Chapter 2 The Systems Development Life Cycle
Dr. Degan Kettles

Learning Objectives:
Be able to explain the SDLC to non-IS people
Become familiar with the high-level steps in the SDLC
Be able to estimate the duration of a project or phase based on industry standards

The Systems Development Life Cycle (SDLC) is a set of phases necessary to design and deliver an IT solution to an organizational problem. The phases that make up the SDLC are: Planning, Analysis, Design, Implementation, Support (See Figure 1).

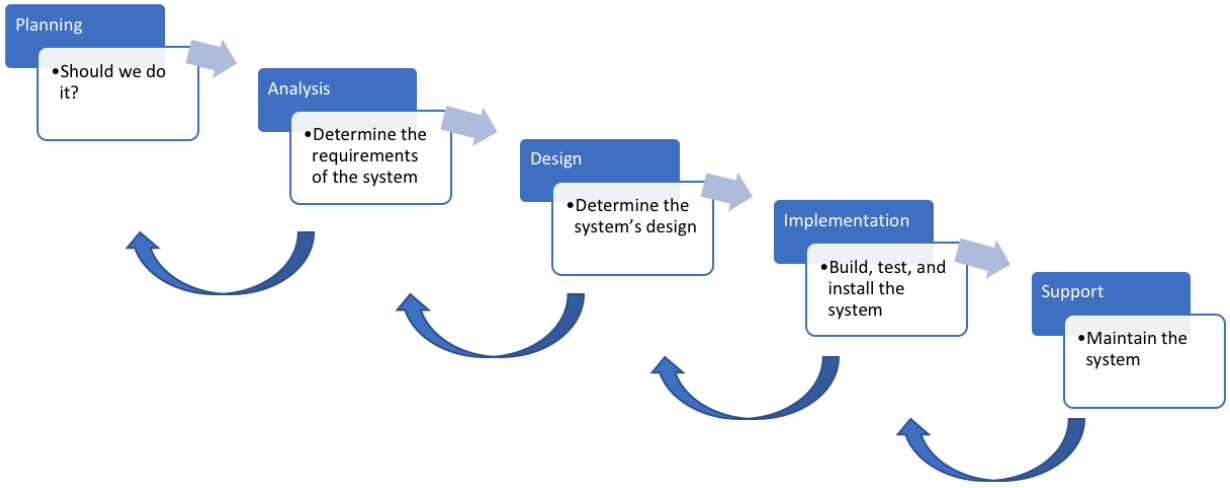

Figure 1. The Systems Development Life Cycle (SDLC)

This list of items varies slightly from author to author in the existing body of literature on the subject, but the basic ideas remain the same. Each of these words has special meaning when discussing the construction of solutions to organizational problems, and each of these words will be discussed later in detail. At a high level, however, these words imply a process that begins with deciding if you should spend time on the problem (or opportunity), digging deep enough to understand what it would take to provide a solution to the problem, designing a solution to the problem, building and deploying the solution, and then supporting the deployed solution.

To further understand the SDLC, it is also helpful to know the proportion of a project's total time that is spent in each phase. Although the exact amount of time will change from project to project, an average for each phase is shown in Figure 2. This means that whether a project takes 10 days or 100 days to complete, the proportion of time spent in an individual phase should remain relatively similar. Using these proportions, it is possible to estimate the time it takes to complete a project if given any single proportion. For example, if you know that it took one day to do planning, then you could estimate that that project would take 10 days to complete the project. If x is the days it took to complete the project that you are trying to determine, then .10*x=1.

