• 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!
  o Encode network traffic so bystanders cannot snoop
  o Can also be used for files
  o Unintended consequences: also very useful for criminals
  o Will not discuss technical details here – discussions in most MIS textbooks are oversimplified.

**Class 23, 25 – Networking Technology**

Note: these classes cover roughly the material in RTP chapter 5, including the appendix. However, my emphasis will be different, and I'll give some details not in the text.

Broadly speaking, in terms of major impact and commercialization:

- The 1980’s was the decade of personal computing (single-user, desktop computers)
- The 1990’s was the decade of networking

Many of the relatively recent advances and technologies we now associate with computers are really networking-based.

Basic definitions:

- Something that connects two computers is a **link**
- Many computers connected by many links comprise a **network**.
- Each computer on the network is called a **node**.
- Generally speaking, data should be able to get from any network node to any other node.
- There are many different shapes (or topologies) that can be used to build a network
  - Today the predominant network topology is a collection of interconnected “stars”.
  - At one time, interconnected “buses” and “rings” were also popular, and they are still in use (see Figure 5.1). Note that most wireless networks, where everybody broadcasts information via radio, effectively have a “bus” organization.
- Some nodes on the network are specialized computers that serve primarily as connection points whose job is to make sure data gets sent to right place. Some common names:
  - Switches
  - Hubs
  - Routers
  Most of this equipment is not that visible to most people, although a lot of people now own routers or wireless routers for their homes.

Kinds of links

- Link speed is usually measure in bits per second (b/s), with the usual prefixes K (kilo/1,000), M (mega/1,000,000), G (giga/1,000,000,000), etc. Note that network engineers usually use the **decimal** form of these prefixes, not the binary ones.
- Wires (usually copper) these can be used in many ways.
  - Twisted-pair wires, such as
    - Regular telephone wiring
Ethernet wires, which currently come in three flavors, 10 Mb/s, 100 Mb/s, and 1Gb/s.

- Coaxial cables, like those used for cable TV. These have higher theoretical capacity but are harder to work with (they are stiff and hard to connect together).
  Wires can carry a lot of data for short distances, but slow down for longer distances

- Optical fiber (carries light pulses)
  - Invented about 30 years ago
  - More economical than wire for high data rates and long distances. Links can have capacities in the many Tb/s
  - More difficult to work with than either twisted-pair wires or coaxial cables. In particular, it’s hard to “splice” two of them together (however, that also makes them more secure).

- Broadcast electromagnetic waves (radio/infrared/microwave) – “wireless”
  - Microwave links (can be directional – can be formed into a narrow beam)
  - Satellite links
  - Within-building (“wi-fi”) broadcast: capacities typically about 11 Mb/s right now
  - Wider area wireless: slower than wi-fi right now (cell-phone modems); this technology is just emerging, but I believe that demand will make it successful.
  Current services are advertised at $80/month for 400-700 Kb/s access, with approximately the same coverage area as standard cell phones.

A quick history of computer communications:

- The first large-scale electronic networks built were telephone networks. But they were not used by computers initially — because computers didn’t exist! In fact, “computer” was actually a job title for a person who did mathematical calculations for engineers and scientists.

- When computers started, each organization had its own computer in its own room. Data got in and out of the computer room by being physically carried as punched cards, printed reports, magnetic tape etc. (eventually floppy disks, too) – later called “sneakernet”.

- People began placing I/O devices outside the computer room, connected by wires: printers, card readers, terminals (=printer + keyboard or screen + keyboard), etc.

- Technology was invented to encode (modulate) data into sounds the telephone network could carry. The data would be “demodulated” back into bits at the other end (thus the term “modem” – modulator/demodulator); see TRP p. 461.
  - This allowed people to have terminals at home and work over telephone lines
  - Many other business applications involving sending or receiving data from remote locations
  - Early modems were slow (100 b/s = 0.1 Kb/s in the 1960’s). This gradually increased to about 56 Kb/s today.
  - The technology is still widely used, but in decline

- In the late 1960’s, interest was growing in large general-purpose data networks independent of the telephone network.
  - Before, these existed only for specialized application (mostly military)
  - ARPANET – the (defense department) Advanced Research Projects Agency
    Network was built in the early 70’s
  - This network gradually evolved into “the internet”
The internet had a fairly small user base until the mid 80’s. Then it began to gather momentum. In the early 90’s, the “world wide web” became very popular and drove a massive expansion of the internet (along with the “.com” boom). In the 90’s there was a general telecommunications boom of which the internet boom was a big part. Numerous firms tried to secure their place in the boom by building lots of network links, especially in North America. A lot of network capacity was built. About the same time technology appeared that greatly increased the amount of data an optical fiber could carry (by simultaneously sending multiple information streams using light beams of several different colors). Things cooled off a lot, but internet use continues to climb.

