

## **Classes 21-23: Ethics (from RT Chapter 3)**

### **Ethics...**

- ...is the branch of philosophy concerned with right and wrong (which I won't try to define here!)
- Ethics and legality should not be confused with one another, although they are (we hope!) related

The existence of modern information technology raises some ethical issues (RT Section 3.1, pp. 62-68; in particular, see the checklist on RT p. 63):

- *Privacy*
  - It is becoming increasingly feasible and cost-effective to gather huge amounts of data, especially on individuals
  - What kind of data should be gathered? How long should it be kept?
  - Are individuals aware of what kind of data is being gathered?
  - Who controls this information?
  - How is it used or shared?
  - How secure is the gathered data?
  - How much should employees be monitored?
  - What kind of information should be made widely available?
- *Accuracy*
  - How accurate is your data?
    - Consumers Union consistently finds that credit reports maintained by information brokers like TRW contain errors.
  - Who is responsible for making sure information is accurate? Can injured parties seek compensation for damage caused by inaccurate data?
  - What are the sources of inaccuracy? Are they intentional or accidental?
  - Is new technology making it easier to distribute false or distorted information?
- *Property*
  - Who "owns" the information? What does it mean to "own" information?
  - What are "fair" prices for information exchange?
  - How does one deal with information piracy? Is piracy justified if information prices or ownership are "unfair"?
  - Should employees use corporate information infrastructure for personal purposes?
- *Accessibility*
  - Who has access to information?
  - Is technological change benefiting or hurting particular groups (for example, people with disabilities, people with lower incomes, people in specific geographical regions) by changing their (relative) degree of access?

Expansion of discussion of privacy and property:

### *Privacy:*

- Evolving technology is making it easier to amass ever-growing amounts of data about individuals

- This information is often sold and exchanged by organizations without our knowledge, sometimes intentionally and sometimes unintentionally
  - Massive TJX (T.J. Maxx, Marshall's, Bob's...) data breach revealed in 2007; see RT pp. 60-61. Not only was data compromised, but the company was keeping credit card data it was supposed to delete once transactions were completed.
  - JetBlue violated its own security policy in 2003 by transferring data on 1.5 million customers to a security contractor. This contractor matched the data with other sources to add social security numbers, income, occupation and other data (in old book)
- Is a reasonable degree of personal privacy being eroded?
- Corporations are now required to have *privacy policies* that are distributed to their customers
  - These are dense, long, legalistic documents. Does anybody read them?
  - How about regulatory standards? Fairly weak at present. A bureaucracy/agency to monitor standards compliance would of course have its own drawbacks.
- Combining data from various firms and/or government agencies can allow extremely detailed data to be synthesized
  - At the most basic level, acquire data tables from different sources and then do “join” operations on them
  - *Data aggregator* firms like ChoicePoint (see RT p. 64) specialize in this synthesis
  - In 2004, ChoicePoint fell victim to a simple scam, selling very detailed data on 150,000 people to identity thieves (in old book)
    - The thieves set up over 100 fake companies that each bought relatively small amounts of data from ChoicePoint. Then the thieves did their own join and union operations to assemble a large volume of identity theft data.
    - See RT p. 67 for a security breach incident involving another aggregator, LexisNexis (but this was more of a classic security incident)
  - Data exchange can also have benefits: inadequate data exchange between US government agencies is often blamed for allowing the 9/11/2001 attacks.
  - What is the right balance?
- Monitoring: evolving technology means that we may be monitored more than we might expect from past experience.
  - Security cameras (in the UK, for example, there are estimated to be some 4 million security cameras in operation)
  - If you carry a mobile phone, your movements can be tracked quite accurately as long as it is on; you do *not* have to be making a call
  - Keystroke monitors – can capture every action an employee takes on their computer/terminal (can also be planted by hackers)
  - Do truck drivers, police, and taxi drivers want GPS units reporting where they are every second? (Recent conflict in Philadelphia.)
    - GPS-based car insurance rates?
  - It is legal (but not always expected) for companies to monitor/read their employee's e-mail. The rationale is that the companies own the infrastructure through which employees send and receive e-mail. For some people, it may be important to maintain separate work and personal e-mail accounts. Is there a

good reason e-mail communication should be less private than telephone and postal communication?

