IS 240 -- Information Systems Analysis and Design

Spring 2000

Phil Agre

phone: (310) 825-7154
e-mail: pagre@ucla.edu
home: http://dlis.gseis.ucla.edu/pagre/

This class reconstructs the classical computer science topics of "systems analysis and design" -- mapping information flows and data modeling -- within a framework derived from industrial design. Compared with the traditional approach, our focus of attention will shift from systems to services, mainframes to networks, the desktop to the street, organizational workplaces to institutionally organized relationships, cognition to physical activity, and individual users to communities of practice. The class will be organized around presentations by interdisciplinary teams, with minimal lecturing and written work. We will attend closely to the design process, and the teams' own experiences will become raw material for their projects.

Two books are required:


Donald A. Norman, *The Invisible Computer*, MIT Press, 1999. This is a polemic against the personal computer and in favor of a new generation of diverse and specialized computing devices.

Another book is recommended:

Jeffrey L. Whitten and Lonnie D. Bentley, *Systems Analysis and Design Methods*, fourth edition, Irwin McGraw-Hill, 1998. This is a thorough introduction to the conventional practice of systems analysis and design. If you are going to work with people who have the conventional training then it will be useful reference book. But as I say, I regard this material as out-of-date.

Here are summaries of the group projects from week to week:

Assignment for week 2: Team-Building Exercise. Everyone writes down their skill set and gets copies of everyone else's. Class members then form themselves into teams. Each team's members discuss their past and future, and how they complement one another. They draw a diagram that gives clear form to the conclusions they have reached, and they design a presentation around it.

Assignment for week 3: Seeing information happen. Each team gets a distinct assignment, all of
which involve going out in the world and watching information happen. Bring back what you've observed and show us. If you use what you've learned in other classes about information seeking then that's great. But we really want you to be observant and name things, and learn how to show what you've seen in a way that changes how other people see the world.

**Assignment for week 4: Growth of the technology.** Each team again gets a distinct assignment, this time involving library work on the state of information technology ten years from now. Because of Moore's Law and related phenomena, we can predict reasonably well the quantitative properties of computing. Processors, for example, will be 100 times faster. What about mass storage, memory chips, wireline and wireless bandwidths, penetration rates of the technologies both domestically and globally, and so on? What important standards will be widely deployed by then? Show us what you've found.

**Assignment for week 5: Layering.** Building on last week, we will do an exercise about the concept of a platform: a service upon which a diversity of other services can be built. The hard part is figuring out what belongs in the generic service, and what the interface should look like between the platform and the services that are built on it. This is going to be a central concept for design in the future. By this time we will have discussed several examples of platforms.

**Assignment for week 6: Show us your collaboration patterns.** All the while you've been documenting your team's work process. This might mean keeping notes, taking pictures, drawing diagrams, videotaping, saving your work, etc. You have probably also settled into something of a routine. Show us how you work together. Along the way we will offer several ideas about what to look for. For example, where is the borderline between "routine" and "improvised"? This will be important in the coming weeks as we mess with the traditional concepts of systems analysis. This is the first week of a six-week iterative design exercise.

**Assignment for week 7: Ontology of collaborative work.** Data modeling is the only idea from traditional systems analysis that is intellectually hard, so we will spend some extra time in class working an example of it. Then your assignment will be to model the data that will be required to implement one or more of your prospective services. Whereas earlier assignments have called on you to invent your own representation schemes, for this assignment we'll have you use a conventional notation scheme for data models. Having done so, sketch an information service that you might like to design to support collaborative work activities.

**Assignment for week 8: Service design.** Now it is time to spell out the details of a potential information service for collaborative work. To prepare for this, think about your service from several angles. What institutional roles are people playing as they use the service? Where are they located? What actions do they need to take? Who gets access to what information? Who creates the data that goes in the databases? Which parts of the data should be structured, and what should the structure be? What existing services does your service exchange data with? And so on. Having explored all of these questions, draw a dataflow diagram for the service you are designing, decomposing the design into enough component processes and databases to get a clear idea of its
Assignment for week 9: Information design. Using cardboard, crayons, glue, and other materials found in kindergarten classrooms, build a mockup of one or more of your services. Show what your service will look like in practice, and tell us how it is comprehensible. We will have discussed some examples of information design, including several that have nothing to do with computers. We want computers to be more like the diagrams in Edward Tufte's books. We also want them to be more like the information appliances that Norman argues for.

Assignment for week 10: Service architecture. Having sketched first the insides and then the outsides of your service, it will be time to return to the inside, applying serious architectural concepts this time. How are the processes and databases that comprise your service divided among the various networked devices that they run on? You will have been reading Messerschmidt throughout the quarter, and this is where you will apply everything in that book.

Assignment for finals week. We don't imagine that anyone will be around during finals week to see your work, and so instead we will have each group videotape a final presentation that we can put on the Web. This will include your service mockup, its information design and internal architecture, how it works cognitively, how the information flows, and generally how it works as a service in the full sense.
Information Studies 240

Spring 2000

Team-Building Exercise

We will divide the class into five teams of four or five members. Because the design exercises in this class will require a wide range of skills, every team's members should be consciously chosen to have complementary skills. Do not simply jump at the chance to work with your friends. To provide a rough sense of what interdisciplinary teams in the real world are like, we will do a team-building exercise. During the first class session, everyone will be asked to prepare a single page describing their own skill set. Include the full range of skills, in the broadest sense, that might contribute to the design of new information services. You can choose what information you want to disclose.

Information that you might want to share includes the following: programming, experience with PowerPoint and the like, a past job where you balanced a budget or wrote a business plan, knowledge about information organization, database skills (especially including XML), extensive knowledge of existing information resources, graphic design, Web site construction, having taken a traditional systems analysis course, photography, digital image manipulation, field interviewing, any sort of new media art or design, model-making, brainstorming and other advanced meeting skills, project management, giving demos, public speaking, getting projects done on schedule, MIS, industrial design, user interface design, and business analysis.

Because some of the projects will require your team to talk to people and learn about their informational lives, you should also include social worlds that you have access to: your workplace, spouse's or friends' professional worlds, an "other life" such as a hobby or political or cultural activity, a past career where you're still in touch with the people, and so on. You're not absolutely promising that you can deliver those people, just that you have good relations with them and think they might be open to helping with a simple project.

We will immediately make copies of these pages for everyone in the class. Spend the evening studying these pages and thinking about whose skills would complement your own. Then, starting on Wednesday morning, form yourselves into teams. This will no doubt require phone calls, e-mail, meetings in hallways, and so on. Don't start forming your teams until Wednesday.

