Reasons not to build your own bug tracking system [closed]

Asked 16 years, 3 months ago Modified 11 years, 11 months ago

Viewed 20k times

54

Closed. This question is off-topic. It is not currently accepting answers.

Want to improve this question? Update the question so it's on-topic for Stack Overflow.

Closed 12 years ago.

Improve this question

Several times now I've been faced with plans from a team that wants to build their own bug tracking system - Not as a product, but as an internal tool.

The arguments I've heard in favous are usually along the lines of :

- Wanting to 'eat our own dog food' in terms of some internally built web framework
- Needing some highly specialised report, or the ability to tweak some feature in some allegedly unique way

- Believing that it isn't difficult to build a bug tracking system

What arguments might you use to support buying an existing bug tracking system? In particular, what features sound easy but turn out hard to implement, or are difficult and important but often overlooked?

bug-tracking

Share
Improve this question
Follow

edited Oct 31, 2011 at 21:45

community wiki
5 revs, 5 users 100%
Matt Sheppard

We built our own. It wasn't that hard. It dramatically improved customer service because it could be integrated with our own custom built CRM as well. We already had a customer database...adding a few extra fields was not hard. Our company was very successful. It was a HUGE time saver and customer service advantage. Not having to use multiple systems was great. With modern technology putting custom forms and reports on the web is trivial. The commercial systems are complex and had steep learning curves..Engineers can easily overthink things and make them complex. Just keep it simple. – David J Mar 17, 2019 at 11:31

35 Answers

Sorted by: Highest score (default)

1 2 Next

81

First, look at these Ohloh metrics:

```
     Trac:  44 KLoC, 10 Person Years,   $577,003
 Bugzilla:  54 KLoC, 13 Person Years,   $714,437
  Redmine: 171 KLoC, 44 Person Years, $2,400,723
   Mantis: 182 KLoC, 47 Person Years, $2,562,978
```

What do we learn from these numbers? We learn that building Yet Another Bug Tracker is a great way to waste resources!

So here are my reasons to build your own internal bug tracking system:

1. You need to neutralize all the bozocoders for a decade or two.
2. You need to flush some money to avoid budget reduction next year.

Otherwise don't.

Share Improve this answer Follow

edited Nov 20, 2012 at 21:19

community wiki
2 revs, 2 users 88%
Constantin

Can't believe Redmine ha so many lines of code.. I thought it's Ruby. – kizzx2 Nov 10, 2010 at 16:45

40

I would want to turn the question around. WHY on earth would you want to build your own?
If you need some extra fields, go with an existing package that can be modified.
Special report? Tap into the database and make it.

Believing that it isn't difficult? Try then. Spec it up, and see the list of features and hours grow. Then after the list is complete, try to find an existing package that can be modified before you implement your own.

In short, don't reinvent the wheel when another one just needs some tweaking to fit.

Share Improve this answer Follow

answered Sep 15, 2008 at 11:05

community wiki
Lars Mæhlum

19

Programmers like to build their own ticket system because, having seen and used dozens of them, they know everything about it. That way they can stay in the comfort zone.

It's like checking out a new restaurant: it might be rewarding, but it carries a risk. Better to order pizza again.

There's also a great fact of decision making buried in there: there are always two reasons to do something: a good one and the right one. We make a decision ("Build our own"), then justify it ("we need full control"). Most people aren't even aware of their true motivation.

To change their minds, you have to attack the **real** reason, not the justification.

Share Improve this answer Follow

edited Jan 1, 2013 at 11:07

community wiki
3 revs, 3 users 75%
MattW.

14

Not Invented Here syndrome!

Build your own bug tracker? Why not build your own mail client, project management tool, etc.

