When developing a new system - should the db schema always be discussed with the stakeholders? [closed]

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

Viewed 1k times

2

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

I'm several layers/levels above the people involved in the project I'm about to describe.

The general requirement is for a web based issue management system. The system is a small part of a much larger project.

The lead pm has a tech pm who is supposed to handle this portion of the project. The lead pm asked me if it's

normal for the help information to not be in the context of where the help was requested. The lead pm was providing feedback about the site and wanted modal dialogs and such for error messages and wanted me to take a look. I'm looking at the system and I'm thinking...

- a new app was developed in cold fusion!?!?
- the app has **extremely** poor data validation
- the app data validation page navigates away from the data entry form
- the app help page navigates away from the form
- the db schema was not discussed between the developer and the pm
- the db schema was not discussed because it does not exist
- there is a menu page - i.e. once you go to a page, you have to go back to main menu and then go to the next page you want
- the lead pm does not know what the dbms is...
- there is a tech pm and she does not know what a dbms is...
- the lead pm has wanted to fire the tech pm for a long time, but the tech pm is protected...
- the lead pm suggested that the exact functionality desired exists in several proprietary projects (several of which are open source - bugtracker, bugzilla, etc.), but the tech pm and dev wouldn't listen.

I have two questions?

Do I

- fire the dev?
- fire the tech pm and the person protecting her?
- fire the lead pm?
- download and configure bugtracker/bugzilla for them and then fire all of them?
- download and configure bugtracker/bugzilla for them and then go have a beer to forget my sorrows?

and isn't it SOP for the db schema to be discussed and rigorously thought through very early in the project?

EDIT:

I used to work with a wide variety of clients with disparate levels of technical knowledge (and intelligence). I always discussed the db schema with the stakeholder. If they didn't know what a schema was, I would teach them. If they didn't have the background to understand, I would still discuss the schema with them - even if they didn't realize we were talking about the schema. In most of the projects I've been directly involved in, the data is the most important part of the system. Thoroughly hashing out the schema/domain model has been critical in getting to a good understanding of the system and what things can be done and reported on. I have high regard for the

opinions of the posters on SO. It's interesting to note that my approach is not the usual course.

BTW - the sad thing is that the project uses tax payer funds and the IT portion is a collaboration with a prestigious university... the dev and tech pm are long time employees - they are not inexperienced. I find it especially sad when I know intelligent and hard-working people who are jobless and people like these are employed.

When I was younger, I would report this type of ineptitude up the chain and expect appropriate action. Now that I'm up the chain, I find myself not wanting to micro-manage other people's responsibilities.

My resolution was to have two beers and get back to my responsibilities...

database database-design project-management database-administration

edited Dec 30, 2008 at 0:34

asked Dec 29, 2008 at 22:17

mson
7,824 ● 6 ● 42 ● 70

@mitch - the internets is a place best interfaced anonymously – mson Dec 30, 2008 at 5:06

@mitch - there are a lot of crazies on the internet. i don't need fame, i don't care what other random internet people think of me, i don't need projects, i don't need references, i don't need work - and i'm almost to the point where i don't need additional money. – mson Dec 30, 2008 at 7:33

I wish I had your problems. =(– Erik Forbes Jun 24, 2009 at 6:49

10 Answers

Sorted by: Highest score (default)

5

Okay, the first thing, to answer your question: No NO, a thousand times NO! The users are not people you should be discussing db schemata with; in general, you'd as well discuss calculus with a cow. Even if they have the technical background, what about the next time the requirements change; should they be involved in the schema update?

More generally, this sounds like a case where the technical leads let the problem get out of touch with the "customers" or stakeholders. If you're being asked to actually *fix* the problem, I'd suggest you need to build a GUI prototype of some sort, maybe even just a storyboard, and walk through *that*. then you'll have an idea where things stand.

Extended: yes, it WOULD be normal to discuss the DB schema within the project. I'd say you do need to think

seriously about some, um, major counseling with the leads.

Extended more: I understand your point, but the thing is that the database schema is an implementation detail. We're so used to databases we let ourselves lose track of that, and end up with applications that, well, look like databases. But the database isn't what delivers customer value; it's whether the customer can do the things they want. If you tie the ways the customer sees the application to the DB schemata, then you tie them to one implementation; a change, such as denormalizing a table in order to make a more efficient system, becomes something you have to show the customer. Better to show them the observables, and keep these details to ourselves.