This will obviously be hard work, and part of our purpose is to smash you into the practicalities of coordinating complex activities using communications media as they exist right now in April 2000. Please observe the team-building process as it is going on. Take notes. Later we will draw on these experiences as raw material for design exercises.

Once you have formed your team, it will be time to start on your presentation for the second class (April 11th). This will be a pure design project -- nothing about technology. Get to know your
team members -- whether in person or in some other way is up to you. Discuss your backgrounds, your current situations, and your plans. Talk about your skills, your goals, and what interests you. Compare and contrast your lives. (Most groups will probably want to stick to professional concerns, but it's up to you where to draw the line.) Then come up with a diagram that neatly summarizes how your lives relate to one another. Organize a seven-minute presentation in which the team introduces itself to everyone else, with the diagram helping to draw the presentation together conceptually.
Assignment for Week 3: Seeing information happen.

In the old days, a computer was a big box with fat wires coming out of it. People used this box by typing at a keyboard that was attached to something called a "terminal". You don't even want to know what that was. Neither the computer or the terminal ever moved. To use this primitive "desktop" setup, you had to sit still in a chair. I am not making this up. Nowadays, the physical form of computing is considerably more diverse and flexible. Powerful computing devices can be carried in a pocket, sewn into clothing, or embedded into just about any artifact. These devices can communicate by wires or wirelessly. Reasonably good continuous speech recognition is commonplace, and display screens are flat and often flexible. All of these devices continuously talk to one another, and they all have access to a vast number of complex information services.

As a result of these innovations, the design space is infinitely larger, and it becomes especially important to understand how a computing device "fits" into the world around it. This fit can be understood on different levels: ergonomic, cognitive, teamwork, institutional, economic, and so on. A device must fit into the rhythms, patterns, and cycles of life. It must coexist with all of the other devices that an individual or community might employ. It must be comprehensible, and it must be meaningful as a cultural symbol. A device that is perfect for stamp collectors might be useless for ecologists. Services for a culture that takes trains may not fit into a culture that drives cars. Devices for people whose social world is largely bounded by their geographic neighborhood might not be useful to people whose professional networks are global in extent. As the space of possible designs for a device becomes larger, therefore, it becomes important to learn more about the activities and relationships within which it will be used.

This week's exercise is intended to give us a fresh look at information in the world. Our goal for this week is not to invent something, only to see what's already there. Each group will be assigned to go out in the world and look at information as a phenomenon of the physically realized social world. You may well have done such exercises before in a class such as information seeking behavior; if so then great. But our goal here is not to apply any single theoretical framework for understanding information-in-the-world. Rather, our goal is to see something new. You can do any of these assignments in a few minutes, and you will probably get "enough" to give a presentation. But it will be a boring presentation, because in a few minutes you can only see the things that you already have names for. Your task, therefore, is to see those things, and then to keep looking. This is sustained looking, looking plus brainstorming, and the goal is to see things that you haven't seen before. If you haven't seen them before then we probably haven't seen them either. Put names on them. Prepare a presentation in which you identify several phenomena, naming each to give us a sense of the general category that you've formulated, and given an example or three of each to give us a concrete sense of what you're talking about. Your presentation can use drawings, pictures, stories, video, play-acting, or whatever changes the way your audience sees the world in seven minutes.
These exercises will require you to talk to people and observe their lives. Ethical rules therefore apply. You are welcome to observe and make pictures of people in public places without their permission, so long as you do not make them feel paranoid. That's what it means to be a public place. If you talk to someone, use common sense. Do not represent yourself as anything except UCLA students doing a class project. Say that you're not going to tell anyone their name. If this were a formal research project then you would need to go through a formal "informed consent" procedure, but this is a class and the potential for harm is almost zero. But if anyone says no or otherwise doesn't want to cooperate, that's their right. Don't interview any children except your own. You will probably have an easier time talking to people you already know, other things being equal, but that's not necessarily the case. Use your judgement.

First group:

Go to a place where people do lots of complicated things. A train station. A courthouse. Westwood Village. Someplace where you will see things that are not going to be obvious to everyone already. Watch. See how information happens in that place. Think broadly about everything information means there. Look at signs. How do people know where to go? What is on their minds? What are the most common activities the people are engaged in, and what kinds of information do those activities require? What do they wish they knew? What kinds of information would, if available, cause them to act differently? Have they come to get information? If they had different kinds of information, would they be there at all? Who interacts with whom, and how and why, and what information is part of this? Do the people have plans, do they check hypotheses, do they make mistakes? Is there a difference between newcomers and oldtimers? How does someone learn to conduct themselves in this place? Those are just a few questions aimed at stirring up your thinking as you watch. You might know the answers just by watching the people, since you can draw on your own experience of doing what they're doing. Or you might have to camp out, or interview people. See what you see. Show us.

Second group:

Talk to three very different people who use many sorts of information in their lives. Show us how the different sorts of information fit into their lives. Rhythms, cycles, patterns. Roles, tasks, relationships. Interruptions, boundaries, improvisation. Not just a list of different kinds of information and their uses, but the interactions between among them. Show the different information uses forming a mosaic, a patchwork, whatever metaphor works for you. You can't show the full complexity, but evoke aspects of it somehow. Draw them out and name them. Compare and contrast. Do some of the same themes emerge for all three people? How are they the same and different? You can send three different team members to talk to the three individuals if you like, but spend some time together, synthesizing what you've found. Otherwise your presentation will just be Part I, Part II, Part III, with nothing connecting them.

Third group:
Look at signs, or "signage" as the architects call it. How are they meant to be used? What do they convey? Can they be interestingly categorized? What makes them good or bad? You'd be welcome to find manuals of signage or talk to architects, but most importantly look at signs. Learn to look at the world as a bunch of signs with buildings and streets etc attached to them. What are their purposes? What do their designers think about the people who use them? What questions do real people have in their minds, that the signs answer or don't answer? These questions are useless as generalizations or abstractions, of course, so work from examples. "Read" the signs the way a literary critic would read a poem: over and over, closely, backwards and sideways, until it gives up a deeper level of meaning. Can signs be biased? Ideological? What representational schemes do they employ, and what representational skills do they presuppose? How does one learn to use them? These are giant questions that couldn't be answered in seven hours. So work at it until you've punched through to some fresh observations. Then put names on a handful of them, and look again, now that you have these new names in your head. Find more examples of what you've named. Then see what you see. Show us.

