The *System Development Life Cycle*

What is the Project Life Cycle?

When a Systems Analyst develops a new information system or augments an existing system, it will require a number of different but related sets of activities. The sets of activities can be grouped into what we call ***phases*** which are groups of tasks that are trying to reach a similar goal such planning, analysis, design, build, testing, implementation, and support. When you combine all the phases of a project, we call it the *System Development Life Cycle (SDLC)* or the *Project Life Cycle.*

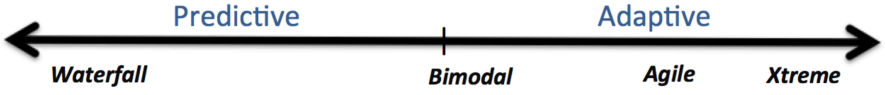
It's possible you've already heard about life cycles in previous classes or in job experience. *Agile* is a type of project life cycle as is *Waterfall.* Our goal in this reading is not to endorse one type but to present a clear definition of common types, their rationale, and the positives/negatives of both so that you can better approach your projects when planning and executing them. Most SDLCs contain the following phases and goals:

- ***Project Initiation*** – identify the problem and secure approval to develop the system
- ***Inception/Planning*** – this involves planning, organizing, and scheduling the project
- ***Analysis*** – discover the problems and their root causes in order to come up with the best solution that will support the business process.
- ***Design*** – this phase focuses on configuring and structuring the new system components based on the requirements gathered in the planning/analysis phases.
- ***Build/Construction*** – while some life cycles include build and test as part of Implementation, we will break them out so you understand the differences between them. In this phase we are realizing the designs by actually building and coding the solution. Typically in this phase Unit Testing and Assembly Testing will occur.
- ***QA/Testing*** – in many SDLCs, Quality Assurance (QA) is typically included in Build. One shouldn't decouple Build and Test. They should always be one and Build without Test should not occur. Sometimes people created an additional phase of testing after unit and assembly that includes a more thorough end-to-end system, user testing, security testing, performance testing, load testing, etc…
- ***Implementation*** – deployment and putting the system into operation in the production environment.
- ***Support*** – identify bugs and potential improvements/upgrades, prioritize changes/fixes, and implement.

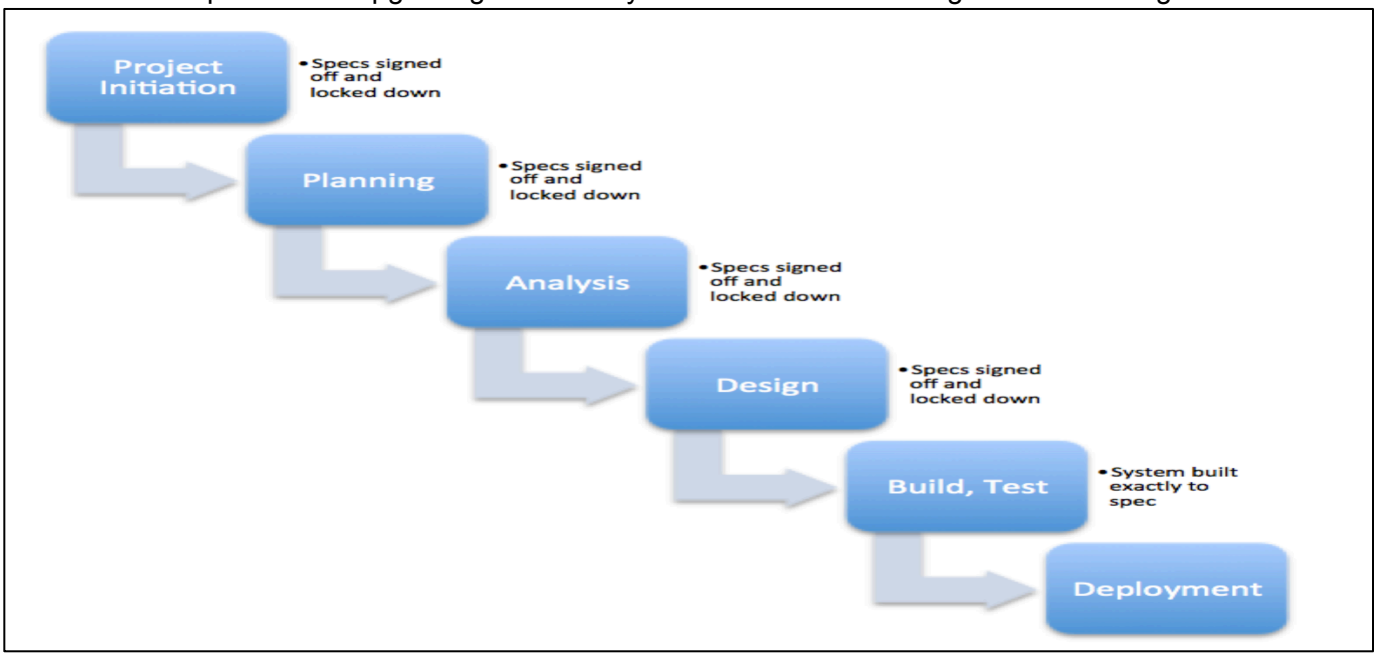
What are types of Life Cycles?

