**What is Verification?**

The verifying process includes checking documents, design, code, and program.

**What is Validation?**

Validation is a dynamic mechanism of Software testing and validates the actual product.

**Verification vs Validation: Key Difference**

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| * The verifying process includes checking documents, design, code, and program | * It is a dynamic mechanism of testing and validating the actual product |
| * It does ***not*** involve executing the code | * It always involves executing the code |
| * Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc. | * It uses methods like Black Box Testing, [White Box Testing](https://www.guru99.com/white-box-testing.html), and non-functional testing |
| * Whether the software conforms to specification is checked | * It checks whether the software meets the requirements and expectations of a customer |
| * It finds bugs early in the development cycle | * It can find bugs that the verification process can not catch |
| * Target is application and software architecture, specification, complete design, high level, and database design etc. | * Target is an actual product |
| * QA team does verification and make sure that the software is as per the requirement in the SRS document. | * With the involvement of testing team validation is executed on software code. |
| * It comes before validation | * It comes after verification |

**What is Verification?**

***Definition****: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.*

Verification is a static practice of verifying documents, design, code and program. It includes all the activities associated with producing high quality software: inspection, design analysis and specification analysis. It is a relatively objective process.

Verification will help to determine whether the software is of high quality, but it will not ensure that the system is useful. Verification is concerned with whether the system is well-engineered and error-free.

***Methods of Verification :***[***Static Testing***](http://toolsqa.com/software-testing/static-testing/)

* *Walkthrough*
* *Inspection*
* *Review*

**What is Validation?**

***Definition****: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.*

Validation is the process of evaluating the final product to check whether the software meets the customer expectations and requirements. It is a dynamic mechanism of validating and testing the actual product.

***Methods of Validation :***[***Dynamic Testing***](http://toolsqa.com/software-testing/dynamic-testing/)

* *Testing*
* *End Users*

**Difference between Verification and Validation**

The distinction between the two terms is largely to do with the role of specifications.

***Validation****is the process of checking whether the specification captures the customer’s needs. “Did I build what I said I would?”*

***Verification****is the process of checking that the software meets the specification.  “Did I build what I need?”*

|  |  |
| --- | --- |
| ***Verification*** | ***Validation*** |
| *1.***Verification** is a static practice of verifying documents, design, code and program. | 1. **Validation** is a dynamic mechanism of validating and testing the actual product. |
| *2.*It does not involve executing the code. | 2. It always involves executing the code. |
| *3.*It is human based checking of documents and files. | *3.*It is computer based execution of program. |
| *4.***Verification** uses methods like inspections, reviews, walkthroughs, and Desk-checking etc. | *4.***Validation** uses methods like black box (functional)  testing, gray box testing, and white box (structural) testing etc. |
| 5. **Verification**is to check whether the software conforms to specifications. | 5. **Validation** is to check whether software meets the customer expectations and requirements. |
| *6.*It can catch errors that validation cannot catch. It is low level exercise. | *6. It can catch errors that verification cannot catch. It is High Level Exercise.* |
| 7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc. | *7.*Target is actual product-a unit, a module, a bent of integrated modules, and effective final product. |
| 8. **Verification** is done by QA team to ensure that the software is as per the specifications in the SRS document. | *8.****Validation****is carried out with the involvement of testing team.* |
| 9. It generally comes first-done before validation. | *9.*It generally follows after **verification**. |

# **Differences between Verification and Validation**

Prerequisite – [Verification and Validation](https://www.geeksforgeeks.org/software-engineering-verification-and-validation/)  
**Verification** is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.  
Verification means **Are we building the product right?**

**Validation** is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.  
Validation means **Are we building the right product?**

The difference between Verification and Validation is as follow:

| **VERIFICATION** | **VALIDATION** |
| --- | --- |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does not include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. | Validation is executed on software code with the help of testing team. |
| It comes before validation. | It comes after verification. |

# **What is SDLC? Understand the Software Development Life Cycle**

STACKIFYAPRIL 6, 2017[DEVELOPER TIPS, TRICKS & RESOURCES](https://stackify.com/developers/)

Streamlined development relies on a consistent methodology and a clearly-defined process from getting from point A to point B. If you’re just getting your feet wet in the wide world of development, you need to understand the Software Development Life Cycle or SDLC.

## Definition of SDLC

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time. SDLC includes a detailed plan for how to develop, alter, maintain, and replace a software system.

SDLC involves several distinct stages, including planning, design, building, testing, and deployment. Popular SDLC models include the [waterfall model](http://learnaccessvba.com/application_development/waterfall_method.htm), [spiral model](http://searchsoftwarequality.techtarget.com/definition/spiral-model), and [Agile model](http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/).

## How SDLC Works

SDLC works by lowering the cost of software development while simultaneously improving quality and shortening production time. SDLC achieves these apparently divergent goals by following a plan that removes the typical pitfalls to software development projects. That plan starts by evaluating existing systems for deficiencies. Next, it defines the requirements of the new system. It then creates the software through the stages of design, development, testing, and deployment. By anticipating costly mistakes like failing to ask the end user for suggestions, SLDC can eliminate redundant rework and after-the-fact fixes.

## Stages and Best Practices of SDLC

Following the best practices and/or stages of SDLC ensures the process works in a smooth, efficient, and productive way.

1. **Identify the**[**current problems**](https://stackify.com/sdlc-phases-identify-problems/)**.**“What don’t we want?” This stage of SDLC means getting input from all stakeholders, including customers, salespeople, industry experts, and programmers. Learn the strengths and weaknesses of the current system with improvement as the goal.
2. **Plan.** “What do we want?” In this stage of SDLC, the team defines the requirements of the new software and determines the cost and resources required. It also details the risks involved and provides sub-plans for softening those risks. In this stage, a Software Requirement Specification document is created.
3. **Design.** “How will we get what we want?” This phase of SDLC starts by turning the software specifications into a design plan called the Design Specification. All stakeholders then review this plan and offer feedback and suggestions. It’s crucial to have a plan for collecting and incorporating stakeholder input into this document. Failure at this stage will almost certainly result in cost overruns at best and total collapse of the project at worst.
4. **Build.** “Let’s create what we want.” This SDLC stage develops the software by generating all the actual code. If the previous steps have been followed with attention to detail, this is actually the least complicated step.
5. **Test.** “Did we get what we want?” In this stage, we test for defects and deficiencies. We fix those issues until the product meets the original specifications.
6. **Deploy.** “Let’s start using what we got.” Often, this part of the SDLC process happens in a limited way at first. Depending on feedback from end users, more adjustments can be made.
7. **Maintain.** “Let’s get this closer to what we want.” The plan almost never turns out perfect when it meets reality. Further, as conditions in the real world change, we need to update and advance the software to match.

The [DevOps movement](https://stackify.com/what-is-devops/) has changed the SDLC in some ways. Developers are now responsible for more and more steps of the entire development process. We also see the value of shifting left. When development and Ops teams use the same toolset to track performance and pin down defects from inception to the retirement of an application, this provides a common language and faster handoffs between teams. APM tools can be used in development, QA, and production. This keeps everyone using the same toolset across the entire development lifecycle.

Read More: [3 Reasons Why APM Usage is Shifting Left to Development & QA](https://stackify.com/apm-shifting-left-development-qa/)

## Examples of SDLC in Action

The most common SDLC examples or SDLC models are listed below.

* **Waterfall Model.**This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its own mini-plan and each phase “waterfalls” into the next. The biggest drawback of this model is that small details left incomplete can hold up the entire process.
* **Agile Model.**The Agile SDLC model separates the product into cycles and delivers a working product very quickly. This methodology produces a succession of releases. Testing of each release feeds back info that’s incorporated into the next version. [According to Robert Half](https://www.roberthalf.com/technology/blog/6-basic-sdlc-methodologies-the-pros-and-cons), the drawback of this model is that the heavy emphasis on customer interaction can lead the project in the wrong direction in some cases.
* **Iterative Model.**This SDLC model emphasizes repetition. Developers create a version very quickly and for relatively little cost, then test and improve it through rapid and successive versions. One big disadvantage here is that it can eat up resources fast if left unchecked.
* **V-Shaped Model.**An extension of the waterfall model, this SDLC methodology tests at each stage of development. As with waterfall, this process can run into roadblocks.
* **Big Bang Model.**This high-risk SDLC model throws most of its resources at development and works best for small projects. It lacks the thorough requirements definition stage of the other methods.
* **Spiral Model.**The most flexible of the SDLC models, the spiral model is similar to the iterative model in its emphasis on repetition. The spiral model goes through the planning, design, build and test [phases](https://stackify.com/sdlc-phases-identify-problems/) over and over, with gradual improvements at each pass.

## Benefits of SDLC

SDLC done right can allow the highest level of management control and documentation. Developers understand what they should build and why. All parties agree on the goal up front and see a clear plan for arriving at that goal. Everyone understands the costs and resources required.

Several pitfalls can turn an SDLC implementation into more of a roadblock to development than a tool that helps us. Failure to take into account the needs of customers and all users and stakeholders can result in a poor understanding of the system requirements at the outset. The benefits of SDLC only exist if the plan is followed faithfully.

The Software Development Life Cycle (SDLC) is the software development world’s spellcheck — it can flag errors in software creation before they’re discovered (at a much higher cost) in successive stages. But it’s much more than that, of course: SDLC can also lay out a plan for getting everything right the first time.

The SDLC process involves several distinct stages, including planning, analysis, design, building, testing, deployment and maintenance. What's the best SDLC methodology? Here are six methodologies, or models, to consider.

**Software development lifecycle methodologies**

* Agile
* Lean
* Waterfall
* Iterative
* Spiral
* DevOps

Each of these approaches varies in some ways from the others, but all have a common purpose: to help teams deliver high-quality software as quickly and cost-effectively as possible.

Reviewing a brief description of the six most common SDLC methodologies may help you decide which is best for your team:

**1. Agile**

The Agile model has been around for about a decade. But lately, it has become a [major driving force behind software development](https://sdtimes.com/agile-is-becoming-the-de-facto-standard-for-software-development-according-to-versionone/) in many organizations. Some businesses value the Agile methodology so much that they are now applying it to other types of projects, including non-tech initiatives.

