requirements documents or requirements that reside only in a database. In either case, the requirements are in a database that can be sorted and filtered and reports can be generated. However, where organizations are really lacking is in the ability to create good requirements which is as much art as it is science. Without requirements management, it makes no difference where your requirements are. A bad requirement in a spreadsheet is just as harmful as one in a database, document or tool.....(10/10/02) Judy Murphy

Author's Response: Judy: You are clearly right that content quality matters much more than volume or packaging. A choice between a few good requirements in a document and many poorer requirements in a database is not really a choice at all. David Gelperin....(10/10/02)David Gelperin

Very good ideas floating around here. I am not at all familiar with the requirements tools (automated) on the market. I would like to say, though, that the attributes I would be looking for in a tool would facilitate ease of use, searchability along the lines that David brought up, traceability, redundancy checking, and change control. As any automated solution is going to, by nature, force a formal schema on the requirements, I suggest that the Planguage authored and championed by Tom Gilb in his book Competitive Engineering might be a helpful framework for writing requirements that can be supported by such a tool. - JB Dubowski....(10/09/02) Joseph Dubowski

David, Certainly voluminous documentation for REQUIREMENTS generally aren't of much help. But if we look back to the very purpose of documentation; it is to STREAMLINE the development process. Therefore, as an organization matures in terms of PROJECTS, these REQUIREMENTS documents also need to be relooked & revised based on the types of Projects undertaken. Therefore documents can be categorised based on the type of projects undertaken. One of the other factors that affects these documents are the awareness of these documents that the end-user who is going to use it. Therefore, it makes more sense to draft these documents based on the end-users feedback & should always be involved as when changes are needed. Quality can be only successful when people embrace it and share a common vision.....(10/08/02) Mahendra Gupta

Author's Response: Mahendra: I am currently reading Alistair Cockburn's "Agile Software Development". As usual he writes beautifully and has many important things to say. If you are not familiar with the Agile Philosophy, I recommend this book. You may find some things difficult to accept, but I think this philosophy should be understood more widely. I bring all of this up because you comment that the purpose of documentation is to streamline the development process. Those in the Agile school would disagree. They would say that documentation slows down development. Important issues should be researched and debated and the appropriate role of documentation is one such issue. Thanks for your comment. David Gelperin....(10/10/02)David Gelperin

We recently used an Access database (customized and used on other projects) and utilized the Word Merge function to export the Requirement text fields into a Word document template. Once it was set up, some minimal configuration by some developers, it was great for updates and reviews. There were some

limitations with several users accessing the same database, used a Microsoft share function to help eliminate that issue, but the database still needed to be locked for mass updates or creating new tables. Another limitation was version control. We have also tracked Requirement changes in a bug tracking tool which allowed us to follow through to updating the document...which is useful after a Requirements Document is signed.....(10/08/02) April Anderson

Author's Response: April: Thanks for sharing the details of your experience. David Gelperin....(10/10/02)David Gelperin

Having worked in some companies with long, painful requirements documents and others with none at all, it seems I have constantly been thinking at some level about how to solve the problem of documenting them. While my tool of choice has ended up being a database, or Excel with hyperlinks when time is very short, the more important thing to me has been what to document. Many requirements documentation efforts become massive, time-sucking monsters because this question never gets asked. In my last company we were defining requirements for a feature in our product that supported DWDM Optical networking equipment. There ended up being thousands of "requirements" which were all nicely organized in an access database, but they addressed so many issues, many of which were too detailed for our application, they were never used. The rest of the company realized that digesting all this data would take all the time in the development plan. In the end, the product that was built had to be changed significantly because it had not met the real requirements. When I look back on it a five-page Powerpoint presentation with diagrams of DWDM equipment in real-life networks would have been much more effective. I will throw in my vote for diagrams as an effective medium for documenting some projects' requirements. I apologize for drifting a bit from the subject, but I think there can be a link between what needs to be documented and the best method for documenting it. As my eighth-grade English teacher used to say as we were starting an exam, "write all that is needed and only that which is needed." In too many cases I have witnessed, requirements became the goal instead of a tool for making the development process work better. One danger of moving requirements to a database format is that the creation of more data becomes easier and the risk of creating more information than is useful to the users is greater. My bet is that, as you were creating a list of wedding invitees, in your mind you had an idea how the "requirements" would support your real project, which was a successful and painless wedding. The proper ordering of meals, how many rooms to block at a nearby hotel, and easy addressing of invitations are examples. These criteria had a big hand in defining the columns to use in Excel, just as the players in and the logistics of a software project would define what needs to be in the requirements. In my last project, noone was interested in documenting requirements, and I was the sole tester with minimal time to spend on such "luxuries." I used a database and started by writing what I believed were the five things our product had to do to be successful. I then shared these with selected team members and got a feel from them where more information was needed. Using such iterative reviews to build the requirements, I ended up with a smaller but comprehensive set of data. Using such an approach, a database was the ideal medium. I believe I also got better buy-in from the other players since they never had to allocate multiple days to review a large, intimidating document.....(10/08/02) John Daughety