Many people when defining the types of life cycles will simplify it into two approaches: Waterfall vs Agile. In reality it's a bit more complicated than just choosing between these two approaches. Also, it's important to note that both types of life cycles contain the phases we've noted above. The key difference is the way each embrace change in the project and how the work (or scope) is broken up in the project. Also there are many flavors of what we call "Agile" projects. What we will do here is break down projects in two approaches: *Predictive* and *Adaptive.*

The ***Predictive Approach*** to a project assumes that we can predict all the steps we need to take and therefore devise an approach or plan based on this. The most common predictive method is known as ***Waterfall***. This method is an SDLC approach that assumes the phases can be planned sequentially at the start of the project and then executed with little to no overlap. The **Adaptive Approach** accepts the likelihood that project scope can and will change due to a number of reasons (changes in personnel, business process, priorities, etc…). To deal with this unpredictability it breaks the scope into smaller pieces and deploys more ***iteratively***. In some cases, Xtreme program can deploy multiple times a day, while Agile deploys in sprints every 2-4 weeks. The Bimodal approach looks at ways to embrace some predictive structure while also using "agile" principles.

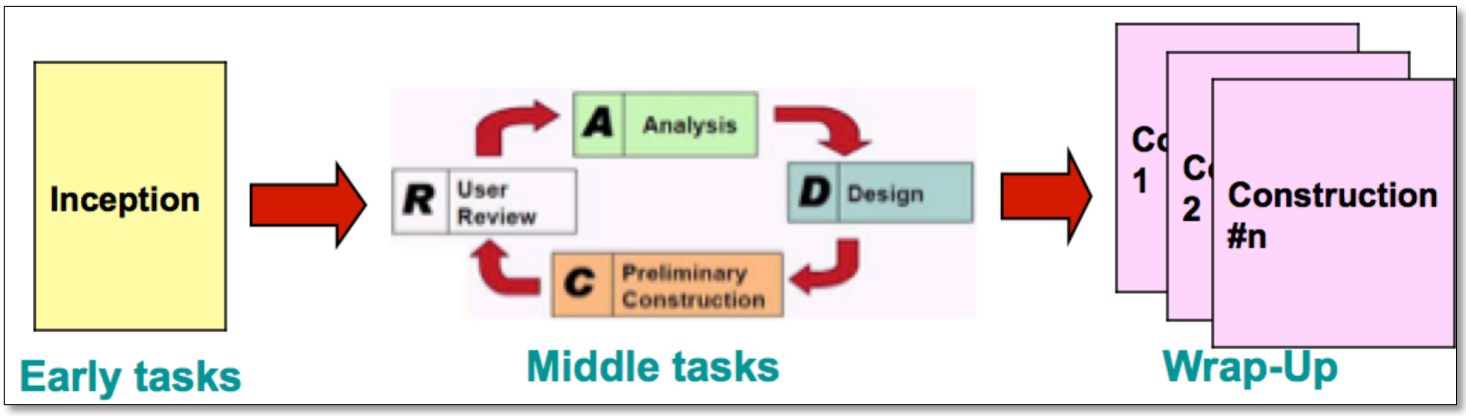


The Waterfall model of SDLC looks something like the figure below. Typically after each phase ends the specifications are locked down and must be signed off before the project can move into the next phase of work. A common issue with this approach is when a scope during a downstream phases (i.e. Build) changes due to a number of reasons, hereby invalidating the work done in previous phases. This causes slowdown and a need for rework which is likely to push the scheduled go-live of your project back and/or cause budget overages. It's important to note that waterfall does work well when projects are shorter in length and the scope is clearly known for example: You're upgrading the OS of your web servers or doing a database migration.



In this class we'll be embrace a more *adaptive approach* that we call an **Iterative Method** many times scope is a moving target. *Iterative* is a term that describes breaking up the project scope is to smaller chunks that are easier to design and build than it would be to complete and deploy all the scope. Agile is actually a type of "iterative" method. The reason we don't strictly push Agile in this class is that Agile thrives on constant and daily communication, which may be difficult in this class. That doesn't mean we can embrace principles of Agile and implement agile concepts. If you google "Agile SDLC", you'll likely find something that looks similar to what we have below. All SDLCs will have some initial inception phase to identify the first sprint or work to complete. In the iterative method we'll not be having 1 big deploy but ***n*** smaller deployments which can be less risky since it builds in repeated testing and learning from constant deployment.

MIS 374 Iterative Method



You may be asking yourself, “Why doesn’t everyone just use Agile or Iterative methods?”. The reason is not because there is a lack of documentation and books about Agile. In fact the reason many companies struggle to embrace Iterative is that many IT people have been trained in the predictive manner. This is hard to break this pattern, so the Iterative approach much be a cultural change within an IT organization and full embraced. Also it’s important to note that not only does your IT group need to embrace the “agile mentality” but your business users must embrace it too. If your IT team is used to 2-4 week sprints but your business users would rather just do requirements gathering sessions a couple times a year, this can’t work. Being “agile” means embracing frequent communication so if you are to be iterative it has to be a complete cultural change in your organization.