But I suspect we're having a terminology clash, too. I would have agreed with you on "domain model." If, by db schema, you mean only those tables and relations visible in the user's view of the system, the "use cases" if you will, then we'd be agreeing.

edited Dec 30, 2008 at 4:19

answered Dec 29, 2008 at 22:29

Charlie Martin
112k ● 26 ● 196 ● 264

Bang on. Well done, sir. – p.campbell Dec 29, 2008 at 23:50

You are absolutely right Charlie. I use db schema/domain model interchangeably in my mind. I know they are technically different things, but to me they have very similar intents - understanding the domain. So right on man - we are on the same page - you are just more precise in your language... – mson Dec 30, 2008 at 7:38

3

The DATA should be discussed with the stakeholders, absolutely yes. The DB SCHEMA should NOT be discussed with the stakeholders except under special circumstances, where the stakeholders are all "database savvy".

So how can you discuss the DATA without discussing the DB Schema? This is the primary use that I've found for Entity-Relationship (ER) diagrams, and the ER model in general. A lot of database designers tend to treat ER as a watered down version of relational data modeling (RDM). In my experience, it can be used much more profitably if you don't think of it as watered down RDM.

What is one difference between ER and RDM? In RDM, a many to many relationship requires a junction box in the middle. This junction box holds foreign keys that link the junction box to the participants in the many to many relationship.

In ER, when applied strictly, junction boxes are unnecessary in many to many relationships. You just indicate the relationship as a line, and indicate the possibility of "many" at both ends of the line. In fact, ER

diagrams don't need foreign keys at all. The concept of linkage by means of foreign keys can be left out of the discussion with most users.

Data normalization is utterly irrelevant to ER diagramming. A well built ER diagram will have very little harmful redundancy in it, but that's largely serendipity and not the result of careful planning.

The "entities" and "relationships" in a stakeholder oriented ER diagram should only include entities that are understood by the subject matter experts, and not include entities or relationships that are added in the course of logical database design.

Values to be held in a database and served up on demand can be connected to attributes, and attributes can in turn be connected to either entities or relationships among entities. In addition, attributes can be tied to domains, the set of possible values that each attribute can take on. Some values stored in databases, like foreign keys, should be left out of discussions with most stakeholders.

Stakeholders who understand the data generally have an intuitive grasp of these concepts, although the terms "entity", "relationship", "attribute", and "domain", may be unfamiliar to them. Stakeholders who do not understand the subject matter data require special treatment.

The beauty of ER models and diagrams is that they can be used to talk about data not only in databases, but also

as the data appears in forms that users can see. If you have any stakeholders that don't understand forms and form fill out, my suggestion is that you try to keep them away from computers, if that's still possible.

It's possible to turn a well built ER diagram into a moderately well built relational schema by a fairly mechanical process. A more creative design process might result in a "better" schema that's logically equivalent. A few technical stakeholders need to understand the relational schema and not merely the ER diagram. Don't show the relational schema to people who don't need to know it.

Share Improve this answer

Follow

answered Dec 30, 2008 at 16:37

Walter Mitty

18.9k ● 2 ● 31 ● 59

2

Well, first you probably should review very carefully the relationship between the tech pm and her sponsor. I'm surprised you say the tech pm is protected when you later imply you can fire the protector. Either she is, or she is not protected. If you can fire the protector, then she is NOT protected.

So it sounds like no-one is protected, and worse - NO-ONE is communicating. I'd recommend the following: call a meeting with the lead pm, the tech pm and the dev. Once together, ask each in turn: "without referencing anything except YOUR work (i.e. you can't blame anyone

else for the duration of this exercise), tell me in 5 minutes or less why I should NOT fire you today".

I realize this is extreme advice, but you have described a HORRIBLE solution to a classic problem. Every aspect of this project and the resulting "code" sounds like a disaster. You probably should have had a greater hand in the oversight of this mess, but you didn't (for whatever reason). I realize that you should expect hired professionals at the PM level to do better than this.

Hence my recommendation for a SEVERE shake-up of the team. Once you put the fear of unemployment one the table (and I'd tell them that you are writing up the failure to communicate for each one), then REQUIRE them to post plans for immediate communication improvement PLUS detailed timelines for fixing the mess by the end of the week.

Then get off your own bum because you're now the LEAD-lead PM on this project.

If they shape up and pull off a comeback on this disaster, then slowly start increasing their responsibilities again. If not... there's always a door.

Cheers,

-R

answered Dec 29, 2008 at 22:51

Huntrods

> the lead pm suggested that the exact functionality desired exists in several proprietary projects (several of which are open source - bugtracker, bugzilla, etc.), but the tech pm and dev wouldn't listen.