Fourth group:

Personal effects. What information-conveying stuff do people carry on their bodies? Get people to go through their wallets, purses, backpacks, pockets, etc, and show you what's in there. Look at their personal effects as a kind of design: vernacular design. How do people design the insides of their wallets, purses, etc? Look for reminders, databases, etc. You'll find address books and notepads, but you'll also find information-carrying objects that are not made of paper. Can things carry information because one has them along at all? Because of which pocket they're in? Or what? Interpret "information" in a broad sense. This exercise is harder than it sounds. You will of course find a bunch of stuff, but in what sense does it convey information? After all, all of the stuff will be familiar. It will be *my* stuff. Little or none of it will be surprising. The "information" will be subtle, and it will be tied to the rhythms, patterns, cycles, relationships, roles, and everything else of the person's life. Keep naming the phenomena and gathering new examples until you're seeing things that are striking, alarming, fresh -- even though they've always been right there for anyone to see. Then show us.

Fifth group:

Talk to some people about making plans. Watch them make plans together. Plans for the evening, for a vacation, for a business, for a meeting. What needs to get coordinated? What information do they need? What constraints do they discover and reconcile? What conventions does their culture or industry or discipline provide for the planning? How do people who know one another well make plans together, as opposed to people who are strangers? Notice yourself making plans. Keep talking about these things with people, and putting names on them, until you spontaneously notice more examples of them. Document their plans and planning processes. Do their stories after the fact convey the real complexity? What is a "plan" anyway? It will depend on the context: some kinds of plans are very formal and specified, whereas others are much looser. This, too, could be a vast, encyclopedic project. That's not the idea. Just sustain your inquiry until you have some
fresh things to show us: names and examples. Then show us. Draw diagrams, take pictures, whatever it takes to change how we see the world.
IS 240 -- Systems Analysis and Design

Assignment for Week 4: Growth of the technology.

Traditionally, systems analysis and design has presupposed a narrow design space of mainframes and terminals. Designing a computer system was a routine matter: map the information flows, model the data, and you're almost done. Not so any more. The design space is much larger now, and yet we cannot free ourselves of the stale habits of the past until we get a concrete sense of what that new design space is like. This week we will gather raw material. To take some examples, Moore's Law says, in effect, that computer processors grow 100 times more powerful every ten years. Memory densities grow at a comparable rate, and mass storage densities grow even more quickly. Computer networks are growing rapidly more capacious, even if it's not obvious that the average household will enjoy these rapidly expanding bandwidths anytime soon. Wireless data services have been slow to take off, but they are grabbing hold ferociously in Europe, and we can now hope for general-purpose wireless packet networking in the foreseeable future if not right away.

This is just a rough sketch; your assignment is to paint a picture of the design space in ten years that is more complete and more compelling. This is library work, and Web work, and it will take you into the trade press. It may require you to talk to technical people to check your understanding of things. It wouldn't even hurt to read the ranting of futurists like Ray Kurzweil. The technology forecasting organization Institute for the Future (IFTF) is respectable; their best-known spokesman is Paul Saffo. And it will make you think about how to present such information usefully. A 100 GHz processor sure sounds cool in the abstract, but what does it mean for us? How can we make the numbers imaginable and useful? Now, I understand that this assignment is, strictly speaking, impossible. Nobody knows what the world will be like in ten years. So what you are really reporting is guesses and predictions, and the experts' are not much better than yours. You are limited, obviously, by the strengths and weaknesses of the public literature. If you have access to specialist reports in this area, by all means please share. But you will not necessarily find predictions framed precisely in terms of "ten years", or 2010, or whatever. Work with whatever predictions you can find, and guess.

It is helpful to distinguish between two kinds of predictions. One kind is quantitative. The fact that we can extrapolate from past quantitative improvements in processor speeds and the like is most unusual: prediction is rarely so reliable or mechanical. Find out the historical growth curves, and then calculate accordingly. You will find many discussions in the literature -- even in publications such as the New York Times -- of these growth curves, and speculations about how long they will last, and whether new inventions will enable us to run ahead of them.

The other kind of prediction is the qualitative organization of architectures and standards. These are remarkably easy to predict as well, but for the opposite reason: because they change so slowly. A standard such as Jini (which allows networked devices to advertise their capabilities to one another), Bluetooth (a spontaneous wireless networking standard meant for things like home...
appliances), or TCP/IP (the protocols of the Internet) can languish in laboratories for a decade or more before the necessary critical mass can be established to for them to "take off" and be widely adopted in the real world. We can therefore safely guess that most of the standards that will be widely adopted in 2010 already exist today, at least in nascent forms in the laboratory, and that few entirely new concepts will enter the pipeline in time to change the world by then.

You have two jobs: present some reasonably solid information, and build a bridge between information and imagination. Design starts with imagination, and good designs will only get built if they can first be imagined. What is Bluetooth, really? What can you do with it? What could you do in a world that was saturated with Bluetooth-compatible devices and services? Okay, so databases will be 1,000,000 times bigger (or whatever the best estimate is). What does that mean? What will fit in that kind of database? Ask around for references that might address the issue. Actually, you have three jobs: the third job is to provide the class with a bibliography, Web links, handouts, and other resources for those who want the details.

Remember that you are not supposed to kill yourself doing this assignment; we only want you to do as much as you can do in the time available. Why are we asking you to hunt down this material rather than us doing it ourselves? Well, we've already done a lot of it ourselves, though surely not as much as you will. Our real reason is that we want you to confront the literature in this area for yourself. A designer in the real world must constantly predict and track various developments that might soon intersect with his or her own projects. If you commit to using a certain standard in your product, for example, you acquire a strong interest in whether that standard will survive at all, how quickly it will be adopted, what directions it will take, and so on. You may be motivated to attend standards meetings, express opinions on heavily-trafficked Web sites, and contribute code to open source projects. The point, then, is that designers are embedded in a social world as much as a technical world. By predicting the future you are joining into the group project of building it. The forms of imagination that you develop will be part of a collective imagination, not least because you have an interest in helping other people to imagine the future in the same way that you do.

Your presentation can take many forms. This material is usually presented in a sober way with PowerPoint slides with statistics on them. If that's your way of bridging information to imagination, go for it. Otherwise, rethink. What picture is emerging from your research? Define the organizing principle of that picture and let it organize your presentation.

Remember to document your work activities while doing this project, and to reflect on the process of research and design. Your experiences will become important raw material for later in the course. Observe your collaboration in all its aspects. Experiment. Watch yourself use different media, various information resources, and so on. What are the real practical problems? How does your group process fit (or fail to fit) with the rest of your life? What sorts of tools would make the whole thing easier? Your goal is to make life better for all of the interdisciplinary teams in the world.
First group: An, Hoffman, Lauruhn, Miller, Sorrell.