How networks work: LAYERING is very important
- Bottom: physical layer – the physical workings of the links (wire, fiber, wireless, etc.)
- Network layer (typically “IP”, standing for “internet protocol”): lets the network figure out what computer the data is meant for.
  - Currently, each computer has a 32 bit “IP address” (usually split into four bytes printed in decimal like 128.6.59.202).
  - The addresses have structure – for example “128.6” in the first two bytes of the address means somewhere at Rutgers (although 165.230 also means Rutgers), the 59 designates a particular “subnet” (roughly the same as a building), and the 202 identifies which computer on the subnet.
  - Note that most computers also have a “hostname” and “domain name” that is easier for humans to remember, like “business.rutgers.edu” or www.amazon.com. While these are related to IP addresses, they aren’t exactly the same. Special computers on the network, called “name servers” provide the translation. Small organizations may not have a name server, relying on a name server from their internet service provider. Large organizations like Rutgers may have dozens of name servers.
  - 32 bits are no longer enough space for an IP address, and we will gradually move from IPv4 (32 bits) to IPv6 (128 bit addresses). Various workarounds suffice for now:
    - Dynamically allocating IP addresses only when computers are actively connected to the network (“DHCP” is a common way of doing this), or
    - Grouping small sets of computers to share a single IP (network address translation or “NAT”)
- Transport layer (typically “TCP”). Specifies how data is split up and logically moved in the network
  - TCP specifies up to 64K (binary K) logical “ports” for each computer on the network. Each port can be used for a different application. For example, standard web page viewing usually uses port 80.
  - For each port, there may be one or more “sessions” or logical connections between to computers. For example, you could have two independent web browser windows connected to the same website from your own PC; each of these windows would represent a different session for TCP port 80 on the server. Each
session is initiated by one computer sending a request to another. If the second computer agrees, the session is opened, and data may flow in either direction.

- For each session, there may be a sequence of messages in each direction.
- TCP is a “packet switched” protocol – messages are cut up into “packets” that might take different paths through the network and are reassembled at the destination. By contrast, telephone networks are “circuit switched” – the whole conversation uses the same route through the network. The size of packets varies by the application and network hardware in use, but they are typically roughly on the order of 1KB.

- **Application layer:** specifies different protocols to move data for various uses. These protocols constitute an “alphabet soup”:
  - First: TELNET (old) – run a terminal session (a text-based interaction between a person and a computer – kind of like the command prompt in Windows)
  - FTP (old) – move files back and forth (still in some use when security isn’t an issue)
  - SSH – encrypted terminal sessions and file transfers. This is how you connect to the “Eden” system to do text-based interactions. This accomplishes the same basic tasks as TELNET and FTP, but is far more secure. Other protocols can be “tunneled” through SSH to make them secure.
  - HTTP/HTTPS – hypertext transmission. This application appeared in the early 1990’s and rapidly evolved into a way of projecting a wide range of graphic user interfaces across the internet. The “S” in HTTPS means secure/encrypted. HTTP is a much easier and more secure way to do arbitrary things on a remote user’s screen than making the user run custom software.
  - SMB, NFS – file sharing. Making disks on a distant computer look like they’re on yours
  - SMTP – sending e-mail to and between mail servers (computers that can route e-mail). This is a “push” protocol: the computer requesting the connection sends the messages; when your computer sends an e-mail, it typically uses SMTP.
  - POP3, IMAP – retrieving mail from e-mail servers. These are “pull” protocols: the computer requesting the connection receives the messages (if there are any)
  - And many, many, more…

- Typically, each protocol uses a single TCP port (or perhaps a few). For example, HTTP usually uses port 80, and SSH usually uses port 22.

