Software Design vs. Software Architecture [closed]

342 votes

246

Could someone explain the difference between Software Design and Software Architecture?

More specifically; if you tell someone to present you the 'design' - what would you expect them to present? Same goes for 'architecture'.

My current understanding is:

- Design: UML diagram/flow chart/simple wireframes (for UI) for a specific module/part of the system
- Architecture: component diagram (showing how the different modules of the system communicates with each other and other systems), what language is to be used, patterns...?

Correct me if I'm wrong. I have referred Wikipedia has articles on http://en.wikipedia.org/wiki/Software_design and

http://en.wikipedia.org/wiki/Software_architecture, but I'm not sure if I have understood them correctly.

architecture definition

edited Nov 30 '18 at 21:00

Yvette Colomb ♦
20.3k 15 71 113

asked Apr 1 '09 at 10:00

Mads Mobæk
18.6k 20 61 72

closed as primarily opinion-based by George Stocker ♦ Dec 19 '13 at 21:22

Many good questions generate some degree of opinion based on expert experience, but answers to this question will tend to be almost entirely based on opinions, rather than facts, references, or specific expertise.

If this question can be reworded to fit the rules in the help center, please edit the question. *

locked by Yvette Colomb ♦ Nov 30 '18 at 21:00

This question exists because it has historical significance, but **it is not considered a good, on-topic question for this site**, so please do not use it as evidence that you can ask similar questions here. This question and its answers are frozen and cannot be changed. More info: help center.

Read more about locked posts here.

41 Answers

1 2 next

329 votes

You're right yes. The architecture of a system is its 'skeleton'. It's the highest level of abstraction of a system. What kind of data storage is present, how do modules interact with each other, what recovery systems are in place. Just like design patterns, there are architectural patterns: MVC, 3-tier layered design, etc.

Software design is about designing the individual modules / components. What are the responsibilities, functions, of module x? Of class Y? What can it do, and what not? What design patterns can be used?

So in short, Software architecture is more about the design of the entire system, while software design emphasizes on module / component / class level.

edited Mar 11 '14 at 19:50

Robert Harvey ♦
149k 33 276 420

answered Apr 1 '09 at 10:19

Razzie
23.8k 10 54 69

116 Also, architecture usually deals with what (is done) and where (it's done), but never with how. That is think is the principle difference - design completes the how that architecture doesn't (and shouldn't) talk about. – Asaf R Dec 29 '09 at 14:01

2 Hi@AsafR ! this made me think of architecture as analysis because analysis deals with what(is done) and design with how.do you think so? – Chriss Sep 21 '13 at 7:51

2 Nowadays people do all the designing, implementing, maintaining the backend servers (probably cloud-

based), and front-end design (Web or mobile) all by themselves. I think they're called full stack developers. Right? – Maziyar Aug 1 '14 at 7:07

1 Architecture is the outline of a system, a structure, a blueprint about the whole thing. Design is just the activity of making a plan. You can design an architecture, you can design a module, you can even design a method. – Evan Hu Jan 10 '15 at 2:53

2 That is because MVC is an architectural design. MVC does not state any details in itself. The "View" can be a website, a winforms, a console application. The model can be almost anything, it does not state anything about where it comes from (database layer or whatever). – Razzie Aug 31 '16 at 6:46

*

80 votes

In some descriptions of the SDLC (Software Development Life Cycle) they are interchangeable, but the consesus is that they are distinct. They are at the same time: different (1) *stages*, (2) *areas of responsibility*, and (3) *levels of decision-making*.

- Architecture is the bigger picture: the choice of frameworks, languages, scope, goals, and high-level methodologies (Rational, waterfall, agile, etc.).
- Design is the smaller picture: the plan for how code will be organized; how the contracts between different parts of the system will look; the ongoing *implementation* of the project's methodologies and goals. Specification are written during this stage.

These two stages will *seem to* blend together for different reasons.

