Software Engineering

Cicio Ionuț

21/12/2024

Contents

1 Systems modeling

"Software engineering is an engineering discipline that is concerned with all aspects of software production" [1].

In the following pages I'll focus on the most **important concepts** of the course. Those fundamental concepts will be treated in **great detail** to give a deep enough understanding of the material.

When designing **complex software** we have to make major **design choices** at the beginning of the project. Often those choices can't be driven by experience or reasoning alone, that's why a **model** of the project is needed to **simulate** and **compare** different solutions. Our formal tool of choice is the **Markov Chain** (treated in Section 1.2.2).

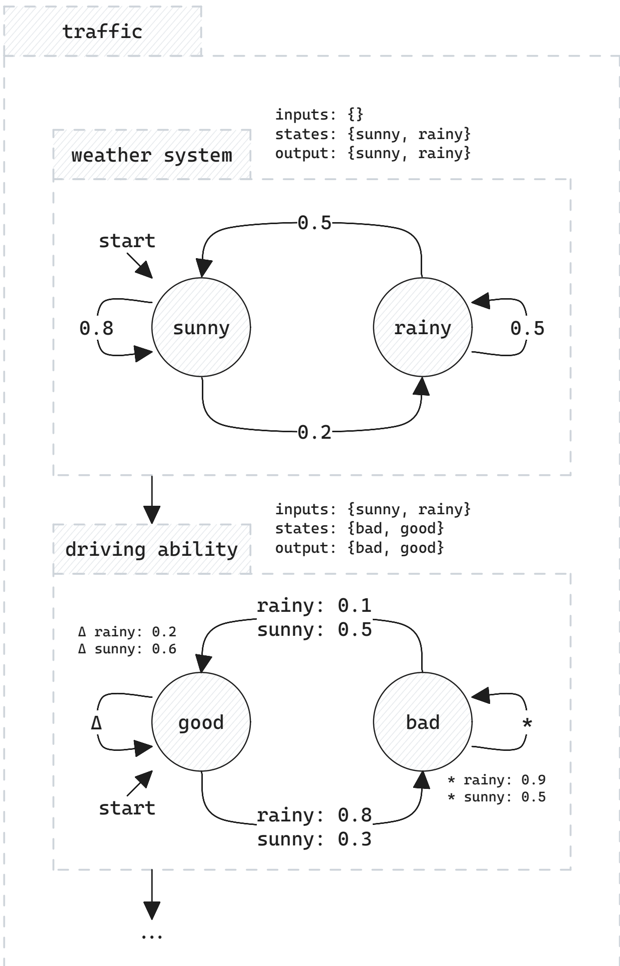

Figure 1: the 'traffic' system modeled with 2 subsystems

1.1 The concept of time

The models treated in the course evolve through **time**. Time can be modeled in many ways (I guess?), but, for the sake of simplicity, we will consider discrete time. Let W be the *'weather system'* and D the *'driving ability' system* in Figure 1, we can define the evolution of D as

$$D(0) = \text{'good'}$$
$$D(t+d) = f(D(t), W(t))$$

Given a time instant t (let's suppose 12:32) and a time interval d (1 minute), the driving ability of D at 12:33 depends on the driving ability of D at the time 12:32 and the weather at 12:32.

1.2 Formal notation

1.2.1 Markov Chain

A Markov Chain is…

1.2.2 DTMC (Discrete Time Markov Chain)

A DTMC M is a tuple (U, X, Y, p, g) s.t.

- $U \neq \varnothing \wedge X \neq \varnothing \wedge Y \neq \varnothing$ *(otherwise stuff doesn't work)*
- U can be either
  - $\{u_1, ..., u_n\}$ where u_i is an input value
  - $\{()\}$ if M doesn't take any input
- $X = \{x_1, ..., x_n\}$ where x_i is a state
- $Y = \{y_1, ..., y_n\}$ where y_i is an output value
- $p : X \times X \times U \to [0, 1]$ is the transition function
- $g : X \to Y$ is the output function

$$\forall x \in X \ \ \forall u \in U \quad \sum_{x' \in X} p(x'|x, u) = 1$$

$$M(0) = x_1$$
$$M(t+d) = \begin{cases} x_1 & \text{with probability} \ \ p(x_1|M(t), U(t)) \\ x_2 & \text{with probability} \ \ p(x_2|M(t), U(t)) \\ \dots \end{cases}$$

It's interesting to notice that the transition function depends on the input values. If you consider the *'driving ability'* system in Figure 1, you can see that the probability to go from `good` to `bad` is higher if the weather is rainy and lower if it's sunny.

1.2.3 An example of DTMC

Let's consider the development process of a team. We can define a DTMC $M = (U, X, Y, p, g)$ s.t.