In the Agile model, “fast failure” is a good thing. The approach produces ongoing release cycles, each featuring small, incremental changes from the previous release. At each iteration, the product is tested. The Agile model helps teams identify and address small issues on projects before they evolve into more significant problems, and engage business stakeholders and get their feedback throughout the development process.

As part of their embrace of this methodology, many teams are also applying an Agile framework known as Scrum to help structure more complex development projects. Scrum teams work in “sprints,” which usually last two to four weeks, to complete assigned tasks. Daily Scrum meetings help the whole team monitor progress throughout the project. And the ScrumMaster is tasked with keeping the team focused on its goal. (For more on Scrum, see the [Scrum Alliance website](https://www.scrumalliance.org/).)

**2. Lean**

The Lean model for software development is inspired by lean manufacturing practices and principles. The seven Lean principles (in this order) are: eliminate waste, amplify learning, decide as late possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.

The Lean process is about working only on what must be worked on at the time, so there’s no room for multitasking. Project teams are also focused on finding opportunities to cut waste at every turn throughout the SDLC process, from dropping unnecessary meetings to reducing documentation.

The Agile model is actually a Lean method for the SDLC, but with some notable differences. One is how each prioritizes customer satisfaction: Agile makes it the top priority from the outset, creating a flexible process where project teams can respond quickly to stakeholder feedback throughout the SDLC. Lean, meanwhile, emphasizes the elimination of waste as a way to create more overall value for customers — which, in turn, helps to enhance satisfaction.

***Ready to fill an open role?***[***Use our candidate finder.***](https://www.roberthalf.com/browse-candidates)

**3. Waterfall**

Some experts argue that the Waterfall model was never meant to be a process model for real projects (check out the discussion on this topic on [StackExchange](https://stackexchange.com/)). Regardless, the Waterfall model is widely considered the oldest of the structured SDLC methodologies. It’s also a very straightforward approach: finish one phase, then move on to the next. No going back. Each stage relies on information from the previous stage and has its own project plan.

The downside of Waterfall is its rigidity. Sure, it’s easy to understand and simple to manage. But early delays can throw off the entire project timeline. With little room for revisions once a stage is completed, problems can’t be fixed until you get to the maintenance stage. This model doesn’t work well if flexibility is needed or if the project is long term and ongoing.

Even more rigid is the related Verification and Validation model — or V-shaped model. This linear development methodology sprang from the Waterfall approach. It’s characterized by a corresponding testing phase for each development stage. Like Waterfall, each stage begins only after the previous one has ended. This SDLC model can be useful, provided your project has no unknown requirements.

**4. Iterative**

The Iterative model is repetition incarnate. Instead of starting with fully known requirements, project teams implement a set of software requirements, then test, evaluate and pinpoint further requirements. A new version of the software is produced with each phase, or iteration. Rinse and repeat until the complete system is ready.

Advantages of the Iterative model over other common SDLC methodologies is that it produces a working version of the project early in the process, and makes it less expensive to implement changes. One disadvantage: Repetitive processes can consume resources quickly.

One example of an Iterative model is the Rational Unified Process (RUP), developed by IBM’s Rational Software division. As explained in this [document from IBM](https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf), RUP is a “process product” designed to enhance team productivity that also “captures many of the best practices in modern software development in a form that is suitable for a wide range of projects and organizations.”

RUP divides the development process into four phases: inception, when the idea for a project is set; elaboration, when the project is further defined, and resources are evaluated; construction, when the project is developed and completed; and transition, when the product is released. Each phase of the project involves business modeling, analysis and design, implementation, testing, and deployment.

**5. Spiral**

One of the most flexible SDLC methodologies, the Spiral model takes a cue from the Iterative model and its repetition; the project passes through four phases (planning, risk analysis, engineering and evaluation) over and over in a “spiral” until completed, allowing for multiple rounds of refinement.

The Spiral model is typically used for large projects. It enables development teams to build a highly customized product, and incorporate user feedback early on in the project. Another benefit of this SDLC model is risk management. Each iteration starts by looking ahead to potential risks, and figuring out how best to avoid or mitigate them.

**6. DevOps**

The DevOps methodology is the newcomer to the SDLC scene. As [this article explains](https://theagileadmin.com/what-is-devops/), it emerged from two trends: the application of Agile and Lean practices to operations work, and the general shift in business toward seeing the value of collaboration between development and operations staff at all stages of the SDLC process.

In a DevOps model, Developers and Operations teams work together closely — and sometimes as one team — to accelerate innovation and the deployment of higher-quality and more reliable software products and functionalities. Updates to products are small but frequent. Discipline, continuous feedback and process improvement, and automation of manual development processes are all hallmarks of the DevOps model.

Amazon Web Services [describes DevOps](https://aws.amazon.com/devops/what-is-devops/) like this: “DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.” So, like many SDLC models, DevOps is not only an approach to planning and executing work, but also a philosophy that demands significant mindset and culture changes in an organization.

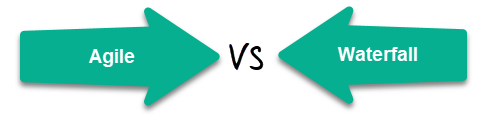
Choosing the right SDLC model for your software development project will require careful thought. But keep in mind that a methodology for planning and guiding your project is only one ingredient for success. Even more important is assembling a solid team of skilled talent committed to moving the project forward through every unexpected challenge or setback.

**What is Waterfall methodology?**

Waterfall Model methodology which is also known as Liner Sequential Life Cycle Model. Waterfall Model followed in the sequential order, and so project development team only moves to next phase of development or testing if the previous step completed successfully.

**What is the Agile methodology?**

Agile methodology is a practice that helps continuous iteration of development and testing in the software development process. In this model, development and testing activities are concurrent, unlike the Waterfall model. This process allows more communication between customers, developers, managers, and testers.

[](https://www.guru99.com/images/1/111517_1049_WaterfallVs1.png)

**Advantages of Waterfall Model:**

* It is one the easiest model to manage. Because of its nature, each phase has specific deliverables and a review process.
* It works well for smaller size projects where requirements are easily understandable.
* Faster delivery of the project
* Process and results are well documented.
* Easily adaptable method for shifting teams
* This project management methodology is beneficial to manage dependencies.

**Advantages of the Agile Model:**

* It is focused client process. So, it makes sure that the client is continuously involved during every stage.
* Agile teams are extremely motivated and self-organized so it likely to provide a better result from the development projects.
* Agile software development method assures that quality of the development is maintained
* The process is completely based on the incremental progress. Therefore, the client and team know exactly what is complete and what is not. This reduces risk in the development process.

**Limitations of Waterfall Model:**

* It is not an ideal model for a large size project
* If the requirement is not clear at the beginning, it is a less effective method.
* Very difficult to move back to makes changes in the previous phases.
* The testing process starts once development is over. Hence, it has high chances of bugs to be found later in development where they are expensive to fix.

**Limitations of Agile Model**

* It is not useful method for small development projects.
* It requires an expert to take important decisions in the meeting.
* Cost of implementing an agile method is little more compared to other development methodologies.
* The project can easily go off track if the project manager is not clear what outcome he/she wants.

**Difference between Agile and Waterfall Model:**

|  |  |
| --- | --- |
| **Agile** | **Waterfall** |
| It separates the project development lifecycle into sprints. | Software development process is divided into distinct phases. |
| It follows an incremental approach | Waterfall methodology is a sequential design process. |
| Agile methodology is known for its flexibility. | Waterfall is a structured software development methodology so most times it can be quite rigid. |
| Agile can be considered as a collection of many different projects. | Software development will be completed as one single project. |
| Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed. | There is no scope of changing the requirements once the project development starts. |
| Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once. | All the project development phases like designing, development, testing, etc. are completed once in the Waterfall model. |
| Test plan is reviewed after each sprint | The test plan is rarely discussed during the test phase. |
| Agile development is a process in which the requirements are expected to change and evolve. | The method is ideal for projects which have definite requirements and changes not at all expected. |
| In Agile methodology, testing is performed concurrently with software development. | In this methodology, the "Testing" phase comes after the "Build" phase |
| Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer's demands. | This model shows a project mindset and places its focus completely on accomplishing the project. |
| Agile methdology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios. | Reduces risk in the firm fixed price contracts by getting risk agreement at the beginning of the process. |
| Prefers small but dedicated teams with a high degree of coordination and synchronization. | Team coordination/synchronization is very limited. |
| Products owner with team prepares requirements just about every day during a project. | Business analysis prepares requirements before the beginning of the project. |
| Test team can take part in the requirements change without problems. | It is difficult for the test to initiate any change in requirements. |
| Description of project details can be altered anytime during the SDLC process. | Detail description needs to implement waterfall software development approach. |
| The Agile Team members are interchangeable, as a result, they work faster. There is also no need for project managers because the projects are managed by the entire team | In the waterfall method, the process is always straightforward so, project manager plays an essential role during every stage of SDLC. |

**Conclusion:**

Agile and Waterfall are very different software development methodologies and are good in their respective way.

However, there are certain major differences highlighted below -

* Waterfall model is ideal for projects which have defined requirements, and no changes are expected. On the other hand, Agile is best suited where there is a higher chance of frequent requirement changes.
* The waterfall is easy to manage, sequential, and rigid method.
* Agile is very flexible and it possible to make changes in any phase.
* In Agile process, requirements can change frequently. However, in a waterfall model, it is defined only once by the business analyst.
* In Agile Description of project, details can be altered anytime during the SDLC process which is not possible in Waterfall method.

One of the first decisions we face for each of our project implementations at Segue is “Which development methodology should we use?” This is a topic that gets a lot of discussion (and often heated debate). If this is not something you’ve worked with before, a definition of development methodology is in order; put very simply, it’s a way of organizing the work of software development. This is **NOT** about a style of project management or a specific technical approach, although you will often hear these terms all thrown together or used interchangeably.

The two basic, most popular methodologies are:

1. *Waterfall:* (ugh, terrible name!), which might be more properly called the “traditional” approach, and
2. *Agile:*a specific type of Rapid Application Development and newer than Waterfall, but not that new, which is often implemented using Scrum.

Both of these are usable, mature methodologies. Having been involved in software development projects for a long time, here are my thoughts on the strengths and weaknesses of each.