Raw computing power. The quantitative predictions here are easy enough because everyone seems to believe in Moore's Law. But so what? What could we do with 100 GHz microprocessors anyway? If steering wheels cost 1% as much as they do now, it would make no difference to the cost and functionality of cars. Perhaps people would burn steering wheels for fuel, but otherwise the world would be much the same. You will encounter a lot of loose talk about how computers will become as intelligent as human beings, or 1% as intelligent, or whatever. This is misleading at best, given that computers work completely differently from people. So find out, what applications of computers, currently in the pipeline, would actually benefit from a 100-fold increase in computing power? Good places to look include speech recognition, simulation, and financial derivatives. But even then, just saying "speech recognition" is only of limited help. Again, so what? What difference does it make to the information services we can deliver?

Second group: Childers, Lippincott, Napper, Tsay, Vasquez.

Databases are obviously crucial for designing information services. How big will they be in ten years? What kinds of information will be in databases that aren't commonly in databases now? Will databases of video be useful in ten years, or is that more like twenty years? How much of a database will someone be able to carry? How many databases will there be in the world, and will we be able to interact with large numbers of them at the same time? Take the conventional scenarios about the whole Library of Congress being available in one big database: are those scenarios true, or is it more complicated than that? What kinds of metadata will be widely standardized, adopted, useful, and consequential in ten years? What will that let us do that we can't do now? One could obviously write a long book about these questions, but we want the concise, riveting, worldview-changing seven-minute version.

Third group: Cleary, Gazan, Hessel, Hunt-Coffey, Plutchok.

In Europe, the wireless revolution is now. Tell us about WAP, the Wireless Applications Protocol. What is it, really, and what are some of the services that are already happening on it? A few sentences will convey the general idea, but it's important to say them because the United States is behind in this area. What can we do with WAP in ten years? Or are there other standards for delivering wireless services that will grow to maturity in that time? Will wireless digital networking really be universal by then, at least in the advanced economies? What kind of bandwidth will we routinely get over wireless connections by 2010? Will we have video cell phones? (Never mind for now whether we would want them.) What can we do with wireless network connections to cars? That sort of thing. In this area you'll find that the ratio of cheap predictions to reliable analysis is quite high. See if you can separate the serious stuff from the hype.

What will the Internet be like in 2010? Will the whole telephone system be running over the Internet? Will millions of people be sending video signals at the same time? Will the Internet fail to scale up and collapse into a chaotic mess? How many devices will be attached to it? Think particularly about the tension between the quantitative aspects of the network (bandwidth, for example), which change quickly, and the qualitative aspects (applications standards, for example), which do not change quickly. The Internet has been growing exponentially in number of hosts, but the Web works pretty much the same way that it worked in 1995, or indeed in 1990, and e-mail works pretty much the way it worked in 1990 as well, unless you count the spread of MIME, though it works much better than it did in 1980. Keep in mind that our goal is not to dive into the technical details, gnarly as they are. Rather, we want to characterize the design space. What does a systems designer need to know to imagine what useful information services will be practical in 2010? That's the focus.

Fifth group: The mystery group, aka everyone else.

Investigate technologies that are related to physical space, especially GIS (geographic information systems) and GPS (global positioning satellite). Many potential information services benefit from knowing where people are, or from tracking vehicles or goods. Many other services deliver information that is indexed by physical space: weather, environmental data, travel-related information, services related to security and safety, and so on. Will we be able to build a GPS tracking device into a cell phone? How wide a variety of GIS resources will be available? All of this stuff exists now. The question is whether it is cheap, practical, widely used, and generally ready for prime time.
IS 240 -- Information Systems Analysis and Design

Assignment for week 5: Layering

Modularity is a fundamental concept of system design. The most important modules are so-called "platforms": hardware or software standards that a wide range of other services can be built on top of. (The term "platform" is often restricted to hardware, such as the Palm Pilot, but we are using the term in a broad sense.) Examples of platforms include the Internet, the IBM PC, the cellular telephone (actually several different cellular telephone architectures), Java (and its execution environment), and XML (and its associated software tools). By providing a clearly defined functionality that the world already wants for other reasons, platforms make it much easier to develop new information services.

Perhaps the most important kind of platform is a network service "layer". Messerschmitt explains the concept of a layer at some length, but here is a simple version. Networked services are complex, and so to manage that complexity they are organized in layers. Each layer provides a distinct and well-defined service. In particular, each layer defines a clear contract that governs its interactions with the service layers upon which it is built, and the service layers that are built upon it in turn. An example of a service layer is IP, the Internet Protocol, which is a protocol for moving packets of information ("datagrams") from point A to point B across a range of interconnected networks. IP is a "middleware" layer, meaning that it presupposes the existence of lower layers (the individual subnetworks, which move bits from one machine to another within a network) and is itself presupposed by higher layers (such as TCP, which controls the flow of packets through IP). Another example of a service layer is SSL, the Secure Sockets Layer for the secure transmission of sensitive information between clients and servers on the Web. SSL is built on top of TCP and IP, and other, more complex service can be built on top of SSL in turn. All modern networked services are organized in layers.

The design of service layers is not only technically complicated; it is also socially complicated. The whole point of a service layer is support a wide range of applications, and so the designer must learn precisely what services the potential applications need and then reconcile all of these potentially conflicting needs in the design. This can be hard. The applications may not have been imagined yet, the applications designers may work in different disciplines and may therefore not speak the same language, and the needs of the various applications may change over time. A good service layer is parsimonious: it is cleanly and simply defined, and represents a powerful and easily understood abstraction. An overly complex service layer invites trouble, and should be broken into more natural parts or rethought from scratch. Yet the boundaries between service layers are not always easy to define. The best way to design the boundary between TCP and IP, for example, was not at all obvious at first. IP, for example, is a so-called "best-effort" protocol -- packets submitted to a host or router that implements IP are not guaranteed to arrive in the correct order, or indeed at all. This sounds strange, but in retrospect it is a natural consequence of the demand for parsimony: by making fewer guarantees, IP can remain simple. TCP, which is built on top of IP, does offer guarantees, and part of its task is to resend packets that are not
acknowledged.

In this assignment, we will explore the process of designing a service layer that can provide a platform for a wide range of other services. This is usually an advanced topic, but I want us to begin with it because of the central importance of networked applications architecture to the future of computing. Much of what we do in this assignment will make more sense after we discuss data modeling and other traditional technical subjects. But we will proceed anyway.