As Omer van Kloeten says elsewhere, pay now or pay later.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

community wiki
4 revs, 2 users 64%
Stu Thompson

1 You ask rhetorically, but this *is* the primary reason I've seen in many places. The bug tracker is in-house precisely because it needs to *integrate* with all the other in-house stuff, often in a single monolithic application with poor development practices. – bignose Apr 28, 2009 at 7:03

2 There are integrated products out there too. And then there is the concept of "middleware". I stand by my answer. – Stu Thompson Apr 28, 2009 at 10:02

12

There is a third option, neither buy nor build. There are piles of good free ones out there. For example:

- Bugzilla
- Trac

Rolling your own bug tracker for any use other than learning is not a good use of time.

Other links:

- *Three free bug-tracking tools*
- *Comparison of issue tracking systems*

Share Improve this answer

Follow

edited Aug 20, 2015 at 4:41

community wiki
4 revs, 2 users 62%
Anthony

Free yes, but IMHO neither bugzilla or trac are good
– Orion Edwards Oct 7, 2008 at 20:11

10

I would just say it's a matter of money - buying a finished product you know is good for you (and sometimes not even buying if it's free) is better than having to go and develop one on your own. It's a simple game of **pay now** vs. **pay later**.

Share Improve this answer

Follow

answered Sep 15, 2008 at 11:03

community wiki
Omer van Kloeten

Because *free software* doesn't exist... – alternative Jul 24, 2011 at 14:40

It's not that it doesn't exist, but usually when you're working at a company that uses free software, you would want to buy a support package, unless you want to start fixing bugs in it yourself (i.e. pay later). – Omer van Kloeten Jul 25, 2011 at 6:07

9

First, against the arguments in favor of building your own:

> Wanting to 'eat our own dog food' in terms of some internally built web framework

That of course raises the question why build your own web framework. Just like there are many worthy free bug trackers out there, there are many worthy frameworks too. I wonder whether your developers have their priorities straight? Who's doing the work that makes your company actual money?

OK, if they must build a framework, let it evolve organically from the process of building the actual software your business uses to make money.

> Needing some highly specialised report, or the ability to tweak some feature in some allegedly unique way

As others have said, grab one of the many fine open source trackers and tweak it.

> Believing that it isn't difficult to build a bug tracking system

Well, I wrote the first version of my BugTracker.NET in just a couple of weeks, starting with no prior C# knowledge. But now, 6 years and a couple thousand hours later, there's still a big list of undone feature requests, so it all depends on what you want a bug tracking system to do. How much email integration, source control integration, permissions, workflow, time tracking, schedule estimation, etc. A bug tracker can be a major, major application.

> What arguments might you use to support buying an existing bug tracking system?

Don't need to buy.Too many good open source ones: Trac, Mantis_Bug_Tracker, my own BugTracker.NET, to name a few.

> In particular, what features sound easy but turn out hard to implement, or are difficult and important but often overlooked?

If you are creating it just for yourselves, then you can take a lot of shortcuts, because you can hard-wire things. If you are building it for lots of different users, in lots of different scenarios, then it's the support for configurability that is hard. Configurable workflow, custom fields, and permissions.

I think two features that a *good* bug tracker must have, that both FogBugz and BugTracker.NET have, are 1) integration of both incoming and outgoing email, so that the entire conversation about a bug lives with the bug and not in a separate email thread, and 2) a utility for turning a screenshot into a bug post with a just a couple of clicks.

Share Improve this answer Follow

edited Jan 1, 2013 at 11:23

community wiki
4 revs, 4 users 73%
Corey Trager

6

The most basic argument for me would be the time loss. I doubt it could be completed in less than a month or two. Why spend the time when there are soooo many good bug tracking systems available? Give me an example of a feature that you have to tweak and is not readily available.

I think a good bug tracking system has to reflect your development process. A very custom development process is inherently bad for a company/team. Most agile practices favor Scrum or these kinds of things, and most bug tracking systems are in line with such suggestions and methods. Don't get too bureaucratic about this.

Share Improve this answer Follow

edited Jan 1, 2013 at 11:28

community wiki
3 revs, 2 users 50%
Slavo

5

A bug tracking system can be a great project to start junior developers on. It's a fairly simple system that you can use to train them in your coding conventions and so forth. Getting junior developers to build such a system is relatively cheap and they can make their mistakes on something a customer will not see.

