

GEM 1501 Problem Solving With Computers

Lecture 12:

Software Engineering

Frank Stephan

Overview

- Summary of previous lecture
- Midterm Test
- Software Engineering

Summary of Previous Lecture

- Comparing Polynomials
- The Class RP
- Fast Probabilistic Pattern Matching
- Introduction to Cryptography
- Public Key Cryptography
- Interactive Proof Systems

Random Polynomial Time

- NP: x can be accepted if there is a solution for problem linked to x ; x is always rejected if there is no solution to problem linked to x .
- RP: x is accepted with probability $2/3$ if there is a solution for problem linked to x ; x is always rejected if there is no solution to problem linked to x .
- Class in RP not known to be in P: Set of all (f, g) such that f, g are multivariate different polynomials of degree n given in a suitable representation perhaps in RP–P.

Cryptography

- **Goals**

- Protecting privacy of communication;
- Detecting falsified messages;
- Enabling electronic commerce and world wide computer networks.

- **Private Key cryptography**

- Often breakable by comparing encoded and plain message
- All aspects of keys must be kept secret by all means

- **Public Key cryptography**

- Encryption Keys can be propagated through public channels
- Decryption Keys are kept private

Software Engineering

- Basic Model (Lifecycle): Requirements Analysis, Design, Implementation, Validation, Maintenance
- Object Oriented Programming
- The Models of Software Engineering

Lifecycle of Software

- **Requirements analysis.** What is needed for the product, what functionality should it have.
- **Design.** Break up the software into independent modules which might be designed by different programmers or subcontractors. Specify what each module has to do and how the information is interchanged between the modules.
- **Implementation.** Programme each module according to the given design. Integrate the various modules into one system.
- **Validation.** Make sure that the software does what it should. This is done by extensive testing; in some few cases also by explicit software verification with technical support for this process.
- **Maintenance.** Adjust the software to new requirements and developments, publish patches on the internet, help clients to get problems solved, repair bugs in the system and redeliver to clients (if necessary), write software connecting the system to other systems for data interchange and so on.

Requirements Analysis for Browsers

- Must display webpages in a browser window and enable navigation for going to further windows or coming back to original window
- Must execute code from remote servers in permitted languages (html, Java Script, Java, ...)
- Should not download and install software on the computer without explicit permission of the user (danger of viruses and spying software)
- Use cryptography for connection to secure sites in order to permit secure transactions for banking or other applications

Design - Product Guidelines

- Decisions on the product appearance. For example browser windows have on the top navigation bars, then the main field to display the information and on the bottom some status information.
- Browsers give some support on webpage writing like error messages for faulty pages (needs to be activated).
- Java Script writes files and reads files only in a specially designed area and these files are called “cookies”. Cookies are kept separately from the other data of a user on a computer so that they cannot infect the computer with viruses and other bad software.
- Secure sites have addresses “https://...” while normal sites have addresses “http://...” in order to give to the browser a hint how much effort should be spent on cryptographical procedures while dealing with a site.

Design - Module Specifications

- The behaviour of modules, functions and data inside the system needs to be specified.
- How to specify the remainder operation:
 - (1) The remainder of a divided by b is that number c such that $0 \leq c < b$ and there is an integer d with $a = bd + c$.
Example: `-1 % 8 == 7;`
 - (2) Java Script definition: `-1 % 8 == -1.`
Advantage of (2): easily implementable by `(-a) % b == - (a % b).`
Disadvantage of (2): computation in finite fields more difficult to implement.
Way out: Use `((a%b)+b)%b` instead of `a%b.`
- Specification has to tell in all cases what has to be done. Product should be reasonable and definitions easy to understand. In doubt some examples have to tell the programmers what is desired.

Specification versus Reference Implementation

- **Specification.**

Tells the programmer in a formal way what the program has to do in each situation. Might be written down in some formal specification language.