Some more notes on layers and protocols

- The standard URL (“Uniform Resource Locator”) used by web browsers has the basic form

  \[ \text{PROTOCOL://hostname/additional-information} \]

  Thus, http://www.rutgers.edu/ means “HTTP protocol, computer with hostname www.rutgers.edu, no additional information”. A numeric IP address can also be used instead of a hostname, as in http://123.57.12.92/obscure-stuff (but is a clue that the site may not be safe to connect to).

- As you move downwards in the protocol layer “stack”, more and more “bookkeeping” data — also called “headers” — get appended around the data you are sending. This means the actual number of bits transmitted can be substantially more than you might think. For example, when a message is divided into packets, information is added to each
packet so that it gets routed correctly and the packets may be assembled again in the right
ordre.

- TCP and IP are usually used together and are known as “TCP/IP”
- You can run more than one network layer on top of a physical layer on the same link (for
  example, IP and AppleTalk)
- You can run several transport layers on top of a network layer (for example, TCP and
  UDP on top of IP)
- And, of course, you can run many application layers on top of a transport layer (SSH and
  HTTP on top of TCP)

Kinds of networks

- LAN – “Local Area Network” on the level of a single building, part of a building, or
  office
- WAN – “Wide Area Network” a somewhat vague term for a network that covers a “non-
  local” geographic area, that is, something larger than a LAN. An example would be the
  network that connects Rutgers’ various campuses (including Camden and Newark).
- Enterprise network – refers to the totality of an organization’s networks, both its LANs
  and its WAN(s) together.
- Internet – multiple networks networked together
  - The idea of *an* internet preceded the current notion of the internet – “the” internet
    came into existence when almost everything got connected into one huge
    network!
  - The “IP” network layer was specifically designed to make it easy to create
    internets. That is why all internets essentially merged into “the” internet that
    grew so quickly in the 1980’s and 1990’s, and conversely why IP is now the
    dominant network layer.
- Intranet – the portion of an organization’s enterprise network that is not accessible by
  arbitrary internet users
- Extranet – when multiple organizations connect their networks in a way that is not fully
  accessible from outside that set of organizations
- VPN – “Virtual Private Network” – an intranet or extranet that physically uses the
  general internet, but is encrypted in such a way that it looks like a private WAN that
  outsiders can’t snoop on (we hope!)

Current network technology

- Most firms now have LANs implemented with copper wire, usually Ethernet, and now
  also building-level wireless
- Many larger firms have WANs containing wire and/or fiber and maybe some satellite or
  microwave links (depending on the firm’s size). The longer links in these networks are
  typically leased from ISP’s (see the next item)
- Internet service providers (ISP’s) are firms maintaining interconnected, overlapping
  networks made primarily of fiber (examples: AOL, ATT, Sprint, etc.), but possibly also
  involving wire, satellite, and microwave links. ISP’s also lease capacity to firms for use
  in WANs. Large- and medium-size firms connect directly to ISP’s.
Also, there are some non-profit alternatives to ISP’s, like “Internet2”, a consortium of large universities like Rutgers, and other large nonprofit organizations.

- Large firms can afford to lease dedicated high-speed connections to ISP’s, like “T3” lines
- The dreaded “last mile”: smaller firms and individual households connect to the ISP’s in various non-ideal ways:
  - By phone and modem (sometimes directly to an employer instead of ISP)
  - Cable modem – signals carried over the same coaxial cable that distributes TV signals. Capacity usually 0.5-7 MB/s, but capacity may be shared with other users in the neighborhood
  - DSL – signals carried over regular phone lines, but not at audible frequencies. About 0.5-1 Mb/s, but occasionally faster. Only works if you are within 2 miles of telephone switching center, but does not have capacity sharing problems.
  - WiMax – (see RTP pp. 223-224) a wireless networking technology with a theoretical reach of up to 70 miles and transfer rates up to 70 Mb/s. However, there is a tradeoff between distance and speed, and more typical performance would be about 10 Mb/s over 6-7 miles, if a “line of sight” connection is possible. This technology is rare in the US and is currently targeted at areas lacking wired infrastructure (especially rural areas). Receiver units can be quite expensive ($349 is mentioned on RTP p. 225).

