Requirements Gathering

Asked 16 years, 3 months ago Modified 7 years, 9 months ago

Viewed 23k times

29

votes

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

How do you go about the requirements gathering phase? Does anyone have a good set of guidelines or tips to follow? What are some good questions to ask the stakeholders?

I am currently working on a new project and there are a lot of unknowns. I am in the process of coming up with a list of questions to ask the stakeholders. However I cant help but to feel that I am missing something or forgetting to ask a critical question.

requirements-management

Share

edited Mar 10, 2017 at 17:28

Bhargav Rao

52k ● 29 ● 126 ● 141

asked Aug 26, 2008 at 22:31

Ryan Sampson
6,807 ● 12 ● 50 ● 55

Comments disabled on deleted / locked posts / reviews

20 Answers

Sorted by: Highest score (default)

20
votes

You're almost certainly missing something. A lot of things, probably. Don't worry, it's ok. Even if you remembered everything and covered all the bases stakeholders aren't going to be able to give you very good, clear requirements without any point of reference. The best way to do this sort of thing is to get what you can from them now, then take that and give them something to react to. It can be a paper prototype, a mockup, version 0.1 of the software, whatever. Then they can start telling you what they really want.

Share

answered Aug 26, 2008 at 22:36

Chris Upchurch
15.5k ● 6 ● 52 ● 66

6 So that, after you write the app, they can tell you that it's not really what they wanted after all. – Robert Harvey Aug 14, 2009 at 18:20

@RobertHarvey only if it's a microsoft product? amirite? all jokes aside. This is the best approach. You just need to work with your customers to get a feel for what it is that they need to do. Create teams from their dedicated power users that understand flows/processes...etc and get them to start outlining what it is they are currently doing, how they want to transform it, how they expect it to end up...etc Create mock-ups and give your users something to play with, even if it's

only visual, let them see how you are interpreting their instructions/requirements. – Mike McMahon Mar 30, 2012 at 0:16

20

votes

See obligatory comic below...

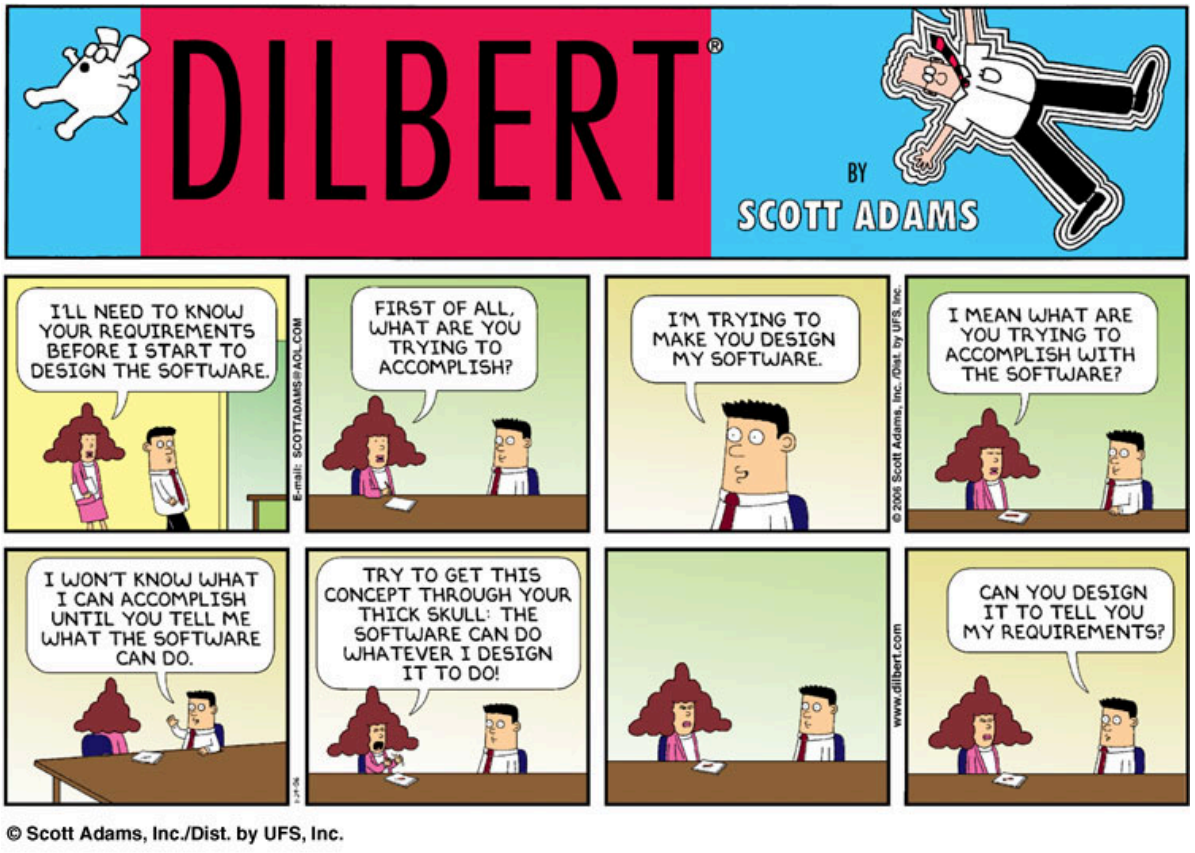
In general, I try and get a feel for the business model my customer/client is trying to emulate with the application they want built. Are we building a glorified forms processor? Are we retrieving data from multiple sources in a single application to save time? Are we performing some kind of integration?