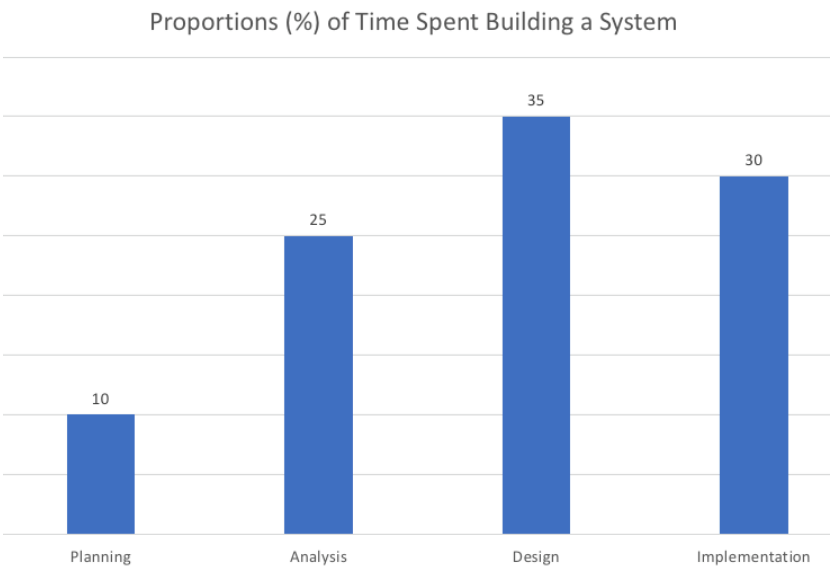

Figure 2. Average Proportions of Project Time Spent in Each SDLC Phase

Building Software Is Like Constructing a Building

Although this process may be clear enough already, I find that talking about a completely different topic, building construction, is helpful in understanding the reasons for the existence of the SDLC. I'll use the story of a house I had built to drive home some points. A few years ago, my wife and I decided to build a new home. We wanted to live in a particular area for two key reasons: the area was close to our friends and we wanted to make a lot of money by buying in the hottest area of town that was going up in value quickly. Our first step was to visit a model home. Of course, we wanted the model home, because it was perfect, but it wasn't available for sale. Instead, there began an extremely long process of making choices that lasted about eight months. First, the salesperson wanted us to pick which lot of land we wanted. Then they presented us with a list of floorplans, or in other words they gave us about 5 styles of houses to choose from that they would be willing to build on the lot. Eventually we made the first two decisions and then the salesperson sent us to a different office that they called the design center. Another long list of choices awaited us. What color do you want the house? What stone choice do you want on the exterior of the house? What carpet style and color do you want? What type of counters do you want? Which door color do you want? What decorative plumbing fixtures do you want in the house? What paint color would you like to use? Would you like to pay an extra $10,000 dollars to have the basement ceiling raised from 8 feet to 9 feet? The questions they posed went on and on like this.

In addition to making choices based on personal preferences, we also were faced with economic dilemmas. Every single feature from carpet to paint had a range of prices associated with it, and picking all the best features in every case could have raised the price of the home by $100,000 dollars, so we had to determine a budget and make trade-offs with our choices.

Over the course of eight months our home was eventually built and we marveled at all the steps that the builder took care of. They scraped off the land, ran utilities to the property, stubbed in water and waste lines on the ground, and then poured concrete over those lines. They managed

completely independent teams of contractors to do each of the following: prepare the ground, pour concrete, frame wooden walls, add electrical wiring to walls, add water and sewage lines in the walls and floors, put drywall on walls, paint walls, hang cabinets, install counters, insert decorative plumbing fixtures, lay carpet, and more. See Figure 3 as an example of the high-level tasks that a builder might manage.

Work Item	Vendor	Labor	Equipment	Materials	Subcontr.	Subtotal	Markup %	Markup	Total
Permits/Fees	City of Los Angeles				$1,500.00	$1,500.00		$0.00	$1,500.00
Excavation		$6,000.00	$8,000.00	$500.00		$14,500.00	15.00%	$2,175.00	$16,675.00
Utilities		$3,500.00	$2,500.00	$2,750.00	$1,000.00	$9,750.00	15.00%	$1,462.50	$11,212.50
Water Well						$0.00		$0.00	$0.00
Septic Tank						$0.00		$0.00	$0.00
Foundation	Connie's Concrete				$3,500.00	$3,500.00	5.00%	$175.00	$3,675.00
Concrete Flatwork	Connie's Concrete				$1,900.00	$1,900.00	5.00%	$95.00	$1,995.00
Framing		$3,500.00	$1,500.00	$9,000.00		$14,000.00	15.00%	$2,100.00	$16,100.00
Roofing	Robert's Roofing				$3,500.00	$3,500.00	5.00%	$175.00	$3,675.00
Windows/Ext.Doors	Wally's Windows				$8,000.00	$8,000.00	5.00%	$400.00	$8,400.00
Garage Door	Gary's Garage Doors				$2,250.00	$2,250.00	5.00%	$112.50	$2,362.50
Siding						$0.00		$0.00	$0.00
Electrical	Ernie's Electric				$18,500.00	$18,500.00	5.00%	$925.00	$19,425.00
Plumbing	Mac's Mechanical				$16,500.00	$16,500.00	5.00%	$825.00	$17,325.00
HVAC	Mac's Mechanical				$23,000.00	$23,000.00	5.00%	$1,150.00	$24,150.00
Insulation		$3,500.00		$1,000.00		$4,500.00		$0.00	$4,500.00
Masonry	Mason's Masonry				$14,500.00	$14,500.00	5.00%	$725.00	$15,225.00
Drywall	Doug's Drywall				$12,500.00	$12,500.00	5.00%	$625.00	$13,125.00
Interior Trim	Doug's Drywall				$9,000.00	$9,000.00	5.00%	$450.00	$9,450.00
Painting	Paul's Painting				$13,500.00	$13,500.00	5.00%	$675.00	$14,175.00
Floor Coverings	Carl's Carpets				$16,500.00	$16,500.00	5.00%	$825.00	$17,325.00
Cabinets	Ken's Kabinets				$22,500.00	$22,500.00	5.00%	$1,125.00	$23,625.00
Appliances	Abby's Appliances	$2,500.00		$11,500.00		$14,000.00	15.00%	$2,100.00	$16,100.00
Landscaping	Sonny's Sodding				$2,750.00	$2,750.00	5.00%	$137.50	$2,887.50
Overhead Costs		$10,000.00				$10,000.00	20.00%	$2,000.00	$12,000.00
Other						$0.00		$0.00	$0.00
						$0.00		$0.00	$0.00
						$0.00		$0.00	$0.00
						$0.00		$0.00	$0.00
						$0.00		$0.00	$0.00
						$0.00		$0.00	$0.00
TOTALS						**$236,650.00**	**7.71%**	**$18,257.50**	**$254,907.50**