If it's junk you can just throw it away but you can give them a feeling of there work already being important to the company if it is used. You can't put a cost on a junior developer being able to experience the full life cycle and all the opportunities for knowledge transfer that such a project will bring.

Share Improve this answer Follow

answered Sep 15, 2008 at 11:35

community wiki
Garry Shutler

5

We have done this here. We wrote our first one over 10 years ago. We then upgraded it to use web services, more as a way to learn the technology. The main reason we did this originally was that we wanted a bug tracking system that also produced version history reports and a few other features that we could not find in commercial products.

We are now looking at bug tracking systems again and are seriously considering migrating to Mantis and using Mantis Connect to add additional custom features of our own. The amount of effort in rolling our own system is just too great.

I guess we should also be looking at FogBugz :-)

Share Improve this answer Follow

answered Sep 15, 2008 at 12:14

community wiki
dcraggs

5

Most importantly, where will you submit the bugs for your bug tracker before it's finished?

But seriously. The tools already exist, there's no need to reinvent the wheel. Modifying tracking tools to add certain specific features is one thing (I've modified Trac before)... rewriting one is just silly.

The most important thing you can point out is that if all they want to do is add a couple of specialized reports, it doesn't require a ground-up solution. And besides, the LAST place "your homebrew solution" matters is for internal tools. Who cares what you're using internally if it's getting the job done as you need it?

Share Improve this answer Follow

edited Jan 1, 2013 at 11:29

community wiki
3 revs, 2 users 62%
Loren Segal

4

Being a programmer working on an already critical (or least, important) task, should not let yourself deviate by trying to develop something that is already available in the market (open source or commercial).

You will now try to create a bug tracking system to keep track of the bug tracking system that you use to track bugs in your core development.

First: 1. Choose the platform your bug system would run on (Java, PHP, Windows, Linux etc.) 2. Try finding open source tools that are available (by open source, I mean both commercial and free tools) on the platform you chose 3. Spend minimum time to try to customize to your need. If possible, don't waste time in customising at all

For an enterprise development team, we started using JIRA. We wanted some extra reports, SSO login, etc. JIRA was capable of it, and we could extend it using the already available plugin. Since the code was given part of paid-support, we only spent minimal time on writing the custom plugin for login.

Share Improve this answer Follow

edited Jan 1, 2013 at 11:38

community wiki
2 revs, 2 users 74%
Ram Prasad

1 +1 for "create a bug tracking system to keep track of the bug tracking system that you use to track bugs in your core development." LOL – Dubs Jan 20, 2010 at 21:04

3

Building on what other people have said, rather than just download a free / open source one. How about download it, then modify it entirely for your own needs? I know I've been required to do that in the past. I took an installation of Bugzilla and then modified it to support regression testing and test reporting (this was many years ago).

Don't reinvent the wheel unless you're convinced you can build a rounder wheel.

Share Improve this answer Follow

answered Sep 15, 2008 at 12:17

community wiki
Mark Ingram

2

I'd say one of the biggest stumbling blocks would be agonising over the data model / workflow. I predict this will take a **long** time and involve many arguments about what should happen to a bug under certain circumstances, what really constitutes a bug, etc. Rather than spend months arguing to-and-fro, if you were to just roll out a pre-built system, most people will learn how to use it and make the best of it, no matter what decisions are already fixed. Choose something open-source, and you can always tweak it later if need be - that will be **much** quicker than rolling your own from scratch.

Share Improve this answer Follow

answered Sep 15, 2008 at 11:07

community wiki
Bobby Jack

2

At this point, without a large new direction in bug tracking/ticketing, it would simply be re-inventing the wheel. Which seems to be what everyone else thinks, generally.

Share Improve this answer Follow

answered Sep 15, 2008 at 11:14

community wiki
Tony Pitale

2

Your discussions will start with what consitutes a bug and evolve into what workflow to apply and end up with a massive argument about how to manage software engineering projects. Do you really want that? :-) Nah, thought not - go and buy one!