- It is common for organizations to keep extensive records of internal e-mails, because they are considered “business documents” (unlike much of the communication they replace, like phone calls and face-to-face conversations). E-mails have become an important source of evidence in prosecutions, lawsuits, and the like.
- Unexpected information release
  - Certain kinds of information (like campaign contributions, property deeds, salaries of public employees) are part of the public record
  - Ordinarily, they are available, but require some effort to get at
  - Some organizations collect this information and either sell it or even make it freely available on the web
    - Example: fundrace.org (in old book) – you can view the campaign contributions of all your neighbors (and see their occupation too)
- New information technologies evolve through innovation in the commercial sphere (although often based on basic research from universities etc.).
  - Lack of precedent and ownership of infrastructure means tend to mean that corporations tend to have the legal upper hand in privacy conflicts with consumers.
  - In the past decade, there was a political aversion to imposing new business regulations; collective security often seemed to take precedence over privacy in governmental management of information, too.
  - How much of this situation is necessary to promote innovation/security?

*Property*: issues regarding ownership and distribution of information

- Trade secrets: information that firms intend to keep confidential or share only with business partners who have agreed not to disclose it.
  - Modern information technology makes trade secrets easier to steal; however, this is primarily a *security* issue (discussed later)
- Copyright and distribution issues: concerns information-based products that firms sell to their customers
  - Examples:
    - Text and graphical material (books etc.)
    - Films and videos
    - Music
    - Database resources
    - Software
  - In the past:
    - Such information was more strongly tied to physical media
    - Physical media were relatively expensive, slow, and/or difficult to copy
      - Quality of copies might be poor
      - It might be hard to make large numbers of copies
      - Copying equipment required major capital investment
      - In some cases, copies could be traced

- Copyright laws have been instituted, dating back as far as 1662, to protect books from massive copying
  - Copyright law augmented at various points in the 18<sup>th</sup>, 19<sup>th</sup>, and 20<sup>th</sup> centuries
  - Since traditional printing presses are large and fairly easy to trace, such laws were mostly adequate for about 300 years.
- Modern information technology has altered the situation:
  - Information more easily separated from the physical medium
  - Can be stored on hard disks etc. and transmitted over high-bandwidth networks
  - Modern input and output devices make quality, hard-to-trace physical reproduction feasible at low cost
    - CD and DVD burners
    - Laser printers
    - Scanners
- Result: massive “piracy” of copyrighted material in some areas
  - Music
  - Film/video
  - Software
- Copy protection technology is only partially effective. In principle, information that reaches the user in unencrypted form can always be copied.
- Piracy uses both physical media and networks (sharing sites like Kazaa, Napster, etc.)
- Music and text/graphics may now be distributed very effectively without the blessing of a mainstream “publisher”. Video will reach that point soon. This phenomenon raises the issue of whether publishers remain necessary.
  - They act as gatekeepers or certifiers of content quality. But how reliable?
  - They can provide marketing/promotion resources
  - They still control the primary physical distribution channels, but there are now effective competing channels
- But how to ensure that musicians, authors, filmmakers etc. are paid for their work? Creating “content” still requires talent and a lot of labor (especially film/video)
- High content prices may not be sustainable in the new environment

### ***Classes 23-26: Security (also from RTP Chapter 3)***

Modern information technology has made it much faster, easier, and cheaper to

- Store
- Move
- Organize
- Manipulate/process

... information than with older “manual” technology.

Unfortunately, the same technology can also make it faster, easier and cheaper to

- Abuse

- Corrupt
- Destroy
- Distort
- Falsify
- Steal

... that same information!

The internet magnifies the problem because millions of unsecured or partially secured computers are now in potential communication.

An electronic business environment also makes impersonation of others – identity theft – relatively easy.

Basic security terminology:

- *Threat*: some danger to proper operation of your information systems.
- *Exposure*: something bad that could happen to your information systems. General categories of exposure include (for example)
  - Loss of data
  - Improper release of data
  - Disruption of normal operations at your organization
- *Risk*: likelihood that something bad will actually happen
- *Controls*: procedures, devices, software etc. that you put in the system to reduce risk
- Hypothetical illustrative example:
  - A hurricane is a potential threat
  - Your data center's exposures to hurricanes consist of disruption of normal operations, and possible loss of data (due to flooding and power failures)
  - Your risk depends on how likely hurricanes are at your location
  - The corresponding controls might consist of anti-flood pumps, plus a backup power system connected to both your computers and the pumps. Another control might be regular off-site backups of your data.

