Final Project Draft

Sabbir Ahmed

November 1, 2021

1 Introduction

In software engineering, design patterns are general, reusable solutions to commonly occuring problems [1]. It is generally considered good practice to integrate design patterns into software products, especially large projects, since it allows the developers to focus their time and attention towards specific implementations. The purpose of this draft is to introduce an open-source project and present analysis on the software patterns within the implementation.

2 Structural Simulation Toolkit (SST)

The software that is being focused on in the final project is Structural Simulation Toolkit (SST). It is a simulation framework that prioritizes high performance computing (HPC) models [3]. Since SST is a large scale project with many stable extensions implemented for its kernel, the scope of the project will be limited to specific sections of the core repository. The repository is hosted on GitHub [2].

3 Software Patterns Present in SST

The following patterns can be observed to have been already implemented in the project:

1. Factory method pattern
2. Singleton pattern
3. Template method pattern
4. Builder pattern

Other patterns are present in the project, such as C++ idioms, such as Include Guard Macro,

3.1 Factory method pattern

The factory method pattern is observed in the

3.2 Singleton pattern

The singleton pattern is present in the `Factory` class. On the repository, the class can be located at `src/sst/core/factory.h`. The class is used to instantiate other concrete simulation classes. SST requires simulation objects to be synchronized throughout the kernel, especially since they can be running on a distributed system where locks can become major issues. The software forces these simulation objects to be singletons.

3.3 Template method pattern

This pattern is implemented throughout the project through C++ function templates. The synchronization subclasses, which can be found at `src/sst/core/factory.h`, are implemented as template subclasses with variadic arguments. This approach allowed for various

3.4 Builder pattern

4 Suggested Software Patterns in SST

4.1 Strategy pattern

4.2 Decorator pattern

References

[1] Design patterns. https://sourcemaking.com/design_patterns.

[2] sstsimulator/sst-core. https://github.com/sstsimulator/sst-core.

[3] The structural simulation toolkit. http://sst-simulator.org/.