Once the general businesss model is established, I then move to the "must" and "must nots" for the application to dictate what data I can retrieve, who can perform what functions, etc.

Usually if you can get the customer to explain their model or workflow, you can move from there and find additional key questions.

The one question I always make sure to ask in some form or another is "What is the trickiest/most annoying thing you have to do when doing X. Typically the answer to that reveals the craziest business/data rule you'll have to implement.

Hope this helps!



Share

edited Mar 30, 2012 at 0:27

Neysor

3,911 ● 11 ● 35 ● 66

answered Aug 26, 2008 at 22:36

Dillie-O

29.7k ● 14 ● 102 ● 141

1 McConnell says, this kind of problems are quite common, and are called "wicked problems" (ie: you can describe them only by solving them). – smok1 May 27, 2009 at 7:58

1 Another comic related to how requirements are communicated. cavdar.net/uploads/2011/10/requirements_accavdar.jpg – E-rich Oct 13, 2011 at 15:09

12 votes

Steve Yegge talks fun but there is money to be made in working out what other people's requirements are so i'd

take his article with a pinch of salt.

Requirements gathering is incredibly tough because of the manner in which communication works. Its a four step process that is lossy in each step.

- I have an idea in my head
- I transform this into words and pictures
- You interpret the pictures and words
- You paint an image in your own mind of what my original idea was like

And humans fail miserably at this with worrying frequency through their adorable imperfections.

Agile does right in promoting iterative development. Getting early versions out to the client is important in identifying what features are most important (what ships in 0.1 - 0.5 ish), helps to keep you both on the right track in terms of how the application will work and quickly identifies the hidden features that you *will* miss.

The two main problem scenarios are the two ends of the scales:

- **Not having a freaking clue about what you are doing** - get some domain experts
- **Having too many requirements** - feature pit. - Question, cull (prioritise ;)) features and use iterative development

Yegge does well in pointing out that domain experts are essential to produce good requirements because they know the business and have worked in it. They can help identify the core desire of the client and will help explain how their staff will use the system and what is important to the staff. Alternatives and additions include trying to do the job yourself to get into the mindset or having a client staff member occasionally on-site, although the latter is unlikely to happen.

The feature pit is the other side, mostly full of failed government IT projects. Too much, too soon, not enough thought or application of realism (but what do you expect they have only about four years to make themselves feel important?). The aim here is to work out what the customer *really* wants. As long as you work on getting the core components correct, efficient and bug-free clients usually remain tolerant of missing features that arrive in later shipments, as long as they eventually arrive. This is where iterative development really helps.

Remember to separate the client's ideas of what the program will be like and what they want the program to *achieve*. Some clients can create confusion by communicating their requirements in the form of application features which may be poorly thought out or made redundant by much simpler functionality then they think they require. While I'm not advocating calling the client an idiot or not listening to them I feel that it is worth forever asking *why* they want a particular feature to get to its underlying purpose.

Remember that in either scenario it is of imperative importantance to root out the quickest path to fulfilling the customers *core* need and put you in a scenario where you are both profiting from the relationship.

Share

edited Aug 3, 2011 at 16:23

answered Aug 27, 2008 at 0:30

Quibblesome

25.4k ● 10 ● 62 ● 104

8

votes

Wow, where to start?

