The Craftsman: 48
Brown Bag V
Statenum

Robert C. Martin
11 March 2006

...Continued from last month.

Sept, 1944.

"Dr. Oppenheimer, General Groves, thank you for coming." The morning sunlight streamed through the southeast windows of the Oval Office, providing a cheery glow that matched President Wallace's mood. "Yesterday I had a most interesting meeting with Dr. Von Braun. It is his opinion that we can build a fleet of huge atomic powered interplanetary space ships to rescue a remnant of humanity from the impact of Clyde." Oppenheimer squirmed uncomfortably in his chair, but Groves kept his attention squarely on the President. "Gentlemen, I don't pretend to know if he is right. What I do *know is that we need more bold ideas like that. I want you to scour project Nimbus and find them; the bigger and bolder the better!"*

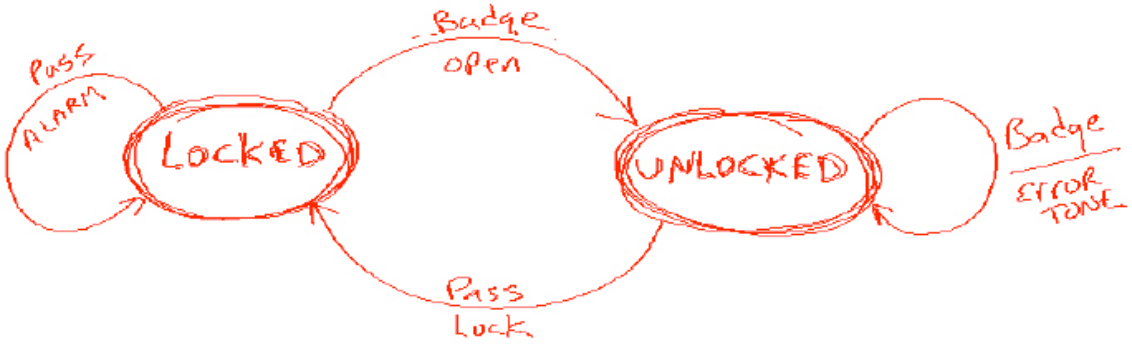
Friday, 1 Mar 2002, 1102

I was running a couple of minutes late for our regular 11:00 meeting. When I walked in the room there were quite a few people in that little conference room. Jerry, Avery, Adelade, and Jasper were watching Jasmine who was just starting to draw something on the wall.

Jasmine fixed me with those deep green eyes of her and said: "It's about time, Hotshot. I was just about to answer that question you asked yesterday."

"Sorry." I murmured as I squeezed in beside Avery. He rolled his eyes meaningfully while subtly cocking his head towards Jasmine. I gave him a pained smile and turned my attention back to her.

She spoke while she drew the following picture on the wall. "OK, here is the finite state machine for the security gate up on the bridge."



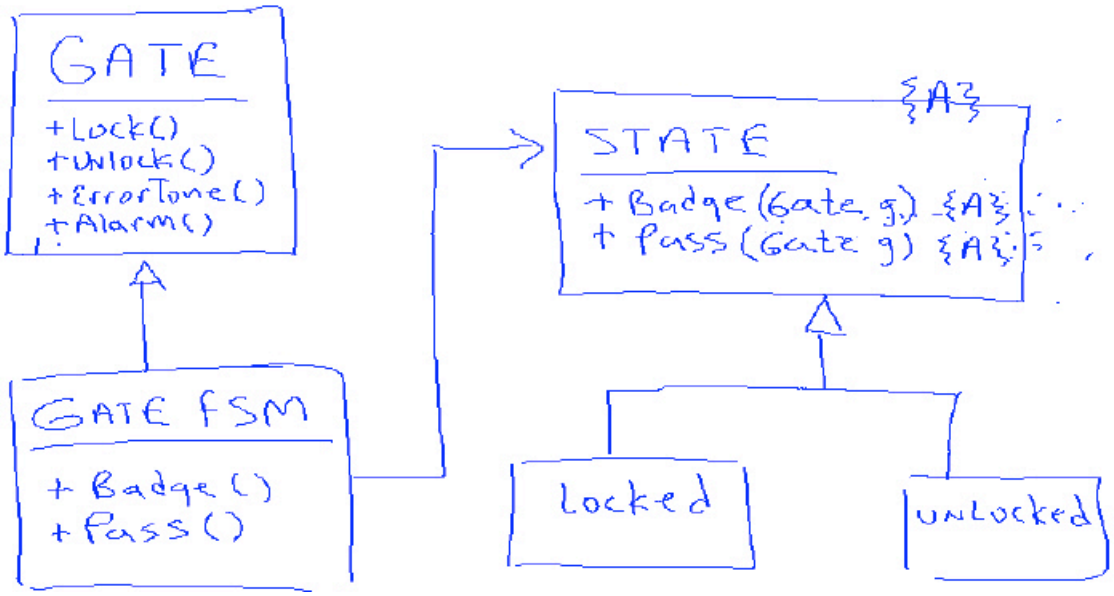
Avery said: "Excuse me, but that's not really how it works, it..."

"Yeah, yeah, I know." Jasmine replied testily, "But I don't have enough wall space or time to draw the complete FSM. Just bear with me. This *simplified* machine starts out in the LOCKED state. If you wave a valid badge at it, the gate opens and the FSM enters the UNLOCKED state. Once you pass through the gate, it automatically closes and locks and the FSM goes back to the LOCKED state.

"The two side transitions show the error cases. The one on the right shows the case of someone continually waving their badge at an open gate. The gate sounds an error tone in an attempt to get the nitwit to stop waving his badge and go through the gate. The transition on the left shows the case where someone forces entry onto the bridge. An alarm sounds."