Figure 3. Sample Schedule of Construction for a Home Builder[1]

As time to move into the house approached, we noticed that some things in the house weren't done as they needed to be. In the family room, the ceiling in the room was sagging downward. In another room, the intersection of two walls was uneven and zig zagged back and forth. The builder wanted to leave things that way, but we said we wouldn't buy the house that way, and so they fixed it.

Eventually we did buy the house, but it wasn't really done yet. We didn't have a lawn, so we needed to get that put in and then water and mow it regularly. The basement wasn't finished and so we started adding rooms ourselves. In short, the maintenance phase of home ownership began as soon as the house was in our possession. If you don't yet own a house, you'll realize that maintenance is a daily aspect of home ownership.

So, what were the points of this story that relate to systems development? In answering that, I'd like to flip the perspective from that of myself as the home buyer to that of the home builder. Consider what they did, what their risks were, and what they had to manage. Their process

[1] https://commons.wikimedia.org/wiki/File:Estimating_Spreadsheet.png

started by understanding the problem that we, the customers had. We wanted to live close to our friends and get a house that would appreciate in value. The builder had to worry about a lot of things: What was our budget? What features did we want in our house? How could he build a house like the ones he was showing us in the pictures (he needed blueprints of the internal structure)? How could he simultaneously coordinate completely different sub-contracting companies across an *entire* neighborhood on a variety of different tasks? How could he ensure that the completed house was what we wanted so that we didn't walk away from the deal?

The preceding questions and issues relate closely to what any IS professional faces on a daily basis. Organizations have individuals in them (or external to them if we consider their customers) that face problems (or opportunities), and solutions need to be provided to them. Those solutions require starting at the concept phase, architecting a solution, building a solution, and then supporting a solution. Regardless of whether the organizational problem is small or large, critical thinking in each phase of the SDLC will facilitate good outcomes.

The Phases of the SDLC summarized.

The following is a brief summary of what happens in each phase of the SDLC. These are not intended to be comprehensive because they will be greatly expanded upon (you'll learn about alternatives for performing each phase) in upcoming chapters. To make the description of each phase less abstract, I'll relate the phases of the SDLC to the ride sharing application Uber.

The first phase in the SDLC is ***Planning***. This sounds like it would mean creating some type of mockup or blueprint, but in the SDLC it means something completely different. In this context it means doing whatever *due diligence* is needed in order decide whether to invest any further time in addressing the problem. Frequently this phase involves an end user submitting a formal written request for an IT department to do a project for them. A screening committee is likely to evaluate a pool of submitted requests for IT solutions, and it will make a decision about which ones to pursue based on the merits of the requests. The approaches they use to evaluate requests include performing a potential cost/benefit analysis, analyzing strategic priorities, or evaluating risk exposure (such as cybersecurity issues). It is possible that a request will be approved or even rejected (See vignette 1). If it is approved, it goes to the next phase of the SDLC, which is analysis. If it is not approved, no further work is done. See Figure 4 for an overview of the Planning Phase.