1. Smaller projects often don't have enough scope to separate out planning into these to stages.
2. A project might be a part of a larger project, and hence parts of both stages are already decided. (There are already existing databases, conventions, standards, protocols, frameworks, reusable code, etc.)
3. Newer ways of thinking about the SDLC (see Agile methodologies) somewhat rearrange this traditional approach. Design (architecture to a lesser extent) takes place throughout the SDLC *on purpose*. There are often more iterations where the whole process happens over and over.
4. Software development is complicated and difficult to plan anyway, but clients/managers/salespeople usually make it harder by changing goals and requirements mid-stream. Design and even architectural decisions *must* bemade later in the project whether that is the plan or not.

*

Even if the stages or areas of responsibility blend together and happen all over the place, it is always good to know what level of decision-making is happening. (We could go on forever with this. I'm trying to keep it a summary.) I'll end with: Even if it seems your project has no formal architectural or design stage/AOR/documentaiton, it IS happening whether anyone is consciously doing it or not. If no one decides to do architecture, then a default one happens that is probably poor. Ditto for design. These concepts are almost **more important** if there are no formal stages representing them.

edited Dec 24 '09 at 16:21

answered Dec 24 '09 at 15:39

Patrick Karcher

18.9k 4 45 61

55 votes

Architecture is strategic, while Design is tactical.

Architecture comprises the frameworks, tools, programming paradigms, component-based software engineering standards, high-level principles..

While design is an activity concerned with local constraints, such as design patterns, programming idioms, and refactorings.

edited Apr 18 '12 at 13:53

answered Dec 24 '09 at 15:44

Chris Kannon
4,345 4 21 32

1 Agreed.. it perhaps: Architecture comprises the frameworks, tools, programming paradigms, component-based software engineering standards, high-level principles.. While design is an activity concerned with local constraints, such as design patterns, programming idioms, and refactorings. – Chris Kannon Dec 24 '09 at 18:19 *

38 votes

I found this as I was looking for simple distinction between architecture and design myself;
What do you think of this way of looking at them:

- architecture is "what" we're building;
- design is "how" we're building;

edited Mar 28 '12 at 18:22

Bo Persson
78.6k 17 118 185

answered Mar 28 '12 at 17:36

George S.
423 4 7

4 What we're building are client's requirements. How we're building it depends on both architecture and design. So no, this is completely wrong. – Marek Nov 6 '14 at 19:23

1 @Marek I don't see what's wrong with that. Architecture is what to build, what the client wants, what it should generally look like, what components it should be made of, etc. Design is how these things are then actually made: The actual implementations of components, algorithms, etc. – RecursiveExceptionException Jun 10 '16 at 15:58

21 votes

1. Architecture means the conceptual structure and logical organization of a computer or computer-based system.*

   Design means a plan or drawing produced to show the look and function or workings of a system or an object before it is made.

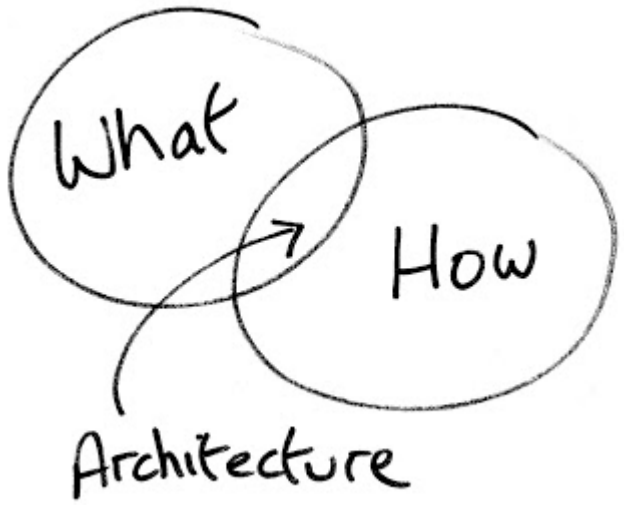
2. If you are “architecting” a component, you are defining how it behaves in the larger system.

   If you are “designing” the same component, you are defining how it behaves internally.

> All architecture is design but NOT all design is architecture.

