KELLTON

WHITEPAPER

Software development team structure:

Our definitive guide

Executive Summary

Behind every successful software product is an efficient, well-structured team. Unfortunately, many companies fall at the first hurdle when choosing a structural model for their software development teams. Inefficient processes, poor communication, and a lack of clear objectives are among the common pitfalls they experience.

However, these issues can be addressed by selecting an approach where everyone is on the same page and working synchronously rather than in silos. Software development is a dynamic process, with team members working on multiple fronts simultaneously. Defining a structure ensures everyone is aligned towards a common goal and delivers effective outcomes.

This white paper covers the standard software development team structures and aims to lead companies toward choosing one that suits their requirements. It further explores why the agile development model is a cut above traditional models and what principles to implement to ensure maximum efficiency.

Table of Contents

Importance of a well-structured software development team

No team at any level of any function can perform up to par without the right organizational patterns, processes, and strata in place. Software development teams are no exception.

A software development team—underpinned by the proper structure—is essential as it ensures that members working on the project are doing so in a way that will most likely lead to success. Everyone understands their role, what they are responsible for, and what the project aims are.

When companies prioritize team structure, they seep balance and coherence into workflows and position every member to deliver high standards consistently throughout the project's lifecycle. Defining a structure helps prevent poor staffing decisions and leads decision-makers to ensure the right proportion of expertise at all levels.

The right structure also breaks down silos and mobilizes team members to collaborate frequently. Problems tend to arise when people work in isolation. If no one else knows an issue, no one can help to fix it. Once everyone knows what everyone else is doing, decisions are taken swiftly, and problem resolution is sped up.

The three types of software development team structures commonly used

1. Generalist structure
2. Specialist structure
3. Hybrid structure

There are many different philosophies on the best way to structure software development teams, but there are some standard organizational models that offer a starting point. We take a look at some of the most common team structures and discuss their pros and cons.

Generalist structure

The generalist team structure is a dynamic that orients every team member toward multiple functional areas of the business. No one has a specialty; each individual contributes to the team's success by sharing their knowledge and skills in different areas. This structure requires team members to be comfortable with accountability and ambiguity. Because the scope of each person's responsibilities is so broad, they need to manage their time well and feel confident making decisions on the fly.

Teams built on the generalist structure thrive in collaborative environments where trust binds everyone. These teams tend to have a higher retention rate because individuals experience growth personally as well as professionally in their respective roles. The downside, however, is the lack of depth of knowledge in any functional area, leading to incompetencies.

PROS

- Low cross-team dependencies;
- As every individual understands the project, it is easier to focus on development and churn out definite outcomes.

CONS

- With no specialization, the need to onboard new team members always remains to bridge the talent gap.

Specialist structure

As the name suggests, the specialist structural model comprises individuals with specific areas of expertise. Since team members are deeply skilled at what they do, this model leads to more efficient and effective software development outcomes. The specialist team structure also makes managing workflows easier and ensures tasks are completed on time.

The specialist structural model, however, can be inflexible and limit the options for professional advancements. Also, this model can create high dependencies. If a team member leaves, it can be challenging to bridge the gap under the pressure of targets.

PROS

- In-depth specialization.
- Ability to address complex project needs with confidence.

CONS

- Since every team member works independently, the components they build may not fit the whole.
- Lack of general knowledge may create a communication vacuum.

Hybrid structure

A hybrid team structure combines elements of the generalist and specialist models. In this structure, a project manager usually oversees the overall project, but the team is also self-organizing and autonomous.

Typically, these teams will start working on the project and narrow it down to more specialized features. This type of team structure is often used in large organizations where there is a need for both flexibility and control. The hybrid model, however, creates communication barriers. With many hybrid specialists on the team, it can be difficult to agree on certain issues and make decisions faster.

PROS

- The amalgam of specialists and generalists ensures meaningful outcomes and that there is a general understanding of the project across all fractions of the team.
- More streamlined, effective product development that unlocks customer-centricity and greater value.

CONS

- Since individuals come from different workflows, finding common ground can be a pain point.
- Coordination can be difficult.
- A hybrid structure demands significant investments in terms of time and money.

The right questions to ask when establishing a team structure

Now we know what a structured software development team entails and the common structures most organizations adopt, we must establish one by asking the right questions. This can be challenging, but the payoff is worth it.