- **Reference Implementation.**

When designing a new programming language, the authors of the language might put up a reference machine on the internet which can compile or interpret any program supplied to it. Then every company manufacturing a compiler or interpreter for this language has to make a product which behaves exactly like the reference machine although it can be faster or having a better performance or giving more support to users while writing the program.

Implementation

- First write modules according to specification and then integrate the various module to one.
- Choice of programming languages and style can give better performance.
- Structured programming: logically well designed programs have less errors.
- Restrictive programming languages: forces programmers to more discipline and compilers detect more errors.
- Documentation: allows other people to modify and improve the programs years after they are written.
- Object-oriented programming: data and methods to modify data are kept together.

Validation

- Verify software formally (if possible). Needs some technical support and can also be time-consuming.
- Programming in pairs: every module is programmed independently by two programmers; then it is tested systematically whether both implementations do the same.
- Design test data which also could handle special cases in which programmers are likely to make errors.
- Try to automatize the testing so that the software can run on many different inputs.
- Try to test the various modules separately in order to identify errors as early as possible.

Testing and Microchips

- The design of chips is automatized and many testing is done by simulation before the actual production in order to find design errors.
- Chip process produces more faulty than correct chips — therefore there are testing circuits on each chip which support the testing of every newly produced chips.
- Testing routines discover the faulty chips before they are installed into computers. Therefore the delivered hardware is quite reliable.

Maintenance

- Products come often in versions, software has often updates on the internet.
- Tabbing in browsers became popular between 2007 and 2008. The new versions of Firefox and Internet Explorer were equipped with tabbed browsing after some update automatically.
- When opening a new tab, text explaining the features to a user appears: “You’ve opened a new tab — With tabs you can: Use one Internet Explorer window to view all your webpages. Open links in a background tab while viewing the page you’re on. Save and open multiple webpages at once by using favorites and home page tabs. ...” Making the users familiar with this new feature is also part of maintenance.

Maintenance at Windows

- Operating System Windows

Windows 1.0, Windows 2.0, Windows/286, Windows/386, Windows 3.0, Windows 95, Windows NT 4.0, Windows 98, Windows Millenium Edition (Me), Windows 2000, Windows XP, Windows Vista

http://en.wikipedia.org/wiki/History_of_Microsoft_Windows

- Besides new versions (to be bought by customers) providing updates and patches of existing versions in order to deal with new challenges like discovered security bugs and viruses; one can install mechanisms updating the versions automatically in intervals

<http://www.microsoft.com/>

- Maintenance and development of new versions are both two aspects of the same game.

Programming and Assignments

- Requirements Analysis: Written down for you.
- Design: Some students found nice algorithms for the Tower of Hanoi assignment and the ATM Withdrawal Game; the solution to the latter does not need precomputed data.
- Implementation: Code the algorithm into the parts left out for this in the assignment. This means to translate the algorithm into code and to typeset the program, it also includes checking for syntax errors.
- Validation: Use of testing and verification routines provided by the assignment permits to detect many errors before handing in the assignment. For calculating the order of functions, an extra testing routine was made as the results were difficult to evaluate by hand; furthermore, the random examples are not that random in order to make rare data with high mistake probability to come up more often.

Testing Assignment 4

This assignment asks students to program a routine which convert a number into a text, so `numberwrite(34)` should first output “34” and then the converted text “thirty four”.

The testing routine runs 20 tests of randomly chosen numbrers:

```
var x; var y;
for (y=0;y<20;y=y+1)
  { switch(Math.floor(Math.random()*4))
    { case 0: numberwrite(Math.floor(Math.random()*1000)); break;
      case 1: numberwrite(Math.floor(Math.random()*200)); break;
      case 2: numberwrite(Math.floor(Math.random()*20)); break;
      case 3: numberwrite(10*Math.floor(Math.random()*50)); break; } }
```

A quarter of the test cases are numbers below 1000;

a quarter of the test cases are numbers below 200;

a quarter of the test cases are numbers below 20;

a quarter of the test cases are multiples of 10.