Author's Response: John: Great comments. You provide so many insights, that I have attempted to summarize them below. Please straighten out my misunderstandings or omissions. 1) Concentrating on project objectives along with good judgment are necessary for developing just the right requirements. 2) Too many "requirements" can be worse than too few. 3) Some requirements information is best presented by diagrams. 4) Databases can be dangerous by inviting much more information than is necessary. 5) On the other hand, databases can be useful in hiding the total volume of information thereby aiding the iterative reviews used to validate the developing set of requirements. David Gelperin....(10/10/02)David Gelperin

Ofer prat asked, " There are several requirements management tools in the market (e.g. Caliber RM). Does anyone has experience with these tools?" I can't imagine doing requirements analysis without the use of a requirements tool. Our company is currently evaluating CaliberRM and Rational RequisitePro. I really love CaliberRM. It allows me to define a requirement, show its relationship to other requirements, create attributes for requirements, and print out wonderful reports in Word. Let me give you an example: At a former company, we were evaluating two programs that were very different. For each requirement that we had gathered, I set a numerical value as an attribute. The value was "2" if the program could fully meet the requirement; "1" if the program could partially meet the requirement; and "0" if the program could not meet the requirement. I then added up the values for all requirements, and printed a report showing each programs final score as well as the list of "0" requirements.(10/08/02) Diane Evans

David is baiting us on this one. As soon as requirements get somewhat complicated there are a number of other trends often associated. 1) Rate of requirement changes increases; 2) Number of authorized updaters increases. Both of these situations point towards a managed solution. If only we could all afford the cost of purchase, maintainance and training!....(10/08/02) Peter Cornish

You are right about documents scaling up to spreadsheets and up from there to databases. We tried using full requirements management tools such as Rational's Requisite Pro when we were still too small a company to capitalize on its strengths. If the project and project team are too small, the labor involved in creating and maintaining a requirements database is disproportionate to the project budget and timeline. Also, unless all project disciplines are utilizing the database's capabilities, it is not cost effective. We downgraded to using straight word processing documents and found that worked better within the constraints of our company and projects. Now that we have grown larger, documents are not sufficient, as you said, to adequately track and cross-reference different types of requirements. We have now upgraded to using spreadsheets with links and crossreferences to documents. This is a precursor to the requirements databases and their document template support. I assume that we will eventually outgrow the spreadsheet and will have to go to a requirements management software. Based on our experience, however, I do believe that you must use a storage method proportionate to the size of your project and project team. Overkill on the tools will hurt your productivity and profit. Under utilization will do the same. The trick is in matching the level of the tools used to the level of your development project. Laura Murphy
Peripherals Plus Technologies, Inc.....(10/08/02) Laura Murphy

This is an interesting article about requirements aggregation and long windy documents. I totally agree that big unruly requirements doc's are a management nightmare. In my experience, 100+ page test requirement documents in word that frame the application under test into a very tight little box tend to be extremely time intensive for the benefit they provide. The problem is the requirement creep that is constantly occurring. The document maintenance is a total chore. I'm inherently lazy myself, so I like to rely on human intuition for much of my testing, but that's not all, I use a spreadsheet as a guide/checklist for things that require detailed testing. I just combine the elements by using a series of flags to identify the commonalities between testcases. Each testcase then has three textual columns, a "common steps" column that contains the basic test steps, a "unique steps" column for differentiating this testcase from the last, and an "expected" column to really drive home the requirements. This way, I don't focus on the little itty bitty trees dotting the forest, I focus on the forest and move through the land. I'm in the process of

writing testcases/requirements for a very complicated tripple database merge and tripple platform user authenticated session translation. The scenarios are many, and the testing is complicated and easy to get lost in... that's why I do it in excel. As for databases, I find databases are too slow to update manually without building a GUI, which takes too much time. The only time I use a database is when I can plug it into an automated tool. I've had success using MySQL as a backend datastore and results capturing medium for an automated frontend (SilkTest and PHP). I still find that for RAD type environments like where I work, a spreadsheet with commonalities identified and split apart (like database normalization) is the best solution for aggregate requirements writing for me.....(10/08/02) Clay Givens