“Risk analysis” process:

- Assess information assets and their exposures
- Estimate probability of each threat
- Consider how to *mitigate* risk via controls
- *Evaluate* cost effectiveness of controls

Risk mitigation strategies:

- *Acceptance*: “live with it”. Examples:
  - A data center in the Nevada desert might choose not to protect itself against floods – they are too unlikely to justify a large expense to protect against
  - You might accept the risk that your top corporate officers could misuse sensitive information – they also have a compelling legitimate need for the information, so that need could outweigh the risk.
- *Limitation*: implement a control that reduces the risk (try to make cost of control proportional to risk)
- *Transference*: transfer the risk to somebody else – for example, buy insurance

There is no such thing as “total” security:

- Don’t think of security issues as “one-time” problems; it is an ongoing process and a portion of the workforce needs to be dedicated to it
- Need to consider security-related costs when looking at cost-effectiveness of computer technology
- Some otherwise good ideas for IT projects might be too risky/difficult to implement securely
- With awareness and effective countermeasures, security can *usually* be manageable

Unintentional threats (accidents):

- Accidents always were a threat to organizations’ data. Fires and hurricanes can destroy paper files just as easily as computer files
- Centralized systems can be vulnerable to problems at the central site
- Distributed systems can be vulnerable to problems at just one site (if their design lacks suitable backups/redundancy)
- Power failures can do a lot more damage than they used to
- With the introduction of computers, there are a lot of new ways for things to go wrong
  - Hard disk “crashes”
  - Software “crashes”
  - Software “bugs”
  - Etc...
- “Human” error and unintentional failure to follow procedures has always been a problem, but can now have more “leverage”
- Loss of misplacement of mobile devices such as laptops, PDA’s, blackberries, and thumb drives (also: unintentional damage to such devices)
- Countermeasures/controls:
  - Backup, backup, backup, backup, backup
    - Data backup
      - Redundancy is undesirable *within* a transaction database, but it is good to have a recent backup copy (or effective equivalent) for the *whole* database.
      - Can restore the database from a recent backup and a log of recent transactions
      - Can back up data to external media (CD-R, DVD-R, tapes) – protect the media!
      - Unfortunately, hard disks have been growing much faster most other media – external hard disks a good choice for PC’s now (store in fireproof box or offsite).
      - Back up data to another site over a network
    - Power backup devices (generators, “uninterruptible” (battery-supplemented) power supplies [UPS], etc.)
    - Backup of software
    - Have a backup plan for entire hardware system (rent replacement hardware, for example)

- For software developed in-house: proper development, maintenance, and lifecycle procedures to contain damage from bugs (discuss later in course)

Remaining threats are intentional – caused deliberately by people

- Internal to your organization
- External to your organization
  - People/organizations you would ordinarily have contact with
    - Partners
    - Vendors
    - Customers
    - Contractors
  - People you wouldn't otherwise have contact with, generally thieves and vandals

Internal threats and problems – employees and consultants

- The larger the organization, the higher the frequency of
  - Employee mistakes or failure to follow procedures
  - Dishonest employees (rarer, but still a concern)
- Shortcuts or dishonesty by MIS employees may have a lot of “leverage” and may be hard to detect
  - “Trap doors” – some unofficial way to gain access to your system
    - Could have been created by an IT employee to make it easier to work offsite
    - Or for some malevolent reason
  - “Skimming” – classic story of rounding down all interest payments to bank accounts, with the extra pennies going into a programmer's account!
  - “Logic bombs” – secret mechanisms that can sabotage a system. Could be either
    - Active – example: if your server receives a particular command sequence over the web, it deletes critical data and shuts down, or
    - Passive – for example, if the employee does not log in for at least two months, the system deletes critical files.
  - ...
- Countermeasures/controls:
  - Separate functions: for example, most programmers shouldn't have access to real customer data
  - Use data access hierarchies and rules
  - Controls on what data may be taken offsite and how
    - Restrict use of floppy drives, CD/DVD burners, even USB ports to prevent sensitive data being moved offsite (for USB ports, this is called “podslurping”)
    - For individuals allowed to take sensitive data offsite, make sure it is properly encrypted
  - Store data in encrypted form so only authorized users may read it
  - Monitoring (this can take many forms, and has ethical drawbacks)
  - Support your employees – make it easy (or automatic) for them to do backup, install security software etc.