- $U = \{()\}$, as it doesn't have any input
- $X = \{0, 1, 2, 3\}$
- $Y = \text{Cost} \times \text{Duration (in months)}$

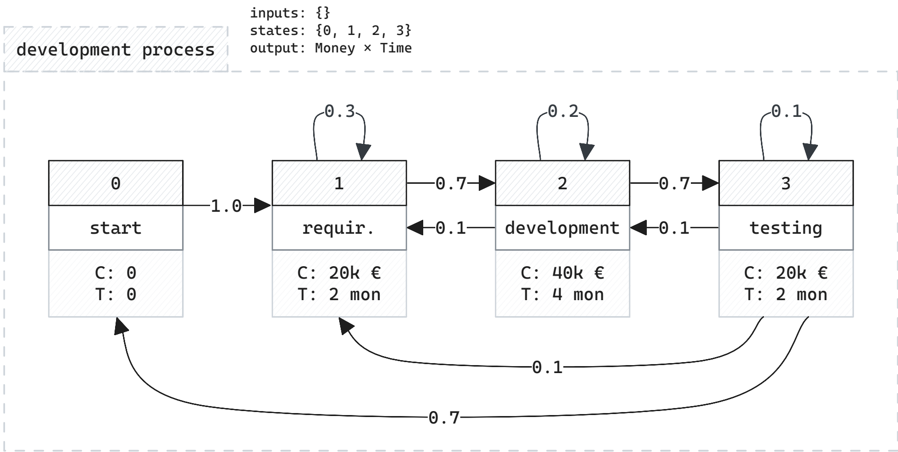

Figure 2: the model of a team's development process

$$g(x) = \begin{cases} (0,0) & \text{if } \; x = 0 \\ (20000, 2) & \text{if } \; x = 1 \\ (40000, 4) & \text{if } \; x = 2 \\ (20000, 2) & \text{if } \; x = 3 \end{cases}$$

1.3 Network of Markov Chains

TODO...

2 C++

2.1 Intro to #include <random>

2.1.1 Seed & std::default_random_engine

2.1.2 Distributions

2.1.2.1 std::uniform_int_distribution<>()

2.1.2.2 std::uniform_real_distribution<>()

2.1.2.3 std::bernoulli_distribution<>()

2.1.2.4 std::poisson_distribution<>()

2.1.2.5 std::geometric_distribution<>()

2.1.2.6 std::discrete_distribution<>()

2.2 Intro to data structures

2.2.1 std::vector<T>()

2.2.2 std::deque<T>()

2.2.3 Sets

2.2.4 Maps

2.3 I/O

2.3.1 #include <iostream>

2.3.2 Files

3 Models

3.1 First examples

Now we have to put together our **formal definitions** and our `C++` knowledge to build some simple DTMCs and networks.

3.1.1 A simple Markov Chain

Let's begin our modeling journey by implementing a DTMC M s.t.

- $U = \{()\}$
- $X = [0,1] \times [0,1]$
- $Y = [0,1] \times [0,1]$
- $p : X \times X \times U \to X = \mathcal{U}(0,1) \times \mathcal{U}(0,1)$
- $g : X \to Y : (r_0, r_1) \mapsto (r_0, r_1)$
- $X(0) = (0,0)$

```
#include <fstream>
#include <random>

typedef long double real_t;

int main() {
    std::random_device random_device; ❶
    std::default_random_engine random_engine(random_device()); ❷
    std::uniform_real_distribution<real_t> uniform(0, 1); ❸

    const size_t HORIZON = 10; ❹
    std::vector<real_t> state(2, 0); ❺

    for (size_t time = 0; time <= HORIZON; time++)
        for (auto &r : state)
            r = uniform(random_engine);

    return 0;
}
```

Listing 1: `software/1100/main.cpp`

3.1.2 Connected Markov Chains

Now let's model a system with two DTMCs M_0, M_1, and lets define the functions

$$U(M_0, t+1) = \cdots \quad U(M_1, t+1) = \cdots$$

3.1.3 Different types of connections

3.2 Traffic light

3.3 Network controlled traffic light

3.4 Statistics

3.4.1 Expected value

TODO: 'mean' trick, ggwp

$$\varepsilon_n = \frac{\sum_{i=0}^{n} v_i}{n}$$

$$\varepsilon_{n+1} = \frac{\sum_{i=0}^{n+1} v_i}{n+1} == \frac{\left(\sum_{i=0}^{n} v_i\right) + v_{n+1}}{n+1} = \frac{\sum_{i=0}^{n} v_i}{n+1} + \frac{v_{n+1}}{n+1} =$$

$$\frac{\left(\sum_{i=0}^{n} v_i\right)n}{(n+1)n} + \frac{v_{n+1}}{n+1} = \frac{\sum_{i=0}^{n} v_i}{n} \cdot \frac{n}{n+1} + \frac{v_{n+1}}{n+1} = \frac{\varepsilon_n \cdot n + v_{n+1}}{n+1}$$

3.4.2 Probability

3.5 Transition matrix

3.6 Complex systems

3.6.1 Insulin pump

3.6.2 Buffer

3.6.3 Server

4 Exam

4.1 Development team (time & cost)

4.2 Backend load balancing

4.3 Heater simulation

4.3.1 Eulero's method for differential equations

Bibliography

[1] I. Sommerville, *Software Engineering*, 10th ed. Boston: Pearson Education Limited, 2016.