The Waterfall Methodology

***Waterfall***is a linear approach to software development. In this methodology, the sequence of events is something like:

1. Gather and document requirements
2. Design
3. Code and unit test
4. Perform system testing
5. Perform user acceptance testing (UAT)
6. Fix any issues
7. Deliver the finished product

In a **true** Waterfall development project, each of these represents a distinct stage of software development, and each stage generally finishes before the next one can begin. There is also typically a stage gate between each; for example, requirements must be reviewed and approved by the customer before design can begin.

There are good things and bad about the Waterfall approach. On the **positive** side:

* Developers and customers agree on what will be delivered early in the development lifecycle. This makes planning and designing more straightforward.
* Progress is more easily measured, as the full scope of the work is known in advance.
* Throughout the development effort, it’s possible for various members of the team to be involved or to continue with other work, depending on the active phase of the project. For example, business analysts can learn about and document what needs to be done, while the developers are working on other projects. Testers can prepare test scripts from requirements documentation while coding is underway.
* Except for reviews, approvals, status meetings, etc., a customer presence is not strictly required after the requirements phase.
* Because design is completed early in the development lifecycle, this approach lends itself to projects where multiple software components must be designed (sometimes in parallel) for integration with external systems.
* Finally, the software can be designed completely and more carefully, based upon a more complete understanding of **all** software deliverables. This provides a better software design with less likelihood of the “piecemeal effect,” a development phenomenon that can occur as pieces of code are defined and subsequently added to an application where they may or may not fit well.

Here are some **issues** we have encountered using a pure Waterfall approach:

* One area which almost always falls short is the effectiveness of [requirements](https://www.seguetech.com/software-requirements-development/). Gathering and documenting requirements in a way that is meaningful to a customer is often the most difficult part of software development, in my opinion. Customers are sometimes intimidated by details, and specific details, provided early in the project, are required with this approach. In addition, customers are not always able to visualize an application from a requirements document. [Wireframes](https://www.seguetech.com/blog/2013/05/09website-wireframes-do-you-need-them) and mockups can help, but there’s no question that most end users have some difficulty putting these elements together with written requirements to arrive at a good picture of what they will be getting.
* Another potential drawback of pure Waterfall development is the possibility that the customer will be dissatisfied with their delivered software product. As all deliverables are based upon documented requirements, a customer may not see what will be delivered until it’s almost finished. By that time, changes can be difficult (and costly) to implement.

The Agile Methodology

[**Agile**](https://www.seguetech.com/blog/2013/04/05/what-is-agile-software-development) is an iterative, team-based approach to development. This approach emphasizes the rapid delivery of an application in complete functional components. Rather than creating tasks and schedules, all time is “time-boxed” into phases called “sprints.” Each sprint has a defined duration (usually in weeks) with a running list of deliverables, planned at the start of the sprint. Deliverables are prioritized by business value as determined by the customer. If all planned work for the sprint cannot be completed, work is reprioritized and the information is used for future sprint planning.

As work is completed, it can be reviewed and evaluated by the project team and customer, through daily builds and end-of-sprint demos. Agile relies on a very high level of customer involvement throughout the project, but especially during these reviews.

Some**advantages of the Agile approach**are easy to see:

* The customer has frequent and early opportunities to see the work being delivered, and to make decisions and changes throughout the development project.
* The customer gains a strong sense of ownership by working extensively and directly with the project team throughout the project.
* If time to market for a specific application is a greater concern than releasing a full feature set at initial launch, Agile can more quickly produce a basic version of working software which can be built upon in successive iterations.
* Development is often more user-focused, likely a result of more and frequent direction from the customer.
* For more Agile Development benefits, please see [8 Benefits of Agile Software Development](https://www.seguetech.com/blog/2013/04/12/8-benefits-of-agile-software-development)

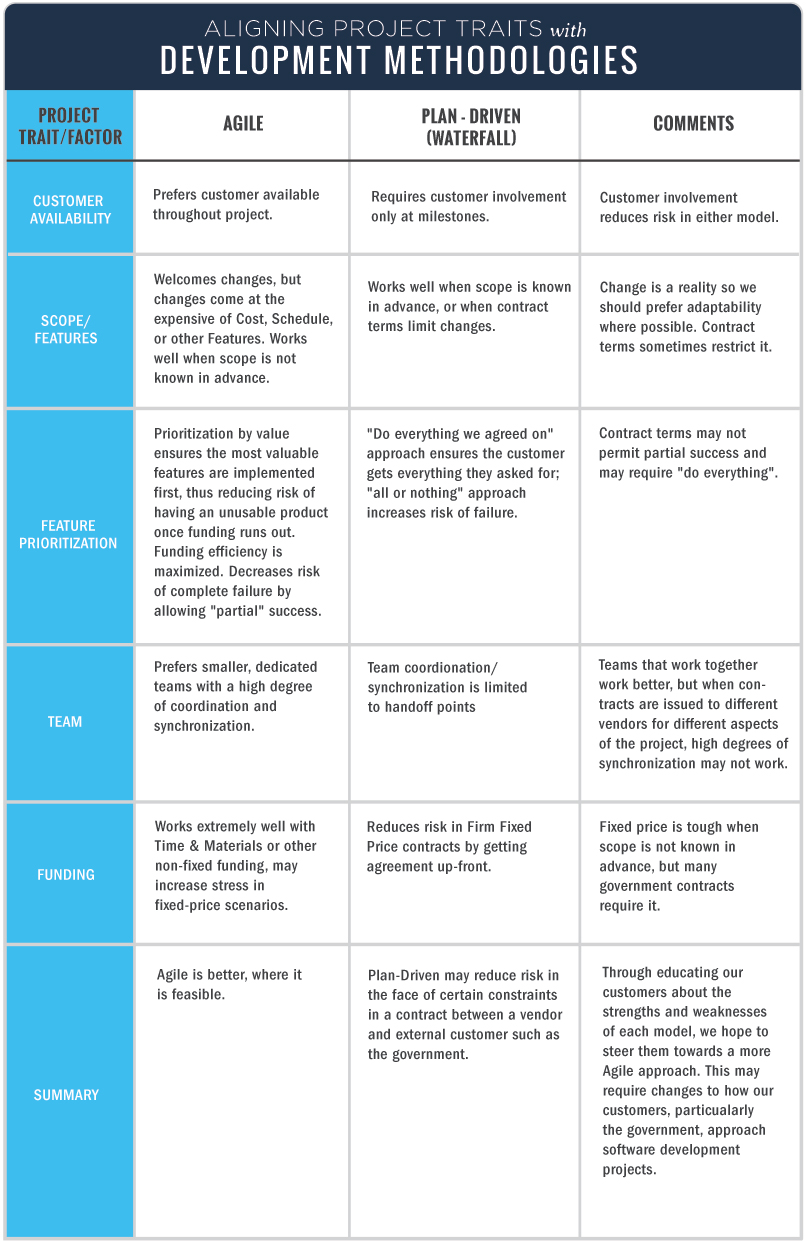
And, of course, there are **some disadvantages**:

* The very high degree of customer involvement, while great for the project, may present problems for some customers who simply may not have the time or interest for this type of participation.
* Agile works best when members of the development team are completely dedicated to the project.
* Because Agile focuses on time-boxed delivery and frequent reprioritization, it’s possible that some items set for delivery will not be completed within the allotted timeframe. Additional sprints (beyond those initially planned) may be needed, adding to the project cost. In addition, customer involvement often leads to additional features requested throughout the project. Again, this can add to the overall time and cost of the implementation.
* The close working relationships in an Agile project are easiest to manage when the team members are located in the same physical space, which is not always possible. However, there are a variety of ways to handle this issue, such as webcams, collaboration tools, etc.
* The iterative nature of Agile development may lead to a frequent refactoring if the full scope of the system is not considered in the intial architecture and design. Without this refactoring, the system can suffer from a reduction in overall quality. This becomes more pronounced in larger-scale implementations, or with systems that include a high level of integration.

Making the Choice Between Agile and Waterfall

So, how do we choose? First, we change the game a little (which is what most software development organizations do) by defining our own process. At Segue, it’s called our ***Process Framework***, and it’s a variation on the traditional Waterfall methodology. Our modifications include use of prototyping where possible to provide the customer a better view of their finished product early in the design/development cycle. This helps to improve the team’s understanding of requirements and communication with the customer. After the primary framework of the application is completed per high level requirements, we continue to develop and also to reach out to the customer for refinement of requirements. In this way, we strive to be as iterative as possible without compromising our overall system architecture.

We consider the following factors when considering which methodology to use:



# 10 Differences Between Agile and Waterfall Methodology

The traditional waterfall methodology for software development is rapidly losing its popularity as Agile methodology is now being increasingly adopted by companies worldwide for software development.

Waterfall basically is a sequential model where software development is segregated into a sequence of pre -defined phases – including feasibility, planning, design, build, test, production, and support. On the other hand, Agile development methodology follows a linear sequential approach while providing flexibility for changing project requirements, as they occur.

Here are the top 10 differences between Agile and Waterfall Methodology:

1. The software development process is divided into different phases in the Waterfall model while Agile methodology segregates the project development lifecycle into sprints
2. Waterfall is a structured software development methodology, and often times can be quite rigid, whereas the Agile methodology is known for its flexibility
3. According to the Waterfall model, software development is to be completed as one single project, which is then divided into different phases, with each phase appearing only once during the SDLC. However, the Agile methodology can be considered as a collection of many different projects, which are nothing but the iterations of the different phases focusing on improving the overall software quality with feedbacks from users or the QA team
4. If you want to use the Waterfall model for software development, then you have to be clear with all the development requirements beforehand as there is no scope of changing the requirements once the project development starts. The Agile methodology, on the other hand, is quite flexible, and allows for changes to be made in the project development requirements even after the initial planning has been completed
5. All the project development phases such as designing, development, testing, etc. are completed once in the Waterfall model while as part of the Agile methodology, they follow an iterative development approach. As a result, planning, development, prototyping and other software development phases can appear more than once during the entire SDLC
6. One of the major differences between Agile and Waterfall development methodology is their individual approach towards quality and testing. In the Waterfall model, the “Testing” phase comes after the “Build” phase, but, in the Agile methodology, testing is typically performed concurrently with programming or at least in the same iteration as programming
7. While Waterfall methodology is an internal process and does not require the participation of customers, the Agile software development approach focuses on customer satisfaction and thus, involves the participation of customers throughout the development phase
8. The Waterfall model can be regarded as a stringently sequential process, however, the Agile methodology is a highly collaborative software development process, thereby leading to better team input and faster problem solving
9. The Waterfall model is best suited for projects which have clearly defined requirements and in which change is not expected at all, while Agile development supports a process in which the requirements are expected to change and evolve. Thus, if you are planning to develop a software that would require frequent overhauls and has to keep up with the technology landscape and customer requirements, Agile is the best approach to follow
10. The Waterfall model exhibits a project mindset and lays its focus strictly on the completion of project development, while Agile introduces a product mindset that focuses on ensuring that the developed product satisfies its end customers, and changes itself as the requisites of customers change

**1. What are the most important values you demonstrate as a leader?**

The most important value that I have is my integrity. I demonstrate honesty and trust in all my actions to establish credibility as a leader. By having this conviction behind my words and actions, those who I lead are gain bought into the direction I take them.  
  