We will proceed as follows: I will sketch the workings of a brand new service layer that we will call the "Suitcase Layer" (abbreviated SL). A suitcase, roughly speaking, lets someone take their data with them wherever they go. (For those who know about such things, SL will be presumably built on top of a distributed object management layer such as CORBA or COM, assuming that the world has finally gotten around to adopting an object management standard by 2010.) By creating the illusion that one's data is always present, just as one's purse is always on one's shoulder, it makes computing (somewhat) location-independent and device-independent. Applications and devices that are SL-compliant allow a person to carry an elaborate computing environment from place to place. Wherever the person might be, their computing environment will spontaneously reconstruct itself using whatever devices happen to be present. Applications developers will rejoice because they can presuppose that data, application, and device have been brought together.

Let me start with an example. A major problem in restaurants, particularly someplace like California, is that diners have dietary requirements that are too complex to negotiate with the waiter. "Does that soup have any dairy products in it?" Only at the finest restaurants will waiters know things like this. So let us employ SL to create a menu that automatically configures itself to suit the requirements of a particular diner. We will have to start by defining data structures to represent these requirements. To keep it simple, we will imagine that an authority has defined a vocabulary of potentially sensitive ingredients, and then we will provide people with an interface that lets them check off the ingredients that they will not eat. These might include whole categories such as meat and dairy products that some people avoid for health reasons, or ingredients that some people are allergic to (eggs, cilantro), or ingredients that are proscribed by some people's religions (beef, pork, shellfish). (Observe that the categories are naturally hierarchical: meat includes beef, which includes veal.) It doesn't matter where this information is stored; the individual's suitcase will make the information available whenever it is needed.

To use a suitcase, the would-be diner carries a device that is the size of a double-thick credit card; this device could be carried in a wallet or incorporated into some other device. (Swatch has recently announced a similar device that is incorporated into a watch.) Even though it has no switches, data ports, display screens, or moving parts, it is constantly active. It talks to wireless networks, and especially to the spontaneous wireless networking capabilities of any and every device that supports the Suitcase protocol. One of these SL-compliant devices is made from electronic paper: a computer display screen that looks and feels like parchment but includes a bunch of digital electronics and can change its "pixels" from black to white and back. Participating restaurants "print" their menus on these devices by entering the ingredients and
descriptions of the dishes into another device that is capable of exchanging data wirelessly with the menus. The electronic menus look and feel exactly like traditional paper menus. People who do not carry suitcases get a menu whose display is perfectly traditional -- the one-size-fits-all menu that most restaurants use now. When the menu detects the presence of a suitcase, however, it downloads the diner's list of proscribed ingredients and displays a customized menu. Dishes that cannot be made without the proscribed ingredients are not displayed; dishes that do not contain the proscribed ingredients are displayed as usual; and dishes that usually contain the proscribed ingredients but can be prepared without them are displayed in a modified form. Waiters carry SL-enabled order pads, just to make sure that each diner's requirements are transmitted correctly to the kitchen.

Now, in reading this description you will no doubt observe that nobody would be foolish to build a system like this on a special-purpose, stand-alone basis, what Messerschmitt calls a "stovepipe". But that's not what we're proposing. Imagine a world in which the Suitcase Layer is already well-established in the market, incorporated in dozens of applications, and supported by dozens of devices. Many other platforms are also widespread in this world, including the electronic paper technology. In that world, the incremental cost of building the dynamic menu application is not great. An entrepreneur could implement a prototype in an afternoon from readily available components, and the prototype could be used to raise funds and sell the concept to restauranteurs. It is not at all clear that the restauranteurs would buy the concept, of course. They would have to contend with more special orders in their kitchens, and they would have to represent their dishes in a structured data format such an XML document type. They might regard this as a hassle and a risk to their trade secrets, or they might regard it as a competitive necessity. We would have to see.

In any case, our job in this assignment is to work out some of the details of SL. Each team should, independently of the others, choose a community of practice, sketch out some SL-enabled services that the people in that community might find useful, pitch the concepts to some members of the community, iterate their designs a few times, pick the most promising one, and work out some of the details. Having done so, prepare a seven-minute presentation that tells about the community, explains the service concept, tells us something you've learned about how the service would fit into the life of the community, and (most importantly) specifies in as much detail as you can the demands that your service will make on the design of SL.

Let me explain these concepts in a little more detail. First, communities of practice. Systems analysis and design have historically been focused on geographically localized workplaces; their "users" have either been individuals or small numbers of workers operating within a rigid division of labor. That is the industrial automation paradigm, and it is an anachronism. Another approach is to design for communities of practice. A community of practice might be defined as those people who share some important thing in common. All of the world's truck drivers form a community of practice. So do all of the world's moms, all of the world's stamp collectors, and everyone in the world who uses the Apple Macintosh. The important thing about communities of practice is that they think together. They may not have an elaborate sense of collective identity --
an organization, a flag, a concept of "us" and "them" -- but they will always have institutions (formal or informal) through which they compare notes, spread news, and accumulate a collective memory. That is what Internet discussion groups are for; it is also what professional associations are for, and quilting bees, and after-work beer-drinking sessions. To design for a community of practice means to be aware of, and try to support and amplify, the community's ways: its work practices, forms of association, genres of communication, and so on. One no longer designs for "stamp collectors" as individuals; rather, one designs for "the stamp collectors' community". The distinction is subtle, and in some cases we admit that it has no important consequences. But it is often terribly important, especially when our design processes implicitly set out to prevent, substitute for, or suppress the mechanisms by which a community of practice thinks together.

You'll start by choosing a community of practice to work with. Your best choice is a community that a member of your team is located on the edge of. If you are a stamp collector, in other words, don't try to design for the community of stamp collectors. You are too close to them. But if your spouse is a stamp collector, then that's ideal. You will have access to the community, but you will have the intellectual distance that you will need to be analytical about it. Alternatively, you can choose a community that a team member belongs to, but then make a big point of distinguishing between the community member's perspective and everyone else's. In any case, you should probably choose a community of practice whose members move around, and who generally might find suitcases useful.