`What` part is the Design, the `How` is the concrete implementation and the intersection of `What` and `How` is Architecture.

Image for differentiating Architecture and Design:



There are also design decisions, that are not architecturally significant, i.e. does not belongs to the architecture branch of design. For example, some component's internal design decisions, like- choice of algorithm, selection of data structure etc.

Any design decision, which isn't visible outside of its component boundary is a component's internal design and is non-architectural. These are the design decisions a system architect would leave on module designer's discretion or the implementation team as long as their design don't break the architectural constraints imposed by the system level architecture. *

The link that gives good analogy

edited Mar 2 '16 at 14:47

André Cadete
13 5

answered Dec 19 '13 at 4:56

TryinHard
2,881 2 21 44

15 votes

I'd say you are right, in my own words;

Architecture is the allocation of system requirements to system elements. Four statements about an architecture:

1. It can introduce non-functional requirements like language or patterns.
2. It defines the interaction between components, interfaces, timing, etc.
3. It shall not introduce new functionality,
4. It allocates the (designed) functions that the system is intended to perform to elements.

Architecture is an **essential engineering step** when a complexity of the system is subdivided.

Example: Think about your house, you don't need an architect for your kitchen (only one element involved) but the complete building needs some interaction definitions, like doors, and a roof.

Design is a informative representation of the (proposed) implementation of the function. It is intended to elicit feedback and to discuss with stakeholders. It might be good practice but **is not an essential engineering step**.

It would be nice to see the kitchen design see before the kitchen is installed but it is not essential for the cooking requirement:

If I think about it you can state:

- architecture is for a public/engineers on a more detailed abstraction level
- design is intended for public on a less detailed abstraction level

edited Sep 13 '12 at 13:45

andand
11.4k 9 40 71

answered Mar 16 '11 at 10:03

user662182
181 1 3

14 votes

My reminder:

- We can change the Design without asking someone
- If we change the Architecture we need to communicate it to someone (team, client, stakeholder, ...)

answered Nov 23 '12 at 23:31

Peter Gfader
5,779 7 46 53

6 votes

I think we should use the following rule to determine when we talk about Design vs Architecture: If the elements of a software picture you created can be mapped one to one to a programming language syntactical construction, then is Design, if not is Architecture.

So, for example, if you are seeing a class diagram or a sequence diagram, you are able to map a class and their relationships to an Object Oriented Programming language using the Class syntactical construction. This is clearly Design. In addition, this might bring to the table that this discussion has a relation with the programming language you will use to implement a software system. If you use Java, the previous example applies, as Java is an Object Oriented Programming Language. If you come up with a diagram that shows packages and its dependencies, that is Design too. You can map the element (a package in this case) to a Java syntactical construction.

Now, suppose your Java application is divided in modules, and each module is a set of packages (represented as a jar file deployment unit), and you are presented with a diagram containing modules and its dependencies, then, that is Architecture. There isn't a way in Java (at least not until Java 7) to map a module (a set of packages) to a syntactical construction. You might also notice that this diagram represents a step higher in the level of abstraction of your software model. Any diagram above (coarse grained than) a package diagram, represents an Architectural view when developing in the Java programming language. On the other hand, if you are developing in Modula-2, then, a module diagram represents a Design.

(A fragment from http://www.copypasteisforword.com/notes/software-architecture-vs-software-design)

answered Jul 14 '11 at 2:58

Enrique Molinari
209 3 6

*

5 votes

Personally, I like this one:

"The designer is concerned with what happens when a user presses a button, and the architect is concerned with what happens when ten thousand users press a button."

SCEA for Java™ EE Study Guide by Mark Cade and Humphrey Sheil

answered Apr 12 '10 at 17:04

Tavi
64 1 6

5 votes

I agree with many of the explanations; essentially we are recognizing the distinction between the architectural design and the detailed design of the software systems.

While the goal of the designer is to be as precise and concrete in the specifications as it will be necessary for the development; the architect essentially aims at specifying the structure and global behavior of the system just as much as required for the detailed design to begin with.