Some internal threats involve employees taking/selling sensitive data outside the firm. These threats are very similar to external threats:

External threats – business partner, vendor, and customer issues:

- If you interact electronically with vendors, customers, and partners, you may be exposed to their security problems as well as your own
  - Example: LexisNexis breach, RT p. 67
- Exacerbated by recent “outsourcing” and cooperation trends like
  - EDI (Electronic Data Interchange): firms automatically share data they believe are relevant. For example, we may let our suppliers see our parts inventories so they can plan better
  - ASP (Application Service Providers) – outsourcing of specific applications such as payroll
- Web commerce technology can make improper/questionable monitoring of customers practical/profitable. Examples include cookies (see RT p. 78) and web bugs
  - Cookies are mini-files maintained by your browser under control of remote sites. They allow websites to “remember” information about you, so you don’t have to re-enter it every time you visit the site. But they can be manipulated to monitor your behavior (to varying degrees, depending on your browser’s security features)
  - “Web bugs” are invisibly small “graphics” in HTML-formatted e-mails. If you view the e-mail, the planter of the bug can tell you received it, and your IP address. That is why many mail programs now ask you first before showing the graphics in HTML-format mail.
- In an e-business environment, it may be harder to tell legitimate vendors, customers, and partners from crooks masquerading as such. Vendors and customers are harder to impersonate in person or even over the phone.
- Countermeasures?
  - Limit access
  - Investigate partners
  - Try to use reputable vendors/partners
  - Encryption
  - Consumer awareness

Other external threats

- Two motivations
  - Personal gain – thieves
  - Malice/troublemaking – hackers etc. (harder to understand)
- These threats always existed, but computer technology – and especially network technology – makes attack much cheaper, faster, and easier
- Various forms of undesired software
  - Pestware – software that tries to take over functions from your preferred software, or makes itself hard to dislodge (the architecture of Windows was designed to hide key functions from obvious view, and is therefore very pestware-friendly)
    - Adware – software that “pops up” undesired advertising in various forms; this is the most common form of pestware and usually relatively benign
  - Malware – malevolent software that sneaks into your computer

- Spyware – software that sends information from your system to others without your consent
    - Snoopers and sniffers: monitoring networks as others’ data passes by (especially passwords)
      - Wireless networks especially vulnerable, if not encrypted
    - Keyloggers (used in LexisNexis incident) – record all your keystrokes
    - “Screen scrapers” – capture contents of screen
    - Programs that look for files that may contain personal information
  - Spamware – subverts your system to send spam
  - Pestware/malware may get placed on your computer through malicious websites or e-mail attachments
- Hacking: gaining access to private systems and data (and possible abusing/damaging them)
  - Port scans: try to connect to a large number of different TCP port on a single target system. Which ports respond and how give a “fingerprint” of the kind of system and what software it has
  - Bug exploitation (usually in operating systems, browsers, and e-mail programs).
    - Example:
      - A website has a form which allows people to enter some data (a review of a movie, for example)
      - You find out/suspect the web server software has a “buffer overrun” bug
      - You send a “movie review” that is much longer than the “buffer” reserved for it on the web server
      - Your “review” actually contains a machine-language computer program
      - Your “movie review” overwrites part of the web server program
      - If that part of the program happens to get executed, your uploaded program is now in control
- Spam
  - Time-wasting
  - Nowadays, contents usually criminal/dishonest/fraudulent
  - “Social engineering” and “phishing” – faking messages from tempting or official sources to induce people to run booby-trapped software, reveal passwords, or disclose other confidential information
    - Recent development: “spear phishing” – selecting a specific person so that you can send a more convincing phishing attack.
  - Unintended consequences: would the inventors of e-mail have guessed that the vast majority of all e-mail would eventually consist of unsolicited offers for fraudulent loans, prescription drugs without prescriptions, get-rich-quick schemes, stock pump-and-dump campaigns, and impossible anatomical “enhancement”? Unfortunately, the cost of sending spam is too low.
  - Legislation (“Can Spam”) has not been effective in reducing spam
    - Not a priority for law enforcement agencies
    - Spam sources often in other countries from recipients

