MCMR Design

Weiwei Jia
www.jiaweiwei.com
Updated at Apr. 4, 2014

---

1 Introduction

MCMR is Mock Coffee Machine Running. Fig. 1[1] shows the statecharts[2] of MCMR. This document aims to realize a software to simulate what the statecharts says, which just describes how coffee machine works well automatically.

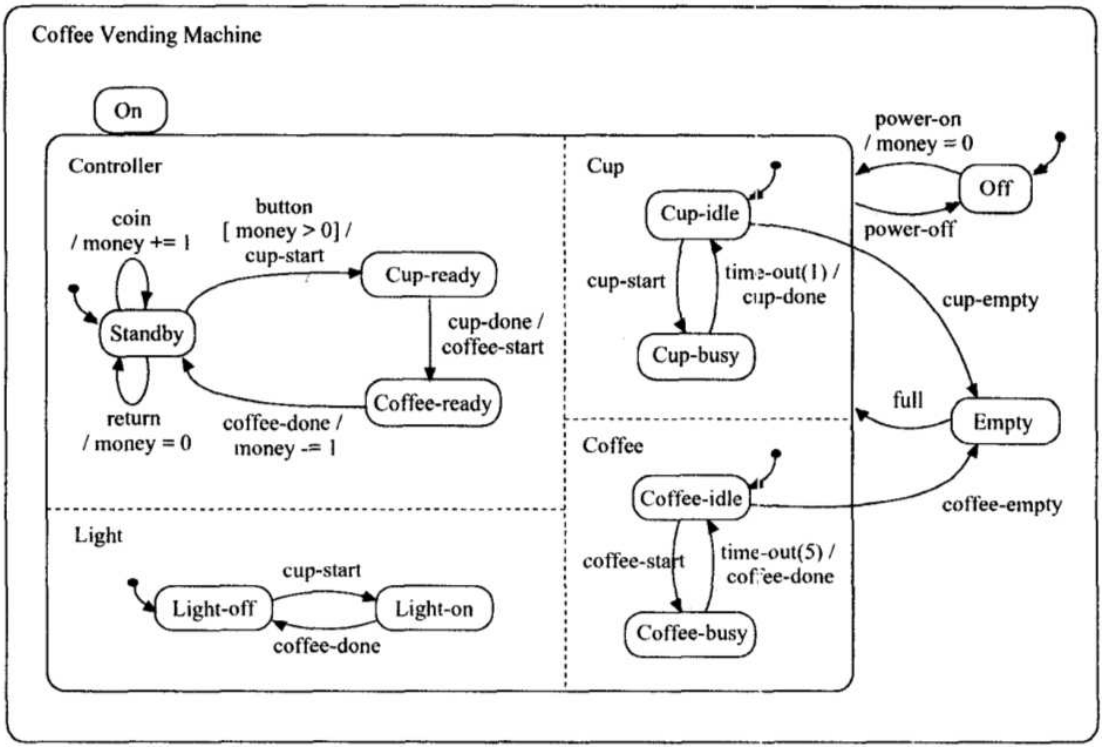

Figure 1: Mock Coffee Machine Running StateCharts

2 General design

As is seen from the statecharts diagram, there are three states for the Coffee Vending Machine, which are ‘On‘, ‘Off‘ and ‘Empty‘. ‘On‘ could transfer to ‘Off‘ when power-off is set. On the contrary, ‘Off‘ turns to ‘On‘ after power is on and money is zero. Meanwhile, ‘On‘ becomes ‘Empty‘ once cup/coffee is empty. ‘Empty‘ transfers to ‘On‘ when cup/coffee is full. Actually, there are three conditions for the transition between ‘On‘ and ‘Empty‘.

- Cup and Coffee are empty.
- Cup is empty and Coffee is not empty.
- Cup is not empty and Coffee is empty.

2.1 Internal On state

Once the Coffee Vending Machine is On, there would be four processes/threads are started simultaneously, which are Controller process/thread, Light process/thread, Cup process/thread and Coffee process/thread. Now, let's analyze each of them.

- Controller process/thread

  1. In Standby state in default. After one does coin action, money would plus one.
  2. After one does button action with money more than zero, it would transfer to Cup-ready state, which leads to cup-start (See Cup and Light processes/threads).
  3. After cup-done is set (See Cup process), Cup-ready would turn to Coffee-ready, which leads to coffee-start (See Coffee process).
  4. After coffee-done is set (See Coffee process), Coffee-ready would transfer to Standby state with money minus one.
  5. After one does return action, it transfers to Standby itself, which leads to money is zero.

- Cup process/thread

  1. In Cup-idle state in default. After cup-start is set (in Controller process), it would transfer to Cup-busy.
  2. With one second duration, Cup-busy would turn to Cup-idle and cup-done is set in the same time.
  3. It would go to Empty state when there is no cup (cup-empty is set).

- Light process/thread

  1. In Light-off state in default, it would become Light-on state with cup-start (Set in Controller process/thread).
  2. It would go back to Light-off state with coffee-done is set.

- Coffee process/thread

  1. In Coffee-idle state in default. After coffee-start is set (in Controller process), it would transfer to Coffee-busy.
  2. With five seconds duration, Coffee-busy would turn to Coffee-idle and coffee-done is set in the same time.
  3. It would go to Empty state when there is no coffee (coffee-empty is set).

3 Detail design

3.1 How MCMR works

1. Initialize corresponding stuffs, such as fields in the global memory control block and so on.
2. The system is in Off state in default. Do power action, it goes to On state with money is initialized with zero.
3. For rest parts, please see 'Internal On State' section for details.

3.2 How processes communicate with each other

We use memory share technology in MCMR because it is faster.

3.3 Related data structures

The global data structure.

```
-------------------------------------------------------------------------
struct mcmr {
int money;
int power_on;
int power_off;
int cup_empty;
int cup_start;
int cup_done;
int coffee_empty;
int coffee_start;
int coffee_done;
int time;
struct coffee_vending_machine cvm;
struct contorller controller;
struct light light;
struct cup cup;
struct coffee coffee;
struct action *do;
};
-------------------------------------------------------------------------
```

Actions Coffee Vending Machine does

```
-------------------------------------------------------------------------
struct action {
int (* coin) (struct mcmr *);
int (* button) (struct mcmr *);
int (* cup) (struct mcmr *);
int (* coffee) (struct mcmr *);
int (* time) (struct mcmr *);
int (* power) (struct mcmr *);
int (* return) (struct mcmr *);
};
-------------------------------------------------------------------------
```

The Coffee Vending Machine data structure

```
-------------------------------------------------------------------------
struct coffee_vending_machine {
int on;
int off;
int empty;
};
-------------------------------------------------------------------------
```

The Controller data structure

```
-------------------------------------------------------------------------
struct controller{
int standby;
int cup_ready;
int coffee_ready;
};
-------------------------------------------------------------------------
```

The Light data structure

```
struct light {
int light_off;
int light_on;
}
```

The Cup data structure

```
struct cup {
int cup_idle;
int cup_busy;
};
```

The Coffee data structure

```
struct coffee {
int coffee_idle;
int coffee_busy;
};
```

4 References

[1] Hong, Hyoung Seok, et al. ”Static semantics and priority schemes for statecharts.” COMPSAC-NEW YORK- (1995): 114-114.
[2] Harel, David. ”Statecharts: A visual formalism for complex systems.” Science of computer programming 8.3 (1987): 231-274.