Software design period...what do other developers do? [closed]

Asked 15 years, 8 months ago Modified 2 years, 9 months ago

Viewed 522 times

1

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 2 years ago.

Improve this question

I'm a new software architect/lead, coming up with software design for a team of software developers. I'm coming up with the requirement spec, interface header files, and visio software design docs, and build plan, etc.

My question is: what do the rest of the team do during this period? I'm certainly engaging them in the design, but we dont need the whole team actively working on what I'm doing all the time.

Are there any good books for new software architect?

project-planning

Share

Improve this question

Follow

edited Apr 3, 2009 at 19:07

Jon B

51.8k ● 31 ● 136 ● 163

asked Apr 3, 2009 at 19:03

sivabudh

32.6k ● 65 ● 166 ● 231

11 Answers

Sorted by: Highest score (default)

6

Generally the various stages overlap, so there will be some coding during design etc. There are a lot of things to do besides that. They can be reviewing unfamiliar technology that is going to be used, setting up source control system, reviewing business requirements, reviewing your documents to make sure they make sense and are clear. There is a lot of other work to be done besides programming.

Share Improve this answer

Follow

answered Apr 3, 2009 at 19:10

kemiller2002

115k ● 28 ● 199 ● 253

4

What a software team does while the lead does the design is very different from company to company. On my

company we try to work on the design while the developers are finalizing other projects or solving bugs.

Another approach that I've taken when starting a whole new project is to get the developers to work on the design as well - people with a good understanding of the requirements can help you designing smaller parts of the system and writing the specs for them. Others can work on mockups, frameworks. This worked rather well for the small software team I led in a previous job (4 developers in total).

I also found it useful to have other team members research parts I'm unsure of (or even validating that things I think should work will indeed work), such as:

- Investigating whether an external API provides the features we need
- Writing a small proof of concept or technology demonstrator
- Create an API mockup (header file, interface or REST endpoint) to investigate whether the API looks useful.

edited Feb 27, 2022 at 13:38

answered Apr 3, 2009 at 19:11

Guss
32.2k 19 112 140

As other have said, you typically want a ramp-up period during the first part of the project, and through the first iteration. **You're planning on building this iteratively, aren't you?** Start with a core team (nor more than 3-4 people, since you're going to need to communicate heavily with each other) to help you explore the requirements, get a **basic data model** in place, identify and setup any **frameworks**, identify and setup **build** and **test** tools. Some coding activities typically take place in the design phase: for **UI mockups**, **run-ahead prototypes** of technically sensitive areas (whatever risks you have should be mitigated by explirative coding: be they **new technologies**, undocumented interfaces to **integrated systems**, or **unstable requirements**).

But coders in the design phase should help with the design, in order to get their buy-in, and to help train up the rest of the team during the first iterations. Your role during this is to ensure that the major nonfunctional requirements (e.g. are known, prioritized, are met by the design, and can be tested). You should also collaborate with the project lead or whoever else is responsible for staffing and financing in order to sketch out the iterations and the staffing levels needed. Ensure the solution can be built iteratively, and aim at implementing only a basic structure during the first iteration, both to build confidence, and to eliminate risks. (Sometimes, you can push major risks to the second iteration, and focus the first towards confidence and team building.)

And of course, be sure you are not designing every detail. You should be able to use every design artifact in the next iteration (and elaborate them later as needed). Since design decisions are expensive to change, try to postpone them. However, some influence the entire solution (for instance, the data model, or your approach to security) and absolutely must be at least outlined up front. This isn't waterfall. This is just not closing your eyes and hoping a viable architecture will emerge by magic.

But design proceeds throughout the iterations. It's just that you do less of it as you go along, and with lesser impact on the solution (unless you're unlucky... and then things get expensive).

Share Improve this answer

Follow

answered Apr 3, 2009 at 20:29

Pontus Gagge

17.3k ● 1 ● 42 ● 52

1

Stop doing the useless things you do and just start coding with them! ;)

Share Improve this answer

Follow

answered Apr 3, 2009 at 19:13

Andriy Volkov

18.9k ● 9 ● 69 ● 84

This is totally NOT useless. Exaggerating a bit, almost anybody can code. Writing proper specs and designing a

good software architecture, however, is the hardest part of any project... – Varkhan Apr 3, 2009 at 19:17

I just don't know how can you do that w/o falling into the pit of Waterfall – Andriy Volkov Apr 3, 2009 at 19:33

+10, Varkhan. Having 4 years in software development (coding) experience and tried out designing/writing specs for the first time, I realize designing IS harder than coding. However, designing FIRST before coding is definitely the way to go! (and makes development much more fun~)
– sivabudh Apr 3, 2009 at 19:34