David, When any body of knowledge becomes too large to be held in the head at one time, it needs to be organized. A spreadsheet may be appropriate in some cases, but I have found that hypertext (which is itself a sort of database management system) to be very effective, especially for tracability (including to the design and implementation) and for capturing dependencies and other relationships.[para] The common idea (repeated below) that diagrams always beat text is simplistic. Diagrams are a good way to illustrate relationships, but they lack the expressive, explanatory, and (unless you are using existential graphs) reasoning power of natural language. A requirements spec. (or design) in pictures alone is almost certainly incomplete.[para]Finally, of course, there are the mathematics of formal methods if your problem demands a great deal of rigor.....(10/08/02) Andrew Raybould

While I was hopeful that the article would present a feasible method of displaying requirements data in a visual format, or possibly a process for documenting requirements without cumbersome documentation at all (pipedream), I think the concept of using a spread sheet or relational database is a wise one. It is prudent anytime you need to document functionality that involves multiple subsystems and their boundaries. By filtering based upon relationships you can identify issues earlier. You can also more thoroughly assess the project and hopefully you and the developers can better estimate the time it will take to get all those new features to market. Two drawbacks to this form of documentation are (1) it can be just as timeconsuming to design and maintain as conventional documents, and (2)you still need to pick a format to take into a meeting since you may not have control of the projector.....(10/08/02) Sam Shober

Hi. There are several requirements management tools in the market (e.g. Calibar RM). Does anyone has experience with these tools?....(10/08/02) ofer prat

The answer to David's question seems very obvious to me: Requirements management tools. The key word is "management". I think it better describe all actions performed on the initial requirement during the project life cycle. A tool can solve all the problem that David have mentioned (queries on different criteria, different formats/templates for different purposes, calculations capabilities) and many others, especially used in parallel with other tools for version control, change management, design, automated testing. Anyway, even alone, a requirement management tool combine the advantages of the database, with the capabilities of word processors, under a user friendly interface, and with an internal logic, which contains necessary rules for requirements management, and which could be customized to meet satisfy specific environments, business rules and even templates. I know there are many of those, but the only one I'm familiar with are Rational product, who has RequisitePro for requirements management. In fact the use of specific templates, spreadsheets or database, with specific queries, under specific business rules

in every organization are core functionalities of any requirements tools, but without (or less) automation and user friendly interface. The only “problem” could be the costs of such tools. If for large and expensive projects, costs associated with automated tools could be afford, for smaller projects costs could be prohibitive. I hope in near future, more this tools will be available, at low costs and even free, like many defect tracking tools, which seems to be one step further than requirement management in automation. I don’t want to be misunderstood; I don’t say that a requirement management tool will ever replace the specialists, as some could believe. It will only help him/her organizing and formatting the data and focusing on important problems.(10/08/02) Daniel SUCIU

Managing requirements differs from publishing requirements. We like to publish requirements in a document (easy to use). However if you manage requirements (create, change, discuss, postpone) it just a question of manipulating the attributes related to the requirements. For this purpose a document is error prone. So why don’t use a repository. Some tools are able to use a repository for managing the requirements and at the end a document can be generated.....(10/08/02) Hans Thelosen

Do you think using Business Process Model and Business Function Model in Designer could help you capture your requirements in a meaningful way.....(10/08/02) sameer nigam

I found the column really interesting, as one of those simple things that it’s surprising it wasn’t thought of before. I work in a rather big telecom company where projects have a very wide scope and must be broken into pieces to be defined/implemented by groups specialized in each of the areas. The process works, but it is really cumbersome as all involved people (>80) have to ‘fish’ for requirements through the whole document even when only a small part affects their area, not to mention the overhead associated with changes after document approval. The issue at question might seem trivial, as it is only a matter of technology/tools and the data manipulation possibilities they offer, but I feel such a simple change would have a highly noticeable productivity impact.....(10/08/02) Alberto López Navarro

I have to say that the best help I have EVER had for writing requirements (or any other documentation) was from a course in Information Mapping. It has really focused the way I write things, and has made the documents so much quicker to read. Its designed to make it quick to scan and therefore incredibly quick to find the information that you need. I’d really recommend it. For more information on the method and process (and some examples) please see: <http://www.infomap.com/>....(10/08/02) Fiona Williams

