embarcadero®

Team Collaboration in the Software Development Life Cycle

INTRODUCTION

Software has always required some amount of collaboration and cooperation to produce — now more than ever. Only the simplest of applications or utilities are one-person projects, and even then others are needed to maintain them, extend them, modify them, or fix them.

The benefits of solid team collaboration are fairly straightforward. If one player has trouble, other members of the team can step in and take up the slack. Team collaboration is a necessity in today's environment of rapidly changing customer demands and requirements, where products are expected to be brought to market within very limited time frames while still maintaining a high level of quality.

In order to support the kind of collaborative, productive environment that meets the challenges of today's software development realities, organizations must embrace culture change and adopt software tools that emphasize safely sharing and modifying code, facilitate communication among all levels of the organization, and avoid bureaucratic rules that work against modern software development life cycles (SDLC) such as Agile, Scrum, DevOps, and other rapid-iterative process models.

Beyond this, according to the Project Management Institute, poor communication can result in project failures about one-third of the time. This impact to your organization's bottom line can make the difference between your overall success or failure. The investment made in collaboration tools can mitigate these communication problems as long as management supports their developers with a commitment to open communication and information sharing.

COLLABORATIVE ENVIRONMENTS

In software development models that encourage team collaboration, teams work together toward shared goals. Instead of compartmentalizing design, development, testing, and operations, members of each team are involved with every stage of a project. If done correctly, this can result in the following:

- Better communication
- The ability to adapt quickly to changing requirements
- Easier to spot pitfalls
- Better understanding of limitations
- Better identification of possible improvement areas
- Faster delivery
- Higher quality output
- Easier to support, resulting in fewer down-the-line costs

SDLC MODELS

Before we get too far ahead of ourselves, let's look at a few models designed to create better collaboration. SDLC refers to the end-to-end process used for the designing, developing, testing and deployment of software. Some form or other is used in most organizations, either explicitly defined or as a matter of course. The segments of SDLC include:

01 Planning
02 Defining
03 Designing
04 Building
05 Testing
06 Deployment

While there are many SDLC models, we will focus on a comparison among only a few of the most well-known methods. These are "Waterfall," which is the traditional siloed methodology and, in contrast, three collaborative approaches: Agile, Lean, and DevOps.

WATERFALL

Originally referred to as a ***linear sequential life-model*** and expressed in a paper by Winston Royce describing a methodology and company that wasn't working well, the Waterfall model was unfortunately adopted as a prescriptive solution despite Royce himself indicating that it caused massive problems. Waterfall is so named because work in a project cascades downhill, with each segment passing results to the next group upon completion.

Waterfall is a document-intensive process that begins with creating a requirements document. This document is then passed onto developers, who build the project and create an implementation, which is delivered to QA for testing and finally, assuming everything is acceptable, deployment. Maintenance, provided by operations, is considered more of an after-thought. In Waterfall, each team works in isolation. If issues arise during testing, the project is returned to development, and the process starts over.

