Managing feature creep in GUIs [closed]

Asked 16 years, 3 months ago Modified 3 years ago Viewed 2k times

13

Does anyone have any practical suggestions about how to manage feature creep in GUIs?

I'm getting strong pressure from both internal and external sources to add, modify, tweak, etc. I always cringe when someone approaches me with the words "wouldn't it be nice if...?". I can't just turn around and yell "NO" at them, because often they are my superiors or customers.

Instead, I'm looking for suggestions to help explain why it's a bad idea to be constantly adding new features, and in doing so, manage their expectations of the final product.

project-management

Share

Improve this question

Follow

edited Dec 7, 2011 at 15:43

community wiki
5 revs, 4 users 62%
TK.

This question is off-topic because it is not within the scope of questions appropriate for this site, as defined in What topics can I ask about here? Please also see: What types of questions should I avoid asking? You may be able to get help on another Stack Exchange site. – Makyen ♦ Aug 18, 2017 at 23:25

I'm voting to close this question as off-topic because it's about process and not programming – Charlie Jun 27, 2019 at 16:30

28 Answers

Sorted by: Highest score (default)

15

Have feature requests handled in a formal process, normally through the project manager and whoever analyzed the requirements originally. Its always better to palm those sorts of decisions off to someone that isn't the developer, assuming that whoever is going to do that job is actually capable of it.

If you're freelance then obviously charge for changes to the requirements, and if you're an internal development team, then you could consider inter-department billing to make sure people think about what they want to spend money on.

Finally, **expect** requirements to change and feature creep to happen. If you code without considering what changes might be requested, or your process and/or deadlines are so inflexible that you can't adjust to this, then you'll find that the project will become a nightmare.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:10

community wiki
Marc Gear

What if there are no other members of the team who aren't part of the developers? How should the developer handle such thing? – Randell Aug 17, 2009 at 7:52

14

What I do is keep feature ideas on index cards and post the cards somewhere visible. When someone asks, "Could it also do XXX?" I write a new card. This is a better relationship building move than screaming "NO!" :-) It also has the advantage of not losing potentially good ideas. OTOH, I'm under no compulsion to implement it right then. The suggester knows they've been listened to, I know I won't forget, I can get back to work, and we can all get together to make priority decisions at a better time than when my brain is in CodeLand.

Share Improve this answer Follow

answered Sep 17, 2008 at 14:05

community wiki
Kent Beck

10

All right then, I'll be the voice of agile here. The problem can't be solved at the end of that process, it has to be avoided by managing the project differently.

Aside from a specific methodology, the trick is to put those decisions into the hands of the customers. You have a list of things to be done. When they want to change that list, you ask them which item from the list won't be getting done to accommodate the new item. Or, how much more money they will be giving you to handle it.

Also, you have to do the work in small iterations (a week to a month) so they have chances to readjust in between.

We use SCRUM and its been great. After a couple of iterations all the business-level and process-level items get worked out and you're delivering exactly what they want by the end.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:24

community wiki
Brian

I would concur with the use of Scrum. If you can get the organization to buy into the methodology, its a great way to control feature creep and make explicit decisions on what features are of the highest priority. – Steve Wranovsky Sep 17, 2008 at 14:08

4

Ideally requests like this should be handled by the person in charge of the Functional Design. Wether you like it or not, changes will happen (from the first letter in the functional design to the last byte of code and beyond) and there will always be requests for extra features. So make sure your design is up for such a dynamic process.

This will probably sound like a very lame solution (and I doubt it's good practice), but I have been struggling with the same issues in the past. The fact that it happened in a very small company (lack of 'layers' in management) made it worse, since I was in charge of development, functional design, technical design and managing my own projects.

What worked for me is to deflect the problem back to the person asking (wether it was a superior or the customer). Hand over the functional design, a prototype printout or whatever describes the current situation and ask them to figure out 'how' and 'where' this mighty new feature should be implemented.

Both the superiors and the customers were then 'forced' to take it back to their own people, discuss it in meetings and whatnot. Usually this means that you don't hear from it a second time. In the cases where it did come around, it was actually a concept that worked.

answered Sep 17, 2008 at 13:12

community wiki
Twan

+1 for giving it back for them to review what they're asking for – Randell Aug 17, 2009 at 8:09

2

Your company appears not to be defining requirements clearly before starting a project, and this will only end in tears.

My policy is to get a clear breakdown of all requirements in advance and have all parties know the implications of intruding these requirements.