Most network connections carry a fixed charge per month, without tracking the exact number of bits sent – one reason we have so much “spam”!

Uses for networks are expanding all the time. For a fairly current catalog, see Section 5.4 of RTP.

Data transfer calculations – how much time will it take to move a data file? (The file could contain text, data tables, video, audio, etc.)

- Calculate the amount of data to be moved in bits
- Divide by the speed of the transmission line in bits per second
- Convert from seconds to larger time units like minutes or hours if necessary
- Remember:
  - File sizes are commonly in bytes, with binary-style K, M, G etc.
  - Transmission line speeds are usually in bits per second, with decimal-style K, M, G etc.
  - This mismatch is annoying, but unfortunately, it is the common convention.
  - It’s easiest to convert the file size to decimal bits, and then divide.
  - The protocol stack will add header information that will cause the real download to take longer
  - Network congestion or malfunctions could cause even more delays

**Sample file transfer calculation:** Suppose we want to do “video-on-demand” downloads of 4 GB movies in DVD format (binary-style GB). How long would that take over a 1 Mb/s DSL line?

Size of movie = (4 GB)(1024 MB/GB)(1024 MB/KB)(1024 B/KB)(8 bits/B) = 3.44 \times 10^{10} \text{bits}
Seconds to transfer with DSL = \((3.44 \times 10^{10} \text{ bits})/(1 \times 10^6 \text{ bits/sec})\) = \(3.44 \times 10^4 \text{ sec}\)

\[= (3.44 \times 10^4 \text{ sec})/(60 \text{ sec/min} \times 60 \text{ min/hr}) = 9.54 \text{ hours} – \text{probably not acceptable!}\]

Note that actual transfer times would be somewhat larger due to overhead (headers) added by the application, transport, network, and physical network layers, and because of possible network congestion.

**Classes 26-27 – Kinds of Information Systems**

This material consists of a condensed overview of RTP chapter 8 (which overlaps extensively with material from the first two class meetings of the semester), and then topics from RTP chapter 9.

*Transaction processing systems (TPS)*

- Systems that record everyday business events. Examples:
  - Receiving a shipment
  - Shipping an order
  - Ringing up a sale
- Typically, most TPS processing is “online” – recorded in the database essentially instantaneously. There is still limited use of “batch processing” in which similar transactions are grouped together for efficiency. Batch processing was much more common in the early days of business computing (the 1950’s and 1960’s). Modern systems tend to be all online, or a blend of online and batch.

*Functional information systems:* systems that support a particular functional area like accounting, manufacturing, HR, etc.

“Classic” MIS – systems that provide standard reports for monitoring the state of the business/organization. Such reports are often *routine*, meaning they are delivered on a standard weekly, daily, or monthly cycle. Exception reports are issued only when the MIS thinks something unusual is happening (for example, there are too many quality rejects on the production line).

Integrating TPS and MIS systems that serve different divisions, functional areas, or geographic areas is probably the central problem in modern IT management. Integration problems include

- Systems that communicate infrequently
- Systems that communicate incompletely
- Systems that cannot communicate without manual intervention

*ERP (Enterprise Resource Planning)* systems attempt to bypass integration problems by providing a single system to manage all of a firm’s TPS, MIS, and functional-area IT needs.

- The vendors (SAP and Oracle/PeopleSoft are the leading vendors) essentially try to design a system so powerful that it can adapted to the needs of most organizations by
  - Using some of the available modules, and disabling others
  - Applying “configuration” settings to tailor the behavior of the modules in use.
• ERP systems can contain thousands of tables
• When ERP works, the benefits can be enormous (example, Under Armour, Inc. on RTP p. 249).
• However, the configuration process may be long, painful, and extremely expensive. Extensive, costly customization is often needed, or the customer must change their business processes to fit the system (example: Stanford University on RTP pp. 250-251).

Data warehousing is a common alternative to complete ERP integration. A firm periodically (nightly, weekly, monthly) copies critical data from one or more TPS systems to another system called a “data warehouse”. The copy may then be used for reports, analysis etc.
• The copying step provides a clearly delineated, relatively simple interface between the systems
• Customization is simpler than for a more tightly integrated approach
• After copying is complete, analysis of the warehouse data and normal transaction processing don’t interfere with one another, meaning both can proceed faster
• Drawback: the warehouse data may be somewhat out-of-date, so the warehouse cannot be used to track very rapid or recent developments
• Note: while normalization is critical for TPS databases, it may not be for the warehouse. One may want to use a “denormalized” data representation in the warehouse, allowing large queries to be done with fewer joins, and thus more quickly.

