Documents for a project? [closed]

Asked 15 years, 10 months ago Modified 13 years, 7 months ago

Viewed 8k times

12

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 10 years ago.

Improve this question

I work for a CMMI level 5 certified company and one thing I hate about is the amount of documents we prepare (As a programmer I already hate documents). We have lots and lots of documents like PID(project initiation doc), Business requirements, System requirements,tech spec, Code review checklist, issue logs, Defect logs, Configuration management plan, Configuration management check list(s), Release documents and lots...

Almost 90% of these docs are just done for the sake of QA audit :) .. What do you think are the most important documents for a project? What documents can be used in the long run by another developer?

Please share your good practices here. I would like to use them for my own projects or the company I am planning to start in the long run.

Thanks

project-management project

Share

Improve this question

Follow

edited Feb 17, 2009 at 18:54

George Stocker

57.9k ● 29 ● 180 ● 238

asked Feb 10, 2009 at 16:42

Shoban

23k ● 8 ● 66 ● 107

See stackoverflow.com/questions/522684/… – S.Lott Feb 11, 2009 at 11:49

7 Answers

Sorted by: Highest score (default)

11

The key document is a good functional spec. There should be ***one and only one*** reference document for a system.

Overdoing documentation proliferates a large number of small requirements and spec documents every time someone changes a system or interface. For a system of any complexity, before long you have your spec

distributed around several hundred assorted word, excel, visio and even powerpoint files. When this happens you lose clarity about what is current or even whether you have located and identified all pertinent documentation.

The BRD-SRD-Tech spec progression is based on an assumption that the business signs off the BRD, a business analyst signs off the SRD against requirements documented in the BRD and the technical specification is signed off against the SRD. This generates a web of sign-offs, multiple documents with redundant information and makes it difficult and clumsy to keep the spec documents up to date.

Because of this, subsequent requirements documentatation tends to take the form of a series of change request and supplemental requirement and spec docs, each with their own sign-off and audit process. You gain CYA and audit trail (or at least the appearance of an audit trail), but you lose clarity. There is now no definitive reference document for the system and it is difficult to establish what is current or relevant to any particular activity. The net result is that your business analysis process gets bogged down in forensic research, which adds overheads and latency to delivery schedules.

A spec document should be built in such a way that there is one definitive reference for any given system or subsystem. The document should be kept up to date and versioned. Get a good technical documentation tool like Framemaker, so your process can scale, and the

document has some structural integrity of the sort lacking on Word.

Share Improve this answer

Follow

edited May 23, 2017 at 11:47

Community Bot

1 ● 1

answered Feb 10, 2009 at 16:47

ConcernedOfTunbridge Wells

66.5k ● 15 ● 148 ● 198

4

For me the only real document I ever use is a spec. The more detail the better. However it doesnt need to be all completed at one time, and it doesnt need to be particularly formal. What is far more useful to me than documents that are checked and signed and double checked and double signed is always being able to get the latest version of a document. And being able to talk to people about what they have written, and get a decision in the case of any ambiguity. this is far more useful to me than anything else.

To sum up: a spec is the only document I have ever found useful, however it pales in comparison to having a project manager who knows the proposed system inside out, and can make sensible decisions based on what they know.

Share Improve this answer

Follow

edited Feb 10, 2009 at 16:59

answered Feb 10, 2009 at 16:48

Jack Ryan
8,472 ● 4 ● 41 ● 76

2

Documentation is like tofu -- most people hate it until they realize that under the right conditions, it can be really good.

The problem is that what you consider documentation is mostly made for documentation's sake. You, as a developer, don't see any immediate value in the documents you produce because you know you can do your job without all the TPS reports which you're required to make.

Unfortunately, I'm going to wager that there's not a lot you can do about in a company where you're being forced to eat raw tofu all the time. You'll probably just have to suck it up and write the docs which your company requires, but you can at least do one thing... you can write documents which at least are useful to *you*, and you can pass them along with your code for others who will maintain it.

