Software Architecture

What is software architecture?

The software architecture of a system depicts the system's organization or structure, and provides an explanation of how it behaves. A system represents the collection of components that accomplish a specific function or set of functions. In other words, the software architecture provides a sturdy foundation on which software can be built.

A series of architecture decisions and trade-offs impact quality, performance, maintainability, and overall success of the system. Failing to consider common problems and long-term consequences can put your system at risk.

There are multiple high-level architecture patterns and principles commonly used in modern systems. These are often referred to as architectural styles. The architecture of a software system is rarely limited to a single architectural style. Instead, a combination of styles often make up the complete system.

What is software design?

Software design is the process of conceptualizing the software requirements into software implementation. This is the initial phase within the software development life cycle (SDLC) (/software-integrity/resources/knowledge-database/software-development-life-cycle.html)—shifting the concentration from the problem to the solution.

When conceptualizing the software, the design process establishes a plan that takes the user requirements as challenges and works to identify optimum solutions. The plan should determine the best possible design for implementing the intended solution.

Software design includes all activities that aid in the transformation from requirement specification to implementation. Major artifacts of the software design process include:

Risk Management +

Software Vulnerabilities +

- **Software requirements specification.** This document describes the expected behavior of the system in the form of functional and non-functional requirements. These requirements should be clear, actionable, measurable, and traceable to business requirements. Requirements should also define how the software should interact with humans, hardware, and other systems.
- **High-level design.** The high-level design breaks the system's architectural design into a less-abstracted view of sub-systems and modules and depicts their interaction with each other. This high-level design perspective focuses on how the system, along with all its components, implements in the form of modules. It recognizes the modular structure of each sub-system and their interaction among one another.
- **Detailed design.** Detailed design involves the implementation of what is visible as a system and its sub-systems in a high-level design. This activity is more detailed towards modules and their implementations. It defines a logical structure of each module and their interfaces to communicate with other modules.

What's the relationship between software architecture and software design?

Software architecture exposes the structure of a system while hiding the implementation details. Architecture also focuses on how the elements and components within a system interact with one other. Software design delves deeper into the implementation details of the system. Design concerns include the selection of data structures and algorithms, or the implementation details of individual components.

Architecture and design concerns often overlap. Rather than use hard and fast rules to distinguish between architecture and design, it makes sense to combine them. In some cases, decisions are clearly more architectural in nature. In other cases, decisions focus heavily on design and how it helps to realize that architecture.

An important detail to note is that architecture is design, but not all design (https://en.wikipedia.org/wiki/Software_design) is architectural. In practice, the architect is the one who draws the line between software architecture (architectural design) and detailed design (non-architectural design). There are no rules or guidelines that fit all cases—although, there have been attempts to formalize the distinction.

Current trends in software architecture assume that the design evolves over time and that a software architect cannot know everything up front to fully architect a system. The design generally evolves during the implementation stages of the system. The software architect continuously learns and tests the design against real world requirements.

What problems does architecture analysis solve?

Software defects that lead to security problems come in two major flavors:

1. bugs in the implementation and
2. flaws in the design (http://searchsecurity.techtarget.com/opinion/Opinion-Software-insecurity-software-flaws-in-application-architecture).

Implementation bugs in code account for at least half of the overall software security problem (https://www.synopsys.com/blogs/software-security/myth-5-its-all-about-finding-bugs-in-your-code/). The other half involves a different kind of software defect occurring at the design level. The division of design flaws and bugs is about 50/50. Both need to be secured to ensure your software's well-being. You can institute the best code review program on the planet, with the strongest tools known to humanity, but it's unlikely that you will be able to find and fix flaws this way.

4 ways to identify flaws

1. Analyze fundamental design principles.
2. Assess the attack surface.
3. Enumerate various threat agents.
4. Identify weaknesses and gaps in security controls.

It is far more cost-effective to identify and remediate design flaws early in the design process than to patch flawed design implementations after deployment. Architecture risk analysis (/software-integrity/software-security-services/software-architecture-design/risk-analysis.html) (ARA), Threat Modeling (/software-integrity/software-security-services/software-architecture-design/threat-modeling.html), and Security Control Design Analysis (/software-integrity/software-security-services/software-architecture-design/security-control-design-analysis.html) (SCDA) are useful in finding and fixing design flaws.

SCDAs are a light-weight approach to ARA. They take less time to conduct and can be carried out by a much larger talent pool than traditional ARA reviews. Most importantly, the lightweight approach is efficient enough that it can be scaled to cover an entire application portfolio.

Organizations failing to integrate architecture and design reviews in the development process are often surprised to find that their software suffers from systemic faults both at the design level and in the implementation. In many cases, the defects uncovered in penetration testing (/software-integrity/managed-services/penetration-testing.html) could have been identified more easily through other techniques—earlier in the life cycle. Testers who use architecture analysis results to direct their work often reap greater benefit.

BLOG (HTTPS://WWW.SYNOPSYS.COM/BLOGS/SOFTWARE-SECURITY/CATEGORY/SOFTWARE-ARCHITECTURE-AND-DESIGN/)

NEW Want to secure your apps? Build security in with the right toolchain (https://www.synopsys.com/blogs/software-security/application-security-toolchain/)

OWASP Top 10 web application security risks (https://www.synopsys.com/blogs/software-security/owasp-top-10-application-security-risks/)

From mainframes to connected cars: How software drives the automotive industry (https://www.synopsys.com/blogs/software-security/automotive-software-security/)

RELATED RESOURCES

Get support no matter where you are in the SDLC (/software-integrity/software-security-services/software-architecture-design/risk-analysis.html)

Find the weaknesses in the design of your application (/software-integrity/software-security-services/software-architecture-design/threat-modeling.html)

(https://www.synopsys.com/)

PRODUCTS

Software Integrity (/software-integrity.html)

RESOURCES

Solutions (/solutions.html)

Services (/services.html)

CORPORATE

About Us (/company.html)

LEGAL

Privacy (/company/legal/privacy-policy.html)

Semiconductor IP (/designware-ip.html)
Verification (/verification.html)
Design (/implementation-and-signoff.html)
Silicon Engineering (/silicon.html)

Support (/support.html)
Community (/community.html)
Manage Subscriptions (https://www.synopsys.com/apps/subcenter/ea1.php)

Careers (/company/synopsys-careers.html)
Global Citizenship (/company/global-citizenship.html)
Investor Relations (/company/investor-relations.html)
Contact Us (/company/contact-synopsys.html)

Trademarks & Brands (/company/legal/trademarks-brands.html)
Software Integrity (/company/legal/software-integrity.html)

FOLLOW

(https://twitter.com/synopsys)

(https://www.linkedin.com/company/s

(https://www.facebook.com/Synopsys

(https://www.youtube.com/user/syno