Share Improve this answer Follow

answered Sep 15, 2008 at 11:20

community wiki
Rikalous

2

Most developers think that they have some unique powers that no one else has and therefore they can create a system that is unique in some way.

99% of them are wrong.

What are the chances that your company has employees in the 1%?

Share Improve this answer Follow

answered Sep 15, 2008 at 13:26

community wiki
LepardUK

2

I have been on both sides of this debate so let me be a little two faced here.

When I was younger, I pushed to build our own bug tracking system. I just highlighted all of the things that the off the shelf stuff couldn't do, and I got management to go for it. Who did they pick to lead the team? Me! It was going to be my first chance to be a team lead and have a voice in everything from design to tools to personnel. I was thrilled. So my recommendation would be to check to the motivations of the people pushing this project.

Now that I'm older and faced with the same question again, I just decided to go with FogBugz. It does 99% of what we need and the costs are basically 0. Plus, Joel will send you personal emails making you feel special. And in the end, isn't that the problem, your developers think this will make them special?

Share Improve this answer Follow

answered Oct 8, 2008 at 11:54

community wiki
Jonathan Beerhalter

What's the 1% you're missing for FogBugz? – Jacob Krall Oct 23, 2008 at 3:05

I didn't mean that as an insult or anything. No product meets 100% of the user's needs. Its just not possible.
– Jonathan Beerhalter Nov 19, 2008 at 13:45

1

Every software developer wants to build their own bug tracking system. It's because we can *obviously* improve on what's already out there since we are domain experts.

It's almost certainly not worth the cost (in terms of developer hours). Just buy JIRA.

If you need extra reports for your bug tracking system, you can add these, even if you have to do it by accessing the underlying database directly.

Share Improve this answer Follow

answered Sep 15, 2008 at 11:46

community wiki
Dan Dyer

1

The quesion is what is your company paying you to do? Is it to write software that only you will use? Obviously not. So the only way you can justify the time and expense to build a bug tracking system is if it costs less than the costs associated with using even a free bug tracking system.

There well may be cases where this makes sense. Do you need to integrate with an existing system? (Time tracking, estimation, requirements, QA, automated testing)? Do you have some unique requirements in your organization related to say SOX Compliance that requires specific data elements that would be difficult to capture?

Are you in an extremely beauracratic environment that leads to significant "down-time" between projects?

If the answer is yes to these types of questions - then by all means the "buy" vs build arguement would say build.

Share Improve this answer Follow

answered Sep 15, 2008 at 11:59

community wiki
fuzzbone

1

If "Needing some highly specialised report, or the ability to tweak some feature in some allegedly unique way", the best and cheapest way to do that is to talk to the developers of existing bug tracking systems. Pay them to put that feature in their application, make it available to the world. Instead of reinventing the wheel, just pay the wheel manufacturers to put in spokes shaped like springs.

Otherwise, if trying to showcase a framework, its all good. Just make sure to put in the relevant disclaimers.

To the people who believe bug tracking system are not difficult to build, follow the waterfall SDLC strictly. Get all the requirements down up front. That will surely help them understand the complexity. These are typically the same people who say that a search engine isn't that difficult to build. Just a text box, a "search" button and a "i'm feeling lucky" button, and the "i'm feeling lucky" button can be done in phase 2.

Share Improve this answer Follow

answered Sep 15, 2008 at 12:46

community wiki
Kinjal Dixit

1

Use some open source software as is. For sure there are bugs, and you will need what is not yet there or is pending a bug fix. It happens all of the time. :)

If you extend/customize an open source version then you must maintain it. Now the application that is suppose to help you with testing **money making applications** will become a burden to support.