A good architect will prevent hyper-specifications - the architecture must not be overly specified but just enough, the (architectural) decisions established only for the aspects that present costliest risks to handle, and effectively provide a framework ("commonality") within which the detailed design can be worked upon i.e. variability for local functionality.

Indeed, the architecture process or life-cycle just follows this theme - adequate level of abstraction to outline the structure for the (architecturally) significant business requirements, and leave more details to the design phase for more concrete deliverables.

*

edited Jun 16 '12 at 22:01

answered Jun 16 '12 at 21:54

Ajay Shendye
51 1 2

5 votes

Architecture is design, but not all design is architectural. Therefore, strictly speaking, it would make more sense to try to differentiate between *architectural design* and *non-architectural design*. And what is the difference? It depends! Each software architect may have a different answer (ymmv!). We develop our heuristics to come up with an answer, such as 'class diagrams

are architecture and sequence diagrams are design'. See DSA book for more.

It's common to say that architecture is at a higher abstraction level than design, or architecture is logical and design is physical. But this notion, albeit commonly accepted, is in practice useless. Where do you draw the line between high or low abstraction, between logical and physical? It depends!

So, my suggestion is:

- create a single design document.
- name this design document the way you want or, better, the way the readers are more accustomed to. Examples: "Software Architecture", "Software Design Specification".
- break this document into views and keep in mind you can create a view as a refinement of another view.
- make the views in the document navigable by adding cross-references or hyperlinks
- then you'll have higher level views showing broad but shallow overview of the design, and closer-to-implementation views showing narrow but deeper design details. *
- you may want to take a look at an example of multi-view architecture document (here).

Having said all that... **a more relevant question we need to ask is: how much design is enough?** That is, when should I stop describing the design (in diagrams or prose) and should move on to coding?

edited Apr 18 '17 at 18:56

answered Apr 24 '13 at 18:34

Paulo Merson

5,574 4 33 38

1 While I agree with the initial definition, it would be nice to add its source: "Paul Clements, Documenting software architectures: views and beyond". For your last question: You never stop designing. That is what Clements tries to point out in the referenced book. Every developer working on the system will design **some** part of it but most of these designs will not be relevant for the architecture. Therefore, if you want to talk about or document the software architecture, you stop as soon as you are discussing parts that are no longer relevant. – Thomas Eizinger Apr 18 '17 at 15:39

1 @ThomasEizinger. I added a link to our book. Good suggestion. And your comment also helps to give proper credit. As to the last question, I meant to refer to the design documentation effort. I edited the paragraph. Thanks! – Paulo Merson Apr 18 '17 at 19:00

3 votes

Yep that sounds right to me. The design is what you're going to do, and architecture is the way in which the bits and pieces of the design will be joined together. It could be language agnostic, but would normally specify the technologies to be used ie LAMP v Windows, Web Service v RPC.

answered Apr 1 '09 at 10:05

MrTelly
13.2k 39 76

3 votes

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them.

(from Wikipedia, http://en.wikipedia.org/wiki/Software_architecture)

Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

(from Wikipedia, http://en.wikipedia.org/wiki/Software_design)

Couldn't have said it better myself :)

answered Dec 24 '09 at 15:45

Larry Watanabe
8,575 8 35 44

---

3 votes

I view architecture as Patrick Karcher does - the big picture. For example, you can provide the architecture to a building, view its structural support, the windows, entries and exits, water drainage, etc. But you have not "designed" the floor layout's, cubicle positions etc. *

So while you've architected the building you have not designed the layout of each office. I think the same holds true for software.

