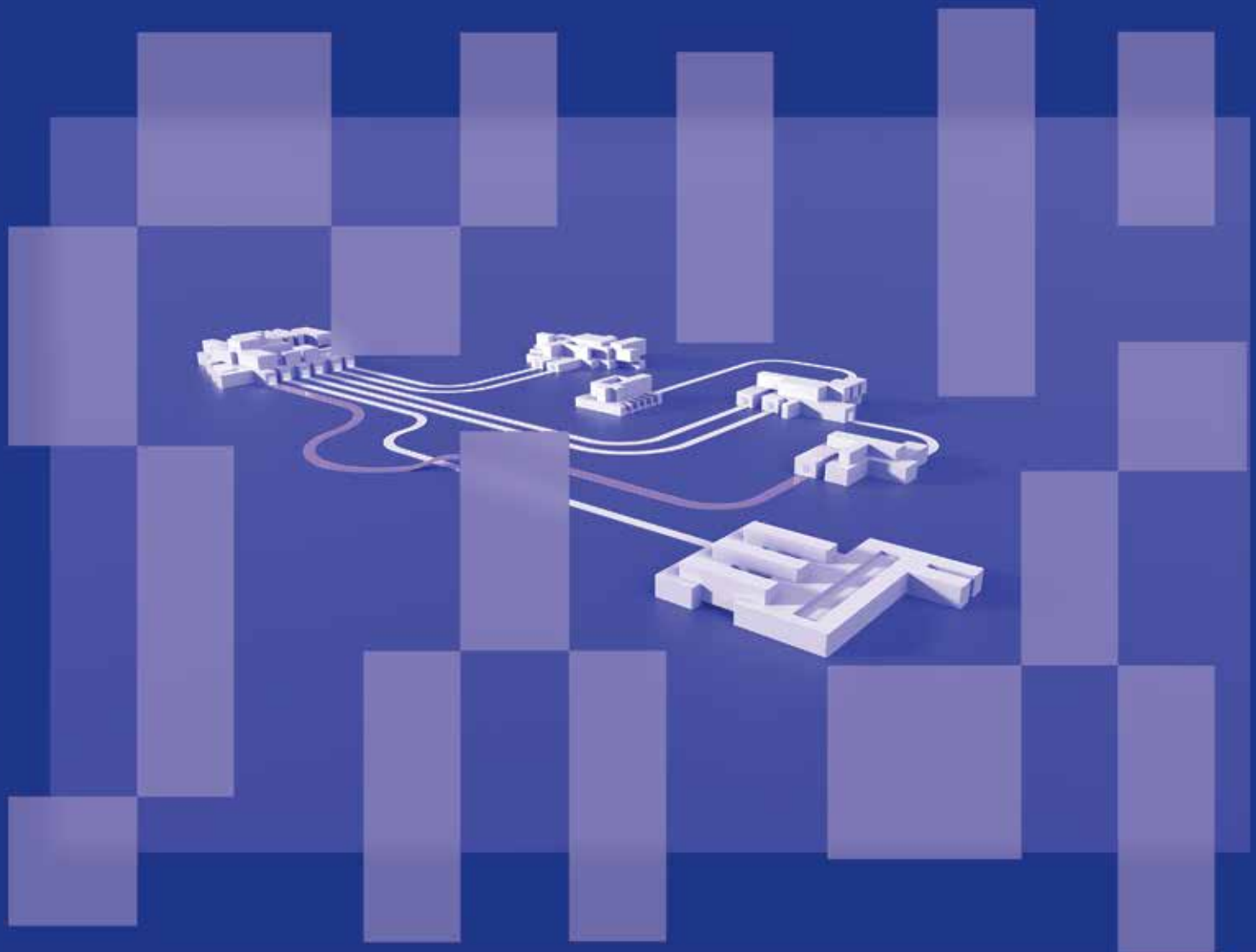
Qt Quality Assurance

WHITEPAPER

Crafting Robust Foundations with Software Architecture

A guide to success

New to Architecture Verification? This guide will explain what software architecture is and why it is important.



"*What is software architecture?*" is a question that might have different answers depending on your knowledge, experiences, and use cases. While definitions may differ, certain fundamental concepts are crucial in software architecture.

