• Access has limited ability to handle concurrency: it is really designed as a relatively small-scale desktop database tool.
• A brief search on the web indicates that the maximum number of users an Access application can handle will typically be about 10, and could be lower.
• The database tables need to be stored on a system supporting Microsoft (SMB) file sharing.
• MS Access allows you to store some objects in one .mdb file, and other objects in another. Typically, you would store all changeable tables in one .mdb file, stored on a file server, and all other objects (such as forms, tables that don’t change, and macros) in duplicated .mdb files on each user’s hard drive. This technique will reduce network traffic.
• MS Access does support the concept of grouping record modifications into a transaction.
• If you anticipate needing more than 10 users or need rapid transaction processing, you should be considering using a more “industrial strength” database engine. There are a lot of these (Microsoft SQL server, Oracle, SyBase, DB2, …), but they are generally much more expensive than MS Access. The general hierarchy that applications go through as they grow is:
  o Ad hoc record keeping in spreadsheets or word processor files (plus macros and/or manual procedures)
  o A system based on a desktop relational database like Access
  o A system based on a concurrent relational database like Oracle

Remainder of Class 22: Review MS Access Queries

Class 23: Second Midterm

Class 24: Integration, ERP, and Decision Support

Because of their limited scope, functional-level systems are relatively easy to design, build, understand, and maintain. That means they are relatively easy to create “in-house”, cheaper to buy from outside, and cheaper/easier to customize.

Integration is a key issue with functional-level systems. At some point, they must inevitably retrieve data from or provide data to other functional-level systems. Integration is hard to do smoothly.

• For each particular kind of data, one system must be declared the “master”
• Data can be periodically copied or “mirrored” from one system to another. For example, sales data may be periodically batch-loaded from sales to the accounting and production departments. This practice means that some information may be out of date, and there may be no consistent “big picture” snapshot of the entire operation
• Systems can query one another for information on an as-needed basis. This practice can result in tighter integration, but also slow performance and excessive network traffic
• In an extension of the above technique, one might create a “federated” portal that makes the entire organization’s data look like one giant database, but would actually decompose queries into subqueries, one for each functional system, and then combine the results. Such solutions can be effective but may have slow performance.
The alternative is to replace multiple functional systems with one larger integrated system. This approach, called *Enterprise Systems*, is becoming the most common and effective approach, but has its drawbacks.

Curiously, ERP evolved from systems designed for the production/manufacturing function.

- Originally, such systems were called MRP (Material Requirement Planning). These systems kept track of how many of each kind of part or subassembly make up each other part/subassembly. Thus, an order for six airplanes could be fairly easily translated into an order for 6072 screws, 88 miles of wire, etc. MRP was mainly concerned with inventory and fairly crude management of production schedules.
- “MRP II” added the ability to estimate financing, labor, and energy requirements.
- Finally “ERP” (Enterprise Resource Planning) evolved to include a transaction base for other functional departments.
- Some related acronyms:
  - CRM: *Customer Relationship Management*
  - PRM: *Partner Relationship Management*
  - SCM: *Supply Chain Management*.

SCM concerns the related flow of

- Physical material
- Information
- Money,

both within a firm’s production process and to/from suppliers and customers. It is particularly fashionable to concentrate on material inventory issues. A claim made by SCM adherents is that automated sharing of information with suppliers and customers can reduce inventories, improve forecasting, and reduce inventory variability along the entire supply chain; that is, improved information flow, facilitated by information systems, can reduce the need for expensive physical inventory “buffers”.

Leading ERP vendors:

- SAP (R/3)
- Oracle
- Computer Associates
- PeopleSoft (being taken over by Oracle)

ERP systems typically consist of “modules” roughly comparable to functional-level systems, but integrated in a single giant database design scheme (picture and entity-relationship diagram with 9000 tables!).