Roles and responsibilities	Project requirements	End-game conditions
Who should lead the team	What are the project requirements	What are the conditions that will signal the project's completion
What roles do you need in a team	What does the end-user or client want from this project	What is the goal associated with the end product
What are each team member's roles and responsibilities	How does this project fit within the broader objectives of the company	Does the product cater to the client's objectives
Who will be responsible for which aspects of the project	What is the budget for this project	How will the team ensure project support once it's delivered

Factors influencing software development team structures

There are many factors to consider when deciding on a software development team structure. We discuss the five most important below:

1. COMPLEXITY OF THE PROJECT

The complexity of a project directly impacts the structure of the software development team. If a project is extensive and intricate, it's important to define a large, high-functioning team with efficacy, competence, and alignment at the core of it. On the other hand, if it is a smaller and more straightforward project, a minor team can generally tick all the right boxes. The table below explains how complexity can influence team structure.

Proof scope/stage	Team size	Team roles
Discovery/proof of concept	Up to 5 specialists	Project Owner, Project Manager, Business Analyst, Software Architect, UI/UX Designer
MVP development	8+ specialists	Project Owner, Project Manager, Business Analyst, UI/UX Designers, Software Engineers, Test Engineers
Product development	No team size limit *(Largely depends on the choice of the team workflow methodology - Agile or Waterfall)	Project Owner, Project Manager, Business Analyst, UI/UX Designers, Software Engineers, Test Engineers, Test Automation Engineers, Performance Engineers, DevOps and Security Engineers

2. SKILLS AND CULTURE

Each software project requires its own set of skills and degrees of expertise. When picking the right team structure to implement, consider project requirements carefully. Companies should consider what culture they want to foster within software development teams. Do they want a more relaxed and informal or more formal and structured environment? This largely depends on what cultural and personal attributes organizations are comfortable with.

3. PURPOSE OF THE TEAM

The team's purpose or mission makes a huge difference when structuring a software development team. A team that is focused on developing new features for a product will likely have a different structure than a team responsible for maintaining and supporting an existing product.

4. BUDGET

Budget influences every decision when choosing a suitable structure for software development teams. When it is determined how much money is on the table, it is easier to decide how many team members can be hired, how many specialists can be secured, and how feature-rich the product can be. At its simplest, a budget establishes the limits of expenditure and helps decision-makers make more conscious decisions without burdening resources over the course of time.

5. WORKFLOW APPROACHES

Whether a company opts for a Waterfall or Agile development approach has a direct bearing on the software development team structure. The choice of project management methodology determines the strength of the team, the responsibilities associated with each role, and the relationships between members. To understand how the waterfall or agile methodology can alter the development team composition, it is important that we learn their key characteristics.

Waterfall vs. Agile: Must-know differences between methodologies

Criterion	Waterfall	Agile
Approach	Waterfall development methodology takes a collaborative approach to software development, with requirements and solutions evolving over the course of an iteration.	Agile uses an iterative approach to software development that allows the product owner to be flexible in terms of the project scope and schedule.
Scope	The scope is defined well in advance. Changes are limited.	Changes are enabled and allowed during the development project.
Mindset	The mindset is project-driven. Accomplishing targets is the prime motive.	The mindset is product-driven. Delivering value for money is the prime motive.
Testing	Tests are done after the building phase.	Tests run simultaneously with software development.
Client involvement	Clients are involved only during the milestone stage when a deliverable is ready.	Clients are involved at every stage of product development.

