CSC 408F/CSC2105F Lecture Notes

These lecture notes are provided for the personal use of students taking CSC 408H/CSC 2105H in the Fall term 2004/2005 at the University of Toronto.

Designing Portable Software

- Portability - the ability to *easily* transfer a piece of software from one hardware or software environment to another
- compare: Interoperatability - the ability to *easily* transfer a piece of information from one software or hardware to another
- There are many examples of good portable/interoperatable software, e.g. GNU C compiler, vi editor, *many* X applications, SQL queries, Java applet/applications, XML/SOAP, Web Services/WSDL, etc.
- Portability increases the market for a piece of software at a much lower cost than creating multiple independent versions
- Design for portability is generally compatible with the good design practices discussed previously
- **Portability generally enhances reusability** Portability has to be designed into the software (c.f. build for reuse), retrofitting portability is expensive

Portability Techniques

- Isolation: Isolate all hardware/environment dependencies
- Parameterization via unique defining declarations

  Use named constants and types

  Unique declaration allows single point modification
- Conditional compilation and macro substitution
- Substitution at the module level
- Standarization

  [Java: Write once, run everywhere] [XML: Write once, exchange everywhere]

  [Web Services: Write once, call everywhere]
- **Avoid embedded and distributed dependencies** Magic numbers in the code, Machine or system dependent programming

Portability Issues - Software

- Variability in programming languages and compilers [Java/.NET story ... a battle behind the giants]
  - Manufacturer specific language extensions
  - Differences in run-time support libraries
  - Failure of compiler to adhere to language standard
  - Different implementation limits and system supplied defaults
- Portability techniques
  - Program strictly according to language standard
  - Avoid manufacturer-dependent programming constructs
  - Avoid system supplied defaults
  - Use software tools (e.g. lint) to verify compliance with the standard
  - Use conditional compilation and macro expansion to isolate unavoidable differences

Portability Issues - Software

- Differences in Interface to Environment
  - Operating System, e.g. Unix(almost Unix, Linux, FreeBSD), DOS/Windows(3.1/95/98/NT/2000/XP/2003/Longhorn), MAC OS . . .
  - Input & Output, e.g. Unix, DOS/Windows, MAC, [Path separator], . . .
  - Database packages, e.g. dbm, Empress, SQL, . . .
  - Graphics, e.g. X11R?, SunView, DOS, Windows, curses, . . .
  - Browsers, e.g. Netscape, IE, Opera, Mozilla, Safari, Firefox . . .
- Portability techniques
  - **Isolate all interfaces to external environment**
  - Use standard internal data representations, convert on input/output
  - Parameterize *all* attributes (constants, types) affecting interface to environment
  - Use tables to encapsulate environment dependent information
  - Use conditional compilation and macro expansion to deal with unavoidable differences

Portability Issues - Hardware

- Data representation
  - Numeric - integers, reals, IEEE vs. DEC vs IBM
  - Character - ASCII vs. EBCDIC, signed/unsigned, collating sequence
  - String - explicit length vs. null term, max length
  - Pointers - range & representations
  - Type size equivalence assumptions
    sizeOf(int) = sizeOf(address), sizeOf(int) = sizeOf(real)
- Portability techniques
  - **Isolate all machine dependent code in separable modules**
  - Parameterize data representation and machine specific information
  - Program to avoid hardware dependent constructs
  - Use conditional compilation and macro expansion
  - Use templates and application generators: e.g., generative programming, automated configuration

Portability Issues - Operating Systems

- Wide variety of Operating Systems

  Unix-like: SunOs, System V, Ultrix, HPUNIX, AIX, IRIX, SEQUNIX, BSD

  Other: DOS, Windows, Next, MAC OS, OS/2, VMS, MVS, . . .

- File System Issues
  - File naming conventions
  - Directory structure, naming, traversing
  - Protection mechanisms, file attributes
  - Limits on concurrently open files
  - Physical structure of files, fixed vs. variable records
  - File access methods: sequential, random, byte, block
- Portability techniques
  - **Isolate all OS-specific code in separable modules**

    Use module level substitution
  - Use conditional compilation and macro expansion

Building Portable Software

- Document rules for customizing software

  e.g. table of options

- Isolate customization to a small number of separate files that may control customization of other files

  e.g. Makefile, config.h

- Try to provide a customization tool (shell script)

  Make config

  configure VMS DECMIPS

- Try to make documentation track customization using conditional text

- Test customization thoroughly

Legal Issues for Software Engineers[ab]

- Contracts - software acquisition or custom development
- Liability and Negligence, Warranties
- Intellectual Property - Copyright, Trade Secrets & Patents
- World Wide Web Legal Issues

---

[a] An informed civilian's comments on legal matters affecting software engineers.