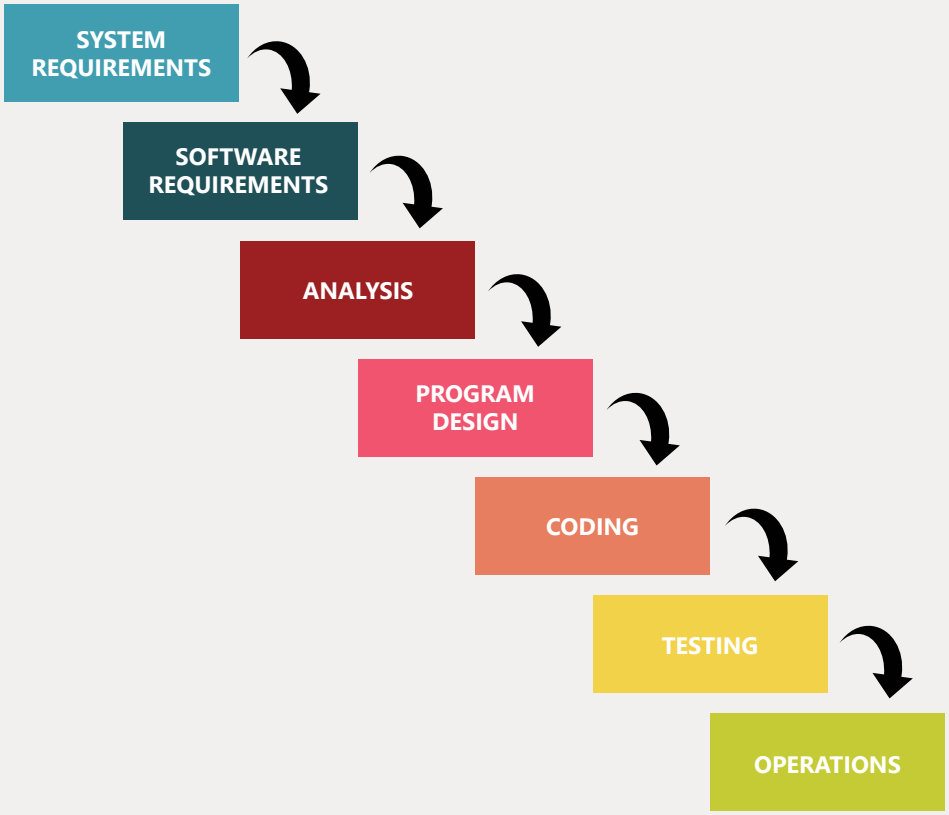


Figure 2: Pipeline stages of the Waterfall Model

[1] Winston W. Royce (1970). "Managing the Development of Large Software Systems" in: Technical Papers of Western Electronic Show and Convention (WesCon) August 25–28, 1970, Los Angeles, USA.

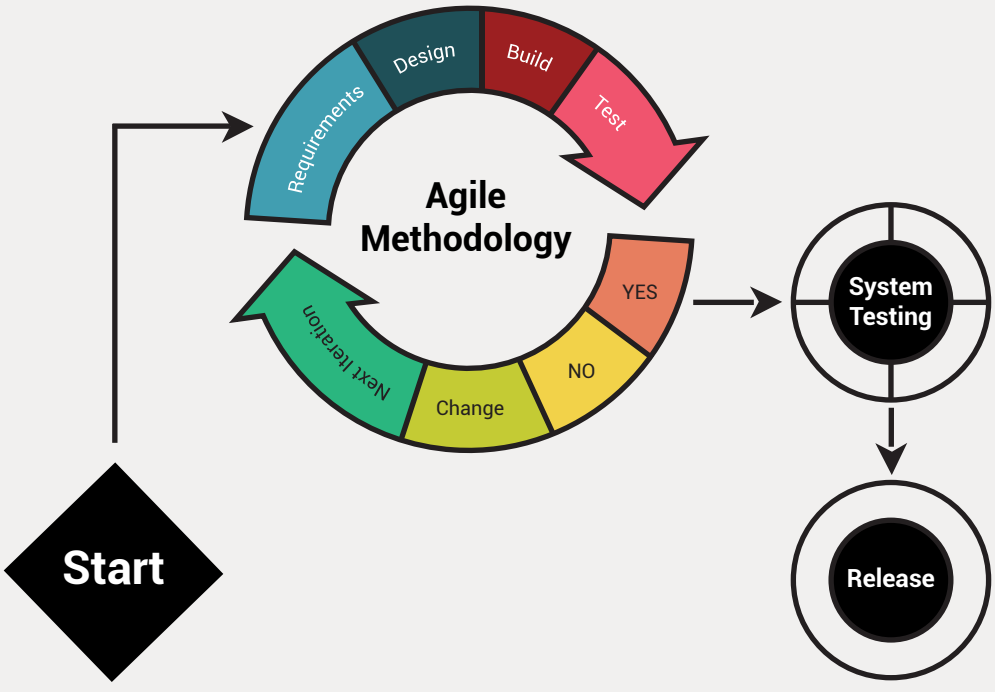
There is little collaboration among groups. Most traditional software development cycles tended to follow this method. Historically, large organizations with clearly defined hierarchical units favored Waterfall methodologies in order to trace project progress and accountability from requirements through design through coding, then test cases, then QA sign-off. Because it is a serial process that assumes requirements will be gathered up-front (and then change very little until the system is deployed), Waterfall development is slow to adapt to changes and may deliver software that no longer meets the needs of the business. The time from inception to the final product is typically at least six months, and often longer. The fact that requirements were rarely revisited after sign-off was one of the major fatal flaws in the Waterfall model.