ERP drawbacks

- Very high cost
- Having to buy modules you will never use
- Loss of flexibility – you may not be able to do it exactly “your way”
- Modules may not have all the functions of more specialized software
• Configuring/customizing ERP is a very specialized skill
• ERP tends to marketed as a way to wipe away all the problems of legacy systems. But the difficulty of “crossing over” to ERP is frequently underestimated. Sometimes the budget is grossly exceeded and the ERP project may even be killed before tangible results are obtained (see TRP pp. 260-261).
  o Third quarter of 1999: ERP/SCM introduction blamed for a 19% earnings drop at Hershey
  o 1999: large numbers of Whirlpool appliance shipments delayed by new ERP
  o 1996/2001: FoxMeyer (a prescription drug distributor) blames bankruptcy filing on ERP introduction

To get around some of these limitations, ERP systems can be interfaced with custom or “best of breed” systems with specialized functions.
• In this case, the ERP system is typically keeper of the transaction-based “master” data.
• Such interfacing has many of the same drawbacks as trying to integrate individual functional-level systems.
• ERP vendors keep adding capabilities and modules to reduce the need for interfacing and capture even more of the business software market.

Personal observation: many business decision makers don’t know (at least any more) as much about designing information systems as you do now. Even if they do, the wider the scope of a system becomes, the more complex it tends to be, and the harder it becomes for non-specialists to understand. Therefore there is a natural tendency of very wide-scope information systems like ERP to be outsourced to firms that specialize in them. There are very strong economic arguments for this arrangement, but it implies a certain loss of control and understanding by the firm using the system. Thus, instead of deeply understanding how one’s information systems are working, one can be reduced to throwing around acronyms and trying to interface between gigantic, monolithic, and perhaps poorly understood applications like R/3. Perhaps we need a more modular approach to building systems out of component parts, somewhat akin to the old functional-level systems, but designed for easier integration with one another.

**Decision Support Systems (DSS):** systems to help with planning and decision making. DSS goes beyond just presenting reports and leaving the decision process itself entirely up to people. Such systems may be needed because:
• The decision may be too complex for unaided human decision makers
• It may be manageable for human decision makers, but not in the time available
• Sophisticated (mathematical) analysis may be required, which is much easier and more reliable if automated.

Decision support systems require *modeling* of the system to be decided about. There are two kinds of modeling:
• Explicit, often using “management science” and/or statistics tools. This type of modeling typically requires mathematical skills and makes the model assumptions explicit. The rudiments of such skills are covered in the “Operations Management” and “Statistical Methods for Business” classes. Such modeling can also be more logic-oriented than
mathematical, based on systems of rules and deductions (see the topic of expert and “intelligent” systems below).

- Implicit/automated – we may use some kind of “canned” methodology to construct the model. In this case there is generally some sort of “learning” or “fitting” procedure that adapts the model to the problem at hand. The model assumptions may not be immediately discernable, and model behavior in unusual situations may be hard to predict. This sort of “magic box” approach is therefore risky. Example mentioned in book: artificial neural networks.

DSS systems typically need to “harvest” data from TPS or other systems. This data is then processed and plugged into the model.

Some standard categories of decision support:

- **Logistical/operational planning systems** – these tend to use mathematical “management science” models and analytical (although more complicated) akin to what’s in the “Operations Management” course, augmented by an interface to the organization’s databases and a suitable user interface.

- **Data Mining** – systems to detect patterns in large volumes of data. These may use a combination of analytical techniques, including statistical methods

- **Expert Systems** – systems that try to mimic the abilities of human experts who may be in short supply.
  - A common approach here is to build a “rule base” or “knowledge base” of rules or facts that you believe to be true about your business. These rule/knowledge bases are far less structured than relational databases. Examples:
    - A defective power supply 20-volt output causes an overheating finisher unit
    - Overheating finisher units cause paper jams

  - The rule/knowledge base can be processed by an *inference engine* that tries to combine the elements of the rule base in an intelligent way. A very simple example:
    - Based on the above information, a defective power supply 20-volt output causes paper jams

  - Such systems have been applied to
    - Medical diagnosis
    - Equipment troubleshooting
    - Online decisions whether to extend credit to customers

  - **Dangers:**
    - A contradiction within the knowledge base can cause the system to go haywire
    - Due to the complexity of the interactions within a large rule base, the system’s behavior in unusual situations may be hard to predict