First, there is a set of knowledge someone should have to do analysis on some projects, but it really depends on what you are building for who. In other words, it makes a big difference if you are modifying an enterprise application for a Fortune 100 corporation, building an iPhone app, or adding functionality to a personal webpage.

Second, there are different kinds of requirements.

- Objectives: What does the user want to accomplish?
- Functional: What does the user need to do in order to reach their objective? (think steps to reach the objective/s)
- Non-functional: What are the constraints your program needs to perform within? (think 10 vs 10k simultaneous users, growth, back-up, etc.)

- Business rules: What dynamic constraints do you have to meet? (think calculations, definitions, legal concerns, etc.)

Third, the way to gather requirements most effectively, and then get feedback on them (which you will do, right?) is to use models. User cases and user stories are a model of what the user needs to do. Process models are another version of what needs to happen. System diagrams are just another model of how different parts of the program(s) interact. Good data modeling will define business concepts and show you the inputs, outputs, and changes that happen within your program. Models (and there are more than I listed) are really the key to the concern you list. A few good models will capture the needs and from models you can determine your requirements.

Fourth, get feedback. I know I mentioned this already, but you will not get everything right the first time, so get responses to what your customer wants.

As much as I appreciate requirements, and the models that drive them, users typically do not understand the ramifications of of all their requests. Constant communication with chances for review and feedback will give users a better understanding of what you are delivering. Further, they will refine their understanding based on what they see. Unless you're working for the government, iterations and / or prototypes are helpful.

answered Sep 20, 2008 at 16:05


Jeffrey Davidson
184 ● 1 ● 4

6
votes

First of all gather the requirements **before** you start coding. You can begin the design while you are gathering them depending on your project life cicle but you shouldn't ever start coding without them.

Requirements are a set of well written documents that protect both the client and yourself. Never forget that. If no requirement is present then it was not paid for (and thus it requires a formal change request), if it's present then it **must** be implemented and must work correctly.

Requirements must be testable. If a requirement cannot be tested then it isn't a requirement. That means something like, "The system "

Requirements must be concrete. That means stating "The system user interface shall be easy to use" is not a correct requirment.

In order to actually "gather" the requirements you need to first make sure you understand the businness model. The client will tell you what they want with its own words, it is your job to understand it and interpret it in the right context.

Make meetings with the client while you're developing the requirements. Describe them to the client with your own words and make sure you and the client have the same concept in the requirements.

Requirements require concise, testable example, but keep track of every other thing that comes up in the meetings, diagrams, doubts and try to mantain a record of every meeting.

If you can use an incremental life cycle, that will give you the ability to improve some bad gathered requirements.

Share

answered Aug 26, 2008 at 23:56

Jorge Córdoba

52.1k ● 11 ● 82 ● 130

3

votes

You can never ask too many or "stupid" questions. The more questions you ask, the more answers you receive.

Share

answered Aug 26, 2008 at 22:34

Robert Durgin

1,830 ● 19 ● 23

3

votes

According to Steve Yegge that's the wrong question to ask. If you're gathering requirement it's already too late, your project is doomed.

Share

answered Aug 26, 2008 at 23:25

Sam Hasler

12.6k ● 10 ● 73 ● 106

wall of text detected at given url...tl/dr – Adam Bellaire Jan 24, 2009 at 14:08

Yeah, Steve Yege has a reputation for writing long blog posts. See the following for his explanation for why he does it on purpose: steve-yegge.blogspot.com/2008/01/… (half as long as the one I link to in the answer, although that's still not short :) – Sam Hasler Jan 26, 2009 at 16:16

I don't totally agree with Steve Yegge here (many commenters to his post didn't), but I think the essay is a useful read to those learning about requirement gathering. If you, personally, don't know how the thing is supposed to work or really care how it's supposed to work on a day-by-day personal basis, it's just not gonna work the way the customer wants it to. A tough pill to swallow, methinks. – J. Polfer Oct 15, 2009 at 20:22

2

votes

1. High-level discussions about purpose, scope, limitations of operating environment, size, etc
2. Audition a single paragraph description of the system, hammer it out
3. Mock up UI
4. Formalize known requirements
5. Now iterate between 3 and 4 with more and more functional prototypes and more specs with more details. Write tests as you go. Do this until you have functional software and a complete, objective, testable requirements spec.

