Programming Embedded Systems

Lecture 8

Overview of software testing (continued)

Monday Feb 13, 2012

Philipp Rümmer
Uppsala University
Philipp.Ruemmer@it.uu.se

Decision coverage (DC)

(a kind of logic coverage)

- **Decisions** D(p) in a program p:
set of *maximum* boolean expressions in p
- E.g., conditions of `if`, `while`, etc.
- But also other boolean expressions:

```
A = B && (x >= 0);
```

Precise definition is subject of many arguments:
only consider decisions that program branches on?

(`B&&(x>=0)` is a decision, `B` and `(x>=0)` are not)

Decision coverage (DC) (2)

- **NB:** multiple occurrences of the same expression are counted as different decisions!
  E.g.

```
if (x >= 0) { … }
// …
if (x >= 0) { … }
```

Two decisions

Decision coverage (DC)

- For a given decision d, DC is satisfied by a test suite TS if it contains at least one test where d evaluates to false, and one where d evaluates to true (might be the same test)
- A test suite TS **achieves DC** for a program p if it achieves DC for every decision d in D(p)

DC example

- Consider decision

```
((a < b) || D) && (m >= n * o)
```

- **Inputs to achieve DC?**

TS achieves DC if it triggers executions

```
a = 5, b = 10, D = true, m = 1, n = 1, o = 1
```

and

```
a = 10, b = 5, D = false, m = 1, n = 1, o = 1
```

Condition coverage (CC)

- **Conditions** C(p) in a program p: set of *atomic* boolean expressions in p
- e.g., in the decision

  ```
  ((a < b) || D) && (m >= n * o)
  ```

  the conditions are

  `(a < b)`, `D`, and `(m >= n * o)`

Condition coverage (CC) (2)

- For a given condition c, CC is satisfied by a test suite TS if it contains at least one test where c evaluates to false, and one where c evaluates to true (might be the same test)
- A test suite TS **achieves CC** for a program p if it achieves CC for every condition c in C(p)

CC example

- Consider all the conditions in

```
((a < b) || D) && (m >= n * o)
```

- **Inputs to achieve CC?**

TS achieves CC if it triggers executions

```
a = 5, b = 10, D = true, m = 1, n = 1, o = 1
```

and

```
a = 10, b = 5, D = false, m = 1, n = 2, o = 2
```

Modified condition decision coverage (MC/DC)

- For a given condition c in decision d, MC/DC is satisfied by a test suite TS if it contains one test where c evaluates to false, one test where c evaluates to true, d evaluates differently in both, and the other conditions in d evaluate identically in both.
- For a given program p, MC/DC is satisfied by TS if it satisfies MC/DC for all c in C (p)

MC/DC example

- Consider the condition `(a < b)` in
  `((a < b) || D) && (m >= n * o)`

- TS achieves MC/DC if it triggers executions

  ```
  a = 5, b = 10, D = false, m = 1, n = 1, o = 1
  ```

  and

  ```
  a = 10, b = 5, D = false, m = 8, n = 2, o = 3
  ```

Bottom line

- Value of any kind of structural/logic coverage is arguable
- But demonstration is often required

Further reading

- “Introduction to Software Testing,” Paul Ammann and Jeff Offutt; http://www.cs.gmu.edu/~offutt/softwaretest/