If this is true, tell the lead pm to be more assertive; then tell him/her to install bugzilla and be done with it. If the tech pm and dev weren't listening because of stubbornness, they need a little chat...

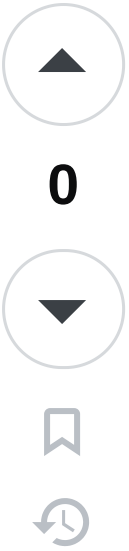
Either way, I'd say you have a problem with your organization... How many thousands of dollars were lost because of a case "not developed here"? However, given that it reached the point of implementation, there are problems further upstream than the development level...

As far as discussing the db schema with everybody, I'd say no. Everyone who can positively contribute should be involved after the application requirements have been gathered.

answered Dec 29, 2008 at 22:39

Austin Salonen

50.2k ● 15 ● 111 ● 140



Wow, sounds like a disaster. Let me address your points in rough order:

1. First, people develop in languages they find comfortable. If someone is still comfortable in an older environment when much better alternatives exist, it is a sure sign that they have little appetite for skill acquisition.
2. Data validation prevents people from going too far down a path only to find it is a blind alley. Lack of validation means the developer isn't thinking about the user. Also, it is *not* something tacked on at the end...it simply doesn't work that way.
3. Web "dialogs" cannot be "modal" in the sense you are thinking. However, it is easy enough to pop up an additional window. Help on a page should almost always use a pop up window of this sort.
4. Data validation should NEVER navigate away from the page where data is entered - this is horrible UI design.
5. The DB schema is kind of the least of your problems. If the developer is responsible for delivering functionality and is clearly competent in data schema design, I wouldn't think it critical to discuss the nuances of the schema with the lead PM. It *should* be discussed among various code-level stakeholders and it must be capable of handling the requirements of the work. However, the important thing from the PM's perspective isn't the schema so much as the

operational aspects. Of course, if you have no faith in the developer's ability to construct a good db schema, all bets are off.

6. If you seriously don't know what the dbms is, you may have a serious problem. Do you have a standard? If everyone else in the extended project is using MS SQL Server and this guy chose Oracle, how do you transfer expertise and staff into or out of this project? This is a sign of an organization out of control.
7. There are two reasons for ignoring alternative proprietary products. First, they may not truly meet your needs. Second, the tech PM and developer may simply be featherbedding or engaging in some nasty 'not invented here' justification for wasting your resources. The problem is that you aren't likely to have enough insight, at your level, to know the difference between the two.
8. With respect to firing the Dev...is it possible to help him by sponsoring some additional training? If this person is otherwise a good employee and knows your business well, I'd be very hesitant to fire them when all that is needed is a push in the right direction.
9. The tech PM sounds like she really isn't doing her job. She is the logical person to point out the flaws I am writing about and pushing for improvement. The real question, vis a vis her position, is whether she

can learn to be a better advocate for your organizational interests.

10. The lead PM sounds too passive as well. Comments made above regarding the tech PM apply here as well.

11. If bugtracker, etc. really work then it would make sense to go that route. However, you might want to be a bit more circumspect about firing people.

Share Improve this answer

Follow

answered Dec 29, 2008 at 22:41

Mark Brittingham
28.9k ● 12 ● 82 ● 111

0

First off, I agree with Charlie Martin about the db schema.

Second,

It sounds like the developer on the project is very green - is this his/her first programming job? If so, I would only fire the dev if their resume says something else.

I don't know how involved the lead/tech pms are expected to be in a project, but it sounds like the responsibility is dev > tech pm > lead pm. If that is the case, then the tech pm completely dropped the ball. You may want to find out why the ball was dropped and fire/keep her based on that, but a botched job like that is reprimand time where I work.

Finally, imho, the "protection" stuff is b.s. - you need to reward and reprimand people based on their quality and value, not who their aunt is.

Good luck! Cheers!

Share Improve this answer

Follow

answered Dec 29, 2008 at 22:55

willoller

7,312 ● 1 ● 37 ● 63

0

Wow. I feel your pain.

Looks to me as if the first source of your problem is the techPM who is "protected". Why is she protected and by whom? I once was on a project where the ceo's secretary became first the business analyst and then (after he quit) the project manager because they were having an affair. She didn't know what language we programmed in and thought requirements were a waste of time. Since she was protected by someone as high as possible in the organization, the only real solution was to look elsewhere for employment.

You seem to think you can fire her and her protector so it may be someone lower than you but above the lead PM so he couldn't do anything about it but you can? Yes, you should fire the two of them.