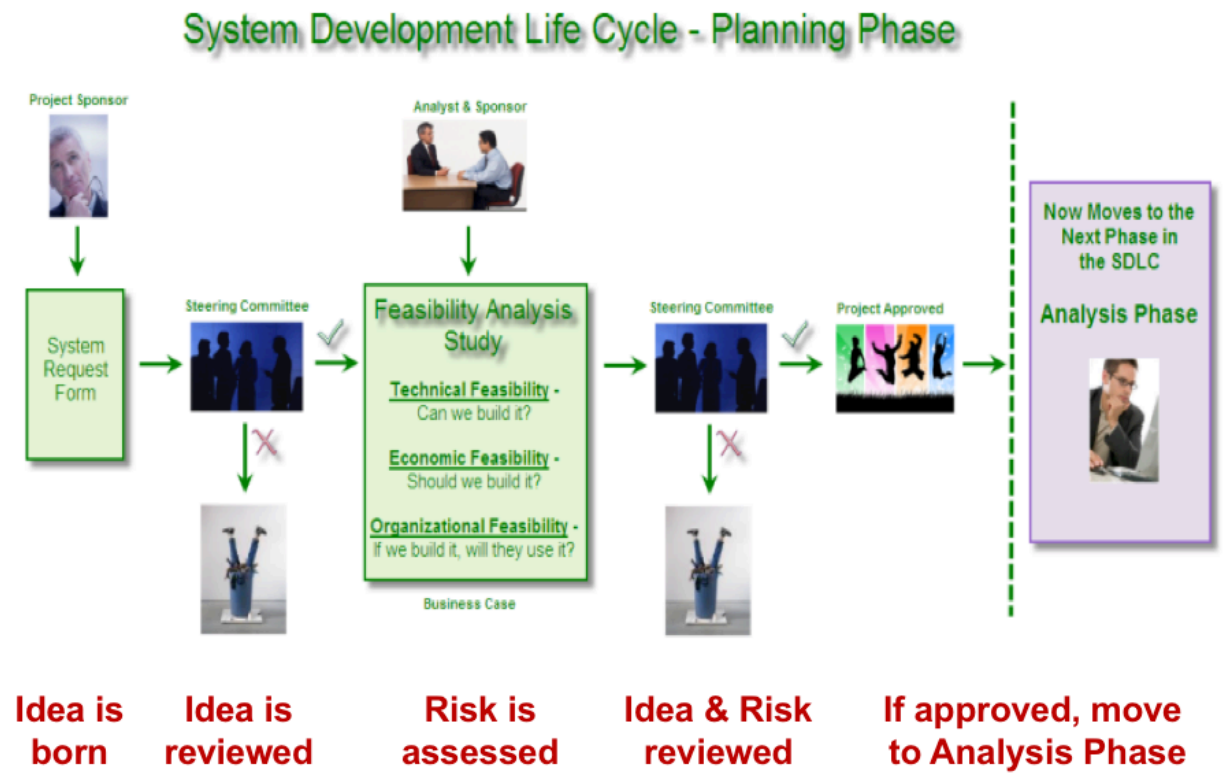

Figure 4. How a Request Gets Approved or Denied in the Planning Phase

Vignette 1: The Internal University Project. Once while working for a division within a university, a manager asked me develop an application that would reduce the amount of time her employee spent on a particular task. I estimated that it would take me three to four weeks to build a solution to her problem. Before beginning, however, I interviewed the employee to determine several facts including how much time she spent usually performing this task. It turned out that she spent no more than three days a year on the task, and the need to perform the task was going to go away within three years. I went back to the manager and explained how the cost/benefit of the solution made it unwise to pursue. She agreed and the project ended there.

If Uber were developed by proceeding through the steps of the SDLC, the planning phase for Uber would involve starting with a high-level explanation of what the business model would be as well as the application that would support it. Then, a cost/benefit case would be made for investing a certain amount of time and money into building the application. Estimates of income

would be evaluated against estimates of cost. Once it was concluded that the benefit would outweigh the cost, the next phase of the SDLC would be begin, Analysis.

The second phase of the SDLC is ***Analysis***. Analysis in the SDLC is the process of determining the features and functionality of a software solution to be delivered. This is also sometimes called determining the *requirements* of a system. Typically, the process of determining requirements involves conducting interviews, collecting relevant screenshots/reports/documents, and the creating visual models that capture different aspects of a current system (if one exists) including processes and data content (See Exhibits 1-4). For example, if an organization needs to reengineer an internal process that requires a dozen workers and hundreds of steps to complete, then interviews followed by diagrams showing process flows and data entities will be required to capture and effectively communicate the current state of the process (or work system). A written narrative of such a complex process is often insufficient for understanding (not to mention completely unavailable in most cases). Using models and diagrams to understand a current system is really what Analysis is and what makes a person an IS Analyst. Learning the diagramming and modeling techniques required for complex business system analysis is central to a course of study in Systems Analysis and Design.

Once analysis is completed, no actual software has been created, but far more accurate estimates of actual costs and time frames to completion can be created at this point because there is less uncertainty about what the requirements are.