- **Constraint Logic Programming** is a methodology for building logistical/operational planning systems using a technology related to expert systems – using simulated logical deductive thinking to find good or best solutions to planning problems. These might also be called “expert systems”, depending on one’s definition of the term.
o An advantage of these systems is that they can are very flexible in terms of the kinds of data and constraints they can handle. They often are the best solution to decision problems that have very complex constraints and few possible solutions.
o If there are too many possible solutions, these methods may not be very efficient and finding the “best” one. This is contrast to “management science” methods, which are less flexible in handling complicated constraints, but better at determining the best of many possible alternatives.
o Examples of successful application:
  ▪ Allocating gates to flights at airline terminals
  ▪ Determining the NFL schedule (but the referee schedule is created by a more “management science” approach)

**Classes 25-26**
- Go over results of midterm
- Unary (self) relationships – employee/supervisor and course catalog examples

**Class 26-27 – Acquisition and Development**
We now have a rough idea how standard business information systems are typically constructed, with a focus on the relational database elements. Now, we’ll briefly discuss the management and “business process” issues of creating or replacing information systems. These topics are covered (somewhat differently) in TRP chapters 11 and TG6.

Discussion here is written from the standpoint of software, but the same basic principles also apply when system development involves acquiring hardware or developing custom hardware.

I’ll organize the discussion around four main issues:
- The structure of the process by which the information system gets created; here, most experienced people favor various forms of the SDLC (System Development Life Cycle) framework.
- The technical aspects of how the system is implemented
- Who does the implementation
- How the services are delivered

Common problems with developing information systems, especially software:
- Schedules slip
- Budgets are exceeded
- Final product may not meet stakeholders’ expectations
- It is hard to make systems reliable; problems can be very disruptive
- Extra features tend to creep in along the way (example from FAA air traffic control)

**Systems Development Life Cycle (SDLC) framework:**
1. A 5- to 8-stage process. There are many variations in the exact number of steps and their names; see Figure TG6.2 (p. 490) for one example.
2. Each step has a “deliverable” on which all interested parties “sign off”. In the early stages this is a document. Later, it may be some version of the system itself.

3. 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 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. Implementation
5. Cutover (sometimes called “implementation”, just to confuse things)
6. Maintenance

Sometimes there is an additional “evaluation” stage at the end, but perhaps that is best considered something that should be done throughout the other stages:
- How effective is our solution?
- Did we meet budget and deadline goals?
- (Once system is operating) Is it reliable? What improvements can we make?

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?

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
- Can construct dummy screens/forms and reports, or prototype systems
- Leads to a requirement or specification document. This document should be “signed off” the parties involved.

Step 3: Systems Design – specify how you will meet the specification
- Describe as collection of modules or subsystems
• Each module may be given to different 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.
  o Many “computer people” like writing code but not documents, so they may resist this phase
  o 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 do what you want.

Step 4: Implementation – build the system!
• Test things thoroughly as you create them
• Make unit tests to exhaustively test each module before connecting modules together
• Some firms have separate QA developers to test things again

Step 5: Changeover or cutover – start using the new system
• Crucial: final testing before cutover
• Cutover can be really painful, especially if the old system was already automated
• Options:
  o “Cold turkey” – do it all at once; very risky
  o Parallel – use both systems at once
  o Phased – gradual
    ▪ By part of system
    ▪ By part of organization (regions, departments)
    ▪ Can be difficult to implement
• 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, best to collect changes into a release which can be carefully tested
• Install new releases periodically; not too often
• Develop a “QA suite” or regression test to check that bug fixes don’t create more problems or revive old bugs (“rebugging”)
  o Expand QA tests as features are added

Critical to involve users in decision making in most stages (exceptions: implementation and design).

Try to avoid having additional features and capabilities creep in at each stage (“scope 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.
• Inflates cost of making small changes or adjustments.