**2.How have you gained commitment from your team?**

I gain commitment from my teams by influencing and persuading them to set specific objectives and also buy into the process. Once they have established cooperation and cohesion, they are on board to attain the goal.

**3.How can a leader fail?  Give an example of that.**

A leader can fail when they can’t get their team on board with the goals of the organization. Factors outside of a leader’s control may also lead to failures such as available resources, time constraints, and the economy.

In the example you give, make sure that you talk about how dealt with a difficult challenge and how you analyzed the setback. Make sure you explain how to seek honest feedback to ensure that you learned from the failure.

**4. What is the difference between a Team leader and a Team manager?**

A manger is able to handle tasks and responsibilities and ensure that others get their work done. A leader will inspire and motivate their team to achieve their goals.

**5. What is your greatest strength?**

Being able to lead and inspire a team to perform their best and strive to achieve goals. I can do this through relationship building, being passionate about the goals, and influencing those around me.

**6. What would be your greatest weakness?**

When I delegate duties to others that I know I can do better.  However, if I don’t delegate, then I could end up with more work than I can handle myself. I’ve taken courses in time management and learned how to effectively handle tasks to overcome this weakness.

**7. How do you get others to accept your ideas?**

I talk about the benefits of the idea and how to apply it. I would stay open to other thoughts and change my ideas in a way that we can all agree. When you gain buy-in from others, you are much more successful in attaining the goals than when you make it mandatory to follow procedure.

**8. How would you go about praising a team member in public?**

I would use a time when we would be gathered in a group, such as a meeting to bring up the praise to the team member. I would recognize their success in front of the group so others could also learn best practices.

**9. Are you more effective in a group or one on one basis?**

I feel that I am more effective in a group because everyone has some unique quality that they bring to a group. We can develop our interpersonal skills by helping those in the group who need it as well as learning from those who are successful.

**10. How often do you feel it’s necessary to meet with your team?**

I feel that I should meet with my team at least once a week on a set time and day of the week. Communication among team members is critical, and this will give the team an opportunity to get together on a regular basis and talk about their challenges and best practices. Also, when our team reaches a milestone, a new project begins, an award or promotion is given, or when there is a challenging situation, I would want to bring the team together. Everyone will get the same message that way, and we can celebrate successes or come together in challenging times.

**11. Describe a time you took a leadership position when you did not have the title of a leader.**

In this question, take an example from a situation where you were in a group and took responsibility to delegate to achieve goals. Show how you gained from the other members to follow your lead and the result of your leadership.

For instance, in college, we were put into groups of four to complete a marketing project. We had to prepare a 15-page paper and 10-minute presentation on a new product. We want to introduce that outside the U.S. I took the initiative among the group to lead a discussion on how we should split up the work when we meet throughout the semester and deadlines for each person’s part of the work. Because I was the one to take the lead the discussion and had a plan in mind, I gained the buy in of the other members quickly. I took everyone’s e-mail address and created a group email to help us all keep track of our progress and so we could help each other outside of class and our meetings. By the end of the semester, my group achieved a 95% on our project.

**12. How would you go about getting cohesion among a team who disagree?**

I would find common ground between the members who disagree. I would talk about the importance of the overall goal and the implications if we didn’t come together to achieve it. We would then work together to come to an agreement that is a win/win for both sides.

**13. What sort of leader would your team say that you are?**

They would describe me as someone who will clear the way when there are obstacles and always has their back.

**14. How do you motivate your team?**

I find out what motivates them individually so I can speak to how a goal or change is going to benefit them. I ensure that I have the right amount of positive and constructive feedback to help them perform effectively. My actions always match my words so when I speak to my team with conviction; they are on-board with performing their best.

**15. How do you set an example to those for your team members?**

I will perform my best at everything and ensure that my actions match my words. My team sees that my expectations that are set for them are the same expectations I put on myself.

**16. Have you ever been in a mentor to another aspiring leader? How did you go about establishing that relationship?**

Yes, I treated it much like the relationship that I have with my team. I built a strong working relationship with the person, listened to their goals, gave advice, and my personal experience. I shared my best practices and constantly monitored their progress to celebrate their success and move them in the right direction.

**17. What is the most difficult part of being a leader?**

In some ways, although you are part of a group, you are alone. It’s a leader’s responsibility to see the end goal and vision of an organization to lead others towards it. When others do not see it the same way, you have to be the lone voice to bring them back on track.

**18. How do you lead through change?**

As a leader, you have to be the first one to embrace change because if you don’t like those around you will quickly see that. After that, I ensure that I can communicate the change with the conviction that it’s the right path to adopt.I prepare by ensuring that I can answer any questions that may be asked, or have the resources to find the answers. I listen to others concerns about the change and help them through the transition.

**19. How do you measure success for you as a leader?**

By the goals that the team achieves. When someone on the team is successful, then it reflects on my leadership.

**20. What motivates you to be a leader?**

I am motivated by my team’s growth and achievement of their professional and personal goals.

**21. What is a leader’s best asset?**

Their ability to motivate and inspire a team of professionals who can work together to achieve the goals of the organization.

**22. What do you do when you are unsure about how to achieve the goals of the team?**

You have to be open to feedback and be willing to ask for help when you are not clear on how to achieve a goal. I would ask my leader first for their feedback on how they believe I should go about achieving the goals. I would also use all the resources available to me to find the best course of action.

**23. Are you more comfortable with verbal or written communication?**

I am comfortable with both types of communication. However, I feel that verbal communication is more effective.  That’s because when you speak to someone directly, you will be able to see their body language toward the discussion. You are also able to address questions/concerns faster than in written communication.

**24. How would you deliver bad news to your team?**

I would bring them together and state the news. I would explain as much as possible as to why it occurred and what steps we will need to take in the future. I would also open it up to the team to speak about their concerns, answer questions, and share their viewpoints to know how we can avoid a similar situation.

**25. Is competition among a team healthy? Why or why not?**

I believe competition among a team is good as long as it is in good spirit. A team has to have a high level of cohesion among its members to prevent misunderstandings. As a leader, it’s my responsibility to ensure that when there is a competition that it is being monitored to ensure its positivity.

**26. What are the most difficult decisions to make?**

It’s difficult to take the decision to let an employee go. However, if they are not performing the way that they should be, it is the right decision. It is never easy to make a decision that will impact a person’s life.

**27. What kind of criticism you most get?**

I have not received criticism on the same area over and over. I’m always open to personal and professional growth and welcome any opportunity to improve. When I receive criticism, I work on improving that aspect and furthering my growth.

**28. How would you proceed to reorganize your team?**

I would look at the overall goals of the organization and match my team’s strengths up with the reorganization.

**29. Have you ever been a member of a successful team? What was your role in the success of the team?**

Use an example of when you were part of a team and demonstrate the leadership skills that you used to pertain to your role.

**30. How do you build support for ideas/goals with people who do not report to you and you have no authority over?**

In situations where I must build support for my ideas with cross-functional teams, I ensure that I communicate my opinion clearly and effectively. I listen to their feedback and their ideas, and I will make amends if they are necessary to build support or improve on the idea. I foster an environment where an input is sought and validate my idea by explaining why its the best route.

**31. How do you go about resolving conflict?**

I take a mediated approach to conflict. I believe it’s important to listen to both sides and understand where each is coming from. There is usually some common ground between conflict, and I start there and build.

**32. Name a time when an employee disagreed with your directive and how you handled it?**

I heard them out to understand why they disagree. I may have to go back and re-explain the directive and reasons for it. I would listen to their feedback and if it is the right thing to do, take it to change the directive. However, if that is not the case, I would stick to the facts as to why their commitment is necessary.

**33. Who are the most important members of your team?**

Everyone is equally important. Each person contributes something different to the team, and that makes us as a whole stronger.

**34. How do you delegate responsibilities to your team?**

I match up responsibilities with each member’s strengths. If I have a team member who is working on improving an aspect, I will give them the opportunity to take on the task and ensure they have the tools necessary to be successful. I would monitor their progress as well.

**35. Name a time when you had to change a decision due to new facts.**

Pick a situation where you showed that you were open to change and show how you were effectively at changing your decision based on the new facts.

For example, I had created a new spreadsheet for managers to use at the end of the night to keep track of sales for the day. This spreadsheet was due in an e-mail every morning and helped us see how we were doing on a daily basis. A few months later, our point of sales system allowed us to input this information into a program that would allow managers to input sales for the day. With this new technology, I decided to do away with the spreadsheet and had the managers use the program to capture the information and send it to me.

**36. How do you achieve objectives in a fast-paced environment?**

I ensure that the team knows the objectives and the timeliness that have been set. I place milestones so each member can check their progress.

**37. Explain a time when you had to make a decision without all the relevant facts.**

Pick a decision that you would not have all the facts for at the time of the decision. Make sure that you speak about all the different options you had and how you picked the best one out of what you had available. Talk about the results/takeaways.

For instance, I had to decide whether our organization was going to be involved in a new marketing campaign that used social media to advertise our products. At this point, our company did not have relevant information on how successful our previous social media marketing campaigns were. If we were to proceed, I was going to have dedicated at least one member of my team to its success. It would be time-consuming and if not successful, would take up a lot of productivity time. I decided to take part in the campaign because it was relatively inexpensive and the potential to gather information about best practices when launching them in the future. We ended up with a very successful marketing campaign with measurable results.