Exhibit 1. Interviewing a person to understand their work processes[2]

[2] Source: http://allhands.coastguard.dodlive.mil/2014/03/11/evaluations-theyre-not-for-you/

Form **W-9**
(Rev. November 2005)
Department of the Treasury
Internal Revenue Service

Request for Taxpayer Identification Number and Certification

Give form to the requester. Do not send to the IRS.

Print or type See **Specific Instructions** on page 2.

Name (as shown on your income tax return)

Business name, if different from above

Check appropriate box: ☐ Individual/ Sole proprietor ☐ Corporation ☐ Partnership ☐ Other ► ..... ☐ Exempt from backup withholding

Address (number, street, and apt. or suite no.) | Requester's name and address (optional)

City, state, and ZIP code

List account number(s) here (optional)

Part I Taxpayer Identification Number (TIN)

Enter your TIN in the appropriate box. The TIN provided must match the name given on Line 1 to avoid backup withholding. For individuals, this is your social security number (SSN). However, for a resident alien, sole proprietor, or disregarded entity, see the Part I instructions on page 3. For other entities, it is your employer identification number (EIN). If you do not have a number, see *How to get a TIN* on page 3.

Note. If the account is in more than one name, see the chart on page 4 for guidelines on whose number to enter.

Social security number
___ ___ ___ – ___ ___ – ___ ___ ___ ___

or

Employer identification number
___ ___ – ___ ___ ___ ___ ___ ___ ___

Part II Certification

Under penalties of perjury, I certify that:

1. The number shown on this form is my correct taxpayer identification number (or I am waiting for a number to be issued to me), and
2. I am not subject to backup withholding because: (a) I am exempt from backup withholding, or (b) I have not been notified by the Internal Revenue Service (IRS) that I am subject to backup withholding as a result of a failure to report all interest or dividends, or (c) the IRS has notified me that I am no longer subject to backup withholding, and
3. I am a U.S. person (including a U.S. resident alien).

Certification instructions. You must cross out item 2 above if you have been notified by the IRS that you are currently subject to backup withholding because you have failed to report all interest and dividends on your tax return. For real estate transactions, item 2 does not apply. For mortgage interest paid, acquisition or abandonment of secured property, cancellation of debt, contributions to an individual retirement arrangement (IRA), and generally, payments other than interest and dividends, you are not required to sign the Certification, but you must provide your correct TIN. (See the instructions on page 4.)

Sign Here Signature of U.S. person ► Date ►

Exhibit 2. Example of a form that an analyst may collect and study

Exhibit 3. Analyzing a flowchart[3]

[3] Source: Src. https://pixabay.com/en/whiteboard-man-presentation-write-849810/