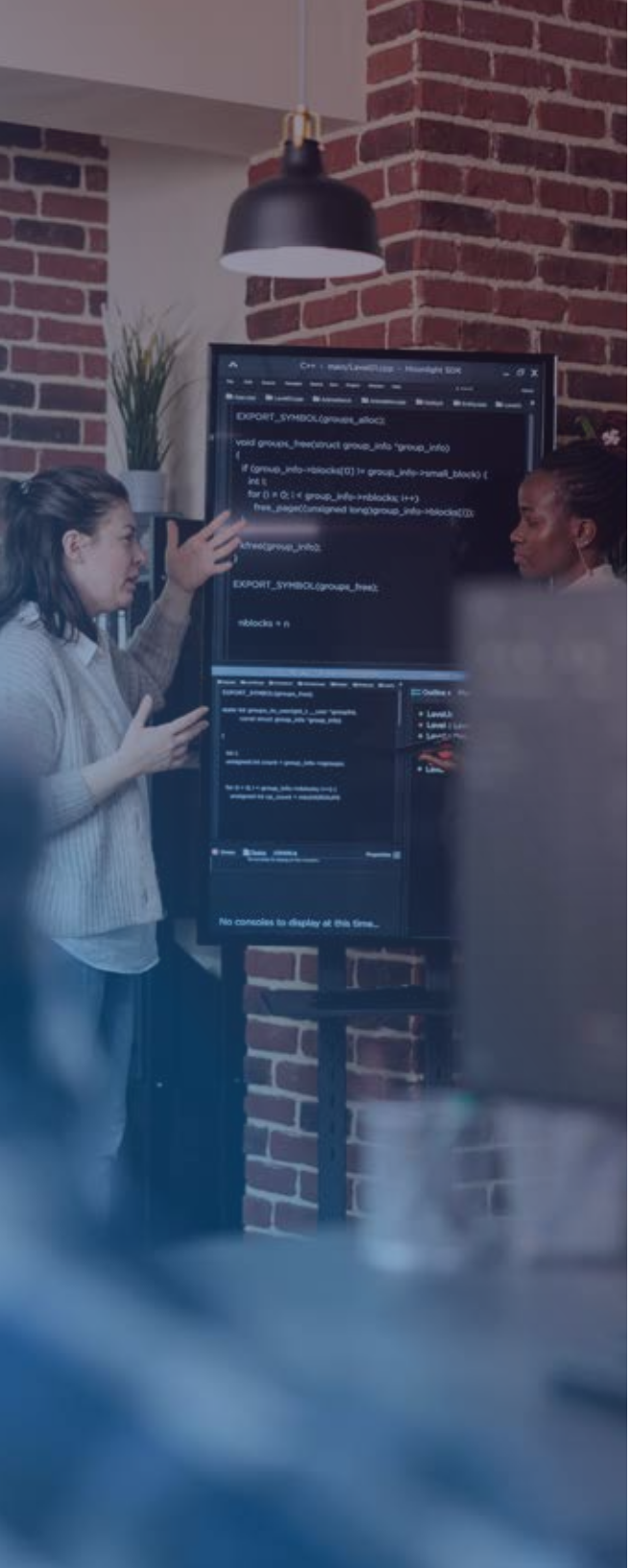
How do differences between Waterfall and Agile methodologies impact software development team structure?

The waterfall methodology proposes a linear system of work that leans on hierarchical relationships between team members. Managers and subordinates work within the limits of their job responsibilities and produce fit-to-purpose solutions. Project managers have significant control over what goes on within a team.

In contrast, agile methodology encourages self-organized, self-managed teams. Every team member chooses how best to accomplish targets rather than being directed by a manager. Scrum Masters are the organizational leaders, and instead of controlling workflows, they are more focused on fostering relationships and nurturing a team environment conducive to growth and productivity.

The table below explains the notable differences between Waterfall and Agile teams.

Waterfall	Agile
Top-down approach to management: The project manager makes decisions on behalf of the team.	**Self-organized teams:** Autonomy prevails within the team. Every member identifies what needs to be done and brings their best self to work.
The focus is on examining and assessing the performance of every team member.	The focus is on reviewing and evaluating the performance of the whole team.
Distinct roles and titles.	Cross-functional talent.
No cap on team size.	Four to ten members at every level of the hierarchy.
The team works on several projects at a time.	The team concentrates on one project at one time.
Synchronization problems occur due to vertical hierarchy; poor coordination affects results.	Smaller team sizes with greater autonomy ensure a higher degree of synchronization, coordination, trust, and transparency.



Why should companies shift to agile over waterfall methodology?

In today's business world, the ability to move quickly and adapt to change is more important than ever. That's why an Agile software development team can be a valuable asset for your company.

Agile teams are typically small, with no more than ten members, and are characterized by their ability to move quickly and adapt to change. That flexibility is what sets Agile teams apart from traditional software development teams, which tend to be much larger and slower-moving.
Moreover, Agile teams are much better equipped to handle changes in scope or requirements than traditional teams. Because they are smaller and nimbler, they can quickly pivot when necessary without getting bogged down in bureaucracy and jeopardizing the quality of work.

Another benefit of working with an Agile team is that they tend to be more cost-effective than traditional teams. Because they require less overhead, companies tend to save money. Additionally, because Agile teams work in short sprints rather than long projects, companies only pay for the work that is being done.