1. Progressively Delayed Release times
2. Increased Bugs
3. Incomplete features
4. Staff stress
5. Staff resignations.
6. Extra charges for expecting more out of the final product than was agreed upon at the declaration of a price (and this is REALLY BAD)

If they don't want to adhere to a system that is sustainable and productive, one might want to opt for #5, or threaten with #5.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:13

community wiki
Kent Fredric

2

For managers:

- the sooner the product is released to the market (assuming it's shrink-wrap), the sooner the company can make money and the better the cashflow.

- don't rule out the new feature outright, but balance it against the value you can derive from doing alternative work; explain the opportunity cost.

For everyone:

- if the new features are in-your-face in the UI, start talking about the effect of visual complexity on the usability - and from that, attractiveness - of the product as whole. But I'm sure you're already doing that. I'll try digging out some references...

Share Improve this answer Follow

answered Sep 17, 2008 at 13:14

community wiki
Mal Ross

2

The best way IMHO is to clearly outline exactly what the cost of implementing the new features will be. "It would be nice if" really starts to dwindle when the user starts to see the cost of such additions.

Disagreeing with the customer about a feature usually gets you nowhere. If you blatently say NO to them they will feel alienated and out of touch with you and your team. The feature probably is a good idea overall, granted you have all the time and money in the world and no technical limitations. In their world being able to see a *fiz* next to a *bar* after they click on a *snip* is a good idea. Of course in our world it means a full table scan, potential security vulnerability, and an all nighter to make sure it's in by the next point release.

If you lay it out for them and explain why it is not a really good idea overall they will usually understand. Don't forget all of the different factors (time/money/cost of adding complexity to the project/risk of slipping deadlines). A reasonable person will understand if you paint the picture clear enough, and you can at least say "I told you so" to an unreasonable person.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:16

community wiki
Jon-Erik

Would be awesome to know how to actually lay it out to them – Randell Aug 17, 2009 at 8:13

1

You cannot handle just feature creep - you need to organise your whole development process in an proper way.

However from your description it seems that you just code what other people ask for and could not re-organise the process. In this scenario your best way to manage the requests effectively by havign a tracking/ticketing system which would allow you to receive requests from other people, prioritise them, estimate them, agree the implementation schedule and track the time you actually spend working on them.

When you will be able to prove with the real-world figures that 'this small button' would take 2-3 days instead of 5 seconds the customer probably believe it should be you will be in much better position to negotiate.

If you will be able to clearly show that the project go-live date will be delayed by two weeks because of the new features you might see those requests simply vanishing.

You have to remember however that 'feature creep' is not always a negative thing. As application matures and grows your customers priorities are changing as well. Failing to acknowledge that could mean that your finished product will not be what they want. Try checking if they would accept trading a new feature for an old one from original specification which is not yet implemeted.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:16

community wiki
Ilya Kochetov

1

I keep a prioritized list of work tasks and my estimates on what will be in build X and how long (roughly speaking) I expect it to take to write tests, implement code and do whatever else is related. I always take their inputs, discuss what they really want/need and insist that we determine where it fits in the grand scheme of things. We talk about the impact to schedule and other tasks.

It keeps the communication line open and clear - there aren't any surprises and the expectations are managed. In the end, it isn't my program - it is the customer's (whoever the customer is) and I want to build them what they want (and need) built.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:19

community wiki
itsmatt

1

The key seems to be in the question.

'**Managing** feature creep'... you do this by implementing a management process that needs to be followed. You can't avoid it (after all, it's frequently the customers requesting it and shouting no at them all the time tends to drive the poor creatures away)... but that doesn't mean it has to be undisciplined. With a procedure in place that entails the person placing the request to give simple things like justification and a preliminary investigation/use-case for the change you start to reduce the number of 'wouldnt it be nice to'. Once you have this in place, your feature creep is managed and you can start prioritising and providing more consistent feedback.

Share Improve this answer Follow

answered Sep 17, 2008 at 14:30

community wiki
workmad3

1

Your users have lots of needs that aren't taken care of. They are suffering. They need attention, and they need *you*. I think feature creep is something that happens when you don't implement *the right features* already.

- Cultivate a close relationship with your users. Let them know you are always interested in their input. Periodically give them a call and ask how your software is treating them.
- Get to know their work habits, standard practices, how they use your software, and how they use their other software. As that information comes in, collect it.
- When feature requests come in, your users won't really know what they want. You know what they want, though, because you have expertise and you've been listening. So, work with them to clarify the problem they're having, then use your collected knowledge to generalize the problem as best you can. Write a solution that solves *that* problem.