Aside from inline documentation, you could set up a wiki to be used by yourself and people on your team. This type of documentation is **searchable**, which is already a big plus to developers, plus it's more of a living document instead of a homework-like paper you had to write. You already post to SO, so just think of your documentation as pooling your knowledge in a more useful place.

Share Improve this answer

Follow

answered Feb 10, 2009 at 17:06

Nik Reiman

40.3k ● 29 ● 107 ● 161

1

> What do you think are the most important documents for a project?

Different people have different needs: for example the documents which the owner needs (e.g. the business contract) aren't the same as the documents which QA needs.

> What documents can be used in the long run by another developer?

IMO the most important document (except for the source code) is the functional specification: because what the software is **supposed** to do (as opposed to, what it **is** doing) is the one thing that can't necessarily be reverse-engineered. See also How does a good developer keep from creating code with a low bus hit factor?

Share Improve this answer

Follow

edited May 23, 2017 at 10:32

Community Bot

1 ● 1

answered Feb 10, 2009 at 16:48

ChrisW

56k ● 14 ● 120 ● 233

User Stories, burndown chart, code

1

Share Improve this answer

Follow

answered Feb 10, 2009 at 16:49

Chris Ballance

34.3k ● 26 ● 105 ● 152

1

I'm a fan of the old 4+1 views:

- Use Case view (a/k/a user stories). There are several forms: proper use cases, forward-looking use cases that aren't as well defined and epics which need to be decomposed.
- Logical view. The "static" view. UML Class diagrams and the like work well here as a design document. This also includes request and response formats for various protocols. Here is where we document the RESTful requests and responses. This includes the REST URI design.
- Process view. The "dynamic" view. UML activity diagrams, sequence diagrams and statecharts and the like for here for design documents. In some cases, simple narratives work well. In other cases, there's a **State** design pattern, and it requires a combination of class diagrams and statecharts to show how the stateful objects interact.

This also includes protocols (e.g. REST). Here is where we define any special processing for the various REST requests.

This also includes an authentication or authorization rules, and any other cross-cutting aspects like security, logging, etc.

- Component view. The pieces we're building for deployment. This includes the stuff we depend on, the structure of the modules and packages, etc. This is often a simple component diagram or a list of components and their dependencies.
- Deployment view. We try to generate this from the code as deployed. Since we're using Python, we use epydoc to create the API documentation. We also use Sphinx to import module documentation into this view of the software.

  This also includes the parameters, settings, and configuration details.

This, however, isn't sufficient.

When projects start, you have to work up to this through a series of sprints.

1. The first sprints build just the use case view.
2. Subsequent sprints build an "architecture" to implement the use cases. The architecture document has 4+1 views, but at a high level of abstraction. It summarizes the structure of the model schemas, the requests and replies, the RESTful processing, other

processing, the expected componentry, etc. It never has a Deployment view. We generally reference operator guide and API documents as the deployment view of an architecture.

3. Then design-and-construction sprints build (and update) detailed 4+1 view documents for various components.
4. Then release sprints build (and update) the deployment views.

Share Improve this answer

Follow

answered Feb 11, 2009 at 12:56

S.Lott

391k ● 82 ● 517 ● 788

1

From the project point of view, the most important documents are those that normally include the word Plan, such as the Project Plan, Configuration Management Plan, Quality Plan, etc.

What you are describing is common in process improvements, and normally responds to two major causes. One is that the system really is overeaching and getting in the way of real work being done. Another is actually answered in your question: it is not that the documents are only done for the sake of audits, and your focus should not just be how usefull is the doc for other developers, but for the project or the company as a whole.

One usually looks at things from it's own perspective, sometimes it's necessary to look at the general picture.

Share Improve this answer

Follow

edited May 26, 2011 at 14:07

Kevin Worthington

457 ● 2 ● 5 ● 17

answered Feb 10, 2009 at 16:50

krusty.ar

4,060 ● 1 ● 27 ● 28