Next Date – Computational Problem

NextDate:

Inputs: integer $month$, $1 \leq month \leq 12$

integer day, $1 \leq day \leq 31$

integer $year$ $1812 \leq year \leq 2012$

Output: the date of the day after the input date.

Next Date – Equivalence classes, 1st version

$$M1 = \{month \,|\, 1 \leq month \leq 12\}$$
$$D1 = \{day \,|\, 1 \leq day \leq 31\}$$
$$Y1 = \{year \,|\, 1812 \leq year \leq 2012\}$$

Next Date – Test Suite, 1st version

case id	month	day	year	expected output
T1	6	15	1912	6/16/1912

That's one test case only!!!!

Next Date – Equivalence Classes, 2nd version

$M1.1 = \{month \,|\, month \text{ has 30 days}\}$

$M1.2 = \{month \,|\, month \text{ has 31 days}\}$

$M1.3 = \{month \,|\, month \text{ is February}\}$

$D1.1 = \{day \,|\, 1 \leq day \leq 28\}$

$D1.2 = \{day \,|\, day = 29\}$

$D1.3 = \{day \,|\, day = 30\}$

$D1.4 = \{day \,|\, day = 31\}$

$Y1.1 = \{year \,|\, year = 2000\}$

$Y1.2 = \{year \,|\, year \text{ is common year}\}$

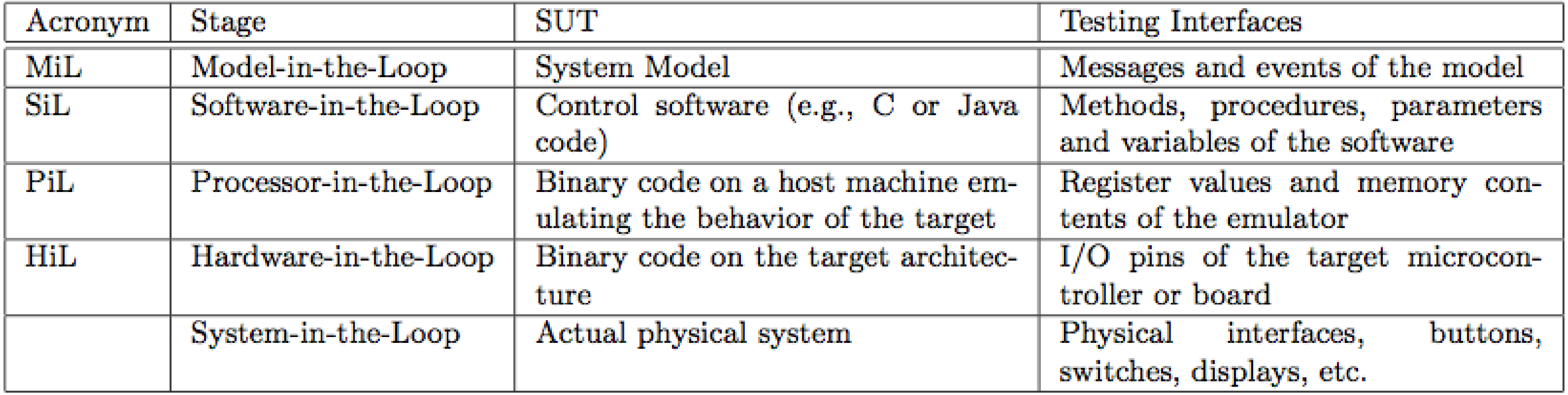
$Y1.3 = \{year \,|\, year \text{ is a leap year}\}$

Next Date – Test Suite, 2nd version: One the blackboard

name	month	date	year	expected output
. . .	. . .	. . .	. . .	

A note on test evaluation

Which system are we testing?



Acronym	Stage	SUT	Testing Interfaces
MiL	Model-in-the-Loop	System Model	Messages and events of the model
SiL	Software-in-the-Loop	Control software (e.g., C or Java code)	Methods, procedures, parameters and variables of the software
PiL	Processor-in-the-Loop	Binary code on a host machine emulating the behavior of the target	Register values and memory contents of the emulator
HiL	Hardware-in-the-Loop	Binary code on the target architecture	I/O pins of the target microcontroller or board
	System-in-the-Loop	Actual physical system	Physical interfaces, buttons, switches, displays, etc.

Observation: different data representations on the interfaces.

Data representations for the naturals

- decimal
- binary
- hexadecimal
- roman
- . . .

ASN.1

Abstract Syntax Notation One (ASN.1) is a standard and notation that describes rules and structures for representing, encoding, transmitting, and decoding data in telecommunications and computer networking.

Example

Testing Technologies uses ASN.1 (among others) to “enable your test environment to speak the language of the system you are testing. The automatic import of interface specifications and the automatic generation of codecs free customers from the burden of dealing with test system implementation issues.”

http://www.testingtech.com/download/flyer/TTbrochure.pdf

JUnit

Does two things, namely

- Test execution:
  - send stimuli to the SUT
  - observe reactions from the SUT
- Test evaluation
  - compare actual result with expected result
  - and give a verdict (pass or fail)

Thus: when encoding testcases in JUnit, we also have to encode changes in data representation.

Lab on Equivalence Class Testing

Data representation in the test suite:

```
6/30/1996
```

Data representation in the program output:

```
30.6.---1996
```

Your job as testers is to encode the change in data representation.