- New communication media spawn new kinds of spam – for example instant messaging (IM) spam, called “spim”.
- Annoyance/vandal attacks – denial of service (DoS)
  - For example, bombard a server computer with bogus messages so it has no time or network capacity to do its real job
  - Often executed through “botnets” – groups of computers that have been infected by “malware” allowing malicious control
- Self-replicating attacks: viruses and worms
  - Once present on one system, they try to use that system to propagate themselves to other systems
  - May move via e-mail and have a social engineering aspect (like much spam)
  - But may exploit a software security hole (like a forgotten trap door) and not require any human participation (this mode of propagation is rarer)
  - Can reproduce very quickly if human participation isn’t needed
- The more powerful software is, the more vulnerable (MS Office macros)
- Trojan horses: software that appears to have one function but has a hidden agenda. For example, a free DVD player program that sends personal data back to an identity thief
- Phishing – gathering personal information under false pretenses
  - Example: e-mails supposedly from your system administrator asking for you to “confirm” your personal information or password
- Pharming – set up a website that looks like a legitimate business, and use it to gather personal data. The business may be made-up, or the pharm may masquerade as (“spoof”) the site of a real business
  - In 2007, somebody discovered a problem in TCP/IP that would have allowed massive redirection of traffic from legitimate websites to fake ones; *i.e.* pharming on an unprecedented scale. The software of all DNS servers in the world had to be “patched” in a very short time frame
    - Once a security patch is made public, hackers can analyze its contents and figure out what vulnerability it fixes
    - Thus, once the patch becomes public knowledge, systems without it become vulnerable
    - So, must propagate patch quickly!
- Many attacks combine categories. Examples:
  - spam + phishing + pharming: spam tells you that you need to update your personal information stored by a large company like Amazon, E-Bay, or CitiBank, and then directs you to a fake version of their website
  - spam + virus + spamware: spam has an attachment that takes over your computer and sends the same spam to everybody in your address book
- Hierarchy among hackers and spammers
  - “Script kiddies”
  - Spam pyramid schemes?
  - I receive many copies of essentially the same spam, purportedly from different people. Often the pictures in hundreds of different spams are identical.

## Countermeasures:

### Controls

- User identification and *access controls*
  - Passwords
    - Make sure they are not vulnerable to guessing (see RT p. 83)
      - To prevent computer password attacks that involve a trying a long list of possible passwords, most systems now force a delay period between rejecting an incorrect password and accepting another password.
      - Many systems require passwords that have a mix of letters and numbers, a mix of upper and lower case, and are not dictionary words. Some now also require special characters like #, %, @, etc.
    - Have a change schedule
    - Problems:
      - You accumulate too many passwords
      - So, you have to write them down or use one password for several systems – so compromising one system can compromise others
      - Vulnerable to snooping interception with some older protocols like TELNET and FTP
  - Password generators: small electronic card that combines
    - Fixed user password
    - Internal passcode
    - Time
  - ... to produce a password with a very limited lifetime
    - Example: “SecurID” and “CryptoCard”
  - Biometrics: promising, but:
    - Expense?
    - Reliability?
    - Technology sufficiently mature?
    - Fingerprint readers were common on laptops for a while, but I suspect they were not used much
- Access controls within a computer system (server)
  - Read, write, execute, (delete) privileges for files or blocks of information
  - Basic levels: user/group/all
  - More advanced: hierarchies and “access control lists” (ACL’s). Hierarchical granting of access – used for example for university class registration data:
    - Registrar can see all registrations
    - Registrar grants access to deans to see registrations for their own schools
    - Deans grant access to department chairs to see registrations for their own departments
    - Department chairs grant access to instructors to see registrations for their own courses
- Restrict physical access (physical controls)
  - Example – US Government systems with classified data are supposed to have no physical connection (other than power cords) to any unclassified system.

- If a computer seems compromised by hackers or viruses, physically detach it from the network immediately
- Some physical controls have limitations (for example, “tailgating” – somebody places their ID in a scanner, an automatic door opens, but somebody follows in close behind)
- Audits/verification
  - Example – user-verified paper voting records (how else could you do a “recount” of an election involving electronic voting machines)?
  - IT audits – performed by IT units within major accounting or consulting firms (examples: Deloitte, Accenture)
- Scanning software and hardware
  - Virus scanners: scan received disk files and arriving e-mail for suspicious patterns
  - Spam filters
    - Many use a Bayesian statistical method: use
 
$$P\{\text{message contains “Viagra”} \mid \text{it is spam}\}$$

to help determine

$$P\{\text{message is spam} \mid \text{it contains “Viagra”}\}.$$

Here, the symbol “|” denotes “given that”.