You could view designing the layout, as "architecting the layout" though ...

answered Dec 24 '09 at 15:45

mr-sk
9,887 5 50 84

---

3 votes

Good question... Although the line between them is hardly a bright sharp line, imho, if you are using both terms, then Architecture encompasses more technical or structural decisions

about how to build or construct something, especially those decisions that will be hard (or harder) to change once implemeneted, whereas Design encompasses those decisions that either are easy to change later (like method names, class <-> file organizational structure, design patterns, whether to use a singleton or a static class to solve some specific problem, etc.) and/or those that effect the appearance or esthetic aspects of a system or application (Human Interface, ease of use, look and feel, etc.)

edited Dec 24 '09 at 15:49

answered Dec 24 '09 at 15:42

Charles Bretana
114k 20 122 199

---

3 votes

Software **architecture** is"concerned with issues...beyond the algorithms and data structures of the computation. *

Architecture is specifically not about…details of implementations (e.g., algorithms and data structures.) Architectural design involves a richer collection of abstractions than is typically provided by OOD" (object-oriented design).

Design is concerned with the modularization and detailed interfaces of the design elements, their algorithms and procedures, and the data types needed to support the architecture and to satisfy the requirements.

"architecture" is often used as a mere synonym for "design" (sometimes preceded with the adjective "high-level"). And many people use the term "architectural patterns" as a synonym for "design patterns."

Check out this link.

Defining the Terms Architecture, Design, and Implementation

answered Dec 24 '09 at 15:50

Tebo
6,831 10 42 61

3 votes

Architecture:
Structural design work at higher levels of abstraction which realize technically significant requirements into the system. The architecture lays down foundation for further design.

Design:
The art of filling in what the architecture does not through an iterative process at each layer of abstraction.

answered Dec 24 '09 at 15:55

Joshua Ramirez
399 3 10

*

3 votes

I really liked this paper for a rule of thumb on separating architecture from design:

http://www.eden-study.org/articles/2006/abstraction-classes-sw-design_ieesw.pdf

It's called the Intension/Locality hypothesis. Statements on the nature of the software that are non-local and intensional are architectural. Statements that are local and intensional are design.

answered Jan 23 '12 at 4:58

LindsayBradford
66 5

3 votes

...long time ago in a faraway place philosophers worried about the distinction between the one and the many. Architecture is about relationship, which requires the many. Architecture has components. Design is about content, which requires the one. Design has properties, qualities, characteristics. We typically think that design is within architecture. Dualistic thinking gives the many as primordial. But architecture is also within design. It's all how we choose to view what is before us - the one or the many.

answered Apr 5 '12 at 15:44

buzzcoda
31 1

---

3 votes

Pretty subjective but my take:

Architecture The overall design of the system including interactions with other systems, hardware requirement, overall component design, and data flow. *

Design The organization and flow of a component in the overall system. This would also include the component's API for interaction with other components.

edited Jan 3 '13 at 9:01

ErikE
33.9k 14 117 162

answered Dec 24 '09 at 15:46

Jesse Vogt
8,863 12 48 65

---

2 votes

Software architecture is best used at the system level, when you need to project business and functions identify by higher

votes

architecture levels into applications.

For instance, your business is about "Profit and Loss" for traders, and your main functions involved "portfolio evaluation" and "risk computation".

But when a Software Architect will details his solution, he will realize that:

"portfolio evaluation" can not be just one application. It needs to be refined in manageable projects like:

- GUI
- Launcher
- Dispatcher
- ...

(because the operations involved are so huge they need to be split between several computers, while still being monitored at all times through a common GUI)

a Software design will examine the different applications, their technical relationship and their internal sub-components.
It will produce the specifications needed for the last **Architecture layer** (the "Technical Architecture") to work on (in term of technical framework or transversal components), and for the project teams (more oriented on the implementation of the *business* functions) to begin their respective projects.

edited May 23 '17 at 12:02

Community ♦
1 1

answered Apr 1 '09 at 11:00

VonC
851k 301 2713 3273

2 votes

if somebody constructs a ship, then engine, hull, electric-circuits etc. will be his "architectural elements". For him, engine-construction will be "design work".

If he then delegates the construction of the engine to another team, they will create an "engine architecture"...