**38. How do you formulate and present arguments to others?**

I look at all sides of an argument first so I know what may come up when I present my position. I base my arguments strictly on facts that are objective.

**39. How did you a handle a time when you had to make an unpopular decision?**

Talk about a decision that you made that was necessary, but not popular with your team. Explain how you communicated the decision, listened to their concern, and stood your ground on the decision.

One possible answer –

Last year I decided to change our commission structure to our sales reps. I felt it was a necessary change because there were too many sales reps who were doing the bare minimum to collect a paycheck. Needless to say, many of the sales reps were upset with the decision. I reiterated the reasons for the change and ensured they had the tools they needed to be successful in the new commission structure. The organization saw an increase in their revenue and sales reps were making 5% more with the new commission structure.

**40. What do you do to remain engaged in a conversation?**

I actively listen by paraphrasing what others say to me. That ensures that I am on the same page as the other person and keeps me attentive to the conversation.

**41. How do you organize projects and tasks?**

I organize them by what is the most important and time-sensitive to complete.

**42. Explain a time when you were not able to meet a deadline?**

Use an example were you where not able to meet a deadline due to outside factors.

For instance, there was a big project that my team was working on, and I had split up the work among some members and myself. During that time, one member of the team had to leave due to their spouse getting a position in another city. He left at a critical time, and I had to re-assign his duties to someone else. I make the new person work to speed with the progression of the project and due to this, was not able to complete it on time. We were still able to complete the project a few days after the deadline even with the change in the team member.

**43. How have you rallied your team in the past in difficult projects/tasks?**

I communicate my confidence in their ability to complete the project. I ensure that I remove as many obstacles as possible and they have all the tools/answers they need to complete the task. I ensure there are clear expectations and open communication.

**44. How do you encourage the development of your employees?**

I develop my employees by being a mentor, giving effective performance feedback on a regular basis, and coaching. I take a personal interest in the development of my employees, and when they see that I am committed to their growth, they are more motivated.

**45. What is the most significant change that you brought to an organization?**

Provide an example that shows how you demonstrated your vision to make a positive change in the organization. Also, talk about the results of the change.

For instance, at my previous organization, the management team came up the ranks and never had formal management training. They did not know how to lead their former peers and were uncomfortable having productivity discussions with their teams. I felt there was a need to train these managers on the skills they would need to be successful. So I made my case to the leadership team on why it is important and provided examples I was seeing. Due to this, all managers go through a rigorous management training program that prepares them for their new role.

**46. Have you been developed an innovative solution to a non-traditional problem?**

In your example, show how you promote change and innovation. Solutions to unique problems occur when there is a constant information flow in all directions to ensure responsiveness to change.

For instance, I was responsible for a sales team in my previous position. A separate production staff handled the orders that my sales team would prepare. This production team had difficulty making the deadlines that my sales team promised their clients. In addition, the product was sometimes not customized to the level the client was looking for. So I decided to change the process that our sales reps put in sales order by having the sales rep communicate with the production team who was responsible for each client’s product. This helped my sales rep create achievable timeliness and a product that was the way the client expects.

**47. What is the role that leadership plays to a manager?**

A leader’s role is to communicate with clarity to the strategic vision to the management team. This vision must be able to be in the form of a clear direction and plans. There should be clear priories, objectives timeliness, accountability, and performance measures.

**48. What leadership style do you use?**

This answer should be based on the type of organization you are joining. You should show that you will be able to change your style in different circumstances.

**49. How would you go about developing your team?**

I encourage training courses, soft skills workshops, on the job mentoring, and coaching.

**50. Have you ever taken on a job that you were unqualified for?**

In your example, show how you are not afraid of taking risks to achieve goals at work. Demonstrate your focus on the job at hand and how it inspired others.

For instance, I took on management responsibilities in my previous position to take the place of my manager who had left. I did not have any management experience, but I knew that the team was not going to be able to be effective without a leader in place. I may have made a few mistakes, but ultimately was successful in taking on that additional responsibility. The upper-level management was impressed by my growth and efforts, so they ended up promoting me into that position.

5 Team Leader Interview Questions and Answers

Whether you are preparing to interview a candidate or applying for a job, review our list of top Team Leader interview questions and answers.

How do you determine if a task or project is at risk?

Team leaders have a responsibility to communicate team goals. They must ensure work is completed by the deadline and at the quality level that's expected by upper management and customers. This question lets you see how applicants monitor tasks and projects, identify issues and make certain the team is helping the company achieve its main goals. What to look for in an answer:

* Applicant's ability to communicate expectations and collaborate with team members
* Critical thinking and problem-solving skills
* Management style

Example: "First, I communicate work expectations clearly to team members, being sure to highlight everything that's been communicated to me from managers and executives. This reduces the chance of failure. Second, throughout a task or project, I monitor key performance indicators to see if the team is on schedule, within budget and producing quality work. I also ask that team members maintain constant communication with me throughout the project. This allows me to anticipate and identify problems before they arise and reduce any negative effects."

What do you do if there's a disagreement within your team?

The success of a team often depends on the ability of its leaders to get everyone working together towards a common goal. A lack of unity can lead to failure. With this question, you can gain an understanding of how the applicant would handle the situation successfully and move the team in a positive direction. What to look for in an answer:

* Applicant's decision-making and problem-solving approach
* Conflict management and resolution skills
* Opinion on the value of teamwork

Example: "Disagreements can happen, especially when people have strong opinions over the way a task should be done. I would first ask the team members to take a step back. I would then lead a team discussion so that we can all understand each other's point of view. If appropriate, I would negotiate a compromise that would put the team in the best position for success. If there is clearly a right way to handle the situation, I will explain why we're choosing one solution over the other."

Are there similarities between being a team leader and a coach?

Great team leaders don't simply dictate work be done from their office. They encourage team members, provide feedback and offer skills development. They're similar to great sports coaches. This question can show you how applicants view leadership and implement their style. What to look for in an answer:

* Applicant's leadership capability and style
* Commitment to developing team members
* Ability to provide constructive criticism

Example: "I do not employ a top-down leadership style because I believe a leader should act like a supportive coach. I give my team members space to unleash their talent while providing constructive feedback, regular performance reviews and training sessions. I aim to run an open, transparent and honest operation. My goal is to know employees well so I can help them grow and improve. This is how the team wins."

What metrics do you use to evaluate team performance?

Team leaders are not just expected to generate and share detailed performance reports with the team and company management. They should also know how to use data-driven insights to boost team efficiency. Look for answers that show a team leader won't rely on instinct to make decisions but rather will examine the numbers to determine the best solutions and changes to increase performance. What to look for in an answer:

* Applicant's method for improving team performance
* Knowledge of performance metrics
* Opinion of data-driven approaches

Example: "Given all the data we have today, it's essential that team leaders analyze and extract insights from that data to increase effectiveness. My opinion is that the metrics a team focuses on should strongly link to a company's strategic objectives. In a general sense, I always look at quantity, quality and time. For instance, to measure the quality of work, it's important to look at the number of errors and customer satisfaction. Another thing I measure is employee morale and engagement because those can make or break a team."

How do you interview prospective team members?

Many companies ask team leaders to participate in the interview process for applicants to their team. Since team leaders are so close to the front line, management values their input. Look for answers that clearly illustrate what the applicant values in team members. What to look for in an answer:

* Applicant's view on the skills and characteristics a team needs
* Ability to interview and identify talented workers
* Opinion on the importance of team culture

Example: "As a team leader, I aim to build a superstar team, not just a team of superstars. I make sure applicants have the necessary hard skills but also look at important soft skills, such as written and oral communication, logical thinking, adaptability, attention to detail and reliability. In addition to questions to check experience and credentials, I would ask behavioral, competency and opinion questions during the interview, as well as a brain teaser or two."

# Team Leader interview questions

Going for a job interview role as a Team Leader? Preparation is key as there are lots of Team Leader interview questions you could be asked. These could be generic about why you chose to apply for the business, or more specific to leading a team, focussing on your previous experiences and your skills.

Whilst you won’t know exactly what questions you’ll be asked until the day itself, you can still prepare by thinking about some of the most common Team Leader interview questions. This will help you consider your answers in advance, so that you can get all the information you need across, and demonstrate that you’re the best person for the job.

## Examples of Team Leader interview questions:

To help ensure you are ready for your next [job interview](https://www.roberthalf.com.au/career-advice/interview), read these six Team Leader interview questions and answers with our advice on how to respond effectively.

## 1. How would your colleagues and team describe you?

Common Team Leader interview questions you might get asked could include how existing colleagues and your team would describe you. The interviewer will be interested to hear what others have to say, but they will also want to establish how you view yourself.

For this question, be honest. Your interviewer may already have received references so you don’t want to make something up that is completely untrue.

If you’re not sure what colleagues would say about you, think back over whether they have provided any feedback to you verbally, or in writing, such as in the form of a reference, or LinkedIn recommendation. If you have too many comments to choose from, see if any of the remarks mention qualities that are also mentioned in the job description. If so, make sure you prioritise these.

## 2. How would you mitigate a dispute between two team members who disagree over their responsibilities?

As well as questions about your past, you may be presented with situational Team Leader interview questions. A key theme that interviewers may want to focus on is conflict management, which is a key skill that every Team Leader needs to have.

Take your time to consider your response. You want to show that you don’t tolerate arguments and can resolve the situation quickly and effectively. Make sure you show that you understand how a dispute can not only impact the individuals, but the team as a whole, along with the project. Explain how you would listen to both sides of the argument and carefully consider their feedback. Demonstrate that you are not afraid to then come to a decision and stick to it.

## 3. What techniques have you used to motivate a team?

As a Team Leader, it’s your responsibility to ensure work gets done by the set deadlines, to ensure your customers remain satisfied. To do this, you need motivated staff.

For this interview question, explain how you would improve motivation levels. For example, do you take a formal approach and give bonuses, share constructive feedback to encourage staff to perform better, or choose an “Employee of the Month”? Or, do you take a more relaxed approach by simply being friendly to your team, organising fun team events, or allowing staff to occasionally leave early after a successful result?

