**Requirements Gathering/Analysis**

This phase is critical to the success of the project.  Expectations (whether of a client or your team) need to be fleshed out in great detail and documented.  This is an iterative process with much communication taking place between stakeholders, end users and the project team.  The following techniques can be used to gather requirements:

* Identify and capture stakeholder requirements using customer interviews and surveys.
* Build multiple use cases to describe each action that a user will take in the new system.
* Prototypes can be built to show the client what the end product will look like.  Tools like [Omnigraffle](https://www.omnigroup.com/products/omnigraffle/), [HotGloo](https://www.hotgloo.com/) and [Balsalmiq](https://www.balsamiq.com/) are great for this part of the process.

In a corporate setting, this means taking a look at your customers, figuring out what they want, and then designing what a successful outcome would look like in a new bit of software.

**Design**

Technical design requirements are prepared in this phase by lead development staff that can include architects and lead developers.  The Business Requirements are used to define how the application will be written.  Technical requirements will detail database tables to be added, new transactions to be defined, security processes and hardware and system requirements.

Let’s look in more detail at some of the activities involved in this stage:

*Risk analysis*

* Threats and vulnerabilities which may arise from interactions with other systems.
* External or legacy code needs to be analyzed to determine if there are security vulnerabilities.
* High-risk privacy projects could require review with a legal department. This review should consider what personal data to collect, how to collect it, and permissions/authorizations to make changes. This type of review is especially necessary with corporate projects.

*Functional Specifications*

* Includes a description of interface requirements such as definition of data entry fields (allow numeric or alpha only, can it be left blank?)
* Important details, like: can date entered be before current date? What timezone will user logins default to?
* Workflow – after clicking approve button, which screen appears next?
* Audit trail for every update on the database. This is where [error monitoring](https://airbrake.io/) and logging tools can be useful.

*Non-Functional Specifications*

* Extensibility of the system – will current system easily allow new enhancements or features with the next rollout? This is critical for any application that you’ll be adding new features and updating often.
* Has the current or future capacity been analyzed for database requirements? Will the current build plan result in capacity issues shortly after you finish building?
* Performance and response time – Has the expected response time been determined?
* Resource Constraints – Are there constraints that need to be taken into consideration in this phase? Common ones include disk space, bandwidth, etc.

**Coding**

This phase is the actual coding and [unit testing](https://www.slideshare.net/assembla/assembla-airbrake-webinar-production-monitoring-and-continuous-delivery-may-2013) of the process by the development team.  After each stage, the developer may demonstrate the work accomplished to the Business Analysts and tweaks and enhancements may be required.  It’s important in this phase for developers to be open-minded and flexible if any changes are introduced.  This is normally the longest phase of the SDLC.  The finished product here is input to the Testing phase.

**Testing**

Once the application is migrated to a test environment, different types of testing will be performed including integration and system testing.  User acceptance testing is the last part of testing and is performed by the end users to ensure the system meets their expectations.  At this point, defects may be found and more work may be required in the analysis, design or coding.  Once sign-off is obtained by all relevant parties, implementation and deployment can begin.  
 **Implementation/Deployment**

The size of the project will determine the complexity of the deployment.  Training may be required for end users, operations and on-call IT staff.  Roll-out of the system may be performed in stages starting with one branch then slowly adding all locations or it could be a full blown implementation.

One of two methods can be followed in a SDLC process.  Waterfall is the more traditional model and has a well structured plan and requirements to be followed.  This method works well for large projects that may take many months to develop.  The Agile Methodology is more flexible in the requirements, design and coding process and is very iterative.  This process works best for smaller projects and expectations of continuous improvement to the application.  Whether you use one over the other will also depend to a large extent on the corporation and skills of the IT dept.