Organizations of all sizes are now embracing collaborative methodologies that reduce siloization, deliver software faster and can adapt quickly to changes in requirements and technology. The following three methods are considered to be collaborative.

COLLABORATIVE METHOD 1: AGILE

Agile takes an approach that assumes that each project needs to be handled separately and that methods need to be tailored to meet the requirements. Within Agile, designers, developers, and test engineers work together as a single team from project design to project completion. Each team member is expected to have knowledge of what other team members do, so if for some reason someone needs to fall out of a project, others can quickly pick up the slack.

Work within the project is divided up into short pieces, known as sprints. In these sprints, all team members work together at once with clearly defined tasks and short-term goals with specifically defined end dates. Dividing work into sprints allows the requirements to change without losing months of effort and code. No work that is not clearly defined at the beginning of a sprint can be worked on, which helps prevent scope creep. If problems arise, they are put aside for a future sprint so that development proceeds to fulfill the goals of the current sprint and no more.



Agile methodology focuses on individuals and interactions to create cohesive teams, with a goal of creating working software that is testable along the way. Agile is designed to solve the communication problem between customers/ software requirements and developer and testing teams. Customers, or someone serving as the "voice of the customer," are included in the process to ensure that the product requirements are being met. Agile is iterative in that as individual pieces are created, it is easy to loop back and readdress issues without slowing the design/development process. As a result, organizations using the Agile method can adapt quickly to change and produce quality results in a short amount of time.

COLLABORATIVE METHOD 2: LEAN

The Lean framework takes an iterative approach to software development. Lean was initially developed in Japan by Toyota (as the Toyota Production System) as a way of providing quality with limited resources by flattening organizational structures. Individual parts of a project are divided into different streams, or kanbans, where work is handled simultaneously but separately. Workflow is managed in each of these streams by placing work-in-progress (WIP) limits to make sure that groups do not become overwhelmed. Lean, like most formal quality methods owing their heritage to Dr. W. Edwards Deming and J. M. Juran, emphasizes defect prevention through statistical process control. In contrast to Waterfall, the time from designing to release is significantly reduced. Lean production uses a "just-in-time" waste-reduction methodology that requires careful management of resources in order to have the correct amount when needed.

Another area where Lean differs from Waterfall is that individuals are made aware of each aspect of work within the stream. Participants within the kanbans are empowered to stop any activity if they spot any problems along the way to ensure an increase in quality before reaching a testing phase. Testing is considered an integral part of the process.

Lean shares some aspects of Agile within each stream, except it has the advantage of being able to separate into smaller groups of individuals who have specialized knowledge that is not easily shared with the larger group. In this way, individuals who can accomplish their work separately can avoid having to wait around while other sprints are waiting to be completed.

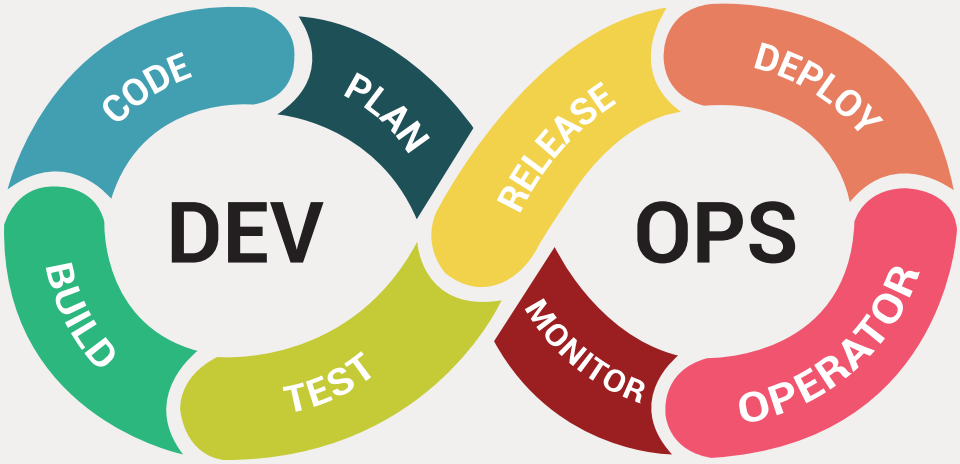
Kanbans also have the advantage of pairing within groups like-minded individuals who collaborate well with each other. For instance, introverted individuals who work well with each other can be kept on different streams than the extroverts.