If you want to take your answer one step further, try to understand how the business you’re interviewing at motivates their staff. If your approach is similar, great. If it’s different, ensure you make it clear that you do have your own tried and tested methods, but you’re open to new approaches.

## 4. What are your preferred methods of communication and why?

It’s important for a Team Leader to be able to communicate clearly with their team members. Before you answer, consider how you communicate on a daily basis. Do you prefer having a lengthy call with someone, do you arrange to see them face-to-face, do you send them a quick bullet-pointed email, or do you use online collaboration tools? Explain why you use the communication methods you do and make sure you use examples.

## 5. Do you use any tools to support you in leading a team?

As a Team Leader, you need to be able to oversee day-to-day operations, delegate tasks to team members and set deadlines. There are an increasing number of tools available, to help track timelines and goals, collaborate more effectively, communicate more efficiently and improve productivity. Outline which tools you have used before and then explain why you have used them. If you have had any positive outcomes as a result of using these tools, explain to the interviewer what you have experienced.

## 6. Explain a project where you led your team to success

Other key Team Leader interview questions you might be asked can revolve around your previous experiences. Don’t simply answer with the first thing that comes to your mind. Spend some time to think about all your projects and which one had the best results, perhaps one where you had to overcome a large unexpected obstacle to achieve success. This is your chance to really stand out, so make sure you explain the project, the results and the key part you played in its success.

By following these six Team Leader interview questions and answers, you’ll be well on your way to interview success. Of course, this isn’t an exhaustive list, but it will certainly help you make a good impression at your upcoming interview.

**How to Answer Interview questions about leadership?**

**Q #1) Differentiate team leader and team manager?**

**Ans)** A Manager handles the everyday jobs and responsibilities of his team members and also ensures that they complete their assigned work on time. Whereas a Leader will encourage and inspire his/her team to finish and work out their goals.

**Q #2) Define a difficult and important task of a Leader?**

**Ans)** The role of a leader involves two duties. As a leader, he should act as a team member and simultaneously he should act as an individual to drive the whole team.

A leader should ensure that all his team-mates are on the same path towards the end goal which in turn leads to organizational success.

**Q #3) How will you guide your team to improve or develop their individual skills?**

**Ans)** As a leader I will suggest my team members attend any training, workshops on soft skills, behavioral courses to improve or strengthen their interpersonal skills and other talents.

**Q #4) What is the motivation for your leadership? As a leader how will you measure your success?**

**Ans)** A Leader gets motivated by his/her team’s achievements like team-mates finishing their targets, achieving their professional as well as personal goals etc.

**Obviously,** a leader’s success lies in his/her team member’s success.

**Q #5) When you come across any bad news then how will you put it in front of your team?**

**Ans)** To deliver any bad news with the team, I will arrange a small meeting with the team and deliver it. Will try to explain all the possible solutions that need to be implemented in the near future so that such mistakes cannot take place.

At last, will ask my team to share their views, ideas, concerns or any suggestions that can be carried out in the next activity to avoid such situations.

**Q #6) What is your approach when you are not clear about achieving your goals on time?**

**Ans)** As a leader you should be ready to accept any comments or opinions from your team. And should not hesitate to seek help from your immediate superiors or your team-mates.

Taking all the inputs into account, you should come up with a new and clear agenda on how to finish your target on time.

**Q #7) How will you encourage your team by sharing the other team member’s success?**

**Ans)** Praising one’s success in front of others will inspire and motivate them to work harder. It encourages and gives confidence to others to perform better. One who gets succeeded in their career should be rewarded.

As a leader, if we link the performance of a team member to any reward or recognition, then it motivates them to work harder.

**Q #8) Is there any strong or unique skill in you that makes me hire you?**

**Ans)** My career objective is to serve a professionally managed company and achieve a challenging position by utilizing my abilities developed through my experience and education and contribute to enhancing the goals of the company. This makes you

This makes you hire me into your organization.

**Q #9) Are you comfortable with group discussions/meetings or one on one?**

**Ans)** As a leader I prefer group discussions because in a group we will be able to know the individual skills and hidden qualities of the team-mates. This can help everyone to improve their interpersonal skills. And as a whole, it will lead to the success of the team.

**Q #10) Explain your strength and weakness as a leader?**

**Ans)** Being a leader one should maintain a strong relationship with the team members and should focus on their goals. Simultaneously a leader should inspire and encourage his/her team to work hard and give their best. If a team performs well in the project, then it shows the strength of a leader who is leading the respective team.

Simultaneously a leader should inspire and encourage his/her team to work hard and give their best. If a team performs well in the project, then it shows the strength of a leader who is leading the respective team.

Even though a leader is capable of finishing a task on his own, he should distribute that particular task among the team. If a leader did not do so, then he will end up with a lot of work on himself and cannot complete the task on time.This can be considered as leader’s weakness.

This can be considered as leader’s weakness.

**Q #11) What is over supervision in your view as a leader? How will it affect an employee?**

**Ans)** Over Supervision is nothing but when a person is given with more direction than what he needs. When an employee is under such situation, they get frustrated and angry.

They stop trying and taking risks. They stop making decisions on their own, and their participation and initiation come down.

**Q #12) What will be the situation when an employee is under supervised?**

**Ans)** Under supervision is nothing but giving directions to an employee in a coaching pattern when he is in need of some direction and close supervision. When people are in such situation, they fail because they did not get the proper guidance which they need and gets frustrated.

**Q #13) What will be the situation if a leader’s supervision and an employee’s working style gets matched?**

**Ans)** When the Leadership style and development level of the subordinate get matched, then there will be a win-win situation. Employee or subordinate should not feel that someone else’s loss is their win or that their win is someone else’s loss.

**Q #14) How will you assist your subordinates based on their competency and commitment levels?**

**Ans)** A leader should provide his subordinate what they cannot provide for themselves.

* For an employee with Low Competence or High Commitment, the leader should provide direction and the subordinate provides his own eagerness and support.
* For an employee with Some Competence or Low Commitment, the leader should provide direction along with motivation and encouragement.
* For an employee with Moderate-High Competence or Variable Commitment, the leader should provide support and encouragement because that particular team member is self-directive, but his assurance is low on one aspect.
* For an employee with High Competence or High Commitment, the leader should delegate the responsibility for task achievement to the team member.

**Note:** Competency deals with Knowledge and Training Skills. Commitment deals with Self Confidence and Self Motivation.

Leadership does not mean “*Different types of guidance for different employees”*, it also means that “*Different types of instructions for the same employees, based on the job”.*

**Q #15) What are the four different leadership styles and brief them?**

**Ans)** Generally the Leadership styles differ in three ways: the amount of direction provided by the leader, the amount of support and encouragement given by the leader, and the involvement of an employee in decision making.

* **DIRECTING:**This is described as one-way communication. The leader will define the roles and goals, guiding the employee what, when, how, with whom, and where to carry out the task. The leader initiates the Problem-solving and decision making. Resolutions and conclusions are announced, and a job is closely observed by the leader.
* **COACHING STYLE:**This is a two-way communication. The leader assigns the task to the employee and also motivates, encourages, praises the employee’s views on how to proceed with the task. Even though the leader approaches the employee, decision-making and problem-solving are up to the leader.
* **SUPPORTING:**In this style, both the leader and the employee identify the problem together and will set the goals. Problem-solving and decision-making shifts between them. The employee will share his view on how to proceed with the job. Then the leader provides any assurance or support, resources etc if needed.
* **DELEGATING:**In this style, the employee and the leader jointly define the problem and goals. The employee prepares the plan about when, how and with whom the task should be done. A leader will monitor the employee’s performance periodically.

In a Leadership interview, you will be more focused on the scenario type questions.

Like, you will be given a scenario and will be asked to give your explanation or views against the same. And you will be judged based on your perception towards the answer.

***Here are few scenario-based questions for the interview***

***Scenario 1)*** You have asked one of your team members to prepare a report for a newly added requirement into your current project. Usually, he completes the task assigned to him on time with encouragement from you. However, this time his report is overdue.

**Q #16) How would you handle this situation?**

**Ans)** You tell him what you are expecting from him and when you want the report to be completed. But discuss with him regarding the reason for not preparing the report on time. Apart from that, you need to keep track on his performance daily.

***Scenario 2)* Recently,** you have been facing a problem with one of your team member. He became lazy and exhausted. Your constant follows up has brought task completion. Because of such experience with him, you suspect that he is not capable of completing a high-priority task.

**Q #17) How will you overcome such situation?**

**Ans)** As a leader I will involve him in Problem-Solving with the task and offer support to him. And will try to utilize his ideal in the task completion. Apart from this, I’ll try to draw out his attitudes and feelings concerning this task assignment.

***Scenario 3)*** One of your senior employees is assigned with a new job which is important to your team in near future. Even though he is excited about the new job, he has no experience with the task.

**Q #18) How will you motivate him to take up the task and proceed further?**

**Ans)** I will discuss the job with him by defining the necessary activities to proceed with it, supporting his ability to do the task. And highlight his outstanding performance in the past. Ask him to share his views on the new job so that they can be implemented.

***Scenario 4)***A highly productive and efficient member of your team has asked for your help on a task. He is familiar to work effectively on his own but some work hurdles have discouraged him to solve the task by himself.

**Q #19) How will you solve his problem and make him comfortable to work?**

**Ans)** I will try to analyze the problems and outline the methods to solve them. Finally, I will help him to determine and implement an appropriate solution to resolve the task on his own.

***Scenario 5)*** A new employee has been assigned to your team to perform a new assignment. He is excited and confident enough to accept the task but he has no experience in that job or task.

**Q #20) How will you deal with that new employee?**

**Ans)** First of all I’ll welcome him into the team and will explain him all the details regarding the new task and will tell him what the job demands from him. As a leader will guide him how to proceed with the task. I will also convey him that what is the expectation on him and will monitor his work frequently and closely.

***Scenario 6)*** Recently you have been appointed as head of a department. Under former HOD’s supervision, the team functioned satisfactorily with his encouragement and support. Since you have taken the authority, the team’s performance has come down.

**Q #21) How will you identify their problem and suggest a solution for it?**

**Ans)** Immediately I will call for a team meeting and discussion regarding the team’s low performance and will suggest the possible solutions overcome those difficulties. Will also support their efforts to indicate any corrective actions. At last, will ask for any inputs and suggestion from the team.