On the other hand, "feature creep" is often the response of a software product to an evolving business. If your customer's business is growing, you're fortunate, because they will spend more money on your work. So relax, they'll pay you! They just need to understand that, the bigger a system gets, the harder it is to change, and sometimes a new small feature necessitates a big rewrite, or a whole new user interface, in order to keep everything working smoothly.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:22

community wiki
easeout

1

You have to be careful to balance a reluctance to avoid feature creep with a tendency to ignore feature requests and feedback.

Every time a user comes to you with feedback, that's an opportunity to improve your product and what you're working on. It may end up that you're adding something interesting to both the user, and your developers; it might actually be fun to work on. And yes, it may be a stupid idea, *as posed* to you. But it's your job to accept the feedback, extract anything positive from it, and shape it into something valuable to your users, the product, your company, and your development team.

That being said, feature creep is a very difficult thing to manage. And how well you manage it depends on your position and who the "creep" is. If you're a mid-to-junior level developer, and the CEO is demanding a feature; well, you're going to be adding that feature. You can try to convince the CEO that it's not a valuable feature, or it won't work, or there are more important things to be working on, or it will negatively impact the schedule. But never do any of that at the time the feature is being requested. All you'll end up with is two people defending their position instead of working together towards a common goal.

Instead, accept the feedback and feature request (or feature demand) at face value immediately. Walk away, think about it openly for a while by yourself. "Could this be valuable?" "Am I missing something in the way the CEO asked for this?" "Is it as hard as I'm making it out to be?" Ask yourself these kinds of questions, and come up with some concrete answers. Then **always** go back to the CEO with follow-up questions. Demonstrate that you've thought about the feature requested, and have actually come up with some ideas, tweaks, enhancements, or objections, etc. This will create an open discussion. One that the CEO hadn't anticipated, but that he most likely would not object to since it was not outright resistance to his idea initially.

Share Improve this answer Follow

answered Sep 17, 2008 at 17:49

community wiki
Mike King

1

One of our financial backers requests features all the time. Sometimes he says can we get the software to do 'x'. If it is possible we tell him yes, and then ask him what timescales he had in mind. If he comes back with ASAP - then we tell him that some other feature will have to give, or extend our deadlines. Thankfully he then normally changes his opinion to sometime in the future.

I think the most important thing is to actually record the idea or request, even if the feature doesnt get implemented straight away.

We use Bugzilla to keep a track of bugs - but also Feature Requests. We have a 'features' worklist (or target version)... that way everyone can see what features we would like to develop in the future and as people have more ideas on a feature they can simple add more to the item in bugzilla.

Every release when we sit down and work out the worklist's for a version we dip our toes into the features list to see if there is anything we can pull in. We do try to pull in a feature when we can and give feedback to people - this shows that features and ideas are not falling on deaf ears.

This feedback in helps people know that we are acknowleding their features requests and we DO get round to implementing them, rather than them just sitting on a list which gets bigger and bigger.

answered Oct 21, 2008 at 14:08

community wiki
autonm

1

1)Increased time before release.

2)Increased cost.

3)Exponential maintenance cost

4)Increased potential for bugs

In order to manage a feature request, ask them to submit a change order. Periodically, review the change orders and send back a statement about each request, "This will take X long to do, implying this Y additional cost. Is this acceptable?" Once the requester has accepted the additional cost, then that's a-ok. Your hands are washed. :)

answered Oct 21, 2008 at 14:24

community wiki
Paul Nathan

You explain "sure, it's doable, would you like to have an estimate of how much it will push out the project

1

completion date? Also, giving you that estimate will add about a day to the project end, as well."

There's nothing wrong with adding features, so long as the stakeholders understand that there is a cost associated with doing so.

Share Improve this answer Follow

answered Oct 21, 2008 at 14:28

community wiki
Andy Lester

1

Suppose you build a product that has exactly one feature and all 100 of your customers love your product and find it easy to use. Now suppose that you add ten more features to your product that only 10 of your customers will use. Now you will find that 90% of your customers have much more trouble using your product because there are ten times more choices to make, and ten times as many things you could do that won't help you. The good stuff has been lost in the noise.

This is of course a simplification but the reality is that most of your users will only use a small portion of the features of your product.

Read some good books on software design and UI design, and get your manager to read them too. Joel Spolsky's books are a good place to start - www.joelonsoftware.com answered Nov 21, 2008 at 23:05

community wiki
Jon

1

If there's a lesson we can learn from the Internet and Web 2.0 kinds of things, it's that people love customization. That's what iGoogle and hundreds of other sites are all about. If you can build customization in to your GUI, chances are your customers will love you for it.