So - it depends on the level of abstraction or detail. One persons' architecture might be anothers' design!

answered Oct 21 '12 at 14:59

Gernot Starke
129 2 3

2 votes

Architecture are **"the design decisions that are hard to change."**

After working with TDD, which practically means that your design changes all the time, I often found myself struggling* with this question. The definition above is extracted from *Patterns of Enterprise Application Architecture*, By Martin Fowler

It means that the architecture depends on the Language, Framework and the Domain of your system. If your can just extract an interface from your Java Class in 5 minutes it is no longer and architecture decision.

answered Mar 24 '13 at 16:17

ekeren
2,235 3 23 49

1

Cliff Notes version:

vote

Design: Implementing a solution based on the specifications of the desired product.

Architecture: The foundation/tools/infrastructure/components that support your design.

This is a pretty broad question that will invoke a lot of responses.

answered Dec 24 '09 at 15:43

kd7
25.8k 10 64 94

1
vote

Architecture is the resulting collection of design patterns to build a system.

I guess Design is the creativity used to put all this together?

answered Dec 24 '09 at 15:43

Mark Redman *
16.4k 17 81 124

1
vote

Software design has a longer history while the term software architecture is barely 20 years old. Hence, it is going through growing pains right now.

Academics tend to see Architecture as part of the larger field of software design. Although there is growing recognition that Arch is a field within it's own.

Practitioners tend to see Arch as high-level design decisions that are strategic and can be costly in a project to undo.

The exact line between Arch and design depends on the software domain. For instance, in the domain of Web Applications, the

layered architecture is gaining the most popularity currently (Biz Logic Layer, Data Access Layer, etc.) The lower level parts of this Arch are considered design (class diagrams, method signatures, etc.) This would be defined differently in the domains of embedded systems, operating systems, compilers, etc.

edited Dec 29 '09 at 13:24

answered Dec 29 '09 at 13:03

LeWoody
2,741 3 18 26

---

1 vote

Architecture is high level, abstract and logical design whereas software design is low level,detailed and physical design.

answered Feb 24 '11 at 5:32

imran *
11 1

---

1 vote

Also, refer to: http://en.wikipedia.org/wiki/4%2B1_Architectural_View_Model

answered Jul 4 '11 at 6:39

Durga Vaddi
11 1

1 Vaddi, please consider summarizing the content as well as posting the link. See this meta post for a discussion on the topic. Thanks! – GargantuChet Jul 5 '11 at 4:21

1 vote

I like Roy Thomas Fielding's definition and explanation about what is software architecture in his paper: Architectural Styles and the Design of Network-based Software Architectures

> A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture.

He emphasizes "run-time elements" and "levels of abstraction".

answered Jan 12 '13 at 14:14

Jacky
5,320 7 25 37

1 vote

There is no definitive answer to this because "software architecture" and "software design" have quite a number of definitions and there isn't a canonical definition for either. *

A good way of thinking of it is Len Bass, Paul Clements and Rick Kazman's statement that "all architecture is design but not all design is architecture" [Software Architecture in Practice]. I'm not sure I quite agree with that (because architecture can include other activities) but it captures the essence that architecture is a design activity that deals with the critical subset of design.

My slightly flippant definition (found on the SEI definitions page) is that it's the set of decisions which, if made wrongly, cause your project to get cancelled.

A useful attempt at separating architecture, design and implementation as concepts was done by Amnon Eden and Rick Kazman some years ago in a research paper entitled "Architecture, Design, Implementation" which can be found

here: http://www.sei.cmu.edu/library/assets/ICSE03-1.pdf. Their language is quite abstract but simplistically they say that *architecture* is design that can be used in many contexts and is meant to be applied across the system, *design* is (err) design that can be used in many contexts but is applied in a specific part of the system, and *implementation* is design specific to a context and applied in that context.

So an architectural decision could be a decision to integrate the system via messaging rather than RPC (so it's a general principle that could be applied in many places and is intended to apply to the whole system), a design decision might be to use a master/slave thread structure in the input request handling module of the system (a general principle that could be used anywhere but in this case is just used in one module) and finally, an implementation decision might be to move responsibilities for security from the Request Router to the Request Handler in the Request Manager module (a decision relevant only to that context, used in that context).

I hope this helps!

*

answered Oct 13 '13 at 21:49

Eoin
507 2 8

1 2 next