We want you to design iteratively. You won't have time to really learn about the community, or really design in a participatory way, or to iterate more than few times, but at least you'll get a rough sense. Start with basic knowledge about the community. Then come up with a few concepts. These concepts should be very simple, back-of-the-envelope one-liners. Pitch the concepts to a few members of the community and listen really hard to what they say. Would they find it useful? Most likely you'll discover that you were clueless about the community, or that you lacked important knowledge about its workings. That's good, because now you'll be shorn of some preconceptions. Rework your concepts or invent new ones. Bring them back to your community members, and sketch them in a little more detail this time. You might scribble a picture or a diagram. Listen again. Back and forth: designing, pitching, and listening. That is the iterative cycle. Go through it a few times -- not enough to really implement your prospective service, just enough to get a solid sense of it.

At some point you will pick the most promising of your candidates, and you will work it out some more. In particular, you will ask what demands it will make of the Suitcase Layer. What kinds of information about the person will have to be carried around? What kinds of devices need to be able to support the protocols, and which kinds of applications? Do multiple people's suitcases need to be able to communicate with one another? Does the suitcase need to support access to huge databases and gigantic information services? Do you need audio? Video? Large, complex documents in different formats? What kinds of information will need to have structure imposed on it (like the menus and ingredients in the example above)? We recognize that you will not be able to answer all of these questions precisely at this point in the course. That's okay. Do
what you can.

Documentation and diagrams will be important. In seven minutes you will be communicating a lot of information, and so you should work hard to define your design concept in a concise and compelling way. Draw pictures of what it might look like. Build a model out of cardboard. Install props at the front of the room to show how it might work. The hardest part to explain might be the demands you'll be making on the design of SL. We can help with this.

We want the five groups to work independently. Once we come together and hear all of the group presentations, we will listen hard to the requirements that everyone is imposing on SL. These requirements will no doubt be hard to reconcile. They may conflict; they may pull the layer's design in five incompatible directions; they may be expressed in different vocabularies that are hard to translate into a single language. Some teams will probably mention issues that other teams will not. Some teams will probably go into more detail than others, or interpret what we mean by "requirements" in different ways. All of this is normal, and our point is exactly to teach what it is like to confront the incommensurable demands that different applications can place on an incipient platform.

Remember that we are designing ten years in the future. Please go ahead and presuppose the technologies of ten years from now: processors that are 100 times faster than we have now; platforms such as spontaneous wireless networking that are widely implemented; giant databases; small batteries; and so on. This will be clearer after the presentations for week 4.

Remember, too, the long-term agenda for the class projects: we are using design to investigate the reconfiguration of interactions among people in a heavily wired world. Some activities that are now conducted face-to-face will happen through electronic mediation, once the capabilities of that electronic mediation improve, and (less intuitively) some activities that are now conducted through electronic mediation will happen face-to-face, now that people are liberated to move wherever they want. If they don't have to sit at a keyboard in front of a clunky terminal any more, they will probably want to go someplace more interesting, such as getting together with the people they want to interact with face-to-face. Think about these phenomena in the community you are studying. What if the people had lots of portable computing and high-quality interaction media? What if all their information was in digital form, and what if they had suitcases to haul it all around with them? Would they all disperse across the landscape, or would they all concentrate in one place? How are their activities tied to particular places, and what sorts of places would they (re)invent for themselves if their activities were less tethered to anachronistic computers?
IS 240 -- Information Systems Analysis and Design

Assignment for week 6: Show us your collaboration patterns

For week 6 we are making the transition into the term project. From week 6 forward, each project will build on the others in a relatively orderly way, culminating in a final presentation during finals week. Feel free to think ahead about your approach to all of the assignments. For example, for week 7 we'll sketch some services in terms of the entities they will represent and then do some data modeling, drawing on the reading from Simsion. (This is a change from the syllabus.) Then we'll work on issues of information flow and databases, based on the reading from Gane and Sarson, as well as on Messerschmidt. Then information design, based on readings soon to come from Charlotte. Then internal architecture, based on Messerschmidt. If you want to read ahead and think about the project as a whole then that would be ideal.

I have already told you about the problem that we will be investigating in this term project, but let me try to explain it another way. Contrary to the slogans of some early enthusiasts, it seems clear that new information services do not make bodies obsolete. We will not immerse ourselves in preservative as we surrender our minds to something called "cyberspace"; nor will most of us move to Bermuda and conduct all of our activities and relationships "virtually". Instead, we will continue to live in the same physical and geographical world as everyone else. We will still engage in face-to-face interactions, and we will still move from one place to another. But much will certainly change, and our job is to conceptualize those changes now that the illusions of cyberspace have gone.

Think of it this way: we endlessly manage a huge number of relationships with other people, and we endlessly move among a huge number of activities. Some of the relationships can be conducted without ever seeing the other person, interacting through media or intermediaries, whereas others involve moving back and forth between activities that require copresence and activities that do not. Every activity binds us to certain artifacts, be they a computer or a book or a car or a machine, as well as to certain people. As new information services come along, the bindings get reshuffled. Activities that once bound us to a desk, and thus to an office and a building, now bind us to a portable computer, and thus to a suitable chair and perhaps to an occasional electric socket and a compatible wireless beacon. Likewise, activities with other people that had once required copresence now require only simultaneous (or even asynchronous) access to cell phones or the Internet. But it does not follow that we spend less time being copresent with people. Because we have all of these relationships to manage, we may want or need to spend more time being copresent with some people, and we might take advantage of our unbinding from desktops and offices to achieve this. What is more, new possibilities arise for activities to move to different places, such as airplanes, where they had not formerly been possible, and where perhaps no useful activities had taken place in the past. On the other hand, activities now threaten to spill over the boundaries among places, so that the boundaries between home and work begin to blur. Using design we can hope to amplify the opportunities that this new reality presents, and
to alleviate the dangers.

Your final project will take the form of a design study that works through in a thoughtful way one single theme that you have articulated within the broad field that I have just sketched. Your task is not to support every aspect of modern life by designing a does-it-all utility belt. Rather, you should look and think about work in teams -- teams such as yours -- and then pick a theme where a new information service might help. We don't know yet what form these newly designed services might take, and if we have any guesses at this early stage then we should expect them to be superseded by deeper observation and the iterative design process. For the moment, our priority is observation. I'm providing the longer-run perspective to increase the chances that you'll notice relevant things to observe, but the important thing is to document your work activities and ask yourself a bunch of interesting questions about them.