Alternative or complementary approaches:
• Prototyping: rapidly build test/demonstration systems. Users can interact with these systems and suggest changes.
  o Iterate (repeat) until the system looks acceptable
  o Users may produce much more specific and helpful suggestions from interacting with a prototype than from reading a specification document or attending a meeting.
  o But – may tempt you to skip necessary analysis and planning
  o Tends to lack documentation and paper trail
  o Can result in too many iterations
  o Can be combined with SDLC to try to get benefits of both approaches (use in planning and analysis stages)
• JAD – Joint Application Development: try to pack a lot of the process into one gigantic meeting (or a few of them).
  o I don’t have experience with this approach
  o Gigantic meetings can be horrible
• RAD – Rapid Application Development: use specialized software tools to speed up the process.
  o Example: software tools for the design phase that then automatically do the implementation
  o Usually best applicable to narrow, well-defined applications or portions of the application.
    • For example, there are very efficient tools for building user interfaces, so long as those user interfaces stay within certain rules
  o CASE (Computer Aided Software Engineering) tools are an example
  o Can lose design flexibility
  o You can think of MS Access as a simple form of RAD. For database applications that conform to certain restrictions, you can just hook up some tables, queries, forms, and reports, and get something reasonable without any activity that seems like “classical” programming.

Bottom line: SDLC has proven itself in practice.

**How to Build Information Systems**: technically, how do you design and implement? A spectrum of options, running across:
Custom programming

- Advantages:
  - Lots of control
  - Easy to include special features and capabilities

- Disadvantages:
  - Programming is difficult, time-consuming, hard to control, expensive
  - Need to keep programmers around for maintenance phase

Assemble applications from components:

- Examples of common parts to buy:
  - Specialized math/analysis components
  - Security components: encryption/decryption
  - Credit card interfaces; shopping cart

- Advantages:
  - Reduce development time
  - Leverage expertise of others
  - Hide complexity of things like credit card approval – a classic “layering” technique

- Disadvantages:
  - If you only need a few features, expense and interface complexity may not be needed
  - Some loss of control

Note that virtually nothing is “totally” custom-built any more. Even if you develop the “whole” system yourself and do a lot of programming, you are still probably using:

- Standard (or mostly standard) hardware
- Standard operating systems
- Standard programming languages and supporting software libraries
- Standard database engines (like Access, Oracle, etc.)

Purchasing external solution: but something off-the-shelf and configure it for your needs

- Extreme example: ERP software like SAP
- Amount of customization depends on generality of software
  - There are lots of “turnkey” systems for “standard” businesses like auto repair, doctors’ offices, and landscaping. You just enter some basic data and use it.
  - More complex, general software needs more configuration; it can potentially get very complicated and specialized
- Allows you to take advantage of outside expertise
May lose flexibility – you may have to modify your business processes to conform to the system.

General turnkey solutions like SAP can be very expensive.

For simple, standard applications (example: a dentist’s office), turnkey systems will be hard to beat. For very large or specialized systems, the economics are not necessarily so favorable.

Recent development: open source software like Linux. The source code is freely available. Users can understand what the code is doing, make modifications, and submit upgrades back to the central repository.

The general trend in the industry has been for systems to become gradually more interoperable. Examples:
- Adoption of standard networking protocols (TCP/IP)
- Apple Macintosh idea (partially based on earlier developments at Xerox) of a standard general user interface and cutting/pasting between different desktop applications (from different vendors!). This idea spread to Windows and other operating systems.

Perhaps this trend will continue and it will become easier to piece together systems from components.

Who does the development work? Whether you are custom programming, assembling components, or just configuring an off-the-shelf product, some development work needs to be done. Who will do it? Choices:
- Your own employees
- Contract or temporary employees working for salary
- An external vendor, as part of contract to develop/deliver the system

Own employees:
- Advantages:
  - Easiest to integrate with users and managers they’ll need to interact with
  - Understand your problem best
  - Commitment to your firm
  - Cheapest over the long term if their skills are fully utilized
- Disadvantages
  - May be difficult to find and keep
  - May not have the specialized skills needed for every project
  - If they do have specialized skills, you may not be able to use them very efficiently

Contract/temporary employees:
- Advantages:
  - Can take advantage of specialized skills it’s in inefficient to maintain in-house
  - Pay them only when you need their specialized skills; then they move on to another client
- Disadvantages
  - Possible communication problems
o Might be less committed to solving problems
o May not understand your business as well as permanent employees
o Could be more expensive if you need them more than you originally estimated
o May be hard to get back if you need them again.