CRM (Customer Relationship Management) systems – systems focused on customer relationships. Slogan: “treat different customers differently” without a huge investment of labor.
• CRM systems may be interfunctional: they might combine aspects of sales, marketing, and customer service.

SCM (Supply Chain Management) systems – manage flow of materials, information, money and services between parts of the firm, and also between the firm and its suppliers and customers.
• SCM systems may also be interfunctional
• SCM systems are often interorganizational: for example, allow your suppliers to view your inventory position for the components they supply, so they can plan better

EDI (Electronic Data Interchange) denotes automatic data exchange between two distinct organizations. Examples:
• Interorganizational SCM.
• Payroll processing: it is very common for small and medium-sized firms to outsource payroll processing to outside providers. EDI can make this process more efficient.

EIS (Executive Information Systems) – systems targeted specifically at the needs of upper management.
• A common approach is to package the kind of information available in standard MIS reports in a more flexible, interactive form
• “Drill down” – part of a report, chart, or display expands to show more detail when clicked
Although they may be on a larger scale, with more simultaneous users than MS Access can handle, many information systems can be constructed out building blocks similar to those we have seen in Access tables, forms, reports, and queries. That is, the system tracks evolving information through transactions that each modify a limited number of rows of various tables, and retrieves, processes, and aggregates information through queries much like those we have used in MS Access. Thus, the main ingredient one needs to implement such systems is a Relational Database Management System (RDBMS), of which Access is an entry-level example.

Some systems may need to do things that go beyond the capabilities of an RDBMS – that is, they may need to perform calculations or analysis beyond that available in a standard database query, or even in a set of chained queries. Possible Examples include portions of the SCM and CRM systems mentioned above, depending on the sophistication of their analysis. Another kind of system that frequently needs to go beyond the basic capabilities of an RDBMS is a DSS (Decision Support System).

DSS (Decision Support Systems) – systems designed to improve or accelerate a specific decision or planning process.

- Sometimes, “decision support” can be accomplished by simply aggregating, arranging, and presenting data in the right form, with only the kinds of capabilities we find in RDBMSs like Access.
- Often, more sophisticated capabilities are needed.

Why are DSS systems needed? (Expansion of RTP p. 275)

- The number of possible alternatives may be too large for a human decision maker to consider unaided (example: allocate production of 200 products may each be made in any of 10 plants, and then shipped to 1000 customers – what is the cheapest production and shipping plan?).
- The time available to make the decision without computer assistance
- One may have to coordinate information from multiple, possibly remote sources
- Sophisticated analysis or highly specialized knowledge may have to be applied (operations research methods, statistics, expert knowledge, etc.)
- (Not explicitly in RTP book) Computing may be a more cost-effective substitute for human labor in at least part of the planning decision process.

Some kinds of specialized analysis a DSS might provide:

- “What if”, simulation, and sensitivity analysis – try to predict what will happen if a certain decision is taken
- Constructing and recommending a plan of action (or a partial plan of action); for example, this might involve using an optimization model such as some of you may have encountered in the Operations Management course (but probably not using a spreadsheet).

To provide such capabilities, DSS systems typically need some kind of modeling of whatever is being decided about, beyond the simple data relationship modeling entailed in entity-relationship diagrams. There are two main kinds of modeling:
• Explicit, often using various mathematical tools. Examples (which may be used alone or together):
  o Statistical models
  o Operations Research/Management Science (OR/MS) mathematical models, including optimization models (simple example: “Solver” models like those in the Operations Management course), probabilistic models, and simulation techniques (simple example: “YASAI” analyses from the Operations Management course).
  o Constraint logic programming (CLP) models: similar in spirit to classic optimization modeling – with a more powerful, “free form” modeling vocabulary (trade-off: less efficient solution tools).
    ▪ Example of CLP constraint: the variables $x_1, x_2, \ldots, x_n$ must all take different values from one another

• Rule systems/expert systems (see below)

• Implicit/automated
  o 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.
  o 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: neural networks (RTP p. 292).
  o The best use for these technologies may be in “data mining” systems that try to find useful patterns in huge datasets. Any patterns discovered may then be further investigated using other techniques.