The assignment for this week is simple enough: show us your work processes as a team in this course. The usual criteria obtain: being observant, putting names on things, performing sustained analysis, consciously designing your presentation so that it has a clear form, using diagrams to make that form easy to grasp, and documenting your work. The goal is not yet to represent your work processes from the texts; rather, we want you to observe with fresh eyes the teamwork issues that millions of people are facing every day. Think about what it means, and what it *could* mean, to "show" your work processes. Iterate the design of your presentation so that you learn from the structures that show themselves in your first drafts -- the structures that you discover in a draft design are more important than the structures that you have in your head, and design proceeds by drawing those structures out and making them into organizing principles for the design. If they are good structures then they will provide clear guidance for every aspect of the design on every level, and if they are bad structures then the whole thing will fall apart and you'll go back to observation and first principles.

Here are some things you might look at. Although our task right now is just to observe and show us what you see, it might also be worth looking forward to the design problems that observations such as these might present...

* Learning from one another. A good way to learn is by working together with someone whose skills differ from yours. You will naturally take up different roles in the activity, but you will have a close-quarters look at their skills and ways of thinking. How could we amplify this learning? The answer might include both technology and skills. Is there a risk that technology-mediated collaborations will suppress the spontaneous apprenticeship that happens in face-to-face collaborative work?

* Coordination. You're working "together" but not always copresently. How do you stay coordinated? One obvious aspect of coordination is scheduling meetings, which is always a drag. But there are lots of more subtle aspects of coordination as well. For example, you might have decided upon a division of labor, but then questions arise, or unexpected interactions among the subtasks. We can easily imagine tools that impose a lot of structure on your joint activities, but
the risk of oppressive overstructuring is severe, and in fact groupware tools routinely fail because they shackle people together too tightly. What is the middle ground like between totally free-form cooperation and fascistically structured cooperation? A few good examples can go far.

* Iterating design. How does your group iterate designs? One picture is the (lone) architect in Schon's chapter, but you're a group and you probably have your own ways of doing things. How do you notice issues and structures and problems and so on? Do you push as hard to iterate your designs -- including design of both technologies and presentations -- as you wish you did? What would help you do more and better? The answer is not necessary a technology.

* Multimedia production. Producing multimedia presentations requires special skills, but it also includes many inputs and joint decisions. Do you iterate once you've committed your work to media, or do the tools tend to set your work in stone? How do divisions of labor work here? Does everyone have to be there together, or does one person do it alone? Does the person who creates the multimedia content end up making a lot of design decisions that had not come up before? What would it be like for the team members to "think out loud" in multimedia? What practicalities come up with digital images, for example, that get in the way of your work, or that otherwise shape it?

* Planning how to work. Do you set a schedule? Do you keep to it? Do you follow conventional meeting structures such as the divergent-then-convergent cycle of brainstorming meetings? What kinds of work are hard to control in terms of time requirements? How do decisions get made? How do options get selected among?

* Problems. What recurring hassles have you encountered? This is obviously a good question to ask if we're thinking about designing services that might help overcome those hassles. Some of the recurring hassles will be obvious and familiar enough, but try to throw fresh light on the hassles by insightfully naming new aspects of them. What changes did you make to try to resolve the problems?

* Structure. Some kinds of group work need to be unprogrammed, unstructured, unscripted, undirected; but then you have to get down to work and deliver something. What about the relationships between these two kinds of work? Or are there more than two polar kinds of un/structuredness? Have you settled into routines and habits for dealing with this? Or is your team stuck in the middle ground, unable to become truly unstructured or truly structured?

* Rehearsal. Many teams will have a formal rehearsal before a presentation, and that's good, but when a presentation is looming many aspects of the work include an element of rehearsal. For example, one might try out language or images that might work in a presentation, or try to reason out what the professor wants, or work backwards from the limits of time or equipment on the presentation. How do these many aspects of rehearsal affect your work practices? How could tools help with this?
* Feedback. What do you do with feedback, both written and verbal, on your presentations? Do the various items of feedback come back in your discussions on the next project, and if so how? The idea is that you are supposed to internalize the feedback so that you take it into account almost automatically in the future, and this is supposed to drive you to ever more sophisticated and conscious designing. For example, you might create a draft design and then realize that it runs afoul of a kind of feedback that you got earlier, or even some feedback that someone else got, and this might drive you to iterate that draft.

* Changes in work patterns. Did your team settle into routines for working? Did this routines evolve from week to week? Oftentimes small, accidental features of an early team meeting can settle into a pattern that lasts for months, even though you might not have chosen that pattern. Did you make any conscious decisions to change how you work together?

* Interaction of different factors. Patterns of shared work are shaped by factors on many levels. These might include work and family obligations, personality stuff, the distribution of skills, personal and group values, ideas and assumptions, the practicalities and limitations of technology, the nature of the subject matter, and so on. Design means reconciling multiple constraints, and this includes the design of work activities. The constraints can come from many directions, and it takes a lot of mental work to get all of the disparate constraints into one place so you can consciously work with them.

* Boundary objects. People from different disciplinary backgrounds are often helped to communicate by a shared object: a rock, a diagram, a collection of stories and observations, a pile of computer code, etc. What boundary objects did you have, and how did they help you (or fail to help you) make forward progress?

In a project like this, designing tools to support work activities that we observe closely, many dangers loom. Perhaps the greatest of these dangers is that we will reinvent the wheel, or that we will remain locked into the existing arrangements. Design must always negotiate a tension between observation and imagination. Unsophisticated observation can foreclose imagination by impressing upon us just how interconnected reality is. It is true that technological innovations go unused because their designers are oblivious to the many interconnections of the real situations in which technologies get used. But it is also true that innovations often do get used, either through blind luck or because the designer was able to see the fault line along which one densely interconnected reality was able to shift to another. The most basic intuition is that people adopt technologies because they want them, they want them because they can imagine what to do with them, and they can imagine what to do with them because their uses are aligned in both intellectual and practical terms with the wants and needs that people already have. This may seem obvious enough, but you would be surprised how many innovations are predicated on people discovering entirely new and unsuspected needs. This is why we study existing activities: the successful new tools will be the ones that let people do more of what they already want to do, and it's through observation that we discover what people already want to do, given the layered complexity of life as it already exists.
IS 240 -- Information Systems Analysis and Design

Assignment for week 7: Ontology of collaborative work

The most important task of a systems analyst is to decide which entities the system is going to represent. This is a strange task, one that falls across the boundary between engineering and metaphysics. The basic question is this: what does this particular system think that the world is made of? For the past quarter century, the task of answering this question has been called "data modeling". Although methodologies for data modeling have proliferated, their basic method is to perform a rational reconstruction of the concepts of the people -- the customers, that is, and not necessarily the "end users" -- who want the system built. The customers' nouns become data entities and their verbs become processes. I make data modeling sound easy, but of course it's not. It has the advantage of encouraging the analyst to listen and learn about the world where the system will be used, but it has the disadvantage that the customers' concepts, for all their authenticity, might not add up to a sufficiently solid metaphysics. In this assignment, therefore, we will skip the listening part and focus on our own concepts.