      - Spammers try to confuse these filters by misspelling key words and including large amounts of random text.
  - Watch network for suspicious packet patterns
  - Other forms of monitoring (again, how much is acceptable?)
- Firewalls (hardware and software): block traffic into your network or computer
  - Example – for a home, do not allow any connections initiated from outside; this mode is the default for most home-style routers
  - Example – for medium-sized business, block all incoming connections except SMTP (incoming e-mail) and HTTP (people trying to look at your website) into your mail and web servers (respectively).
- Virtual private networks – use encryption to simulate a private network even if parts are carried over the internet (or some less secure private net)
- Encryption!
  - Encode network traffic so bystanders cannot snoop
  - Can also be used for files on disks, USB memory keys, etc.
  - Unintended consequences: also very useful for criminals
  - Will not discuss technical details of encryption here – discussions in most MIS textbooks are oversimplified.

## **Classes 27-28 – Information Systems Acquisition**

There is a spectrum of ways for firms to construct their information systems:

- At one end of the spectrum, firms can purchase (or lease) “off-the-shelf” or “turnkey” systems, with some minimal configuration
- At the other end of the spectrum, firms can develop their own systems “in-house” or “from scratch”, doing a lot of computer programming

There are many possibilities between these two extremes; examples:

- Buying an off-the-shelf system and adding some custom functionality
- Buying major software modules from various suppliers, and connecting them together with some custom programming.

In practice, nothing is *totally* “from scratch”. Almost any corporate IT project will use

- Mostly standard hardware
- Standard operating systems like Windows or Linux
- Standard database management tools like Access, MySQL, SyBase, or Oracle
- Standard programming languages like C++, Java, JavaScript, Visual Basic
- Etc...

In addition, customized work can also be outsourced: a firm can hire another firm to

- Write customized software, or
- Customize/configure existing software
- Connect standard software modules in a customized way
- Etc...

We use the term “acquisition” to mean any process along this spectrum, from simply buying, to a mostly “from scratch” programming project, and any combination of in-house and outsourced work. “From scratch” work is also called “development”.

RT (Section 10.1) suggests that:

- New information systems should be justified by cost-benefit analyses. In practice, that may be hard to do rigorously.
- New information systems should be “aligned” with the organization’s “strategic plan” – but strategic plans can be very vague, and “alignment” hard to define.

The key points here are that resources for IT acquisition may be limited, so firms should try to prioritize IT projects by:

- Their payoff to the organization (including how closely they are related to any defined strategic goals)
- The amount of effort required.

It’s hard to focus on a lot of complicated projects simultaneously, so it is helpful to have a process that keeps too many projects from going forward at once (see for example the British Telecom example, RT p.299). Adoption of any new system, even if it’s “off-the-shelf”, should be considered a “project”.

Acquisition of new systems often costs much more than expected and can sometimes fail spectacularly – examples from the US government (including the FBI, FAA, and IRS) are well known, but there are many private examples too.

The most proven management process for avoid repetition of “classic” acquisition mistakes, is called SDLC – *System Development Life Cycle*.

SDLC is a cascade or “waterfall” of stages; see Figure 10.2 on RT p. 302.

- I have seen descriptions ranging from 5 to 8 stages. There are many variations in the exact number of steps and their names. Most critically, there are two stages at or near the beginning called “analysis” and “design”.
- Each step has a “deliverable” on which all interested parties “sign off”. In the early stages, this deliverable is a document, most importantly a “specification” and a “design document”. Later, the deliverable may be some version of the system itself.
- If problems are found at any stage, you go back to the previous stage, or perhaps back more than one stage. But the idea is to plan ahead at each stage to reduce the probability of having to go back later, and the severity of the issues that might have to be revisited.

A 6-stage version:

1. Feasibility and planning (called *investigation* in the book)
2. System analysis
3. System design
4. Programming (also called “implementation”)
5. Cutover (sometimes also called “implementation”, just to confuse things)
6. Maintenance

The following description assumes that the project involves a significant amount of custom programming or system configuration. For predominantly “off-the-shelf” adoption projects, some of the steps below can be greatly condensed.