If you need real legal advice, consult a real lawyer

[b] B. Sookman, *Computer Law*, CCH Publishers, 1989+annual updates

Software Contracts

- Agreement by customer to purchase specified piece of software from vendor
- Need to precisely set out
  - Nature of the software being produced, requirements or detailed specifications
  - Other goals affecting the software, e.g. performance, reliability, compatibility, etc.
  - Acceptance conditions, when is software complete?
  - Exception handling - breaches of the agreement by customer or vendor, changes in requirements or specification, cost or schedule overruns
  - Warranties and guarantees provided by the vendor
- Methods of payment
  - Fixed Price
  - Cost plus, vendors expenses + profit
  - Per diem perhaps with an upper bound
  - Progress payments at major milestones

Contracting Issues

- Buyer and vendor cooperate on developing requirements
- Develop system specifications. A complete, correct and unambiguous specification is in the best interest of both parties
- Contract sets out rights *and* obligations of buyer and vendor
- Contract lists deliverables and settles ownership of the software, especially the source code and internal documentation
- Contract species confidentiality obligations of the buyer and vendor
- *Contracting for software is a complicated business, advice from a lawyer specializing in software contracting is essential*

Liability & Negligence

- *Negligence* - Failure to live up to a standard of conduct which reasonable and prudent persons would follow in like circumstances
- Negligence may arise from
  - An act of commission - you do something (e.g. release software known to be buggy) that you shouldn't have done
  - An act of omission - you fail to do something (e.g. test your software adequately) that you should have done
- If a negligent act injures someone (physically or financially) then the person who committed the act may be liable for damages
  - Negligent act may be difficult to document
  - Amount of damages due the injured party may be difficult to determine

Negligence

- Legal view of negligence
  1. Has the perpetrator of the negligent act failed to live up to a minimum standard of care in her/his conduct?
  2. Was the perpetrators act (or failure to act) a **direct cause** of the injury or damage?
  3. Did the injured party fail to exercise reasonable care to prevent the injury (i.e. was there *contributory negligence*) ?
- Problems applying legal concepts to software
  - Lack of demonstrable industry standards for correct behavior
  - Often no witnesses to or records of the negligent act.
    Try to apply legal doctrine *res ipsit locatur*
  - Complex software issues are hard to describe to a non-technical judge and/or jury
  - Disclosures of information can be voluminous and inscrutable

Warranties

- Explicit warranty
  - Contract describes warranty
  - Usually exclude previous or verbal warranties
- Merchantability
  - Vendor knows intended use of software
  - Vendor has a *legal obligation* to supply software that is **suitable for the intended purpose**
  - Buyer's defense: document vendor's knowledge of intended use
- Implicit warranties
  - Industry standards (e.g. spreadsheet should calculate correctly)
  - Pre-contract *written* information that influenced the decision to acquire
  - Advertising for the software in magazines and journals
  - System documentation

Liability Example

- Manufacturer sells hardware to Service Bureau
  Software House sells custom software to Service Bureau
- Customer of the Service Bureau suffers injury due to hardware/software problem
- Who can successfully sue whom for damages?
  - Customer sues Service Bureau
    Must establish negligence on part of Service Bureau
  - Service Bureau sues Manufacturer
    Defense: Service Bureau must test and validate system for self
  - Customer sues Manufacturer or Software House
    Defense: not direct agent causing the injury
  - Service Bureau sues Software House
    Defense: Software was designed to specifications provided by the Service Bureau
    Service Bureau is responsible for testing and validating the software
    Contract precludes responsibility for consequential damages

Intellectual Property Issues[a]

- The goal of intellectual property law is to encourage the development of ideas and devices by giving the originator exclusive rights to obtain remuneration for the use of the idea or device for some period of time
- Algorithms, computer programs and user interfaces are examples of intellectual property
- One of four mechanisms is used to protect intellectual property
  1. Copyrights
  2. Trade Secrets
  3. Trademarks
  4. Patents
- Eligibility of computer software for these forms of protection is contentious and still being decided by courts and legislatures

---

[a] There are many good Web Sites that discuss these issues in more detail, for example `http://www.patents.com`.