Outside vendors/outsourcing:
- Advantages and disadvantages are similar to contract/temporary employees, only more so. Additionally:
  - If the outside vendor promises to develop a system for a fixed-fee contract, you can shift development risk to them
  - But, disagreements could arise whether the vendor has fulfilled the contract requirements.

If you use outside vendors to build the system, or buy an off-the-shelf solution, you may have to invest far less effort in the SDLC process. But you are not totally excused from it! For example, you still need to perform planning and analysis to determine what your needs are, and whether an outside vendor’s product will meet those needs.

**How the system’s services are delivered:**
- Traditional approach: you own the hardware, you have non-expiring licenses for all external software components (at least for the release you’re using), and your own employees operate the system
- ASP (Application Service Provider) approach: you get the service the system provides, but you don’t possess the hardware, software. Instead, an external vendor maintains the system, and perhaps supplies some of the labor to operate it. You may simply access it via your employees’ web browsers.
  - This is a very well-established approach in a few areas, like payroll processing
  - When the internet and web got “hot” in the 90’s, there was a lot of hype about expanding these kind of services
  - Supposedly would let firms cut costs dramatically by focusing on “core competencies” and cutting back massively on their own IT staffing needs.
  - Drawbacks:
    - You need a lot of confidence in the ASP. They have your data and they are operating a key aspect of your business
    - Many firms are reluctant to give up control/ownership of their data
    - Firms are reluctant to be so dependent on an external vendor for their basic operations. Suppose the vendor starts having problems, or raises fees once you’re “locked in” and have lost key IT staff?
  - For these reasons, I don’t expect to see massive growth in ASP’s. ASP activity will be in focused areas which present low risk and where ASP’s are well established and can be cost effective. Examples: payroll and processing rebates.

**Classes 27-28 – Ethics and Security (TRP 12)**

**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 (p. 364)

• **Accessibility**
  o Who has access to information? Is technological change benefiting or hurting particular groups by changing their (relative) degree of access?

• **Accuracy**
  o Who is responsible for making sure information is accurate?
  o Is new technology making it easier to distribute false or distorted information?

• **Privacy**

• **Property**

Privacy and property are probably the most critical.

**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 (JetBlue example, pp. 367-377)

• Is a reasonable degree of personal privacy being eroded?

• Corporations are now required to have *privacy policies* that are distributed to their customers
  o These are dense, long, legalistic documents. Does anybody read them?
  o How about regulatory standards?

• Monitoring: evolving technology means that we may be monitored more than we might expect in the past.
  o Security cameras (for example, in the UK)
  o 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
  o Keystroke monitors – can capture every action an employee makes on their computer/terminal
  o It is legal (but not expected) for companies to monitor/read all of their employee’s 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?

**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.
  o Modern information technology makes trade secrets easier to steal; however, this is primarily a *security* issue (see below)

• Copyright and distribution issues: concerns information-based products that firms sell to their customers
  o Examples:
    • Text and graphics 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
  - Copyright laws were instituted dating back to 1662 to protect books from massive copying
    - Augmented at various points in the 18th, 19th, and 20th centuries
    - Since printing presses are 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. 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 can now be distributed very effectively without the blessing of a mainstream “publisher”. Video will reach this point soon. This raises the issue of whether publishers remain necessary.
    - Act as gatekeepers or certifiers of content quality. But how reliable?
    - Give access to a marketing/promotion resources
    - Still control 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.
  - High content prices may not be sustainable (examples: textbooks, some music)

TRP 12 also has an interesting section on “impacts”, including some good information about ergonomics. I will not dwell on these topics in class, however.

-- 36 --
Security: 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!

