**CPSC 5360\_48F\_1 - Software Engineering**

**Instructor - Dr. Stefan Andrei**

**Assignment 2**

**Automatic Document Generation**

**Due Date: April, 5th 2015**

**Submitted By:**

**Balram Agrawal (L20365062)**

**Shubhashish Mishra (L20365063)**

**Lucky Singh (L20355322)**

**Submitted On: April, 5th 2015**

**Question:**

**Continue the business model of the ‘Automatic Document Generation’ project described in Assignment 1. Based on the use cases description, sequence (or collaboration) diagrams, define a correct and complete state-chart for the ‘Automatic Document Generation’ project. Identify which design patterns are appropriate for implementing the state-chart into Java programming language.**

**Answer:**

Generally, a state in UML is a condition or situation an object (in a system) might find itself in during its life time. We capture the behavior of the subject object through modeling these various states and transitions between them. In order for others to understand them more easily, abstraction also takes place for simplicity reason in the modeling process. So it is important to note that a state machine diagram does not necessarily model all possible states, but rather the critical ones only. When we say “critical” states, we mean those that act as stimuli and prompt for response in the external world.

**State Diagram for Automatic Document Generation System:**

A state diagram is pictorial representation of various *states* of a system in response with a particular *event* or *stimuli.*

An event is a method, message or any external or internal agent to a system, and this causes change in state from A to B, let’s say as below:

e: A->B

B

A

Fig 1: representing change of state (state transition) from A to B with stimuli e

Usually, states are represented with circle and putting state name inside instead rectangular,

In software engineering, UML defines specific rules and fashions to write a complete state diagram. An object is represented as a state at a particular instant of time and its complete life cycle is presented in state diagram.

Objects responses to methods or events called from anywhere.

1. Events trigger objects from one active state to another.
2. If event triggers object to itself, then it’s as self transition.

For constructing a complete state chart for Automatic Document Generation System we go through step by step procedure. Where, we first consider a particular case and make a diagram, again consider another possible case or event and again add up in diagram, and finally we will get a complete diagram for the Automatic Document Generation system.

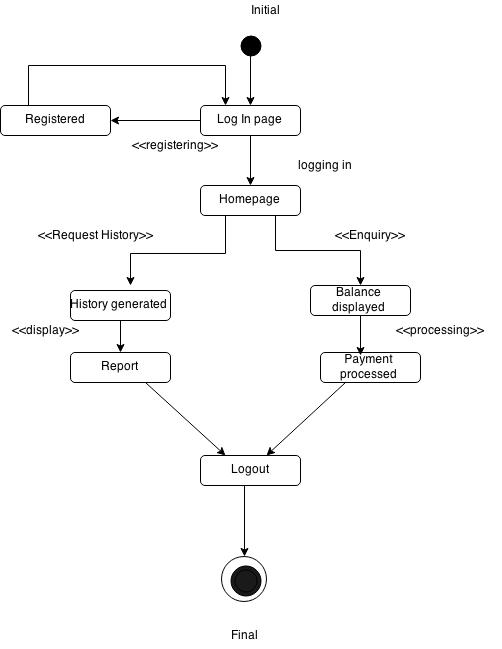


Fig 1: State chart for Automatic Document Generation system for services

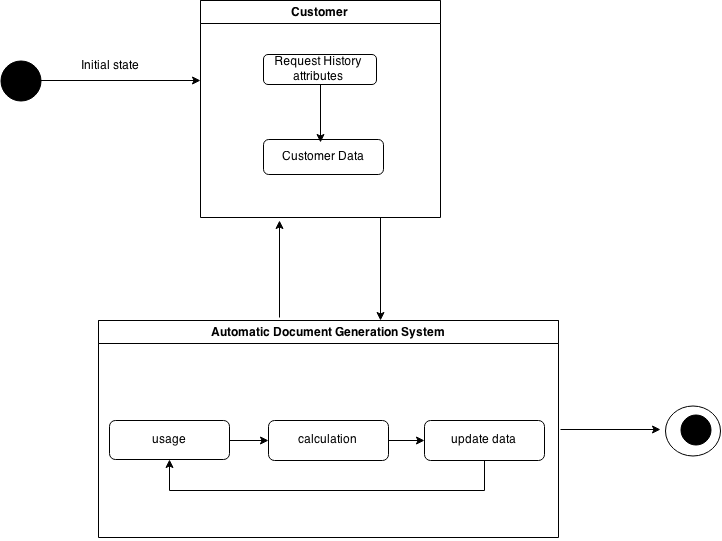


Fig 2: State chart of interaction between two groups of events

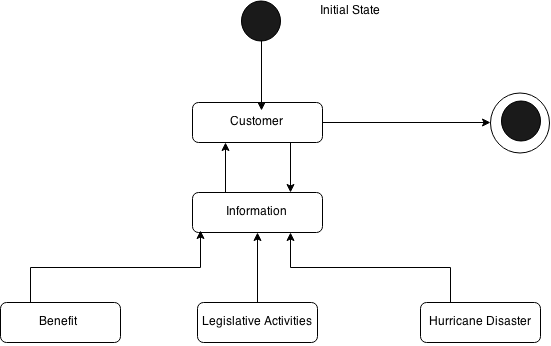


Figure 3: state chart of informing system of Automatic Document Generation System

Following three design patterns that can be used for implementation:

1. **The Façade pattern**: The facade pattern is a software engineering design pattern commonly used with Object-oriented programming. The name is by analogy to an architectural facade.

A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can:

1. make a software library easier to use, understand and test, since the facade has convenient methods for common tasks;
2. make code that uses the library more readable, for the same reason;
3. reduce dependencies of outside code on the inner workings of a library, since most code
4. uses the facade, thus allowing more flexibility in developing the system;
5. **The Player role:** A role is a particular set of properties associated with an object in a particular context. An object may play different roles in different contexts.
6. **The State pattern:** The state pattern is a behavioral software design pattern, also known as the objects for states pattern. This pattern is used in computer programming to represent the state of an object. This is a clean way for an object to partially change its type at runtime.