Share Improve this answer Follow

answered Sep 15, 2008 at 13:33

community wiki
maestroQC

Why would you have to maintain your extensions to the open source software? Just share it with the upstream developers, and if it's something truly useful, somebody else will find an itch to scratch on it. – Jacob Krall Oct 23, 2008 at 3:07

1

I think the reason people write their own bug tracking systems (in my experience) are,

1. They don't want to pay for a system they see as being relatively easy to build.
2. Programmer ego
3. General dissatisfaction with the experience and solution delivered by existing systems.
4. They sell it as a product :)

To me, the biggest reason why most bug trackers failed was that they did not deliver an optimum user experience and it can be very painful working with a system that you use a LOT, when it is not optimised for usability.

I think the other reason is the same as why almost every one of us (programmers) have built their own custom CMS or CMS framework at sometime (guilty as charged). Just because you can!

Share Improve this answer Follow

answered Oct 30, 2008 at 10:58

community wiki
Sarat

1

I agree with all the reasons NOT to. We tried for some time to use what's out there, and wound up writing our own anyway. Why? Mainly because most of them are too cumbersome to engage anyone but the technical people. We even tried basecamp (which, of course, isn't designed for this and failed in that regard).

We also came up with some unique functionality that worked great with our clients: a "report a bug" button that we scripted into code with one line of javascript. It allows our clients to open a small window, jot info in quickly and submit to the database.

But, it certainly took many hours to code; became a BIG pet project; lots of weekend time.

If you want to check it out: http://www.archerfishonline.com

Would love some feedback.

Share Improve this answer Follow

answered Nov 21, 2008 at 16:26

community wiki
Rich Goidel

Link provided in your answer does not work – GibboK Dec 16, 2015 at 22:46

1

We've done this... a few times. The only reason we built our own is because it was five years ago and there weren't very many good alternatives. but now there are tons of alternatives. The main thing we learned in building our own tool is that you will spend a lot of time working on it. And that is time you could be billing for your time. It makes a lot more sense, as a small business, to pay the monthly fee which you can easily recoup with one or two billable hours, than to spend all that time rolling your own. Sure, you'll have to make some concessions, but you'll be far better off in the long run.

As for us, we decided to make our application available for other developers. Check it out at http://www.myintervals.com

answered Jan 2, 2009 at 23:07

community wiki
jjriv

1

Because Trac exists.

And because you'll have to train new staff on your bespoke software when they'll likely have experience in other systems which you can build on rather than throw away.

edited Jan 1, 2013 at 11:41

community wiki
2 revs, 2 users 80%
Marc Gear

1

Because it's not billable time or even very useful unless you are going to sell it.

There are perfectly good bug tracking systems available, for example, FogBugz.

Share Improve this answer Follow

edited Jan 1, 2013 at 11:42

community wiki
2 revs, 2 users 56%
Peter Mortensen

3 It's worth noting that FogBugz was written to initially be internal only because they didn't want to buy one. Not to say that you're wrong but it's funny how the thing we're trying to avoid here is what actually gave birth to FogBugz – Tom Kidd Oct 7, 2008 at 19:47

1

I worked in a startup for several years where we started with GNATS, an open source tool, and essentially built our own elaborate bug tracking system on top of it. The argument was that we would avoid spending a lot of money on a commercial system, and we would get a bug tracking system exactly fitted to our needs.

Of course, it turned out to be much harder than expected and was a big distraction for the developers - who also had to maintain the bug tracking system in addition to our code. This was one of the contributing factors to the demise of our company.

Share Improve this answer Follow

edited Jan 1, 2013 at 11:44

community wiki

2 revs, 2 users 50%
Rangachari Anand

0

Don't write your own software just so you can "eat your own dog food". You're just creating more work, when you could probably purchase software that does the same thing (and better) for less time and money spent.

Share Improve this answer Follow

answered Oct 23, 2008 at 2:57

community wiki
Chris Pietschmann

0

Tell them, *that's great, the company could do with saving some money for a while and will be happy to contribute the development tools whilst you work on this unpaid sabbatical. Anyone who wishes to take their annual leave instead to work on the project is free to do so.*

Share Improve this answer Follow

answered Mar 20, 2009 at 2:25

community wiki
Andy Dent