The lead PM may or may not be salvageable depending on who the protector was. He could have been between a rock and a hard place where he knew what to do but due

to the nature of the relationship between the tech pm and her protector was unable to exert any influence over her and the people who reported to her. I was in that position once where two of my bosses were having an affair with one of my subordinates and it creates all kinds of organizational havoc (which is why the protector must be fired as well as the tech PM). Give him the benefit of the doubt and discuss with him how he would would handle things differently if the tech pm and her protector were out of the way. If you like what you hear, you can keep him but organizationally you will need to step in and make sure that is it clear this person is in charge and no one will be allowed to ignore him. Once a lead has lost authority, he can only get it back with the strong backing from management.

I would also sit down with the lead and the developer and explain exactly what is unacceptable in the project as it currently stands. If the developer feels unable to take direction from the lead (assuming you decide to keep him) or is unable to adjust to a new way of doing business or cannot understand why the code as it stands is unacceptable, cut your losses and get rid of him as well. A new person is likely to work better for the lead if he is salvageable anyway because he won't have a history of ignoring him.

answered Dec 29, 2008 at 23:13

HLGEM

96.4k ● 15 ● 119 ● 189

I wouldn't necessarily think that the db schema should always be shared with stakeholders. Most people wouldn't know what to do with that sort of information. If you're trying to make sure that the product fits the requirements, the requirements should be clearly laid out up front and verified throughout the development of the project.

If you're having problems with the dev, that's just par for the course. Someone more trust-worthy should have been found. If you hired a poor coder, that was your mistake.

There are a few possible solutions:

- Get a better coder. He'll hate working through all the bad code but hopefully he'll slug through it till it's done. Hopefully you're willing to pay him good money.
- Keep the coder and make him fix it all. Hire a new PM that can manage him better. That coder knows his code best and it might take less time for him to just improve it. In the long run, you're better off not keeping a bad coder on payroll so lose him when you're done.
- Suck it up, buy a beer for everyone involved and start over with opensource. You'll probably still need a tech PM to manage the software. You'll also have to forget about doing anything custom at that point. Perhaps a contractor could manage this.

Either way, you're gonna lose some money. Should probably keep a closer eye on things next time.

Share Improve this answer

Follow

answered Dec 29, 2008 at 23:33

thrashr888

1,477 ● 1 ● 17 ● 24

0

I tend to think of it this way. The database schema is there to support the application's data storage requirements. What data the application needs to store will be determined by the end user's requirements. If you're not consulting your end user as to their requirements for the application you're obviously headed for trouble, but provided you have a good handle on their requirements (and likely future requirements) then database schema is a technical decision which can be made by the project team without direct input from the end user/client.

An end user is unlikely to understand the intricacies of tables, fields, normalization, etc, but they'll understand "the system needs to do xyz". Talk to the end users in a language they understand, and let your team make the appropriate technical decisions.

Share Improve this answer

Follow

answered Dec 29, 2008 at 23:45

Jim OHalloran

5,908 ● 2 ● 40 ● 59




My big question is about the relationship between the lead pm and the tech pm's protector: did the lead pm have good reason to fear retaliation from the protector? It's entirely possible that he felt unable to do anything until the situation got bad enough that it was clearly important for people above the protector. In that case, he doesn't deserve any more harsh treatment.

The tech pm is apparently incompetent at her job, and her protector is more interested in favoring her than getting the work done. That suggests to me that they need to be dealt with in some fashion, at minimum with a talk about the importance of getting real work done, at maximum firing both of them.

The dev is likely hunkered down, trying to survive politically, given the climate you've outlined. I can't tell enough about the dev to give any advice.

Therefore, if your description and my amazing psychic powers have given me a clear and accurate picture:

Shield the lead pm from retaliation, and tell him to ditch all the crud and implement an off-the-shelf solution. (If he can't select one himself reliably, he shouldn't be lead pm.)

Discipline the tech pm and her protector. You really don't want to have people wrecking enterprise productivity that way.

The dev is the lead pm's responsibility. Leave it at that. Don't micromanage more than you have to. Have a

couple of beers. Get back to your usual work.

Share Improve this answer

Follow

answered Dec 30, 2008 at 17:19

David Thornley

57k ● 9 ● 95 ● 158

Your pschic powers are on the money. in regard to the dev - I was able to bypass auth/auth measures by directly navigating to pages and access private data... the site is also vulnerable to sql inj. i think your suggestions are good, but i don't try to develop the chaffe. – mson Dec 30, 2008 at 18:07