**Here are few tips to empower your subordinates as a leader:**

* Should encourage your subordinates when they do not expect it and when they are not asking for it.
* An individual who tries and fails in any job can be complimented for having the courage to try.
* Mistakes made by any employee can be treated as learning opportunities instead of blaming them. But ensure that this should not happen all the time.
* Few employees who think that they have to do things perfectly are often afraid to attempt for fear of making mistakes. Such people should be encouraged and given support.

# **Ostrich algorithm**

From Wikipedia, the free encyclopedia

In [computer science](https://en.wikipedia.org/wiki/Computer_science), the **ostrich algorithm** is a strategy of ignoring potential problems on the basis that they may be exceedingly rare. It is named for the [ostrich effect](https://en.wikipedia.org/wiki/Ostrich_effect) which is defined as "to stick one's head in the sand and pretend there is no problem". It is used when it is more cost-effective to allow the problem to occur than to attempt its prevention.

## Contents

* [1Use with deadlocks](https://en.wikipedia.org/wiki/Ostrich_algorithm#Use_with_deadlocks)
* [2Trade-offs](https://en.wikipedia.org/wiki/Ostrich_algorithm#Trade-offs)
* [3References](https://en.wikipedia.org/wiki/Ostrich_algorithm#References)
* [4Notes](https://en.wikipedia.org/wiki/Ostrich_algorithm#Notes)

## Use with deadlocks[[edit](https://en.wikipedia.org/w/index.php?title=Ostrich_algorithm&action=edit&section=1)]

This approach may be used in dealing with [deadlocks](https://en.wikipedia.org/wiki/Deadlock) in [concurrent programming](https://en.wikipedia.org/wiki/Concurrent_programming) if they are believed to be very rare and the cost of detection or prevention is high. For example, if each PC deadlocks once per 10 years, the one reboot may be less painful than the restrictions needed to prevent it.[[1]](https://en.wikipedia.org/wiki/Ostrich_algorithm#cite_note-1)

A set of processes is [deadlocked](https://en.wikipedia.org/wiki/Deadlock) if each process in the set is waiting for an event that only another process in the set can cause. Usually the event is release of a currently held resource and none of the processes can run, release resources, and be awakened.[[2]](https://en.wikipedia.org/wiki/Ostrich_algorithm#cite_note-2)

The ostrich algorithm pretends there is no problem and is reasonable to use if deadlocks occur very rarely and the cost of their prevention would be high. The [UNIX](https://en.wikipedia.org/wiki/UNIX) and [Windows](https://en.wikipedia.org/wiki/Windows) operating systems take this approach.[[3]](https://en.wikipedia.org/wiki/Ostrich_algorithm#cite_note-3)

Although using the ostrich algorithm is one of the methods of dealing with [deadlocks](https://en.wikipedia.org/wiki/Deadlock), other effective methods exist such as dynamic avoidance, [banker's algorithm](https://en.wikipedia.org/wiki/Banker%27s_algorithm), detection and recovery, and prevention.[[4]](https://en.wikipedia.org/wiki/Ostrich_algorithm#cite_note-4)

## Trade-offs[[edit](https://en.wikipedia.org/w/index.php?title=Ostrich_algorithm&action=edit&section=2)]

Although efficient, using the Ostrich algorithm trades correctness for convenience. Yet since the algorithm directly deals with extreme cases it is not a large trade-off. In fact, the simplest and most used method to recover from a deadlock is a reboot.

Some algorithms with poor worst-case performance are commonly used because they only exhibit poor performance on artificial cases that do not occur in practice; typical examples are the [simplex algorithm](https://en.wikipedia.org/wiki/Simplex_algorithm) and the type-inference algorithm for [Standard ML](https://en.wikipedia.org/wiki/Standard_ML). Issues like [integer overflow](https://en.wikipedia.org/wiki/Integer_overflow) in programming languages with fixed-width integers are also frequently ignored because they occur only in exceptional cases that do not arise for practical inputs.

# DEADLOCKS

Computer systems are full of resources that can only be used by one process at a time. Common examples include printers, tape drives, and slots in the system’s internal tables. Having two processes simultaneously writing to the printer leads to gibberish. Having two processes using the same file system table slot will invariably lead to a corrupted file system. Consequently, all operating systems have the ability to (temporarily) grant a process exclusive access to certain resources.

For many applications, a process needs exclusive access to not one resource, but several. Suppose, for example, two processes each want to record a scanned document on a CD. Process *A* requests permission to use the scanner and is granted it. Process *B* is programmed differently and requests the CD recorder first and is also granted it. Now *A* asks for the CD recorder, but the request is denied until *B* releases it. Unfortunately, instead of releasing the CD recorder *B* asks for the scanner. At this point both processes are blocked and will remain so forever. This situation is called a **deadlock**.

Deadlocks can also occur across machines. For example, many offices have a local area network with many computers connected to it. Often devices such as scanners, CD recorders, printers, and tape drives are connected to the network as shared resources, available to any user on any machine. If these devices can be reserved remotely (i.e., from the user’s home machine), the same kind of deadlocks can occur as described above. More complicated situations can cause deadlocks involving three, four, or more devices and users.

Deadlocks can occur in a variety of situations besides requesting dedicated I/O devices. In a database system, for example, a program may have to lock several records it is using, to avoid race conditions. If process *A* locks record *R1* and process *B* locks record *R2*, and then each process tries to lock the other one’s record, we also have a deadlock. Thus deadlocks can occur on hardware resources or on software resources.

In this chapter, we will look at deadlocks more closely, see how they arise, and study some ways of preventing or avoiding them. Although this material is about deadlocks in the context of operating systems, they also occur in database systems and many other contexts in computer science, so this material is actually applicable to a wide variety of multiprocess systems. A great deal has been written about deadlocks. Two bibliographies on the subject have appeared in *Operating Systems Review* and should be consulted for references (Newton, 1979; and Zobel, 1983). Although these bibliographies are old, most of the work on deadlocks was done well before 1980, so they are still useful.

## 3.1 RESOURCES

Deadlocks can occur when processes have been granted exclusive access to devices, files, and so forth. To make the discussion of deadlocks as general as possible, we will refer to the objects granted as **resources**. A resource can be a hardware device (e.g., a tape drive) or a piece of information (e.g., a locked record in a database). A computer will normally have many different resources that can be acquired. For some resources, several identical instances may be available, such as three tape drives. When several copies of a resource are available, any one of them can be used to satisfy any request for the resource. In short, a resource is anything that can be used by only a single process at any instant of time.

### 3.1.1 Preemptable and Nonpreemptable Resources

Resources come in two types: preemptable and nonpreemptable. A preemptable resource is one that can be taken away from the process owning it with no ill effects. Memory is an example of a preemptable resource. Consider, for example, a system with 32 MB of user memory, one printer, and two 32-MB processes that each want to print something. Process *A* requests and gets the printer, then starts to compute the values to print. Before it has finished with the computation, it exceeds its time quantum and is swapped out.

Process *B* now runs and tries, unsuccessfully, to acquire the printer. Potentially, we now have a deadlock situation, because *A* has the printer and *B* has the memory, and neither can proceed without the resource held by the other. Fortunately, it is possible to preempt (take away) the memory from *B* by swapping it out and swapping *A* in. Now *A* can run, do its printing, and then release the printer. No deadlock occurs.

A **nonpreemptable resource**, in contrast, is one that cannot be taken away from its current owner without causing the computation to fail. If a process has begun to burn a CD-ROM, suddenly taking the CD recorder away from it and giving it to another process will result in a garbled CD, CD recorders are not preemptable at an arbitrary moment.

In general, deadlocks involve nonpreemptable resources. Potential deadlocks that involve preemptable resources can usually be resolved by reallocating resources from one process to another. Thus our treatment will focus on nonpreemptable resources.

The sequence of events required to use a resource is given below in an abstract form.

1. Request the resource.
2. Use the resource.
3. Release the resource.

If the resource is not available when it is requested, the requesting process is forced to wait. In some operating systems, the process is automatically blocked when a resource request fails, and awakened when it becomes available. In other systems, the request fails with an error code, and it is up to the calling process to wait a little while and try again.

A process whose resource request has just been denied will normally sit in a tight loop requesting the resource, then sleeping, then trying again. Although this process is not blocked, for all intents and purposes, it is as good as blocked, because it cannot do any useful work. In our further treatment, we will assume that when a process is denied a resource request, it is put to sleep.

The exact nature of requesting a resource is highly system dependent. In some systems, a request system call is provided to allow processes to explicitly ask for resources. In others, the only resources that the operating system knows about are special files that only one process can have open at a time. These are opened by the usual open call. If the file is already in use, the caller is blocked until its current owner closes it.

# **Project implementation: Eight steps to success** Often a smoothly run project gets a black eye because of problems during implementation. Those problems often crop up because we don’t anticipate and plan for the complexity of deploying the solution. For example, you might communicate and plan well for the deployment of a client-server solution, only to discover during implementation that many of your workstations aren't powerful enough to handle the load. This is the type of minor detail that can cause major headaches.

Start at the beginning  
[Part I of this series](https://www.techrepublic.com/article.jhtml?id=u00420030217moc01.htm) on project implementation focused on methodology. Here's how to plan ahead to avoid surprises.

Let’s look at the major steps associated with implementation. Note that many of these activities need to be completed ahead of time. You cannot start planning for implementation while you are actually implementing.