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 these costs when looking at cost-effectiveness of computer technology
• 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
  o Hard disk “crashes”
  o Software “crashes”
  o Software “bugs”
  o Etc…
• Countermeasures:
  o Backup, backup, backup, backup, backup
    • Redundancy is bad within the context of a transaction database, but it is good to have a recent backup copy of 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 than backup options!
    • Back up data to another site over a network
• Power backup devices (generators, UPS, etc.)
• Backup of software
• Have a backup plan for hardware (rent replacement hardware, for example)
  o For software developed in-house: proper development, maintenance, and lifecycle procedures to contain damage from bugs (see material on SDLC above)

Remaining threats are intentional – caused deliberately by people
• Internal to your organization
• External to your organization
  o People/organizations you would ordinarily have contact with: partners, vendors, customers
  o Thieves and vandals

Internal threats and problems – employees and consultants
• The larger the organization, the larger the frequency of
  o Employee mistakes or failure to follow procedures
  o 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, skimming, “time bombs”, …)
• Countermeasures:
  o Separate functions: for example, most programmers shouldn’t have access to real customer data
  o Use data access hierarchies and rules
  o Encryption?
  o Monitoring (this can take many forms, and has ethical drawbacks)
  o Support your employees – make it easy (or automatic) for them to do backup, install security software etc.

External threats – business partner, vendor, and customer issues
• If you share interact electronically with vendors, customers, and partners, you may be exposed to their security problems as well as yours
• Exacerbated by recent “outsourcing” and cooperation trends like
  o 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
  o ASP (Application Service Providers; see above)
• Web commerce technology can make improper/questionable monitoring of customers practical/profitable (cookies)
• In an e-business environment, it may be harder to tell legitimate vendors, customers, and partners from crooks masquerading as such
• Countermeasures?
  o Limit access
  o Investigate partners
  o Try to use reputable vendors/partners
  o Encryption
○ Monitoring (but how much is acceptable?)
○ Consumer awareness

Other external threats
- Two motivations
  ○ Personal gain – thieves
  ○ Malice/troubleshooting – hackers etc. (harder to understand)
- These threats always existed, but computer technology – and especially network technology – makes attack much cheaper, faster, and easier
- Snooping and sniffing: monitoring networks as others’ data passes by (especially passwords)
  ○ Wireless networks especially vulnerable
- Hacking: gaining access to private systems and data (and possible abusing/damaging them)
  ○ Port scans
  ○ Bug exploitation (usually in operating systems, browsers, and e-mail programs)
  ○ “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
    - New development: “spear phishing” – selecting a specific person so that you can send a more convincing phishing attack.
- Spam
  ○ Time-wasting
  ○ Nowadays, usually dishonest/fraudulent
  ○ Unintended consequences: would the inventors of e-mail have guessed that the majority of all e-mail would eventually consist of unsolicited offers for fraudulent loans, prescription drugs without prescriptions, get-rich-quick schemes and impossible anatomical “enhancement”? Unfortunately, the cost of sending spam is too low.
  ○ Legislation (“Can Spam”) has not been effective in reducing spam
- Annoyance/vandal attacks – denial of service (DoS)
  ○ For example, bombard a server computer with messages so it has no time to do its real job
- Self-replicating attacks: viruses and worms
  ○ May move via e-mail and have a social engineering aspect (like much spam)
  ○ But may exploit security hole (like a forgotten trap door) and not require any human participation
  ○ Can reproduce very quickly
  ○ The more powerful software is, the more vulnerable (MS Office macros)
- Many attacks combine categories
- Hierarchy among hackers and spammers
  ○ “Script kiddies”
  ○ Spam pyramid schemes?
  ○ I receive many copies of essentially the same spam, purportedly from different people
Security Technologies/Techniques

- **User identification**
  - **Passwords**
    - Make sure they are not vulnerable to guessing
    - Have a change schedule
    - Problems:
      - You get too many of them
      - Have to write them down or use one password for several systems
      - 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?
    - Ready yet?
    - Fingerprint readers are now becoming common, especially on laptops
- **Access control within a computer system (server)**
  - Read, write, execute, (delete) privileges for files or parts of files
  - Basic levels: user/group/all
  - More advanced: hierarchies and “access control lists” (ACL’s). Hierarchical granting of access – used for example for Rutgers class registration data
- **Restrict physical Access**
  - Example – US Government systems with classified data are supposed to have no physical connection to any unclassified system.
  - If a computer seems compromised by hackers or viruses, physically detach it from the network immediately
- **Audits/verification**
  - Example – user-verified paper voting records
  - IT audits
- **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 determine \( P\{\text{message is spam} \mid \text{it contains the “Viagra”}\} \).
    - Spammers try to confuse these filters by misspelling key words and including nonsense or random dictionary words.
  - 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
- Example – for medium-sized business, block all incoming connections except SMTP and HTTP into your mail/web server.
- 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
  - Unintended consequences: also very useful for criminals
  - Will not discuss technical details here – discussions in most MIS textbooks are oversimplified.