Definitely! What a fantastic definition of a common problem. I’ve always found this to be a difficulty, but never was able to articulate it this way. Thank you for putting the right words out there. I wish I had a good solution to offer. The best template for Requirements that I’ve seen comes from RUP. Everything is numbered and categorized. But, that’s still a “text” document. And I can see the probs that using a spreadsheet may cause. Hmmmm... maybe it’s time for someone to come up with something REALLY innovative - maybe even SEARCHABLE. They might make themselves a million bucks!....(10/07/02) Sandy Flann

Whether one uses word processors, spreadsheets, or databases, one ultimately writes (“to set down in writing”) requirements. From the column title, I expected a proposal such as “drawing” requirements, i.e., graphical rather than textual representation of requirements. After wading through pages of a requirements specification describing shifts between modes of the application, drawing a state-transition diagram greatly simplifies the requirements -- and usually exposes inconsistencies and errors. As a tester, how many times have you drawn a picture to clarify a narrative specification? How many times have you found the developer did the same thing? More importantly, how many times did your picture match the developer’s? My favorite specification tool isn’t a word processor, spreadsheet, or database: it’s a drawing tool. Any kind of drawing tool. Paper and pencil, if necessary. Sure, it helps to have a tool with some knowledge of the representation language (e.g., checking consistency of data flows), but even without it, a picture is superior to narrative English!....(10/07/02) Kevin Priest

Author’s Response: Kevin: I should have mentioned diagrams (state transition, split activity, and essential class) as they are the best way to present important forms of requirements information. But, they are not always the best as you probably know since there is no one best specification method. For example, at a summary level, most non-functional requirements need narrative text descriptions. At a detail level, these requirements are best specified by acceptance/system test procedures. Another example is the detailed definition of composite states e.g. the client is eligible for a particular type of insurance policy. “eligible” is a composite state defined by a logical expression composed of basic states (attribute values) of the client object. The policy may be restricted by any combination of age, sex, marital status, home ownership, etc. etc. Neither diagrams nor narrative text is an effective way to specify logical expressions --- that’s why we have mathematics. Effective requirements specification requires a heterogenous strategy that finds the clearest method of expressing each of the diverse forms of requirements information. That said, we all have our favorites. Thanks for you comments. David Gelperin....(10/10/02)David Gelperin

You’re getting off pretty easy, David. The usual flood of comments is off to a slow start. Probably because what you say is dead on. Recording the requirements of a million dollar project in a flat file is just one more example of the shoemaker himself, not his children, going barefoot. The popular project management software tool that I use doesn’t even have a place to list requirements, much less manage them. Just as the “requirements” driving early MIS projects were simply models of the clerical processes they were to replace, so have our project management tools merely facilitated our continued inattention to the early phases in the project lifecycle.....(10/07/02) Gene Fellner

I was first surprised by the title “Maybe We Shouldn’t “Write” Requirements” ?? Maybe the title should be changed to various methods for capturing requirements or something else, as in the article you do say – requirements should be there, however the tool used to document what they are may differ from Excel to Word, or maybe even power point as more of a conceptual picture that leads down to the requirement. We use Word as our form of requirements document. But inside word there are tables – sometimes embedded Excel sheets, as well as pictures, either from Visio or Power point to get the point across. When we begin testing these documents are used to verify the requirements, by running the associated, inspected, and approved test cases provided.(10/07/02) Chris DeNardis

Author’s Response: Chris: Thanks for pointing out that there is another point on the recording spectrum. We have (1) word processors, (2) spreadsheets, (3) the enriched combination that you

describe,(4)databases and (5) requirements management platforms (which can be viewed as enriched databases). David Gelperin....(10/10/02)David Gelperin

About the Author

David Gelperin (sqegelp@aol.com) was one of the founders of Software Quality Engineering (sqe.com) in Orange Park, Florida. David has more than thirty years' experience in software engineering with an emphasis on quality control. He has been a programmer, project lead, test lead, quality support manager, test consultant, and instructor.

He chaired the development of both ANSI/IEEE standards on software test and catalyzed the launch of Software Test and Quality Engineering magazine. He is chief architect of the following:

Systematic Test & Evaluation Process
(STEP™) test methodology
High-Impact™ technical review methodology
Unique Cause and Pre-emptive Debugging test strategies
Testability Support Model
Ultra-Understandable Usage (U3) Modeling framework

After majoring in math at Carleton College, David received an MS and Ph.D. in Computer Science from the Ohio State University.