The term "architecture" in software development was initially borrowed from the construction industry due to the similarities between the two fields. The waterfall methodology was widely used in software development in the past, and detailed plans were required before writing any code. This approach was similar to the planning required in construction, where the architectural design must be finalized before construction begins. But things have changed.

Modern software development methodologies are designed to be flexible and easy to change over time, so there's less need for strict planning at the beginning. However, the first decisions made regarding how the software is built, can take time and effort to change at a later stage.

If you were to ask what software architecture includes, the answer would differ based on their development lens. This whitepater concentrates on the structural aspects of the software and the role of architecture in it.

What is Software Architecture?

The software architecture is the foundation of a software system and helps understand how the code is structured.

Software architecture refers to the overall design and structure of a software system, which includes how its various components interact to achieve the desired functionality — it is the foundation of a software system.

As defined by the ISO/IEC/IEEE 42010 standard, software architecture refers to the core concepts and characteristics of a system within its surroundings, expressed through its components, connections, and the guiding principles of its creation and development.

The standard definition essentially highlights the following points:

- When you use software, you might not think about how it's built. But, just like a building needs a blueprint, software needs a plan too. That plan is called software architecture.
- The software architecture is like a map showing how the software works and fits into the world around it. It's important to make sure the software works well in the environment in which it is used because it minimizes potential disruptions and maximizes the software's longevity and effectiveness in serving its purpose.
- To help everyone understand the software architecture, a description is created that explains how it works and solves the problems it was created for. This description is like a guidebook that helps people see how the software will meet their needs.
- Different people care about different things when it comes to software. To ensure everyone can understand the software architecture, different scenarios are created that show how it works from different angles. These scenarios help people see how the software will work for them.

Software projects can vary in the level of attention given to design, time spent, focus, and documentation dedicated to different parts of software architecture. However, software architecture primarily comprises vital design decisions that significantly influence the system.

This foundation is a crucial facet of software development, as it directly affects the quality of software developed on top of it. In other words, it has an impact on the system's

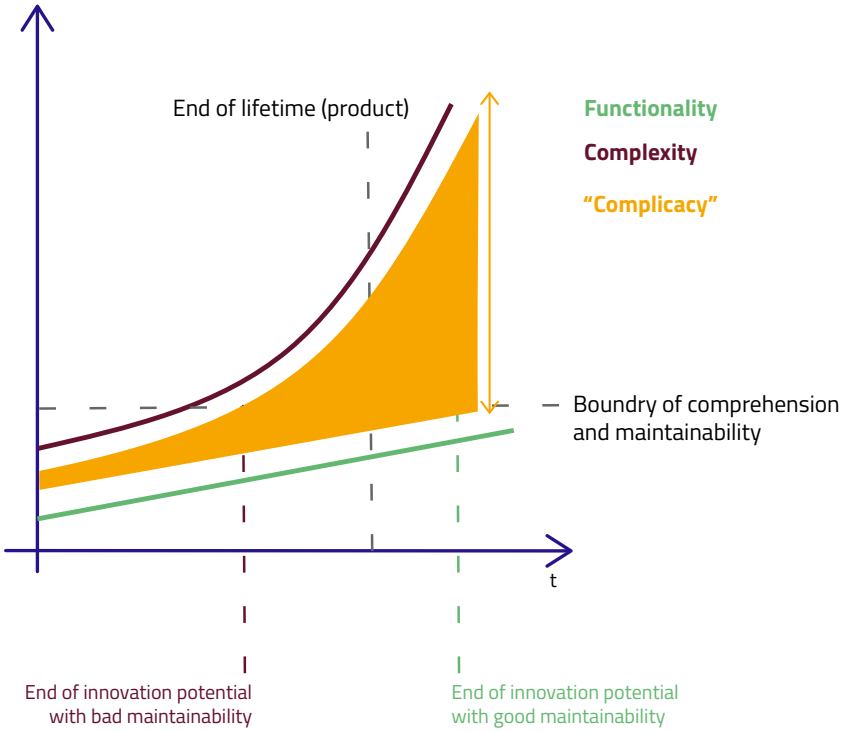
- Performance
- Scalability
- Maintainability

Simply put, how software is created today must be flexible enough to handle future changes. This means that the software should be designed to make it easy to evolve and adapt to these changes rather than completely rebuilding it from scratch.

Why is Software Architecture so Important?