Also, take a look at how other projects successfully manage feature creep. For example, Google lets users submit feature requests, but also shows a list of features already requested. Users can then vote to request that feature as well. Not that I'm a suck up, but take a look at stackoverflow.uservoice.com. They have a similar policy.

It's critical to listen to your users and get their feedback. Expect them to come up with new ideas that are better than yours. Expect them to come up with ideas that you think are dumb. If enough people want it, and it seems reasonable, give them what they want.

edited Dec 11, 2021 at 11:58

community wiki

3 revs, 2 users 94%
Scottie T

0

Create work mandates that define the problem that needs solving. Your work is constrained by only needing to implement that which is necessary to solve the problem.

Any further refinement of the problem then becomes change control.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:08

community wiki
anonymous

0

We follow wireframes in my office. Any change after signoff has to go through a Change Control procedure.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:08

community wiki
Gulzar Nazim

0

Lock-in feature set for a short time frame (Scrum/iteration/agile). As the user starts seeing things working, the necessity or lack thereof of features will become more apparent.

Also, it is helpful to have a person through which all changes come (in Scrum, a really good Product Owner).

answered Sep 17, 2008 at 13:14

community wiki
oglester

0

Show them how **simple GUIs can be effective.** Examples: Google Chrome, Apple's software. You may also want to show examples of bloated software, like Eclipse, Netbeans, Visual Studio... ok, these are actually all software IDEs, but they all have cluttered interface.

answered Sep 17, 2008 at 13:14

community wiki
Paweł Hajdan

0

The trick is to define the project as a sequence of versions. Your initial design is for version 2.0, but, the intended first release is version 1.0. All new ideas (features) are welcomed but since, due to scheduling, version 1.0 is frozen, the new ideas have to go into version 2.0.

Of course as soon as version 1.0 is released you begin bug fixing and coding for a maintenance release of version 1.01 and so on... Perhaps version 2.0 never actually gets released, but is used as an elusive goal and a parking place for features that are good, but not good enough to delay the release of a working version.

Share Improve this answer Follow

answered Sep 17, 2008 at 13:51

community wiki
Seymour

0

The right question to ask is 'How can I give the developers a **stable** environment, while still **responding only** to high benefit feature requests.' A SCRUM like approach would be:

Stable environment:

Have the developers work on a small **fixed** set of features during a small **fixed** iteration interval.

Responding only to high benefit feature requests:

One person maintains a list of prioritized features. New features can **always** be added (Cuts down a lot of politics). However the features selected for the next iteration only are the **high priority** items.

edited Jun 20, 2020 at 9:12

community wiki
jyoung

0

Communication is key. In a relationship with a client, it must be clear to them that when a plan is created with a set of features, that is the set of features. It is only the fault of those interacting with the client who are either misleading the client or are somehow intimidated by the client.

As for developers contributing to feature creep, the key is to find a balance between making decisions on implementation and outright adding new features. Again, communicating with the developer on a regular basis will likely curb an issue here.

answered Sep 17, 2008 at 14:02

community wiki
Tony Pitale

It might not be possible to avoid all feature requests.

0

But try assigning a cost for each feature request. When the next planning meeting or deciding the features for the next release comes around this will help to weed out the unnecessary ones.

Share Improve this answer Follow

answered Sep 17, 2008 at 14:11

community wiki
Thomas Bratt

0

IF you're not the manager or owner of the project, I prescribe the following:

If they want it, do it. Make sure they pay you on payday. I've learned that sometimes the battle to get things to conform to what *you* would like, isn't worth fighting. Enjoy life, after work and plan & code your own personal projects that do things the right way.

Share Improve this answer Follow

answered Sep 17, 2008 at 14:53

community wiki
JasonMichael

0

The answer to your question is broader than just GUIs. Feature/Scope creep will always happen, when someone isn't paying attention to what the contract has stipulated and when there isn't a formal process for handling change requests.

If you lack the ability to implement the formal process or influence its creation, I suggest you get ***all*** feature change requests documented in email, and that you notify your management of the possible consequences in email. This isn't to *get* anyone, but rather to protect yourself from the fallout of the eventual failure.

answered Sep 17, 2008 at 15:02

community wiki
Dan

0

At some point, you have to ship something. Assuming you're going through some sort of a formal test process, as long as the product continues to change, testing is never going to be able to sign off on a working product.

It helps to come up with a timeline describing what features will be released and when. That way the people pushing for the new features have some idea that their requests will be handled. It doesn't mean they're going to be handled right now, but it should provide them some reassurances that the next version will address their concerns.

answered Sep 17, 2008 at 16:25

Follow

community wiki
Scott Marlowe