The Canadian Intellectual Property Office is at `http://cipo.gc.ca`

Copyrights

- Copyright comes into existence when the work is ”fixed in a tangible medium”. Formal registration can be done later.
- Copyright gives the originator of a work the exclusive right to make (or authorize) copies of the work for *one of*

  28 years, 56 years, life of author + 50 years,

  75 years from publication date, 100 years from date of creation.
- Originally copyright only applied to literary, dramatic, musical or artistic works. Copyrighting computer programs as literary works is a recent stretch of the law into new territory.
- Copyright protects against copying of the *expression* of a work.
  Its not clear that a reimplementation of an algorithm constitutes a violation of copyright.
- The doctrine of *fair use* allows the making of single copies of a copyrighted work for *non-commercial, educational and research purposes*

How to Copyright

- Register your copyright with the appropriate copyright registration agent.

  Do this *promptly* after the work is created.

- A *Copyright Notice* consists of

  1. One of ⓒ *or* Copyright *or* Copr.

     (C) although widely used may not be acceptable.

  2. The date of origin of the work.

     Use the *oldest* date that applies if the work contains reused parts.

  3. The name of the copyright owner.

- For software, place the copyright notice

  – On the distribution medium (e.g. printed on a floppy disk or CD-ROM)

  – As an on screen message at the start of program execution

  – Embedded in the object code where it will be visible to a disassembler.

  – On related documentation

Special Issues for Software

- A piece of software may contain trade secrets or patentable ideas.

  Only *published* works are protected by copyright.

  Publication means that others can read the work.

  Care needs to be taken that publishing a work for the purpose of copyright doesn't give away trade secrets or patentable ideas.

  The U.S. copyright law was amended to help solve this problem.

- Work for Hire - Unless there is a written contract to the contrary, the *author* of a work is the *defacto owner* of the copyright to the work.

  An organization needs to be careful to get signed contracts with the programmers it hires, especially part-time and free-lance programmers.

Trade Secrets

- Trade Secrets protect the *idea*, and are not bound to any specific expression of the idea.

  Trade secret protection ceases if the idea is *independently* discovered by any *legal* means

- Trade secrets offer less protection than copyright or patents but offer protection where copyright or patent protection is not available

- The owner of a trade secret must actively and aggressively protect the secret using physical safeguards and contractual restrictions against unauthorized disclosure
- Requirements for trade secret protection violation
  - The information to be protected must not be widely or publically known
  - The information was imparted to the violator under circumstances that implied confidentiality (i.e. a non-disclosure agreement)
  - The violator made some unauthorized use of the information to the detriment of the party communicating it

Trade Marks

- Trademarks are names or symbols that identify a product. The federal government maintains a registry of trademarks.
- Trademark protection gives the registered owner exclusive use of a name or symbol (e.g. $JAVA^{TM}$)
- Trademark protection against use of the same or very similar name (e.g. $IAVA$) depends on degree of similarity and the likelihood that consumers of the trade marked item will get confused.
  $IAVA$ could be used to name an automobile because there's no danger of confusion with the programming language
- Incorporating a company with a given name does *not* automatically give the company a trademark on the name.
  The company has to apply for the trademark separately.
  Trademarks can't be proper names or words in common usage.

Patents

- Patents give the inventor of an invention an exclusive right to profit from the invention for a period of 20 years.
- An Invention is ”any new and useful art, process, machine, manufacture or composition of matter or any useful improvement in any of these things
- To obtain a patent the inventor must fully disclose the invention and the invention enters the public domain when the 20 year patent period has expired.
- Patent protection (when applicable) is the best form of protection for software. Patents protect the *idea* independent of the expression of the idea and provide protection even if the idea is independently rediscovered.
- Obtaining a patent is a slow and costly process.
- Enforcement of patent rights is expensive

What is Patentable?[a]

- Basic requirements for patentability
  1. Invention must be in one of five classes

     1) processes 2) machines 3) manufactures (made objects)