1. **Prepare the infrastructure.** Many solutions are implemented into a production environment that is separate and distinct from where the solution was developed and tested. It is important that the characteristics of the production environment be accounted for. This strategy includes a review of hardware, software, communications, etc. In our example above, the potential desktop capacity problem would have been revealed if we had done an evaluation of the production (or real-world) environment. When you are ready for implementation, the production infrastructure needs to be in place.
2. **Coordinate with the organizations involved in implementation.** This may be as simple as communicating to your client community. However, few solutions today can be implemented without involving a number of organizations. For IT solutions, there are usually one or more operations and infrastructure groups that need to be communicated to ahead of time. Many of these groups might actually have a role in getting the solution successfully deployed. Part of the implementation work is to coordinate the work of any other groups that have a role to play. In some cases, developers simply failed to plan ahead and make sure the infrastructure groups were prepared to support the implementation. As a result, the infrastructure groups were forced to drop everything to make the implementation a success.
3. **Implement training.**  Many solutions require users to attend training or more informal coaching sessions. This type of training could be completed in advance, but the further out the training is held, the less information will be retained when implementation rolls around. Training that takes place close to the time of implementation should be made part of the actual implementation plan.
4. **Install the production solution.** This is the piece everyone remembers. Your solution needs to be moved from development to test. If the solution is brand new, this might be finished in a leisurely and thoughtful manner over a period of time. If this project involves a major change to a current solution, you may have a lot less flexibility in terms of when the new solution moves to production, since the solution might need to be brought down for a period of time. You have to make sure all of your production components are implemented successfully, including new hardware, databases, and program code.
5. **Convert the data.** Data conversion, changing data from one format to another, needs to take place once the infrastructure and the solution are implemented.
6. **Perform final verification in production.** You should have prepared to test the production solution to ensure everything is working as you expect. This may involve a combination of development and client personnel. The first check is just to make sure everything is up and appears okay. The second check is to actually push data around in the solution, to make sure that the solution is operating as it should. Depending on the type of solution being implemented, this verification step could be extensive.
7. **Implement new processes and procedures.** Many IT solutions require changes to be made to business processes as well. These changes should be implemented at the same time that the actual solution is deployed.
8. **Monitor the solution.** Usually the project team will spend some period of time monitoring the implemented solution. If there are problems that come up immediately after implementation, the project team should address and fix them.

[Part I of this series](https://www.techrepublic.com/article.jhtml?id=u00420030217moc01.htm) pointed out the need for planning and communication to help ensure a successful implementation. In this column, we looked at the actual work typically performed in a complex implementation. However, your implementation may not be as complex, and you may not need to look at all of these areas. Nevertheless, there is usually a lot more involved than just throwing the final solution into the production environment. You need to account for the environment the solution will run in, as well as processes and training needs of the client community. If you think through implementation from a holistic approach and communicate well, there is a much greater likelihood that your project will end as a win.

# **Difference between a Data Analyst and a Data Scientist**

**Data Analysts** analyze similar historical knowledge to realize info. Information |the data} generated will not be used more to boost the understanding of the system. Therefore the corporate can keep it safe and take no significant risk to increase their business. Analysts work on historical knowledge and generate the trends of their company.

[**Data scientists**](https://www.geeksforgeeks.org/introduction-data-science-skills-required/) on the opposite hand square measure the extremely experienced (analysts when a few years of experiences may get promoted to scientists) folks of the corporate. They’re the one’s United Nations agency got to take the blame if their information does not exercise correctly for the business. Once analysts generate the data, the person work is to use his/her data and knowledge and take necessary choices to boost the business.

* **Analysts** work on reactive data(historical data). Sometimes they get identical info or results whereas analyzing the info.
* **Scientists** work on prophetic knowledge. What’s going to happen if we tend to try and try this or that.
* **Data analysts** square measure closely associated with business intelligence, whereas knowledge scientists square measure closely associated with business analytics. Therefore merely, analysts work on knowledge to get info.
* The **scientists** work thereon info and their information and expertise to require necessary business choices.

Consider AN example of a social application. Their main customers square measure from European countries. Currently, what AN analyst can do is that he/she can analyze the client behavior (that includes the time of usage, location of the client, event following, etc.). Currently supported these “historical data, ” the analyst can generate {the information|the knowledge|the knowledge} by combining many different data along. Like by combining location and gender of the client, the analyst can return to understand that women use their application quite boys together; however, inbound regions (xyz European country) boys tend to use the appliance additional. Therefore supported this, the corporate can try and enhance their business.

On the other hand, comes the scientists. Currently, scientists use this info and can try and improve the business by their expertise and information so that they will make choices like spreading additional awareness of their application by advertising it additional to some state. His/her main focus is on what’s going to happen “if” the appliance is launched in another country. This is often not low-cost, as advertisements may cost heaps and if the business flops in this country, then the person is that the one was answerable. However, if it is an enormous success, then the market additionally enhances greatly.

**Data Analyst Model:**

* **Managing:**  
  It includes arranging, executing and keeping up information forms for the safe storage of information and data resources.
* **Cleansing:**  
  It is way toward checking information quality and precision by perceiving at that point expelling inaccurate or one-sided information from a database
* **Abstracting:**  
  It is way toward expelling qualities from a dataset to decrease it to a lot of basic attributes for increasingly productive information preparing.
* **Aggregating:**  
  It is way toward gathering data from different information sources to get readily combined datasets for information handling.

**Data Scientist Model:**

* **Descriptive:**  
  What occurred? Example: What is the turnover this month?
* **Diagnostic:**  
  Why did it occur? Example: In your month to month report, you can see that last month’s business execution declined. What caused this?
* **Predictive:**  
  What will occur? Example: Imagine you are a retailer and you need to augment item deals while limiting waste. In what manner can you precisely gauge what amount of stock you need?
* **Prescriptive:**  
  What would it be a good idea for me to do? Example: Based on the traffic expectations, what are the best promoting activities you can set up to augment the prospects-to-lead proportion?

# **Difference Between Programming, Scripting, and Markup Languages**

When it comes to making a website or app coding involves basically three types of languages i.e the programming language, Scripting Language and Markup Language.  
There are a lot of people who consider coding as just developing or making a website but they need to understand every single language fits into a particular category and we need to know which category that language fits into. We will discuss in detail about the difference between these three main categories or pillars of a website or an app i.e. Programming language, Scripting Language and Markup Language.

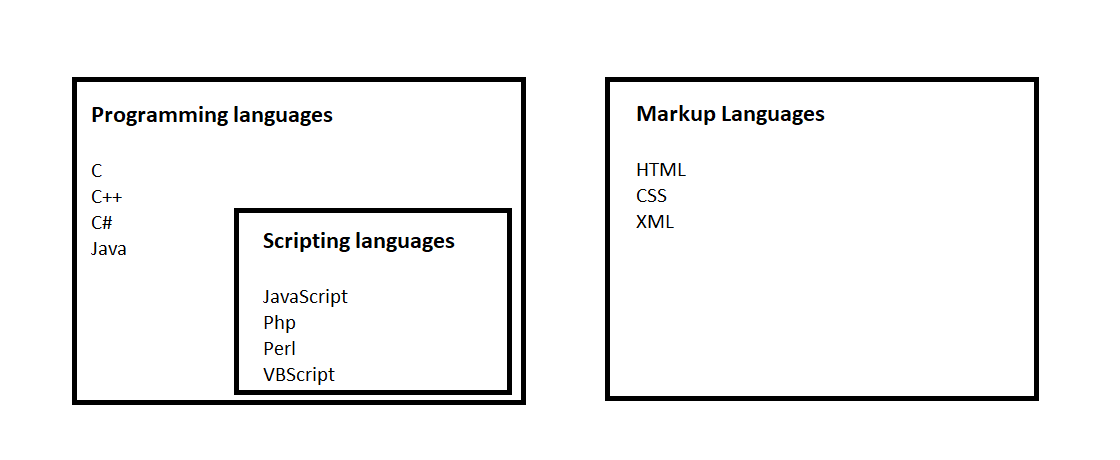


**Programming language:** In a simple language programming languages are set of instructions or code which tells a computer what it needs to do. So basically we provide a logic(loop or ) or instruction to the computer to perform some task to get the desired output from it. When we need to write a CD or burn a CD or when we need to paste something in pen drive these all instruction is given through some software which involves some instructions or set of code and this software communicate to the hardware. Programming languages are high-level languages which convert into machine level language because a computer can only understand machine level language or binary language (0 and 1). So we write the instructions in human-readable form and then we hit compile button to convert this into machine level language which a computer can understand and then the computer perform the task. This conversion is done by the compiler which scans the complete code in one go and if it finds any error it immediately throws all errors. Examples are [Java](https://www.geeksforgeeks.org/java/), [C](https://www.geeksforgeeks.org/c-programming-language/), [C++](https://www.geeksforgeeks.org/c-plus-plus/), [C#](https://www.geeksforgeeks.org/csharp-programming-language/). Programming languages are most widely used to make software or drivers.

**Scripting Language:**As the name suggest, it’s all about giving the script to perform some certain task. Scripting languages are basically the subcategory of programming languages which is used to give guidance to another program or we can say to control another program, so it also involves instructions. It basically connects one language to one another languages and doesn’t work standalone. [Javascript](https://www.geeksforgeeks.org/javascript-tutorial/), [PHP](https://www.geeksforgeeks.org/php/), [Perl](https://www.geeksforgeeks.org/perl-tutorial/), [Python](https://www.geeksforgeeks.org/python-programming-language/), VBScript these all are the examples of scripting language. Scripting languages need to be interpreted (Scanning the code line by line, not like compiler in one go) instead of compiled. There is ***no scope of compiler*** in scripting languages. Scripting languages are most widely used to create a website.

**Markup Languages:** Markup languages are completely different from programming languages and scripting languages. Markup languages prepare a structure for the data or prepare the look or design of a page. These are ***presentational*** languages and it doesn’t include any kind of logic or algorithm, for example, HTML. [HTML](https://www.geeksforgeeks.org/html-tutorials/) is not asking any kind of question to the computer or it’s not comparing things and it’s not asking any logical question. It’s just used to represent a view inside a web browser. It tells the browser how to structure data for a specific page, layout, headings, title, table and all or styling a page in a particular way. So basically it involves formatting of data or it controls the presentation of data. Examples of Markup languages are HTML, [CSS](https://www.geeksforgeeks.org/css-tutorials/)or XML. These languages are most widely used to design a website.

From the above definition, we can summarize Programming language, Scripting language, and Markup languages from below images.



**Conclusion:** So we can say that all the scripting languages are programming languages but all the programming languages are not scripting languages. C cannot be called as scripting languages, it is just a programming language but we can call JavaScript or Php programming or scripting languages. Also, there is no need to compile scripting languages it only needs to be interpreted. Scripting languages are generally slower than programming languages because compiled programs are first converted into machine code. On the other hand, markup languages are just used to define the structure of data which doesn’t require any logic or algorithm.