That's the dream. The reality is usually after a couple iterations everybody goes head-down and codes until there's a month left to test.

Share

answered Aug 27, 2008 at 0:45

Aidan Ryan

11.6k ● 13 ● 59 ● 87

1

vote

[Gathering Business Requirements Are Bullshit](#) - Steve Yegge

Share

answered Sep 16, 2008 at 10:56

Josh Smeaton

48.7k ● 24 ● 135 ● 165

1

vote

- read the agile manifesto - working software is the only measurement for the success of a software project
- get familiar with agile software practices - study Scrum , lean programming , xp etc - this will save you tremendous amount of time not only for the requirements gathering but also for the entire software development lifecycle
- keep regular discussions with Customers and especially the future users and key-users
- make sure you talk to the Persons understanding the problem domain - e.g. specialists in the field
- Take small notes during the talks
- After each CONVERSATION write an official requirement list and present it for approving. Later on it would be difficult to argue against all agreed documentation

- make sure your Customers know approximately what are the approximate expenses in time and money for implementing "nice to have" requirements
- make sure you label the requirements as "must have" , "should have" and "nice to have" from the very beginning, ensure Customers understand the differences between those types also
- integrate all documents into the latest and final requirements analysis (or the current one for the iteration or whatever agile programming cycle you are using ...)
- remember that requirements do change over the software life cycle , so gathering is one thing but managing and implementing another
- KISS - keep it as simple as possible
- study also the environment where the future system will reside - there are more and more technological restraints from legacy or surrounding systems , since the companies do not prefer to throw to the garbage the money they have invested for decades even if in our modern minds 20 years old code is garbage ...

Share

answered May 19, 2009 at 6:03

Yordan Georgiev

5,420 ● 2 ● 59 ● 56

---

1 vote

Like most stages of the software development process its iteration works best.

First find out who your users are -- the XYZ dept,

Then find out where they fit into the organisation -- part of Z division,

Then find out what they do in general terms -- manage cash

Then in specific terms -- collect cash from tills, and check for till fraud.

Then you can start talking to them.

Ask what problem they want you want to solve -- you will get an answer like write a bamboozling system using OCR with shark technoligies.

Ignore that answer and ask some more questions to find out what the real problem is -- they cant read the till slips to reconcile the cash.

Agree a real solution with the users -- get a better ink ribbon supplier - or connect the electronic tills to the network and upload the logs to a central server.

Then agree in detail how they will measure the success of the project.

Then and only then propose and agree a detailed set of requirements.

Share

answered May 27, 2009 at 8:24

James Anderson

27.4k ● 7 ● 54 ● 80

1

vote

I would suggest you to read Roger-Pressman's Software Engineering: A Practitioner's Approach

Share

answered May 27, 2009 at 8:33

TheVillageIdiot

40.5k ● 22 ● 135 ● 191

1 Ouch, the Amazon reviews of that book are brutal.
– Richard Morgan Jul 20, 2010 at 1:28

1

vote

Before you go talking to the stakeholders/users/anyone be sure you will be able to put down the gathered information in a usefull and days-lasting way.

- Use a sound-recorder if it is OK with the other person and the information is bulky.
- If you heard something important and you need some reasonable time to write it down, you have two choices: ask the other person to wait a second, or say goodbye to that precious information. You wont remember it right, ask any neuro-scientist.
- If you detect that a point need deeper review or that you need some document you just heard of, make sure you make a commitment with the other person to send that document or schedule another meeting with a more specific purpose. Never say "I'll remember to ask for that xls file" because in most cases you wont.

- Not to long after the meeting, summarize all your notes, recordings and fresh thoughts. Just summarize it rigth. Create effective reminders for the commitments.
- Again, just after the meeting, is the perfect time to understand why the gathering you just did was not as right as you thought at the end of the meeting. That's when you will be able to put down a lot of meaningful questions for another meeting.

I know the question was in the perspective of the pre-meeting, but please be aware that you can work on this matters before the meeting and end up with a much usefull, complete and quality gathering.

Share

edited Aug 24, 2011 at 19:55

answered Aug 24, 2011 at 17:04

daniloquio

3,882 ● 2 ● 40 ● 56

1

vote

I've been using mind mapping (like a work breakdown structure) to help gather requirements and define the unknowns (the #1 project killer). Start at a high level and work your way down. You need to work with the sponsors, users and development team to ensure you get all the angles and don't miss anything. You can't be expected to know the entire scope of what they want without their involvement...you - as a project manager/BA - need to get them involved (most important part of the job).

