Addis Ababa Science and Technology University

Course: Software Component Design

Software Development Life Cycle (SDLC) Simulation

Project Documentation

Section D

Group Members	**ID**
1. **Yordanos Negusu**	**ETS1370/13**
2. **Yoseph Zewdu**	**ETS1395/13**
3. Rediet Teklay	**ETS1093/13**
4. **Yoseph Shemeles**	**ETS1392/13**

Submission Date: 12/18/24

Submitted To: Mr. Gizatie

Overview

This project simulates the Software Development Life Cycle (SDLC) phases using Python and the **SimPy** library for event simulation. Additionally, it uses **Matplotlib** to visualize the project timeline in a Gantt chart-like bar chart.

The primary goal is to model each SDLC phase (e.g., Requirements Gathering, Design, Development, Testing, Deployment) sequentially and showcase their durations.

Project Phases

The following phases are modeled:

1. **Requirements Gathering** – Duration: 2 units
2. **Design** – Duration: 1 unit
3. **Development** – Duration: 3 units
4. **Testing** – Duration: 2 units
5. **Deployment** – Duration: 1 unit

Each phase records its start and end times, simulating time delays using `env.timeout()`.

Code Implementation

Core Logic

The project implements SDLC phases as functions within a SimPy simulation environment. Each function logs the start and end times and introduces a delay to simulate phase durations.

```
def requirements_gathering(env, team_size):
    start_time = env.now
    print(f"Requirements Gathering started at {start_time}")
    yield env.timeout(2)  # Duration: 2 weeks
    end_time = env.now
    print(f"Requirements Gathering completed at {end_time}")
    phase_times.append(('Requirements Gathering', start_time,
end_time))
```

The same structure is followed for **Design**, **Development**, **Testing**, and **Deployment**.

Simulation Setup

The SDLC phases are run sequentially in a `run_sdlc` function, utilizing the `simpy.Environment`.

```
def run_sdlc(env, team_size):
    yield env.process(requirements_gathering(env, team_size))
    yield env.process(design(env, team_size))
    yield env.process(development(env, team_size))
    yield env.process(testing(env, team_size))
    yield env.process(deployment(env, team_size))
```

Visualization

After the simulation completes, **Matplotlib** generates a Gantt chart-like bar graph representing each phase's start and end time.

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.barh(phase_names, end_times, left=start_times, color='skyblue')
ax.set_xlabel('Time')
ax.set_title('Software Development Life Cycle Simulation')
plt.show()
```

Output

Console Output

```
Requirements Gathering started at 0
Requirements Gathering completed at 2
Design started at 2
Design completed at 3
Development started at 3
Development completed at 6
Testing started at 6
Testing completed at 8
Deployment started at 8
Deployment completed at 9
```

Technologies Used

- **Python 3.x**
- **SimPy** – Discrete-event simulation library
- **Matplotlib** – Data visualization

How to Run

Install Dependencies

```
pip install simpy matplotlib
```

Run the Script

```
python sdlc_simulation.py
```

View Output

- Console logs display phase completion times.
- The timeline visualization appears as a Gantt chart.

Conclusion

This project effectively models the sequential flow of SDLC phases using simulation and provides a clear visual timeline. It can be extended further to simulate **parallel phases**, **resource constraints**, or **team-based dependencies**.