The goal is to cover rare cases more frequently.

Object oriented Programming

- Modules have a clearly defined interface. They implement objects together with methods to modify or enquire information about objects. User functions can make new objects.
- The array object in Java Script can serve as an example. Its methods are mainly accessed by adding a dot and a method name:

| | |
|--------------------------|---------------------------------------------------------|
| <code>x</code> | Object |
| <code>x.length</code> | Length of the object, that is, number of array elements |
| <code>x.pop()</code> | Method moving off the last array element |
| <code>x.join()</code> | Method producing a text-representation of the object |
| <code>x[2]</code> | Second array element |
| <code>x[3].length</code> | Length of the third array element (if an array again) |

- Similar methods in different objects can have the same name:

| | |
|----------------------------|------------------------------------------------------------|
| <code>11.88+33.55</code> | Adding two numbers |
| <code>"This "+"is "</code> | Adding two texts |
| <code>y.length</code> | Length of array <code>y</code> or of string <code>y</code> |

Tower of Hanoi as Object

Implementation of Towers of Hanoi in Variable hanoi such that

```
hanoi.pegnum           Number of pegs
hanoi.ringnum          Number of rings
hanoi.move(a,b)        Moves upmost ring from peg a to b (if possible)
hanoi.display()        Routine to display the towers
new hanoimake(m,n);    Makes object with m pegs, n rings and tower on 0
```

User uses only these interfaces

```
function movemany(hanoi,k,pega,pegb,pegc)
  { if (k>1) { movemany(hanoi,k-1,pega,pegc,pegb); }
    hanoi.move(pega,pegb);
    hanoi.display();
    if (k>1) { movemany(hanoi,k-1,pegc,pegb,pega); }
    return; }

```

```
var hanoi = new hanoimake(3,5);
hanoi.display();
movemany(hanoi,hanoi.ringnum,0,1,2);

```

Implementation of Object

```
function hanmove(a,b)
  { if ((a<this.pegnum) && (b<this.pegnum) &&
      (this.field[a].length>0) && ((this.field[b].length==0) ||
      (this.field[a][this.field[a].length-1]<
      this.field[b][this.field[b].length-1])))
    { this.field[b].push(this.field[a].pop()); }
  else { document.write("Illegal Move.<br>"); } }
```

```
function hanoimake(m,n)
  { var k; this.pegnum = m; this.ringnum = n;
    this.field = new Array(m);
    this.display = handisplay; this.move = hanmove;
    for (k=0;k<m;k++)
      { this.field[k] = new Array(); }
    for (k=0;k<n;k++)
      { this.field[0].push(n-k); } }
```

<http://www.comp.nus.edu.sg/~gem1501/hanoiobject.html>

Information Hiding

- This principle says that a user should only use the official part of an interface to which the programmer of a module has made the commitment not to change this interface.
- For example, only the methods `hanoi.pegnum` and `hanoi.ringnum` but not the values of `hanoi.field` should be used as they are not official. The programmer maintaining the object and its application might rename `hanoi.field` to `hanoi.towers` at some time.
- According to the philosophy of Information Hiding, implementation details not being part of the official interface should even not be visible to other programmers using a module or object in order to make sure that they do not use things which might be changed later.

Models of Software Engineering

- Waterfall Model: each phase of development occurs only once and is then done for ever.
- Spiral Model: the product is developed in cycles of requirement analysis, design of new prototype, implementation of this prototype, validation of this prototype until a state is reached where the prototype has the quality of the final product.
- Evolutionary Model: All parts of the development of a program overlap and the product is improved step by step starting from an ugly and buggy prototype ending up with a good and thoroughly tested program. This is what people do on small projects and how most students solve assignments.
- Companies might make one of these models to their official policy.

Next Week: Artificial Intelligence

- Artificial Intelligence
- What to learn for the examination

Sample questions cover topics, but the actual questions are different and can be more difficult.