Jasmine spun away from the wall to look at her audience. Her long black hair flared out and then fell perfectly around her shoulders. "Everybody got it?" She demanded rather than asked. We all nodded meekly.

"OK," she continued, "now we all know that SMC[1] would generate this using the State pattern. It would create this nice class hierarchy." And she drew the following:



We were all familiar with this so we all nodded meekly as she swung around. She frowned playfully and then went on. "One problem has always been that we had to manage the instances of the `Locked` and `Unlocked` derivatives in `static` variables to prevent us from having to continuously create and destroy them. Another problem has been that we could not make the methods and variables of `GateFSM` private since they are manipulated by the derivatives of `State`."

Jasmine then pulled up some code that she had written previously for this talk.

```
public class Gate {
  void lock(){}
  void unlock(){}
  void errorTone() {}
  void alarm() {}
}
```

```
public class GateFSM extends Gate {
  State state = State.locked;

  public void badge() {
    state.badge(this);
  }
```

[1] http://www.objectmentor.com/resources/downloads/index

```
  public void pass() {
    state.pass(this);
  }
}
```

```
public abstract class State {
  public abstract void badge(GateFSM g);
  public abstract void pass(GateFSM g);

  public static State locked = new Locked();
  public static State unlocked = new Unlocked();
}
```

```
public class Locked extends State {
  public void badge(GateFSM g) {
    g.unlock();
    g.state = State.unlocked;
  }

  public void pass(GateFSM g) {
    g.alarm();
  }
}
```

```
public class Unlocked extends State {
  public void badge(GateFSM g) {
    g.errorTone();
  }

  public void pass(GateFSM g) {
    g.lock();
    g.state = State.locked;
  }
}
```

"Notice the `public static` variables in the `State` class, and the non-private methods and variables of `Gate` and `GateFSM`.

"Excuse me," wheedled Avery, "But you could make those methods and variables private by using inner classes. Here, let me show..."

"SIT DOWN! Yes, I know, I know. I'm just trying to make a point about the code that SMC generates. Bear with me please!"

Avery sat down and started to sulk.

Jasmine gave us all a weak smile, actually it was more like a grimace, and then continued. "Here's how we can implement it using the new `enum` syntax that Alphonse showed us yesterday.

```
public class Gate {
  private enum State {
    LOCKED {
      void badge(Gate g) {
        g.unlock();
        g.itsState = UNLOCKED;
      }

      void pass(Gate g) {
        g.alarm();
      }
    },
    UNLOCKED {
      void badge(Gate g) {
        g.errorTone();
      }
```

```
      void pass(Gate g) {
        g.lock();
        g.itsState = LOCKED;
      }
    };

    abstract void badge(Gate g);
    abstract void pass(Gate g);
  }

  private State itsState = State.LOCKED;

  private void lock() {}
  private void unlock() {}
  private void errorTone() {}
  private void alarm() {}

  public void badge() {
    itsState.badge(this);
  }

  public void pass() {
    itsState.pass(this);
  }
}
```

"First, you'll notice that it all fits in one class. It's small, expressive, we don't have any `static` variables to manage, and all the methods of `Gate` can be `private`. I think this is a pretty cool use for polymorphic `enums` don't you?

We all nodded meekly; except for Avery who kept on sulking.

Jerry stood up and said: "Yes, that's very interesting. The enumeration is like a class, and each enumerator is like a static instance of a derivative of that class. It's kind of hard to get your head around, but it's definitely interesting."

Then Jasper got into the act. "Yes ma'am, that's pretty interesting Jazzie." Jasmine's face reddened, and her jaw clenched, but she held her peace as Jasper gushed on. "But I think the aviator's got a point. The problems with the SMC code can be solved without having to resort to polymorphic enums. Aviator suggested inner classes; but I think nested switch-case statements would work better." And he quickly wrote the following code on the wall.

```
public class Gate {
  private enum State {LOCKED, UNLOCKED}
  private enum Event {BADGE, PASS}

  private State state = State.LOCKED;

  private void lock() {}
  private void unlock() {}
  private void errorTone() {}
  private void alarm() {}

  public void badge() {
    processEvent(Event.BADGE);
  }

  public void pass() {
    processEvent(Event.PASS);
  }

  private void processEvent(Event event) {
```

```
    switch (event) {
      case BADGE:
        switch (state) {
          case LOCKED:
            unlock();
            state = State.UNLOCKED;
            break;
          case UNLOCKED:
            errorTone();
            break;
        }
        break;
      case PASS:
        switch (state) {
          case LOCKED:
            alarm();
            break;
          case UNLOCKED:
            lock();
            state = State.LOCKED;
            break;
        }
        break;
    }
  }
}
```

"See," said Jasper smugly, "no static variables, no private methods, nice and simple."

"You call that simple?" Jerry bantered. "Those nested switch statements make my head hurt. Man, I've maintained big FSMs written that way, and they are no fun at all, let me tell you."

"Yeah," Jasmine chimed in, "And all that logic is sitting there in one method that's bound to grow and grow and grow. Ick!"

"Jazzie...did you just say Ick?"

"Stop calling me Jazzie, Jasper. I've told you before. Yeah, ick! That code is going to rot before you know it."

They kept on like this for a minute or two. Avery left the room in a huff. I finally stood up and tapped on the wall until I had everyone's attention.

"Guys," I said, grinning, "I haven't seen one test. How do we know that any of these FSMs actually work?"

They all groaned and declared the meeting over.

On the way back, Jerry walked up next to me and, shaking his head, said: "I don't know what we're going to do about Avery."