**Class 28 – Brief Discussion of E-Commerce (Time Permitting)**

This mini-lecture relates to Chapter 5 of the text, which attempts to categorize all manner of e-commerce. I will take a different, quicker approach, and try to categorize what I view as the key aspects of successful e-commerce.

In the 90’s, there was a “dot-com boom”. During this boom, many people was assumed that the internet would take over the entire economy and people trying to sell products like dog food over the web would become billionaires. We have since returned to reality. But the internet has had and continues to have a huge impact. A few companies, like Amazon, E-Bay, and Google have been spectacularly successful, and many smaller companies have done well.

I believe the key aspects of successful e-commerce are:
- Making it easier to locate other parties with which to have an economic transaction
- Making it easier to identify desired products and services
- Decoupling the information aspects of a transaction from the physical transfer of goods
- Reducing the costs of communication between economic parties
- Promoting pull-based communication between economic parties

Note that these categories are interrelated and overlapping.

Making it easier to locate parties (firms or individuals) with whom to conduct transactions:
- If I am looking for obscure cross-country ski accessories, I can now locate them quickly even if I live in Alabama. A simple web search will probably identify dozens of firms selling the sort of thing I am looking for. Before the internet, I would have had to go the library and search through national business directories, or try my luck with 800 directory assistance.
- E-Bay’s core business is helping buyers and sellers of possibly obscure used items find one another easily. If I have a very specialized item, services like E-Bay allow prospective buyers from all over the world to find me, instead of hoping that one of them will drive by my yard sale.

Making it easier to identify desired products and services
- Suppose we don’t know what particular product/service we want. It is now relatively easy to
  - Search the entire e-commerce world using a search engine like Google
  - Search within particular suppliers’ websites
Decoupling information and physical aspects of a transaction

- A bookstore or music store stocking all the titles carried by Amazon would be impossibly large.
- We do not have to depend on having goods distributed to our local area before we make a purchase decision. It is now possible to purchase goods not popular in one’s geographical area, and to have much wider choice of products and suppliers than if we depended wholly on local stores.
- For physical goods, online commerce is dependent on an efficient parcel delivery system. Parcel delivery underwent a revolution in the 1980’s with the formation of Federal Express, and information technology was critical in its growth. Parcel delivery has grown symbiotically with web commerce. Parcel shipping still costs more per unit than bulk shipping to retailers, but allows inventory to be kept in a more centralized, more efficient manner. Thus, the overall “supply chain” may be more efficient in many cases.
- Physical stores and showrooms are not going away, though
  - Physical examination of goods very important for some products
  - Immediate delivery can have enormous value

Reducing communication costs

- A supplier can now update pricing and availability information and transmit it into our firm’s information system as frequently, without human intervention
- Our customers can query flight times, inventory availability, pricing etc. without our having to dedicate so many people to physically talk to them
- Customers can lengthily custom-configure products (like computers) without consuming a lot the supplier’s labor

Pull-based communication

- When communication is initiated by the potential consumer of a good or service
- Internet technologies greatly reduce the cost and human effort required for pull-based communication
- Successful e-commerce tends to contain a large “pull” element. Even effective internet advertising (as on Google) is based on determining a level of customer interest, based on their query
- The internet is also tempting for push-based communication because of very low costs. But the results are frequently irritating: spam and pop-up ads.

I believe e-commerce have the potential to make it possible for market economies to function efficiently with smaller, more entrepreneurial, less internally politicized economic units. But it is not clear whether that will happen.