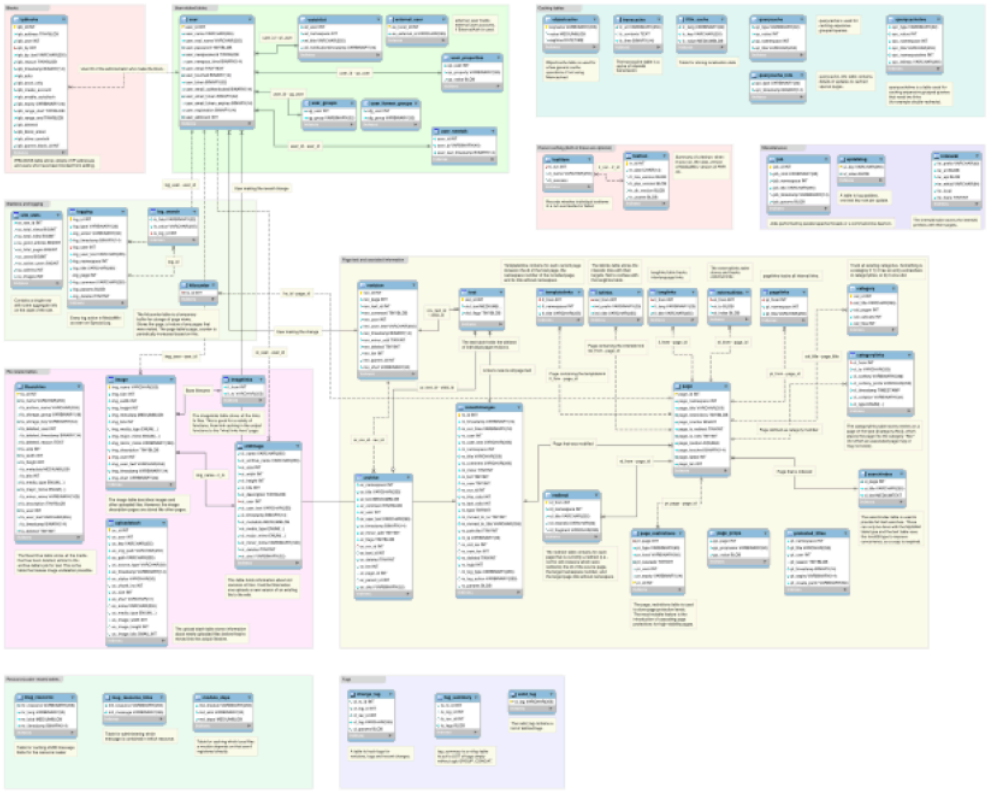

Exhibit 4. A Database Schema from an Existing System to Review

Vignette 2: The Sales Team Process. One organization that sold products worth hundreds of thousands of dollars faced the problem of not being able to see the status of their sales process (gotta close those sales you know or you don't get paid!) related to customers. Each staff member involved in making a sale stored their data about the customer in a different system. I spent several weeks up front interviewing different staff members, documenting their narratives, and then making diagrams that showed how they did their jobs. At the end of this process I analyzed the effectiveness of the processes shown in the diagrams with the sales division manager. The conversation began like this:

Me: Here is how you do this particular process.
Them: No, we don't.
Me: yes, you do.
Them: No, we *don't.*
Me: yes, *you do.*
Them: Ooooh.

Across many consulting engagements I found this to be the case, that managers are often unaware of how their employees actually get work done, and a correct understanding of how they do things is needed to build a satisfactory solution to any problem.

Vignette 3: Estimating Project Costs. All business customers (those paying for the system) want to know how much money a project will cost and when it will be done. Typically, however, the high-level idea discussed in planning doesn't provide enough information to answer that accurately. For example, a broker of mortgages once wanted me to make an application that would display and compare loan options for his clients. This was a complicated task and website that he wanted built, but it took a full eight hours of discussion and mockups to determine what a viable solution would entail. Customers never want to pay for that time, because to them it doesn't count as work. However, that time was needed to create the estimate of future work. Speaking more generally, sometimes when working in the consulting space and after planning and analysis phases were completed (comprising 35% of the theoretical completed project's time and effort) clients decided that the quoted cost to actually build their desired solution was too high and they walked without ever paying a dime.

If Uber were developed by proceeding through the steps of the SDLC, the analysis phase would likely proceed as follows. As a new startup they would not have existing software systems to diagram, but it would likely develop UI mockups as models of the software they want to build. Iterations on those mockups would clarify the features that needed to be created. Written documentation would accompany the mockups in order to clarify how many visual components would work when they were clicked on. Any given activity performed through the software involving the logical flow of steps that happen among riders, drivers, and system administrators could be clarified using diagrams (e.g. normal rides, payment, refunds, canceled rides, etc.).

The third phase in the SDLC is ***Design***. We include in this phase the determination of whether to build, buy, or lease a solution based on the requirements developed in the analysis phase. If it is determined that building is the current course of action, then models pertaining to system design work need to be created. Commonly these include not only the User Interfaces, but also software models such as UML Class and Sequence diagrams, as well as software design decisions such as which software design patterns (or other project standards) will be followed. See Figure 5 as an example of UML Class diagram used as a blueprint to write software code.