COLLABORATIVE METHOD 3: DEVOPS

Whereas Agile focuses on improving the communications between the product requirements and design teams with developers and testers, DevOps focuses on improving the gap between developer/tester teams and operations/IT infrastructure. It is designed specifically toward managing end-to-end processes within IT departments. Like Agile, it focuses on adapting to change but also on constant testing and delivery of software products. DevOps does not have a clearly defined framework, but instead focuses primarily on collaboration itself.

DevOps uses the concept of value stream mapping, which identifies bottlenecks and aims to improve quality and efficiency throughout the entire software development process. It focuses on reducing friction between teams at every stage and handoff, not just between development and operations. This iterative process is never considered to

be complete; when one bottleneck is removed, improvements can be made to the next constraint. At any point, any issues that arise can be routed back through QA and development teams.



It is important to note that all three methods, Agile, Lean, and DevOps, have components that can be used interchangeably if need be.

WHAT DOES "IDEAL COLLABORATION" LOOK LIKE?

If we break down team relationships to a binary level, we can see how to identify and nurture good collaboration between teams.

DESIGNERS AND DEVELOPERS

Ideal collaboration between designers and developers begins with a deeper understanding of how the other team creates and manages their information. This reduces the risk of rework by delivering designs that meet usability and functional requirements and are technically feasible for the developers to implement.

With tools that enable designers to create wireframes, developers can easily translate UI designs and graphics to the development tools. Designers may use templates that define what sort of controls, functionalities and constraints exist so that they deliver models that work for the developers. Developers can then focus on choosing the best language to implement the designers' vision.

DEVELOPERS AND TEST ENGINEERS

In the DevOps model, both developers and test engineers are responsible for QA. Continuous iterative testing reduces friction, catches and resolves issues earlier in the development cycle, and enables the possibility of continuous delivery.

Automating unit tests with quality testing tools can improve collaboration between development and test teams while moving organizations closer to a continuous process. For manual tests that cannot be automated, well-developed testing frameworks will help improve the entire workflow and remove some of the possible bottlenecks. These testing frameworks often require the direct involvement of developers (sometimes called the "Software Development Engineer in Test," or SDET).

Good communication is essential to creating automated tests and testing frameworks that assess the right use cases at the right time. Test engineers can and should help the developers better understand the objectives of the software so that better tests can be written. By testing early, often, and with the knowledge of both developers and test engineers, quality is already in a considerably better state when it gets to QA.

DEV/TEST AND OPS

Working together, development and operations teams can create better unit tests that adhere to system specifications and limitations. While developers are well familiar with unit testing, quite often many ops teams may find this an alien concept. Bringing ops into the process will help them be better able to adapt to automated builds.

The processes used to improve and automate communications between developers and testers should also be used between dev/test and ops. If tools can communicate objectives clearly and beyond the testing process, much of the communication process itself can be automated.

Developers will learn what limitations are in the system and which configurations are necessary, and ops will know what resources to allocate and provision for optimal performance. As the process itself is iterative, this will, of course, need to be monitored and adjusted over time, but solid communication makes all of this easier.

TOOLS FOR BETTER COLLABORATION

The following are tools that can greatly improve communication and collaboration at all levels.

Project Management Tools

- JIRA – Issue-tracking and project management tool for working within Agile environments.
- ASANA – Work management platform.
- Slack – Chat and collaboration tool that enables setting up of individual channels of communication, team, and topic channels for real-time and asynchronous code-sharing, problem-solving, status updates, and more.