Example of DSSs that use OR/MS techniques effectively:
• Waste Management, Inc., RTP p. 280.
• NFL referee scheduling
• Continental Airlines crew planning system (CALEB)

Examples of DSSs that have used CLP methods effectively
• NFL game scheduling
• United Airlines gate assignment at O’Hare airport

ES (Expert Systems) – use a technology that tries to mimic certain kinds of human intelligence, in particular the knowledge of experts who may be in short supply.
• The hear of most such systems is a “rule base” or “knowledge base” of rules you would like to follow or facts that you believe to be true about the application. These rule/knowledge bases are far less structured than relational databases. Examples:
  o A defective power supply 20-volt output causes an overheating finisher unit
  o Overheating finisher units cause paper jams
• The rule/knowledge base is processed by an inference engine that tries to combine the elements of the rule base in an intelligent way. A very simple example:
  o Based on the above information, the system could conclude that a defective power supply 20-volt output causes paper jams
• Such systems have been applied to
  o Medical diagnosis (example, healthgrades.com, RTP p. 289)
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/knowledge base, the system’s behavior in unusual situations may be hard to predict

ES and CLP-based DSS systems are based on similar, overlapping kinds of “artificial intelligence” technology (both could be considered “intelligent systems”). The line between them is fuzzy, but the distinguishing aspect is the kind of modeling used: ES systems tend to model an expert’s thought process about something, rather than the topic itself. For example, in assigning flights to gates at an airport,
- A CLP-based DSS would simply model all the constraints and costs involved, and search all possible solutions for the best one
- A “true” ES would try to model the behavior of an expert human gate assigner.

DSS systems often need capabilities not practical to implement with the basic RDBMS tools (like Access’ tables, queries, forms, and reports).

However, to be readily used, DSSs must somehow be integrated into the rest of a firm’s IT infrastructure
They must be able to extract necessary information from TPSs or data warehouses
Results may need to be sent back to other systems/components
Example: a DSS-produced optimal production plan needs to be stored in the operational information systems used by the manufacturing facilities.

Implementation/integration options:
- Subsidiary modules within an overarching RDBMS-based system:
  - “Stock” approach: some simple production planning modules are provided in many ERP systems (but they may have serious limitations)
  - Or, buy add-on modules that function within a standard ERP/RDBMS environment
  - Or, write your own subsidiary modules. Example: MS Access allows you to write your own programs in the VBA (Visual Basic for Applications) language; these programs reside in “Macro” objects. These programs may be attached to forms, reports, etc. to provide additional capabilities that would be hard to implement otherwise.

  - “Arm’s length” approach: conceive of the DSS as a separate system that communicates with your main TPS and data warehouse systems. It extracts data from TPS and data warehouse systems, and may also send data back once decisions are finalized.

Typically, DSS systems do not make final decisions
- They present possible decisions to the human decision maker(s)
- The human decision makers may make alterations (for example, to account for factors not in the DSS’s model)
- The human decision makers may conclude from the DSS output that some of its model inputs are incorrect; they may adjust the inputs and see how the output changes
- However, for very routine or high-speed decisions, the DSS may actually make the final decision. Example: acceptance/rejection of low-credit-limit credit card applications.
For the typical case that a DSS has extensive back-and-forth interaction with human decision makers, it also should have a good user interface.

- A good interface should make it easier to adjust the DSS model inputs or edit its output
- Graphics may be helpful: for example, display proposed garbage pickup routes on a map

**Class 27 – 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
- 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 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.

RTP (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.

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 the AFLAC blurb, RTP...
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 – for example, the case of the FBI “virtual case file” system described on RTP p. 313. 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 RTP p. 312.

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 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 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?
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, 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 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 things 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 substep 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
- 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 regression test to check that bug fixes don’t create more problems or revive old bugs (“rebugging”)
  - Expand QA tests as features are added

Critical to involve 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 all of them. The system analysis phase and involvement of users will still be critical, especially for ambitious projects.

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.

In Section 10.4, RTP 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 analysis and design phases.