K1

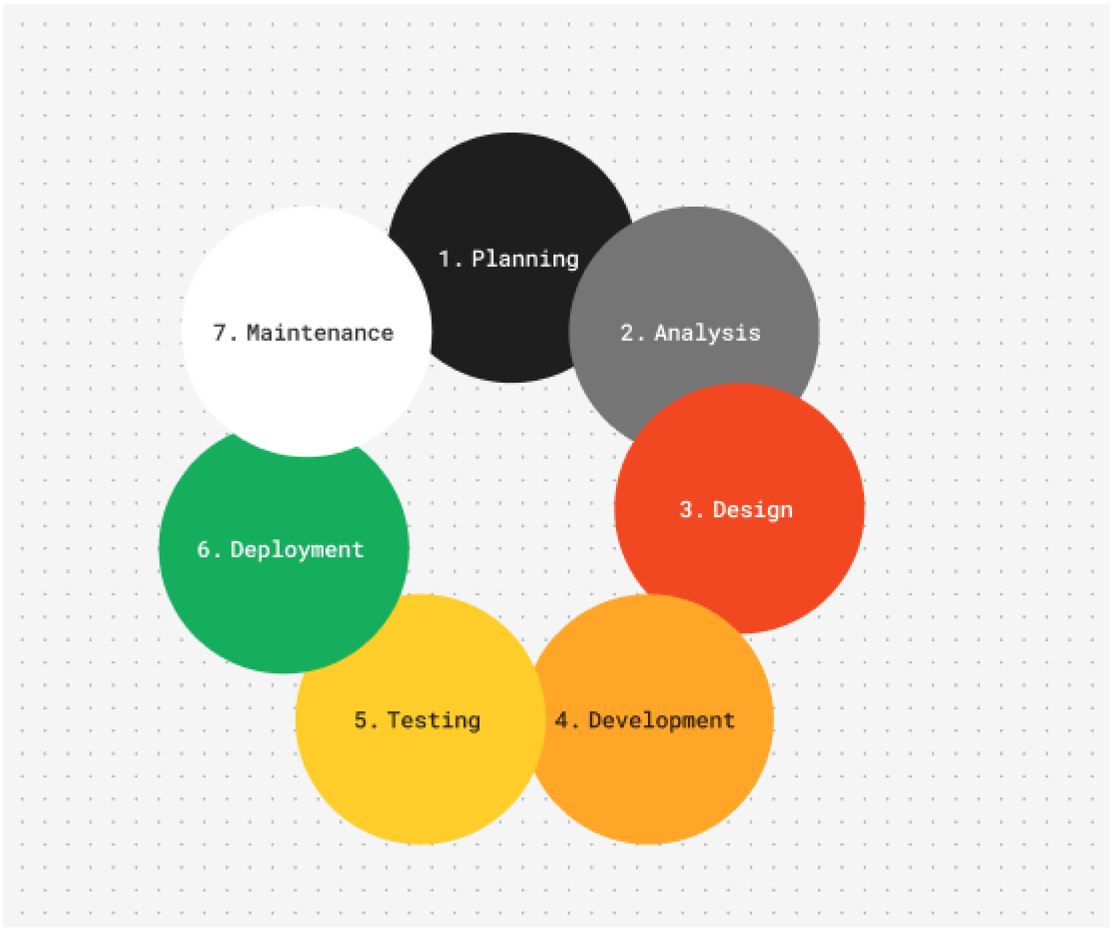
Software Development Lifecycle At Labman

Arthur Clarkson

3rd September 2024

The software development lifecycle (SDLC) is a process that enables development teams to design and build high-quality software in a cost-effective and time-efficient manner. The primary objective of SDLC is to minimize project risks through proactive planning, ensuring that the software meets customer expectations both during production and in its subsequent use. This methodology provides a structured approach by breaking down the software development process into clearly defined steps, allowing for tasks to be assigned, completed, and measured effectively.

The lifecycle can be broken down into the following phases: the planning phase - which is where the project goals, scope and requirements are defined, the analysis phase - where detailed requirements are gathered and system specifications are defined, the design phase - where the architecture and user interface are planned, the implementation/development phase - where the writing and compiling of code is performed according to the design specifications, the testing phase - which ensures that the software functions correctly and meets all requirements, the deployment phase - where the software is released to users and made operational in the production environment and finally the maintenance phase - which allows for ongoing support, such as fixing bugs, making updates and improving performance.



At work we use Salesforce as our Customer Relationship Manager (CRM), it works great, but isn't very user friendly. This is why we use their API to pull out data into our database, and use our intranet to interact with it. For example to create a lead, you can fill out all fields in a form on our intranet, and when you submit it, a method on our business layer will send a POST request to the Salesforce API to their create lead endpoint, and then we will send a GET request to their API to fetch the new lead's data, and save it locally in our database.

Our old lead form had an outdated UI and didn't allow you to edit a lead's details, with this I was assigned the task of creating a new form with an updated UI, with some extra fields that weren't available originally. This was the SDLC for this feature request:

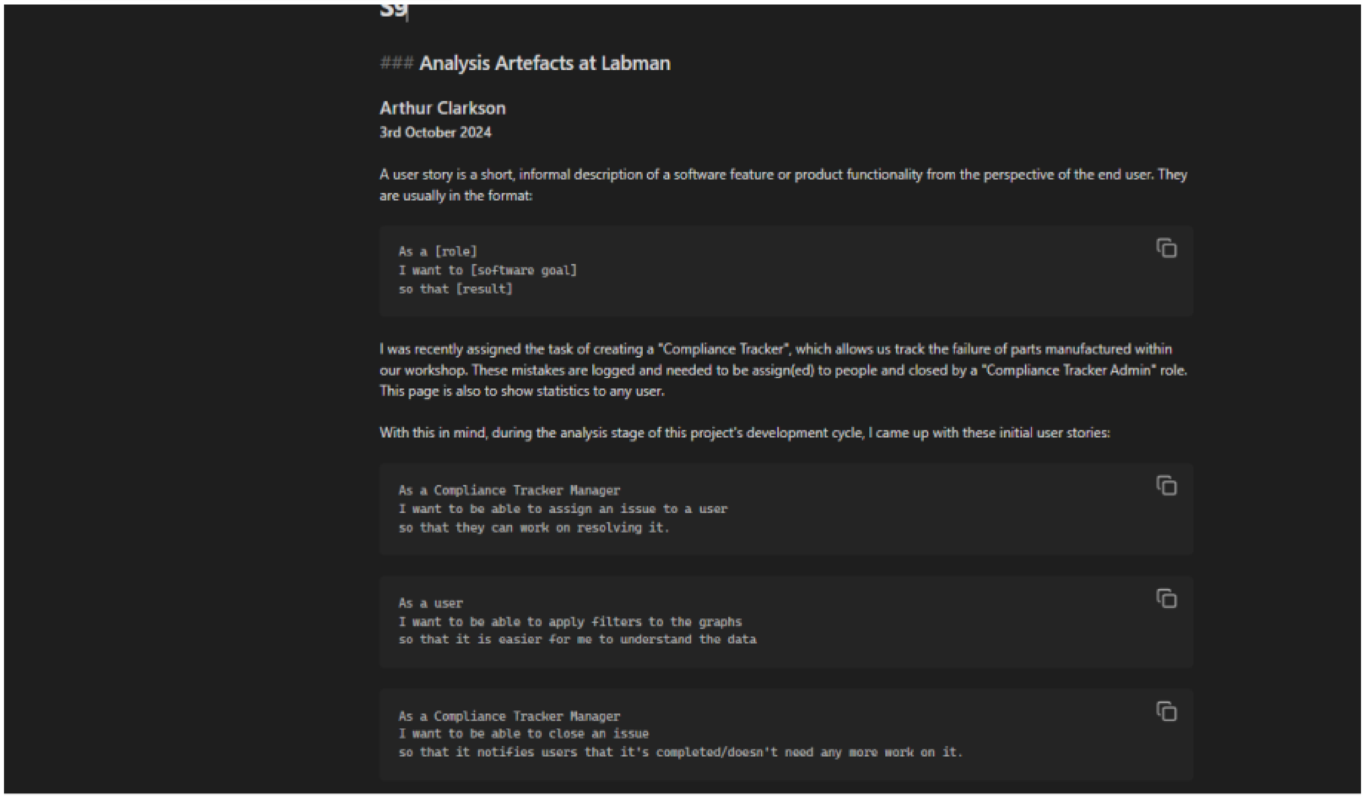
Planning phase:

Every Thursday another software engineer and I have a meeting with the technical director and a sales project manager for intranet salesforce development. They had raised the feature request on our feedback ticketing system, and I wanted to plan it with them in this meeting. We discussed what they wanted on the new form, such as the ability to link a lead to campaigns, and allow for a more "wizard based" approach to the form to make sure users fill out all fields correctly.

Analysis phase:

After this meeting, I had a chat with the other software engineer and we managed to put together the use cases and user stories required for the new feature, giving an estimated time to complete for each one.

See `Analysis Artefacts at Labman.pdf` for more on this:



Analysis Artefacts at Labman

Arthur Clarkson
3rd October 2024

A user story is a short, informal description of a software feature or product functionality from the perspective of the end user. They are usually in the format:

```
As a [role]
I want to [software goal]
so that [result]
```

I was recently assigned the task of creating a "Compliance Tracker", which allows us track the failure of parts manufactured within our workshop. These mistakes are logged and needed to be assign(ed) to people and closed by a "Compliance Tracker Admin" role. This page is also to show statistics to any user.

With this in mind, during the analysis stage of this project's development cycle, I came up with these initial user stories:

```
As a Compliance Tracker Manager
I want to be able to assign an issue to a user
so that they can work on resolving it.
```

```
As a user
I want to be able to apply filters to the graphs
so that it is easier for me to understand the data
```

```
As a Compliance Tracker Manager
I want to be able to close an issue
so that it notifies users that it's completed/doesn't need any more work on it.
```

Design phase:

Our intranet has a pretty good design system, which makes it very easy to develop standardized UIs between modules, with this I was able to put together a mock-up form with all the required fields in html/css, put it on a QA and sent it to our designer to have a look at. With there being so many fields on the form, I wasn't too sure to use a tabbed approach (allows the user to easily navigate the form to fill in an exact field) or wizard approach (allows the user to follow the form in a set path). After talking to our UI designer, he recommended that we used the wizard approach for creating a lead, and tabbed for editing a lead. With all of this I was able to develop myself a technical specification, and it was approved by our technical director.

Development phase:

Following the technical specification and the simple UI design, I was easily able to develop the code required for the task.

Testing phase:

After I'd finished development, I put it on a QA server to allow the stakeholders to have a play with the new form, if an issue/new feature was to be flagged up, I would go back to the planning stage, and decided if it was worth our time to fix/implement within the limited time we had to develop the new feature. If it was worth it, I would design, develop, then test it again.

Deployment phase:

When it was ready for deployment, I sent a deployment plan to our deployment engineer. This contained the scope of the changes, a migration script, and when to deploy it. Once it was live, I notified all sales employees of the changes.

Maintenance phase:

After it was deployed, I was available for a week to be able to fix any major bugs. But for any feature requests / minor bugs, the users were to create a feedback ticket.