System testing

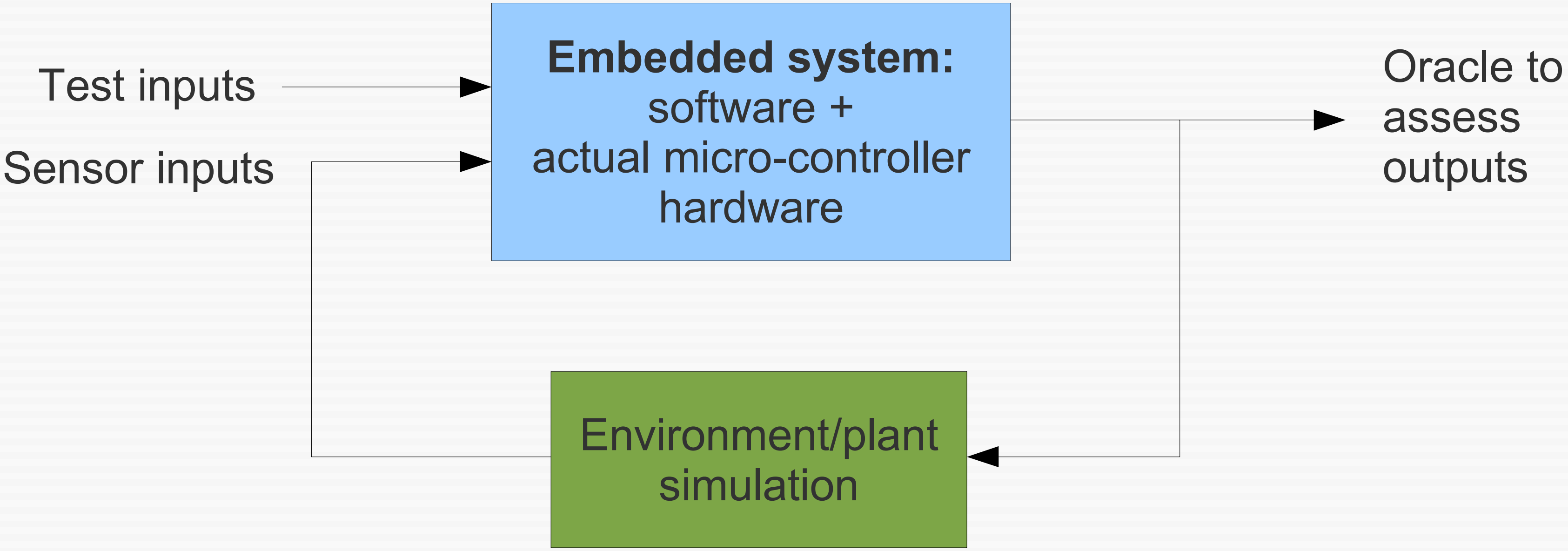
Overview

- No longer consider software units, but system as a whole
- Again, many different variants:
  - Stress testing
  - Usability testing
  - Performance testing
  - ...

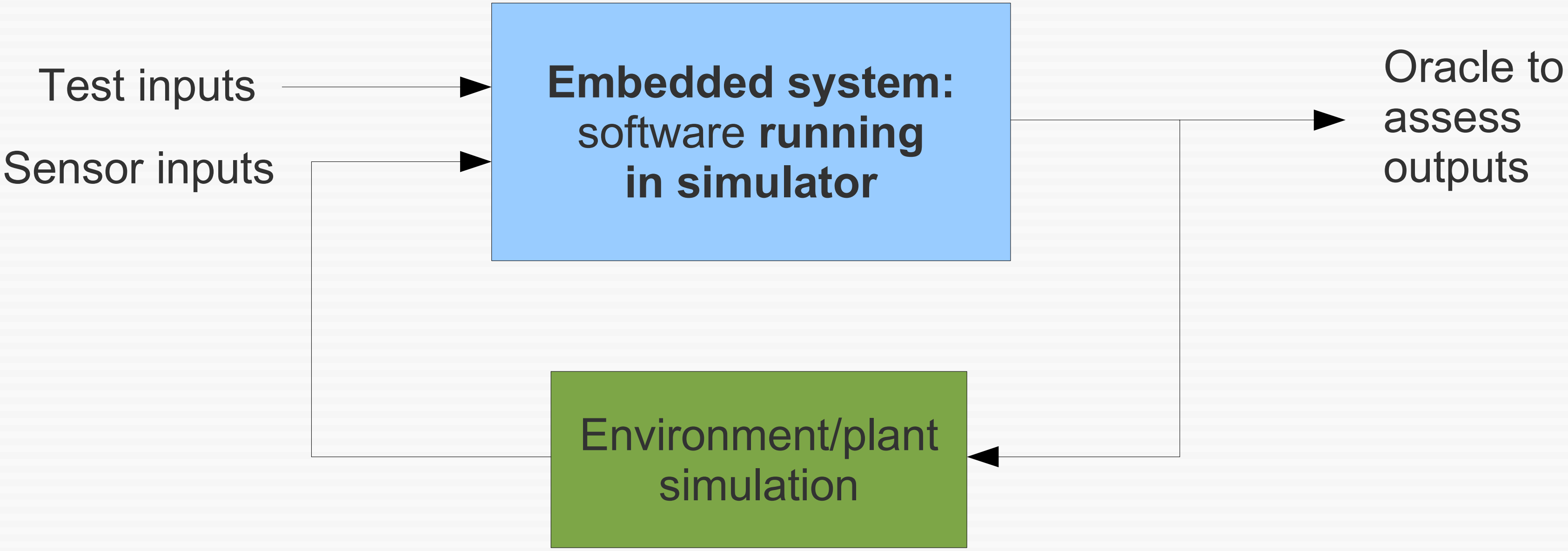
Testing embedded systems

- Embedded systems are reactive: need **stream** of test inputs
- Realistic environment needed:
    - Either actual environment (latest stage of testing)
    - Or a simulation of the environment (cheaper, faster)
- Important setups for embedded systems: **hardware-in-the-loop, software-in-the-loop**

Hardware-in-the-loop (HIL)



Software-in-the-loop (SIL)


Test inputs
Sensor inputs
Embedded system: software **running in simulator**
Oracle to assess outputs
Environment/plant simulation

→ Compare with elevator lab!

HIL + SIL

- Not only test inputs, but also environment simulation has to be chosen by test engineer
    - What if simulation is not realistic?
    - Possible variations of environment can/should be included in simulation
    - Often developed in high-level languages like Matlab/Simulink → See “Model-based ...” course