Before developing software, it's important to have a plan in place. Just like building a house, the foundation of a software system plays a significant role in its quality — the architectural decisions dictate how the software will function. Having good architectures in place helps increase the chances of success for the software system. It ensures that it can function properly and be easily maintained.

The increase of functionality automatically leads to an exponential increase of complexity of a software code. It becomes more and more difficult to understand and maintain.



Even small software systems have an architecture, and it's important to have a good one in place to ensure the system works well and can be easily maintained to make it flexible enough to meet the fast-paced changes in todays world. A well-designed architecture becomes even more crucial with larger and more complex systems.

So, while it's tempting to start coding right away, planning and designing the software architecture is an essential step in the development process. The benefits of a software system's architecture with a solid foundation are numerous. Let's explore these advantages in more detail.

Managing the presence or absence of quality attributes.

Software architecture helps manage the quality of your software.

The design of a software system is critical in ensuring that it works well and meets the needs of its users. There are certain properties of a software system that can be measured and tested, such as:

- how easy it is to maintain
- how well it works with other systems
- how secure it is
- how quickly it performs tasks

These properties are called quality attributes, and they're critical because they affect how happy users are with the system.

Sometimes, different quality attributes conflict with each other, so it's important to make sure the architecture balances them properly. When the architecture is done well, the system meets all the important quality requirements that its users need.

Making it easier for everyone involved to talk to each other

Different perspectives. One goal. Software Architecture makes it easier for people to make decisions which include all needs.

The good news is that software architecture is designed to be easy to understand for everyone involved, even if they don't know a lot about technology. Different stakeholders might care about different things when it comes to software architecture. The relationship between software architects and developers is like that between birds flying over a pond and frogs in that pond—the frogs can see the details of their immediate surrounding, but don't really know much about what happens around them. The birds on the other hand see the big picture, but can't see any details.

When software is designed, different groups of people have different priorities, e.g.:

- The developers focus on implementing their current tasks or features as quickly as possible.
- The project managers might be more concerned about making sure it is available on time and within budget.
- The security team worries about making sure the software is secure so that nobody can hack into it to steal important information.

It is the software architect who cares about making sure it's easy to add new features or fix problems while making sure everyone's priorities are taken into account. For that, a common language and a shared design approach become even more vital. By doing so, stakeholders can talk about their priorities in a way that makes sense to everyone else and the software can be designed to meet everyone's needs, while at the same time remain adaptable and maintainable.

This is especially useful for big and complicated software systems that need to be easier to understand. When stakeholders are figuring out what they want from the system and making important decisions, having a formal software architecture can help everyone negotiate and talk about what is needed.

Exploring methods to improve the accuracy of predicting the duration and cost of a project.

Software architecture gives valuable insights and allows for precise planing.

Project managers need to know how long a software project will take and how much it will cost to plan resources and monitor progress properly. One of the important duties of a software architect is to help project management by providing this information and breaking down the necessary tasks and estimates for those tasks. How the software is designed affects what tasks will be needed for implementation, so the software architect can help project management create the tasks.

Good collaboration between the project manager, software architect, and developers is beneficial for creating accurate estimates. The most precise estimations are attained through team discussions until a mutual agreement is reached. Sometimes, during these discussions, someone on the team provides an insight that no one else has considered, allowing everyone to rethink their position and possibly revise their estimates.

A well-designed software architecture that accurately reflects the project's requirements can help avoid costly rework that would be necessary if crucial requirements were missed. Moreover, a well-thought-out architecture reduces complexity, making it easy to understand and reason, resulting in more accurate cost and effort estimates.

Onboarding new team members to your development team

Software architecture helps teams grow.

A team that works on developing software often needs to bring in new people as they grow. This can be a challenge, as it takes time for new people to learn how everything works. Apart from that, different people might need to work on the software to fix problems or improve it over time—this is where your software architecture becomes even more important.

Essentially, the architecture and implementation must match to make onboarding easy. The more they deviate, the harder the onboarding gets. Imagine having to explain a fluid plan that changes every time you have a new person on the team. It's far from ideal. However, if the team has a good plan for building the software and is committed to the original plan, it is easier for new people to learn everything about the software they need to know.