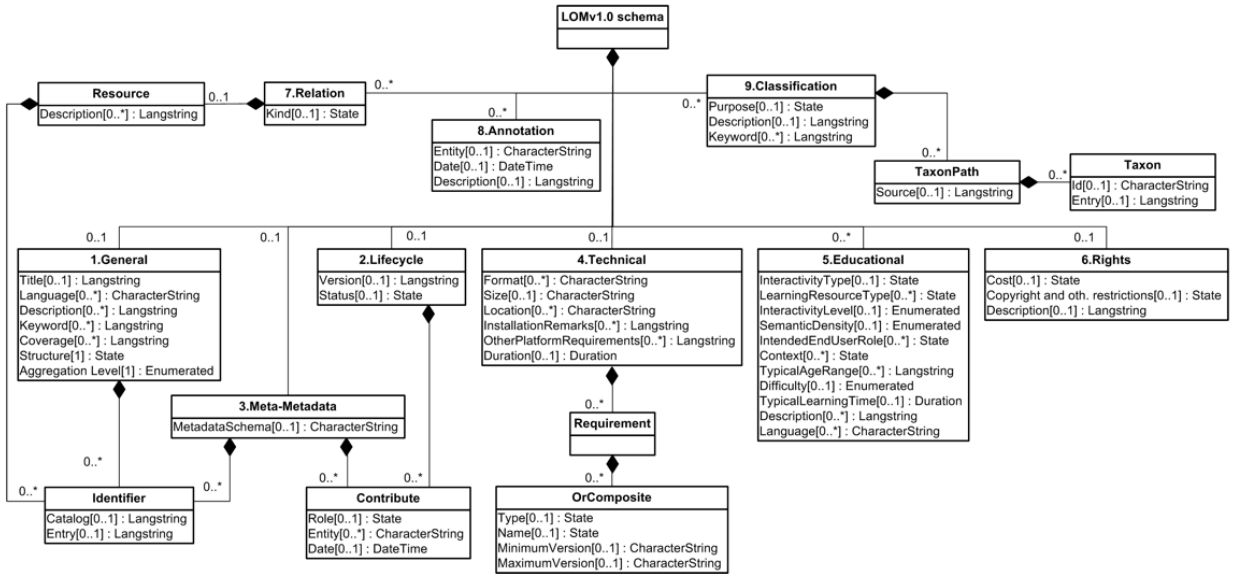

Figure 5. UML Class Diagram Example[4]

If Uber were developed by proceeding through the steps of the SDLC, the design phase would include finalization of the User Interfaces. It would also involve creation of UML Class and Sequence diagrams as blueprints for software code. It would include formally deciding where software languages including Python, Node.js, Go, Java, C, and C++ would be used in their technology stack according to where they would be most appropriate and beneficial. It would involve overarching architectural decisions including the software design patterns to use and the plans for microservices.

The fourth phase of the SDLC is ***Implementation***. Implementation in the SDLC is the process of writing code, testing code, creating deployment plans, allocating and configuring hardware resources such as servers, and physically deploying the software into its production environment. Figure 6 shows an example of a File Transfer Program (FTP), which can be used to upload software code to remote Linux or Windows Servers.

If Uber were developed by proceeding through the steps of the SDLC, the implementation phase would include building out their data center to support the expected traffic they would receive. It would also involve leasing cloud resources in Google's infrastructure as well as Amazon's for additional needed services. It would involve writing code, putting that code in code repositories, writing and performing continuous code testing, and pushing code into production environments.