Share

edited Mar 30, 2012 at 0:05

sarnold

104k ● 23 ● 185 ● 243

answered Jan 14, 2009 at 1:56

meade

23.3k ● 12 ● 33 ● 36

1

vote

There are some great ideas here already. Here are some requirements gathering principles that I always like to keep in mind:

Know the difference between the user and the customer. The business owners that approve the shiny project are usually the customers. However, a devastating mistake is the tendency to confuse them as the user. The customer is usually the person that recognizes the need for your product, but the user is the person that will actually be using the solution (and will most likely complain later about a requirement your product did not meet). Go to more than one person

Because we're all human, and we tend to not remember every excruciating detail. You increase your likelihood of finding missed requirements as you talk to more people and cross-check.

Avoid specials When a user asks for something very specific, be wary. Always question the biases and see if this will really make your product better.

Prototype Don't wait till launch to show what you have to the user. Do frequent prototypes (you can even call them beta versions) and get constant feedback throughout the development process. You'll probably find more requirements as you do this.

Share

edited Mar 30, 2012 at 0:06

sarnold

104k ● 23 ● 185 ● 243

answered Sep 16, 2008 at 6:17

Tawheed

41 ● 3

0

votes

I recently started using the concepts, standards and templates defined by the International Institute of Business Analysts organization (IIBA).

They have a pretty good BOK (Book of Knowledge) that can be downloaded from their website. They do also have a certificate.

Share

answered Sep 16, 2008 at 10:51

whiz

1,073 ● 2 ● 11 ● 19

0

votes

Requirements Engineering is a bit of an art, there are lots of different ways to go about it, you really have to tailor it to your project and the stakeholders involved. A good place to start is with Requirements Engineering by Karl Wiegers:

http://www.amazon.com/Software-Requirements-Second-Pro-Best-Practices/dp/0735618798/ref=pd_bbs_sr_2?ie=UTF8&s=books&qid=1234910330&sr=8-2

and a requirements engineering process which may consist of a number of steps e.g.:

- Elicitation - for the basis for discussion with the business
- Analysis and Description - a technical description for the purpose of the developers
- Elaboration, Clarification, Verification and Negotiation - further refinement of the requirements

Also, there are a number of ways of documenting the requirements (Use Cases, Prototypes, Specifications, Modelling Languages). Each have their advantages and disadvantages. For example prototypes are very good for elicitation of ideas from the business and discussion of ideas.

I generally find that writing a set of use cases and including wireframe prototypes works well to identify an initial set of requirements. From that point it's a continual process of working with technical people and business people to further clarify and elaborate on the requirements. Keeping track of what was initially agreed and tracking additional requirements are essential to avoid scope creep. Negotiation plays a bit part here also between the various parties as per the Broken Iron Triangle (http://www.ambysoft.com/essays/brokenTriangle.html).

Share

answered May 27, 2009 at 8:01

Jonathan Holloway

63.6k ● 33 ● 126 ● 150

0

votes

IMO the most important first step is to set up a dictornary of domain-specific words. When your client says "order", what does he mean? Something he receives from his customers or something he sends to his suppliers? Or maybe both?

Find the keywords in the stakeholders' business, and let them explain those words until you comprehend their meaning in the process. Without that, you will have a hard time trying to understand the requirements.

Share

answered May 27, 2009 at 8:16

Erich Kitzmueller

37k ● 5 ● 84 ● 103

0

votes

i wrote a blog article about the approach i use:

http://pm4web.blogspot.com/2008/10/needs-analysis-for-business-websites.html

basically: questions to ask your client before building their website.

i should add this questionnaire sheet is only geared towards basic website builds - like a business web presence. totally different story if you are talking about web-

based software. although some of it is still relavant (e.g. questions relating to look and feel).

- LM

Share

edited May 27, 2009 at 12:05

answered May 27, 2009 at 7:46

louism

1,639 ● 12 ● 31

0

votes

I prefer to keep my requirements gathering process as simple, direct and thorough as possible. You can download a sample document that I use as a template for my projects at this blog posting: http://allthingscs.blogspot.com/2011/03/documenting-software-architectural.html

Share

answered Mar 25, 2011 at 22:46

allthingscs

297 ● 2 ● 5