Finally, Agile teams are generally more successful than traditional teams, given the right project. It is observed that businesses using agile methodology are more likely to meet their deadlines and stay within their budget.

Software development team structure: Who is who?

A software development team is a collection of professionals who work in tandem to design, create, test, implement, and maintain software products. The makeup of a development team varies depending on the structure chosen but typically includes developers, designers, QA engineers, and managers or product owners.

This section explores the different roles commonly found on a software development team.

Role	Area of function	Key responsibilities
 **Business Analyst**	Builds an abstract product idea and aligns a client's vision with a set of tangible requirements.	• Translates the client's requirements and understands what their needs look like. • Works closely to help the software development team shape the right product. • Understands perspectives and defines paths to suit the project needs.
 **Product Owner**	The ultimate decision-makers, product owners have a say in every aspect of product development. Unlike business analysts, they direct the team to achieve the project without delving deep into the technical side of it.	• Formulates the product strategy while considering business and market needs. • Commits to customer success and ensures the product delivers on expectations. • Managing product backlogs and modeling workflows.
 **Project Manager**	Responsible for the day-to-day execution of tasks on the team level.	• Assigns work responsibilities to team members. • Keeps track of project deadlines and updates statuses. • Fosters teamwork and collaboration. • Looks for opportunities for improvement. • Ensures that the team is delivering more value with every release.
 **UX/UI Designer**	Creates the look and feel of the product and works on user interaction and experience design.	• Undertakes research and maps user personas to tap into the market pulse. • Leverages insights to build user journeys that maximize experience and conversion. • Tasked with product design, wireframing, and prototyping. • Designs intuitive interfaces for the product.

Role	Area of function	Key responsibilities
 **Software Architect**	Makes challenging decisions that impact software design.	• Develops a high-level architectural design for software. • Makes the appropriate selections of technologies and platforms to realize the product vision. • Reviews and establishes code quality standards. • Pairs the right integration patterns to drive consistency.
 **Software Developer**	Responsible for developing the product's front-end and back-end.	• Engineers and stabilizes the product. • Creates elements that users directly interact with. • Implement the core business logic of the product. • Devises and implements app integrations. • Provides technical support during development
 **Testing Engineer**	Tests the product based on functional and non-functional requirements and ensures it performs consistently and securely.	• Examines product performance to check for defects and anomalies. • Fixes errors and bugs to ensure fault-free performance. • Creates different types of testing documentation, including test scenarios and protocols. • Establish and implement quality assurance practices.
 **DevOps Engineer**	By integrating processes, tools, and methodologies, DevOps engineers balance needs across all phases of software development, from coding and deployment to maintenance and updates.	• Facilitates cooperation and coordination between the operations and development teams. • Designs and implements CI/CD pipelines for accelerated delivery.

Adding value with the right software development team structure

It is important to ensure that a team is always operating at its peak. That said while defining a structure, implementing best practices can go a long way in ensuring that a development team is positioned to deliver consistent outcomes and lasting value.

One of the biggest reasons software development projects tend to go off the rails is that the project scope is not defined clearly from the outset. When everyone on the team is working with different assumptions about what the end product should look like, it's only natural that there will be disagreements and confusion down the line. To avoid this, meet regularly with your team at the beginning of the project to develop a clear, concise definition of product requirements. Once you have these documented, make sure everyone is on the same page and refers back to that definition whenever there is any doubt about what needs to be done.

For a software development project to run smoothly, it is essential that everyone involved knows exactly what needs to be done and when it needs to be done. One of the best ways to keep track of all this information is by using project management tools. By assigning tasks to specific team members and setting deadlines, you can ensure that everyone knows exactly what they need to do and when they need to do it—and you can hold them accountable if they do not meet those deadlines.

Miscommunication is one of the biggest enemies of efficiency, so it is important to communicate regularly with your team—in person and in writing. If there are any changes to the project scope or deadlines, make sure everyone on the team is aware of them as soon as possible. Similarly, if anyone has any questions or concerns, address them immediately instead of letting them fester. The sooner you deal with potential problems, the less likely it is that they will turn into actual problems later on down the line.

Get in Touch

North America: +1.844.469.8900
Asia: +91.124.469.8900
Europe: +353.76.604.2716

General Inquiries:
ask@kellton.com | www.kellton.com