[4] Source: https://commons.wikimedia.org/wiki/File:LOM_base_schema.png

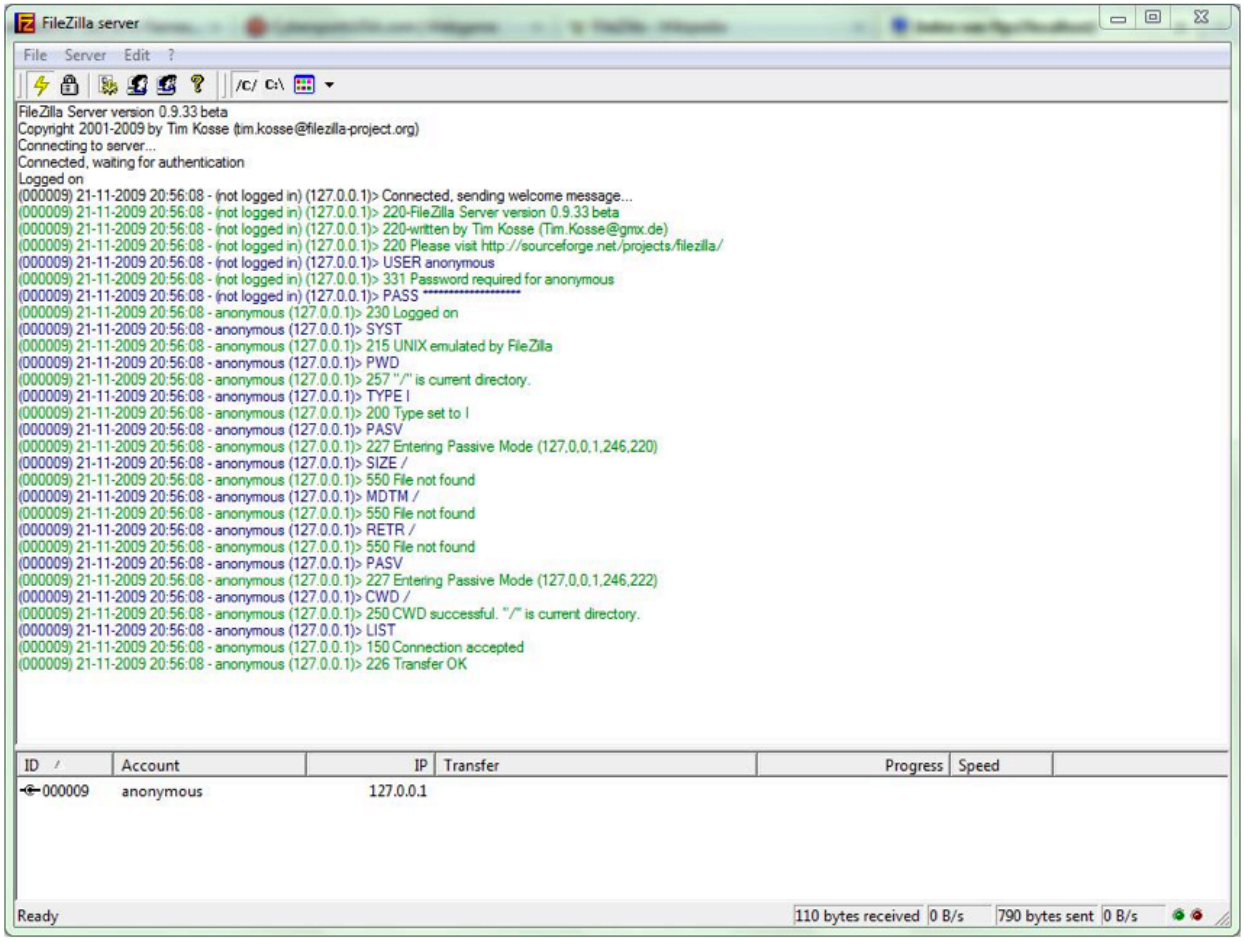

Figure 6. Example of File Transfer Protocol Program (FTP)[5]

The fifth phase of the SDLC is ***Support***. Supporting software includes training end users, maintaining a system, and improving it. The reasons for training users and adding features should be obvious, but the need for resources to maintain a system may be less obvious. Even if nothing changes in the codebase, the hardware that the software runs on will need to be cycled out every few years. Also, bugs will be discovered that need addressing. Often the operating systems, databases, and code libraries need to be updated due to the discovery of security vulnerabilities. All too often these updates cause the original codebase to lose functionality. It is also the case that 3rd-party APIs to services like credit card processing services will change, meaning that an organization outside of your own will force your hand at updating your application so that it will continue to run as before. Another common issue is that as the user base increases, the code needs to be re-written so that it can scale. An example of this is eBay, the original developer wrote the first version using a Perl script over a weekend. That is not the codebase they run on today ;-). Another perhaps comical way to look at the whole support situation is that if support weren't needed, all the developers could be laid off work after deployment happens, and yet if you look at any mature company such as Facebook, they only seem to add developers to the teams working on a given application. It is often said in the

[5] https://commons.wikimedia.org/wiki/File:FTP_Session_in_FileZilla_Server.jpg

software industry that maintaining software costs more than developing it for the first time (See Vignette 4).

Vignette 4: Supporting a Work Flow Application. Once I had a team build an application that would allow the flow of work to be managed as it passed through a series of steps performed by different employees. Specifically, the flow of work related to tasks that needed to be completed to optimize websites for search engine traffic. A senior manager who had recently come from managing software development at a billion firm was highly critical of my team having built the application. He said in essence that managing a flow of tasks was so common to businesses that there was likely already a software package written (commodity software for business) that could have performed the same functionality. The big issue, he emphasized, was that even if it only took a month or two for us to write the software, that we would have to pay people to maintain it forever. There were now only one or two people in the world that knew the code well enough to maintain it. Even if they only made minor updates now and again, they got paid a lot of money and we couldn't lose them. It would be much cheaper, if possible, to just pay a one-time fee for such software, and then get annual updates either for free or for a minor cost.

If Uber were developed by proceeding through the steps of the SDLC, the support phase would include much of the massive amount of work the comprises their daily operations today. They support users in how to use their software. They have an army of people re-writing existing functionality so that it performs better, and they are adding functionality regularly.

Summary Comments about the SDLC

Keep in mind, the SDLC is a framework, and frameworks are never perfect, but they are sometimes useful. One way that the SDLC is useful is in providing a way to organize your thinking related to the numerous techniques that exist related to software development. Literally hundreds of books could be written on different approaches and tools for software development, and the SDLC allows us to reflect back and put them into a simple classification. In the upcoming chapters, I will introduce some of the most popular and useful methods used in the individual SDLC phases, but they are not comprehensive. The techniques you adopt for yourself will be based on numerous factors such as regional preferences, organizational mandates, or

personal preferences. Understanding the SDLC will also allow you to craft your own set of approaches to solving organizational problems and opportunities using IT.

Reflection:

How would you explain what the SDLC is to someone who has never taken an IS class?
What are the main phases in the SDLC?
If a project took 20 days to complete, how many days were spent on planning?
What happens in each phase of the SDLC?
Consider any software application that you are familiar with either in usage or because you were part of the development team, how was each phase of the SDLC executed?