How do you manage customers with regards to changing requirements? [closed]

Asked 15 years, 11 months ago Modified 12 years, 5 months ago

Viewed 6k times

8

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 5 years ago.

Improve this question

Steve Yegge's wisdom notwithstanding, most developers are faced with requirements which were gathered from non-technical customers. Sometimes there are project managers who deal with the customers and translate their requirements, other times not. In any event, the fact that the requirements will change is an inevitability.

Most of what consititues "good programming practice" has to do with developing systems which are adaptable so that they can withstand changing requirements.

Principles like YAGNI, DRY, loose coupling, etc. contribute to this. Iterative development processes such as Agile also attempt to address the concern of trying to hit a moving target, and of course having a system under test makes it infinitely more feasible to make changes.

Nonetheless, it seems that for many of us changing requirements can not only hurt the quality of our software, but can also drain our motivation and make us want to stab someone.

This question is about how to **manage the customer** to make it possible for them to change their requirements in the ways that they need, while discouraging arbitrary or frivolous changes. How do you do it?

- Do you have project managers to insulate the devs from the customer?
- Do you have a formal change management process? Change managers?
- How difficult is it for the customer to get a change when they really need it?
- Conversely, how easy is it for a customer to get a change when it's "frivolous"?
- How much detail do you give the customer when explaining the cost of a change?
- How quickly are you able to give the customer this information after receiving a request for change?

- What factors can torpedo the process (e.g. PM's who can't say no to the customer?)
- **What works for you?**

project-management requirements change-management

Share

Improve this question

Follow

edited May 23, 2017 at 12:08

Community Bot

1 • 1

asked Jan 26, 2009 at 13:10

Adam Bellaire

110k • 19 • 152 • 165

7 Answers

Sorted by: Highest score (default)

9

If you are looking for the ideal world where the customer never changes his mind or you get the ideal spec - **you are in the wrong business**. That being said, the most effective mechanism I have found for managing customer expectations and change requests is to institute an accurate system of measurement.

This is how I run my team:

1) We start with user stories. The customer is involved in writing them and the development team estimates how long each user story will take in a relative manner.

2) Using prior experience I take these relative estimates (story points) and create a rough schedule for when major milestones of the project will be complete.

3) Within these milestones we run 2-week iterations. The customer is involved in setting the approval criteria and whether or not the story has been approved. A simple burn down chart shows the customer how close we are to meeting the launch goal.

4) Often time during the approval sessions the customer will request a change because the feature did not turn out how he expected it (even though it met his original approval criteria). At this time you generate a new story with a new estimate. You can also adjust your milestone dates appropriately. This then puts the ball back into the customers court:

- Often times they realize their change request isn't worth it (they'd have to get approval from their boss) and we'll kill the new feature
- Sometimes it is important so we'll delay the due date to get the feature in
- And finally, there's always the option to kill another not so important feature that will take an equivalent amount of time.

The key is not to run away from change requests, but to establish that every change request has consequences on the product. There's no such thing as a *free lunch.*

Share Improve this answer Follow

edited Feb 18, 2009 at 14:02

answered Jan 26, 2009 at 13:36

Jim

3,170 ● 4 ● 32 ● 38

Speaking for me personally, I am guilty of committing the opposite evil: In the past I have been too permissive with new customer requirements. (My mantra was "No problem!") I'm trying to change that now, hence this question. Thanks for your answer! – Adam Bellaire Jan 26, 2009 at 13:54

Most developers who turn program/development managers have the same problem. I certainly did and learned the hard way. By being strict and following the no free lunch policy you will improve the morale of the team and be able to deliver to the business what they want in a dependable fashion. – Jim Jan 27, 2009 at 15:45

3

I work as indpenedent developer and so contact with customers directly. It's normal that most of the time they have no idea what they actually want. So we start slowly and I give them prototype early on to play with and then the changes will be gradualy made. If I think that customers wants "frivolous" change then I tell him that this change doesn't work or is not needed. If it is 5 min of work then I might even do it anyway.

It helps to add later on to the contract some maintenance clause to get money for those small changes that will

come up later on. For bigger changes you just charge by the hour.

Share Improve this answer

Follow

answered Jan 26, 2009 at 13:18

Riho

4,583 ● 3 ● 35 ● 49

3

Managing the customer is hard, and it is something that very easily can go wrong.

I find that as early as possible you need to **gain the trust** of the customer. For me I think you can do this by:

- Ask the customer to appoint a **product manager** - who is clear thinking enough to communicate the requirements he/she wants, and look to build a strong working relationship with him/her.
- **Really try to understand their business** - you don't need to be a domain expert, but you need to know where the customer is coming from.
- Ask pertinent questions about what they want - **don't assume what they ask for (at first) is what they really want**.
- At first **welcome all changes**. This is not the customer being annoying and fickle, it as an opportunity to better understand what the customer really wants. If this costs you time/money, then you may need to accept it as **a loss leader**.

- **Deliver a prototype early**, and incorporate as much customer feedback as practicable.
- Give the customer **a kick arse product**.

Once you have done this, and the customer trusts you, then you will be in a position to start knocking back unreasonable changes, or ask for extra payments/time for things that were previously considered out of scope.

Of course, you won't be able to build this sort of relationship with every customer, some *are* idiots (in this case see if you can have a different product manager appointed) but you should always do as much as *you* can to build an effective working relationship.

Share Improve this answer

Follow

answered Jan 26, 2009 at 14:47

DanSingerman

36.5k ● 13 ● 83 ● 96

1

You can’t expect customers to know what they want at the start, so you must be adaptable. But also you need to stop change for changes sake.