Well guys I have like 10+ years in coding, architecture and PM and I still have no idea what I'm talking about :) Rudimentary specs are useful sometimes, when there's plenty of time and enough complexity. Otherwise they just get in the way. Can somebody undo the -2 please :)
– Andriy Volkov Apr 3, 2009 at 19:36

I don't know what kind of projects you work on, but dumping 30 engineers onto a year-long project that hasn't been through an analysis stage is completely irresponsible--an utter waste of time. 2 people on a 3 month project--fine. Somewhere in between there is a transition :) – Bill K Apr 3, 2009 at 19:41

1

If there is no overlap with another ongoing project, getting them involved as you're doing is great, maybe push it a little further by having them prototype and present the plus and minus of alternative technologies (APIs, frameworks, libraries, etc...) that your project could use.

Share Improve this answer

Follow

answered Apr 3, 2009 at 19:14

Lancelot

1

As a new software architect, I can recommend some books that helped me understand the role of the architect (but of course not to master it):

- Fundamentals of Software Architecture An Engineering Approach:
  This book gives good modern overview of software architecture and its many aspects, good place to start if you are a beginner or broaden your knowlage.
- Software Architecture in Practice:
  Explains what software architecture is, why it's important, and how to design, instantiate, analyze, evolve, and manage it in disciplined and effective ways.
- Software Architect's Handbook:
  This book takes you through all the important concepts, right from design principles to different considerations at various stages of your career in software architecture. It begins by covering the fundamentals, benefits, and purpose of software architecture.
- Clean Architecture: A Craftsman's Guide to Software Structure and Design:
  Learn what software architects need to achieve and how to achieve it, master essential software design principles and see how designs and architectures go wrong.

- Software Architecture: The Hard Parts:
  An advanced architecture book, with this book, you'll learn how to think critically about the trade-offs involved with distributed architectures.

Share Improve this answer Follow

edited Feb 24, 2022 at 14:31

answered Feb 24, 2022 at 10:42

Uri Loya
1,351 ● 2 ● 19 ● 39

1 Would be great if you could add some descriptions.
– sivabudh Feb 24, 2022 at 12:16

0

Usually there's another project they can work on, but...

I have my team review the project specs/requirements and put together a basic/preliminary structure to get them already thinking through the application and working out specific questions.

When we convene at the table to discuss the plan they already have an idea of what the project is and requires and in some cases, they present questions I may have missed or overlooked.

Share Improve this answer

Follow

answered Apr 3, 2009 at 19:10

jerebear

6,635 ● 4 ● 33 ● 39

0

Although it's too late now, a good way to approach it is to move the architect over before his current project has ended. Start freeing him up at like 25% then work your way up to 75-100% on the new project a month or two before it starts (maybe more depending on how much analysis and customer interaction there is).

On a trivial project (let's say 2 man-years) it might not be necessary, but anything bigger than that can end up in chaos if somebody doesn't at least get the analysis right before everybody jumps aboard.

Share Improve this answer

Follow

answered Apr 3, 2009 at 19:38

Bill K

0

If your team does not have any other projects to work on, ask experienced programmers of your your team to come up with at prototype so that you can create a requirement doc according to the needs of the client.

Also programmers novice to the technologies being used in the team could utilize this time to familiarize themselves with the technologies on which your team is going to develop the project.

Share Improve this answer Follow

edited Apr 3, 2009 at 19:40

answered Apr 3, 2009 at 19:24

Stack Programmer
3,484 ● 1 ● 21 ● 13

0

architect != designer

Chances are that all of your developers can help with the design; let them. Architects don't have to be "lone wolves" and do everything themselves. You lay out the guidelines and the principles and the scaffolding, rough in the wiring, and let your developers flesh out the details - whether it is drawing Visio diagrams or building prototypes to mitigate unknowns/risks.

Migrate towards Agile/XP and away from waterfall methods, and you'll find the team a lot more help.

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 • 1

answered Apr 3, 2009 at 19:50

Steven A. Lowe

61.1k • 19 • 135 • 204

0

When making the general design, it's very handy to have programmers **create proof-of-concepts**. Do that especially with parts of the system that could end up being show stoppers if they don't work in the way you plan to do them, so you can think of alternatives, and adjust the design.

That's going to help you to make the right design-decisions before moving entirely into a certain direction.

Just doing a design, and then moving on and start coding is a sure way to mess up a project. You won't realize that your design is not feasible (or just plain sucks) until you're half-way coding, and by then it's too late to make radical changes.

You'll waste time mitigating non-existing problems during the design, and you'll run into unforeseen problems during implementation.

Share Improve this answer

Follow

answered Apr 3, 2009 at 20:17

Wouter van Nifterick

24.1k ● 7 ● 81 ● 123