Step 1: *Feasibility and Planning* – identify the problem and the general form of the solution

- Identify problem to be solved
- Determine goals
- Evaluate alternatives
- Examine feasibility
  - Technical: do the necessary technology and skills exist? Do we have access to them?
  - Economic: will it be cost effective to develop/acquire the system? Will the system be cost effective in practice?
  - Organizational: will the system be compatible with the organization’s legal and political constraints (both internal and external)
  - Behavioral: will the people in the organization accept the system? Will they be likely to sabotage, override, or ignore it? What kind of training and orientation will be necessary? How will they be likely to use it? Are we attempting technical fix to an organizational problem that would be best addressed another way?
- Sometimes this step can be merged with the systems analysis (next) step

Step 2: *Systems Analysis* – specify exactly what the system will do

- Define inputs, outputs, and general methodology
- Create basic conceptual structure

- Specify in detail how the system will look to users and how it should behave
- Possibly construct dummy screens/forms and reports, or even prototype systems
- Leads to a *requirement* or *specification* document. This document should be “signed off” by the parties involved, especially those who will use the system.

Step 3: *Systems Design* – say how you will meet the specification

- Describe as collection of modules or subsystems
- Each module may be given to different a programmer or team
- Design specifies how modules will communicate (inputs, outputs, etc.)
- Can use specialized/automated design tools
- Can build prototypes
- Leads to a *design document* – a description of how you will create the system. Managers and programmers “sign off” on this document.
  - Many “computer people” like writing code but not documents, so they may resist this phase
  - *But* it is much cheaper and easier to catch big mistakes in a design document than after you’ve started writing a huge program or bought an off-the-shelf product that can’t easily do what you want.

Step 4: *Programming* – build the system! (Also commonly called *implementation*.)

- Test subcomponents thoroughly as you create them
- Make *unit tests* to exhaustively test each module before connecting modules together
- Some firms have a separate group of “QA developers” to test things again, possibly with help from future users. In the book, this sub-step is depicted as a separate stage called “testing”.

Step 5: *Changeover* or *cutover* – start using the new system (sometimes also called *implementation*, just to keep things confusing)

- *Crucial*: final testing before cutover
- Cutover can be really painful, especially if the old system was already automated
- Options:
  - “Cold turkey” – do it all at once; very risky
  - Parallel – use both systems at once
  - Phased – gradual
    - By part of system
    - By part of organization (regions, departments)
    - Can be difficult to implement; allowing for phased cutover may complicate the design of the new system and may require improvements to the old system (even though you plan to stop using it soon).
- Not unusual for organization to “roll back” to an old system (and maybe try again)
- Cutover is much easier if users were already “on board” in specifying the new system
- Preparation/training might be crucial in some cases

Step 6: *Maintenance* – fixing problems, adding features

- Except in emergencies, it's best to collect sets of changes into a *release* which can be thoroughly tested
- Install new releases periodically; not too often
- Develop a “QA suite” or set of *regression tests* to check that bug fixes don't create more problems or revive old bugs (“rebugging”)
  - Expand QA tests as features are added

It is critical to involve eventual system users in decision-making in most stages (typical exceptions: programming and design).

Outsourcing of programming work may obviate a firm from having to follow some of the SDLC steps, but it does not exempt a firm from *all* of them. The system analysis phase and involvement of users will still be critical, especially for ambitious projects. Outsourcing and offshoring can make the feasibility and analysis steps harder by impeding communication.

Try to avoid having additional features and capabilities creep in at each stage (“scope creep” or “feature creep”): decide what you are going to do, how you'll do it, and then “just do it”.

Overall benefits of SDLC as opposed to less structured approaches:

- Easier to estimate time and effort for the project
- Easier to monitor progress
- More control over scope creep
- Can stay closer to budget and deadline
- Easier to integrate work of different contributors
- More communication between users and developers, less disappointment in final results.

Main drawbacks of SDLC:

- Can be cumbersome and slow.
- Can inflate the cost of making small changes or adjustments.

RT also describes some alternatives (or complementary approaches) to SDLC. I don't have experience with them. I do have experience with SDLC, and it is enormously beneficial – especially the formal analysis and design phases.