This is for ‘internal’ customers.

I have found that bargaining with the customer is an effective way to go. They can have whatever feature they want if they wait for it, or if they sacrifice some other (yet to be implemented) features. This forces them to think about the value of the change they are asking for in relation to the system as a whole.

Sometimes this works well and a good compromise is reached. Other times the customer throws their toys out of the pram and goes however high they have to get the feature implemented and quality is reduced.

If the customer is paying, it is a different ball game. They need to be made aware that change costs, and that the cost increases as the product nears completion. This means that you have to do a lot of up front analysis about what you will deliver and make sure the spec is agreed upon. So then you can measure what has changed. This may not be the most effective solution for either party but it does keep things cut and dry. So they are not dissatisfied and you don't end up doing loads of work for free.

Share Improve this answer

Follow

answered Jan 26, 2009 at 14:22

Jeremy French

12.1k ● 6 ● 47 ● 72

1

In software engineering, change is just a fact. It will happen. For us, everything comes at a price. We'll make just about any change the clients wants, but there is always a time estimate and a cost associated with it. Do we ever tell the client no -- not normally, but sometimes the changes request comes in at a very high cost. We do draw the line at potential security threats etc. in which case we calmly explain to them that we can't accommodate the request.

How much do we explain to the customer, we explain where the money is going to be allocated, this much for development, this much for analysis etc. We don't explicitly tell them why something costs the way it does. Now I will admit, this does vary a little with some of our clients. Some of them get very detailed billing as to exactly how many hours are spent where. To get the contract we had to agree to it, although this is very rare for us.

We have sales people who can't say no at times, and that can cause problems. We have spent a lot of time working on that, but unfortunately it still crops up. We combat it by explaining how much money they cost us by quoting something without researching what it will take. Transparency is key at all levels. Everyone has to know how their decisions affect the bottom line.

Do we do frivolous changes? Yep. What you have to remember is that when you bill hourly most of the time a 5 minute change is billed at a full hour, so that is quite lucrative. We explain all of this before like we do with any change request so they are aware of it, but it tends to help discourage such behavior unless it is really important. The fact is that we treat all changes the same. We don't assume we know what is considered frivolous to them no matter how absurd we think it might be. We have a formal change process where the customer asks for something we write it down and get them to sign off that is what we evaluate it and present a Cost of Work Estimate. They either agree in which case they formally

sign a document letting us know it is ok to start, or they rescind the request. We try to be diligent, but we let them know that it will take a few days for us to get a response to their request.

A coworker of mine gave me the best advice I have ever heard about managing customer relations ships. It's a give and take. To make the customer happy, you have to be willing to help them when they need something, but at the same time, you have to be able to say no. When dealing with people they want you to help them, but they also want you to have a spine and stand up for yourself. It becomes a win-win situation that way.

Share Improve this answer

Follow

answered Jan 26, 2009 at 14:36

kemiller2002

115k ● 28 ● 199 ● 253

1

I'd prefer a term of evolving requirements to "changing requirements". Professor M.M.Lehman (http://www.doc.ic.ac.uk/~mml/ and http://en.wikipedia.org/wiki/Meir_Manny_Lehman) has done a considerable contribution into research on software evolution; his works also suggest that not all types of requirements evolve. One might consider themselves lucky if they happen to work on one of these systems, where requirements stay the same (i.e. math libraries etc).

To the rest of us experience suggests that developers prefer as much information about requirements up front

as possible, whereas customers or end-users value an ability to specify or adjust requirements as late as possible into the development process. The former need the detailed information to help planning and designing the solution, the latter can gain a strategic advantage through changing a requirement late, because it gives customer some room for manoeuvre to respond to the changing environment or information gained on as a result of the earlier stages / iterations of the project. A trade-off between the ability to have a detailed plan and change things largely determines the development process itself (waterfall, agile, spiral etc).

Some practical advice managing the evolution of requirements:

- Build in some room into the initial plan to account for evolving requirements, multiple check points or iterations.
- Either put volatile requirements into the beginning of the project so that some kind of prototyping or feasibility study is likely to clarify them or plan for the change late into the project.
- Monitor that the requirements are still relevant.
- Have an up to date, prioritised list of current requirements handy. Nothing else helps to keep evolution in control as a good visibility to all stakeholders of current “must haves” including their relative priority and cost.

- Keep managing customer expectations on how long things are going to take; this also helps to keep focus.
- Introduce a formal process for changing or adding requirements if you need. The process description needs to specify roles of these involved, frequency of reviews etc. It could serve as a good safeguard against some political and most opportunistic but not essential requirements.
- Build in some time for refactoring even for the first version. You're very likely to throw all or part of the solution as a result of additional knowledge gain during the development.

Share Improve this answer Follow

edited Jan 27, 2009 at 19:36

answered Jan 27, 2009 at 16:14

Vlad Gudim

23.5k ● 16 ● 71 ● 92

0

The customer comes to you to do something because they either don't have the time to do it or they don't know what to do **(and want to pay you to do it for them)**. If you have changing requirements, it's because of the latter. In other words, they are paying you to figure out the details! And they know what they like and don't like but they won't know how it works.

Recognize this and whatever the solution needs to be falls in to place.

Share Improve this answer

Follow

edited Jul 3, 2012 at 14:53

user142162

answered Jan 27, 2009 at 19:40

MSN

54.5k ● 7 ● 78 ● 107