Designer/Developer Tools

- FMX Stencils – improves communication and collaboration between designers and developers by using a common language and control types. This reduces miscommunication and expedites prototyping and app development.
- GUI Templates – for RAD Studio can apply consistent theming to app UIs.
- Premium Styles – for RAD Studio can keep design consistent across different target devices while still tailoring to device expectations.
- Live On-Device Previews – in RAD Studio shorten the design/development feedback cycle.

Dev/Ops Tools

- DUnitX and DUnit – Unit test automation tools for RAD Studio assist in collaboration among dev/testing/ops
- Assembla – Secure enterprise-level source code and repository management.
- Ranorex – Automated GUI testing improves app quality and reduces friction between development and test teams.
- RAD Studio – From Embarcadero, RAD Studio is a common IDE for both C++ and Delphi developers.

DISCUSSION/CHALLENGES

The most significant obstacle to adopting a collaborative SDLC methodology is not the technical tool implementation but effective culture change at the team and organizational level. Employees have different personalities and learning and work types and may resist change from familiar processes, so care should be taken to create a safe environment where issues and problems can be raised honestly and quickly and then solved.

In order to jump start collaborative work, many organizations will change their physical office layouts from private offices and cubicles to open work environments. People of similar personality types may adapt well to this environment if placed together. Introverts may work fine with one another as they tend to be focused on the task at hand. However, while extroverts may find some of the best benefits with being able to bounce ideas off of others on a team, they can become easily distracted and disrupt those introverts. Conflict could become problematic. Changing workflow can cause communication problems itself. One possible solution is to use a Lean/Kanban approach within these situations to place people in like-minded streams.

The adjustment period to Agile and DevOps can appear sluggish as teams may first experience reduced productivity until they are comfortable with new tools, processes, and increased empowerment to take the lead on problem-solving. Change feels different, and it takes time for people to adapt to the new collaborative workflow. Improvement typically appears after the first few months, and productivity soars far beyond what it ever had been under the previous system.

When compared with previous models, the cost-benefit analysis should strongly push any organization toward collaborative workflows. This is especially true when one looks at the risk of project failure that accompanied older systems. Being able to provide better customer satisfaction, increase speed to market and also conceivably deliver a better work environment make this decision obvious.

EFFICIENT, SECURE COLLABORATION WITH EMBARCADERO

Embarcadero products—C++ Builder, RAD Studio, Delphi—are built to help enterprise software development teams collaborate efficiently to build and deploy amazing software.

Embarcadero's FMX Stencils, FMX GUI templates, and premium styles enable designers and developers to work closely together with fewer errors between design vision and final app user interfaces.

RAD Studio offers organizations the opportunity to blend C++ and Delphi development expertise in the same project, which means developers can work in their preferred programming language, access a wide variety of libraries in both languages, contribute to the greater project, and compile using the same tool sets. RAD Studio offers several tools to help developers navigate and manage code changes. The Difference and Merge viewer quickly locates differences in the code so developers can identify and understand what their coworkers have added. Integration with popular

source version control systems—including Subversion, Git, and Mercurial—mean test and operations teams have greater confidence in development's commits.

RAD Studio includes command line compilers that are ideal for integrating into continuous build configurations quickly using MSBuild or CMake. External tools such as Hudson and Jenkins, GUI testing with Ranorex, and integrated unit testing powered by DUnitX/DUnit provide a robust foundation for continuous integration. ***In most cases the native applications built with Delphi and C++Builder have limited or no dependencies on Java VM, .NET, or any other specific language runtime. With an installation model close to "x-copy deployment," the cooperation between developers and deployment/installation teams becomes significantly simpler.***

Whether you're just getting started on the journey toward collaborative SDLC methodologies or your organization has fully adopted DevOps, Embarcadero development tools provide the foundation for smart integrated teams to design, code, and deploy better software.

TRY IT FOR YOURSELF **BY DOWNLOADING**
A FULL-FUNCTIONED,
FREE 30-DAY TRIAL OF RAD STUDIO
(NO CREDIT CARD REQUIRED).

About Embarcadero: Embarcadero empowers application developers with tools that solve productivity and go-to-market problems. With Embarcadero's technologies, developers can design, create and deploy applications for all platforms from a single codebase, saving considerable time, money and frustrations.

An overview of Embarcadero products is available at
https://www.embarcadero.com/products

embarcadero®

An Idera, Inc. Company