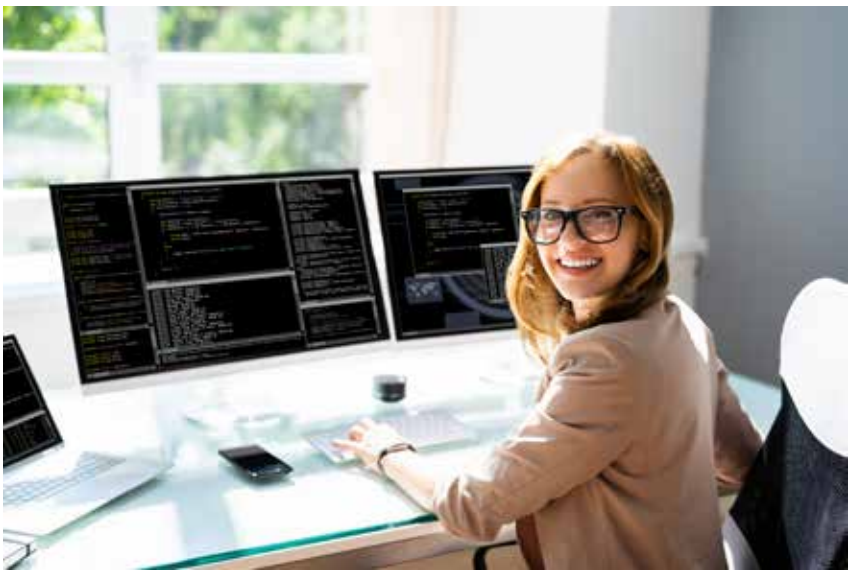
Importance of Conformance Checks in Overcoming Software Architecture Challenges

Software architecture needs to be verified regularly to ensure it can fulfill its important role.

Architecture conformance checking is a process that ensures the way a software system is built matches how it was supposed to be built. Challenges arise as soon as architecture and implementation deviate. The stronger the deviation, the harder the challenge becomes. In this context, conformance checks make it plain for all to see a deviation has happened. Automated architecture conformance checking is not very common, and only a few tools are available to monitor the health of software systems.

Architecture monitoring is a process that helps to make sure that software systems are functioning as they should be. However, many organizations do not use specific automated tools for architecture monitoring, which can lead to problems later in the software life cycle. This is especially true when building new software on top of old, outdated systems.

Sometimes, how software is built can be different from how it was originally planned. This is called software architecture erosion. It can happen when software developers make decisions that don't follow the original plan or established rules.



Understanding software architecture erosion

Software architecture erosion (also known as architecture debt) is when software doesn't turn out the way it was supposed to be designed. When software is created, sometimes shortcuts, temporary fixes, and other compromises are made that can cause problems later on. This can happen when the people building the program don't follow the rules or make decisions that go against what the program was supposed to do. It's worth noting that this doesn't always mean that the developers did a bad job. It might just be that they didn't know about a certain problem or issue.

Realistically speaking, you would have:

- an architecture
- a design
- perhaps a detailed design
- an implementation (sometimes with even multiple layers as models etc.)

The challenge of mixing things can happen at each level and has a different impact when that happens. As a result, it introduces technical debts to the entire process.

The art of architecture is that the most important decisions at the architecture level make it impossible for the lower levels (design etc.) to mix parts together that should not interact when not deviating from the architecture.

The impact of technical debts

Technical debt can harm the long-term success of software systems by introducing undesirable changes to important parts of the software. This can happen when stakeholders have to make quick decisions to get something done fast or when they don't have enough knowledge or resources to create high-quality software. For example, someone might prioritize getting a critical function working quickly over making sure the software is designed well.

Technical debt can also happen when design decisions focus on short-term goals rather than long-term quality. These types of technical debt can cause problems with maintaining and improving the software over time.

That's why it's important for stakeholders to verify their software architecture before making changes to ensure everything is done according to the plan. By doing this, they can save time and money and prevent problems from happening later on.

Conclusion

An effective architecture serves as a unifying force that connects all parties involved, streamlines operations, saves time and money, and enables early detection of potential design flaws.

The architecture is a foundation that binds together different system components, providing a clear development, testing, and deployment roadmap. It creates a shared understanding of the system's structure, functionality, and behavior, enabling cross-functional teams to work together more efficiently and effectively.

With a well-designed architecture, stakeholders can make informed decisions, adapt to changing requirements, and deliver high-quality solutions that meet user needs and expectations.

Qt Quality Assurance

www.qt.io/axivion