     4) compositions of matter 5) new uses of any of the above
  2. Invention must be novel, i.e. not done before.
  3. Invention must be non-obvious.
  4. Invention possesses a degree of utility, i.e. it can't be a theoretical phenomenon.
- Unpatentable
  - Any mere scientific principle or abstract theorem
  - Improved method of calculation
  - An algorithm not embodied in a process or device
- Some software algorithms have been granted *process* patents
- Recent trend has been to allow patents for software algorithms

[a] This subject is contentious and continually changing as courts make new determinations

- Non-obvious means not obvious to a person having ordinary skill in the art to which the invention pertains.
- Novelty is measured against *prior art*[a].
  Has the invention been discovered in the past?
- Patent examiners (at least in the past) have been very unfamiliar with prior art in the software area.
- The choice between Patent and Trade Secret protection is complicated.
  Consult a lawyer expert in this area.
- Laboratory notebooks or other contemporary records are very valuable in resolving disputes about the origin of an invention.
- Don’t publish anything that might be a patentable invention until you’ve received legal advice.

[a] The Software Patent Institute (`www.spi.org`) is building a huge data base of prior software art to help prevent bad software patents

What are Patents Good For

- A patent does **not** necessarily give the owner the right to make, use or sell the invention.
- It gives the owner the ability to **exclude others** from making, using or selling the invention.
- A patent owner may be forbidden from using the invention because it contains or uses inventions patented by others.
- In this case the patent owner has to

  Obtain rights to use the other inventions

  or Wait for the other patents to expire.
- **Patents are about control.**

Software Patent Classification

- Items that are **not** patentable.
  - Functional descriptive material

    e.g. description of a data structure, program listing
  - Non-functional descriptive material

    Descriptive material - music, literary works, data

    Descriptive material on computer media, e.g. CD-ROM
  - Inventions without a practical application

    e.g. Manipulate an abstract idea

    e.g. Solve a purely mathematical problem.
- Patentability Requires
  - A series of steps to be performed on a computer.
  - May involve post-computer physical activities
  - Manipulates data representing physical objects to achieve a practical application.

- Unless the invention relates to a particular programming language or hardware device, it should be described in a language and hardware independent fashion.
- The advice of a competent patent lawyer is *very strongly recommended* in drafting a software patent.

  The required terminology and phrasing is arcane and specific forms are necessary.
- It is possible to do your own patent application[a]

  But there is a significant risk of getting incomplete coverage for the invention.

[a] See for example the self-help guide *Patent It Yourself* published by Nolo Press `www.nolo.com`

Web and Internet Law

- How does the World Wide Web affect intellectual property rights?
- How does the Internet affect intellectual property rights?
- Short Answer
  - They don't.
  - Copyright, Trademark and Patent protection aren't negated by the Web or the Internet.
  - The Web and the Internet make it much easier to violate these protections.
  - Many people equate *available on the Web* as *in the public domain*.
    In general this is wrong!
- Remember that copyright protection arises as soon as an original work is fixed in a tangible medium. Lack of an explicit copyright notice is irrelevant.
- This is an area where the law is rapidly evolving.

Use of things found on the WWW

- In general you need the explicit permission of the copyright owner to make copies of an original work.
- In most cases a WebMaster is not the copyright owner and cannot give you permission to copy something on their web page.
- The doctrine of *Fair Use* allows a *small amount* of copying without permission
  - Copying must be for an approved purpose *educational, review*
  - Only a small portion of the work can be copied.
  - Copying is not Fair Use if it diminishes the market for the item.
  - There is no straightforward definition of what constitutes Fair Use.
- The doctrine of *Implied Consent* makes it legal to view copyright material placed on a publically accessible web site.
  This includes the *transient* copies in the web browser's cache.
  It does *not* grant any further copying rights.

Putting Things on the Web

- You can’t put copyright material on the web without the permission of the copyright owner.
- Even if you produce the material yourself (e.g. pictures from a party), you shouldn’t put pictures of others on the web without their permission.
- Linking to images on other web sites (i.e. an IMG link) is equivalent to putting the images up directly. You still need permission.
  A direct link (i.e. HREF link) is probably safer.
- Links to the home page of other web sites is probably OK.
  Deep links to interior pages of other web sites have been challenged in court.
- Framing (i.e. wrapping your frame around content from some other site) is almost certainly a copyright violation.