Your task this week is to prepare a data model for an information service that might support interdisciplinary team-based work. You'll work back and forth between guessing what entities the service should represent and guessing what it will do. Come up with a few scenarios, and present a data model that would provide the groundwork for them. I do not assume that the meaning of this assignment is obvious, and so I will explain it at some length.

Let us begin with an example. Suppose that one were to design a service for reserving meeting rooms. The meeting room calendar would be visible to everyone who works in a given office, presumably by pulling up a Web page, and anyone would be able to claim a block of time. What entities would such a service need to represent? To find out, one simply pulls out the words from the informal specification I've given. I'll start by repeating the relevant hunk of text:

... for reserving meeting rooms. The meeting room calendar would be visible to everyone who works in a given office, presumably by pulling up a Web page, and anyone would be able to claim a block of time.

Here are the nouns:

- meeting room
- calendar
- someone/anyone
- office
- block of time
- Web page

here is another word that looks meaningful:
visible

here are the verbs:

- pull up (the Web page)
- claim (block of time)
- works (in the office)

We have most of what we need, but we need to clean it up a little:

* The entity "meeting room" seems straightforward. In practice there will be gray areas: how about classrooms, lunch rooms, and so on? Perhaps those decisions will be left to the people who configure the service for a particular office; they will decide what precisely the phrase "meeting room" will mean for them.

* How about the entity "calendar"? We only need to represent it as a distinct entity if there is more than one calendar. One calendar can include many meeting rooms, so do we have any reason to support multiple calendars? I can imagine a reason, for example if the same system is going to support many offices. We should ask the customer.

* The words "someone" and "anyone" are obviously meant as informal glosses. In logical terms they are quantifiers, and they leave implicit what they are quantifying over. The phrase "everyone who works in a given office" sounds like a relationship, so we can say that an entity of "person" can be related by "works in" to an office. Once we start filling in the entity-relationship diagram, we will be forced to ask some useful questions: can the same person work in more than one office? We might also wonder about the connection between this entity of "person" and an entity such as "employee" that another data model might have. Questions will soon arise. Can a visitor reserve a meeting room? How about a temporary employee? An intern? The big boss who works in headquarters?

* The entity "office" is clear enough. We'll have to decide whether every office has exactly one calendar and vice versa; if so then that constraint is easy enough to express in an entity-relationship diagram. Or perhaps we don't want the extra generality of multiple offices and multiple calendars, in which case neither "office" nor "calendar" will be entities. Or perhaps we have some good reason to loosen the mapping between offices and calendars.

* The phrase "block of time" could be an entity. It sounds like it has some internal structure, like a date, a starting hour, and an ending hour. That's fine. In an entity-relationship diagram, this internal structure would be coded as more entities and relationships. So the "block of time" entity could have both a "starts at" relation and an "ends at" relation to the "time" entity, as well as a "occurs on" relation to the "date" entity. This kind of thinking will be alien to those who are accustomed to object-oriented design, but it adds up to the same thing in the end.
* How about "Web page"? Is that an entity? Perhaps not -- it might be the way that calendars are implemented. We have to keep our service layers straight, and "Web page" might be an entity on a different service layer -- the Web, to be precise -- than the one we are designing.

* How about "visible"? At first it sounds like a relationship between the "calendar" entity and the "person" entity. But in an office with 100 employees, that would require us to record 100 relationships between people and calendars. If we really are saying that every calendar is visible to everyone who works in a given office, then maybe it's enough to relate a calendar to an office (and vice versa), and then relate an office to a bunch of people (and vice versa). The "visible" relationship might be redundant. If someone wants to query whether calendar X is visible to person Y, that query could simply be expanded into a two-part conjunction: is there an office O such that X works in O and the calendar for booking meeting rooms in O is Y?

* Let's talk about the verbs. The verbs are probably not part of the data model. They represent processes that we can implement once the data model is defined and the necessary databases are implemented. But it is useful to look at the verbs and make sure that they stand in a coherent relation to the data model. We can ask "who can do it?" and "what can they do it to?". A person can pull up a calendar if they stand in a certain relationship to it, and so on. All looks well. Of course, we may discover extra complexities and ambiguities later on, but hopefully by clarifying our concepts now we have minimized the potential for major surprises.

That's a simple example. Here are some other examples that would be a lot more complicated:

* A system for recording design rationales. As the team talks its way through the design process, someone takes notes on every proposal offered, every issue raised, every constraint discerned, every trade-off defined, every choice made, every reason given, and so on. That way, once the design is completed and the finished system is implemented, the original designers' wisdom doesn't disappear when they do. A later generation of designers who need to modify the system can go back to the design rationale and learn why the choices were made. That way, they can avoid making mistakes that the original designers had thoughtfully avoided. In fact, because the rationale explicitly represents every component of the designed system, it should be possible to hyperlink from a diagram of the system to a complete rationale for the design of each part. The problem with design rationale capture is that the design rationales are wildly complicated. The necessary data model is huge and baroquely metaphysical. This is a case where it's important to be judicious about structure, only imposing structure on those elements of the rationale that someone truly needs to be able to search on, compute with, or otherwise turn into standardized data. After all, if nobody really needs the rationale to become an array of database entries then you can much more easily just roll a videotape during the design meetings, or else hold structured interviews with the designers and capture those on tape. The resulting unstructured data, with perhaps some metadata on top of it, will be much easier to create than structured data, and will present many fewer controversies.

* A system for recording designs, including the partial designs from halfway through the design
process. Here you would need a vocabulary in which all of the designs could be described. You might want entities such as "component", "input", "output", "process", and even "entity" -- after all, you would want to represent the data model for a design. Such a system would be terribly useful: you could have a design editor with a point-and-click interface. The interface could automatically check your design for obvious errors, or it could prompt you to fill in missing information. It could also display the design in many different diagram conventions, because it would have some comprehension of the internal structure of the design.

* A system for recording skill sets. Imagine an organization with 10,000 employees, all of whom work in an endless shuffle of interdisciplinary teams. A program manager might enter the skill set that a team would need taken all together, and the system could propose different groups of people who would cover all those skills. You would probably want to represent availability, geographic constraints, and things like that. Perhaps the skills can be represented with a nice big hierarchical vocabulary, or perhaps skills are more complicated. The question, however, is not whether "skills" as such are complicated in general, as a matter of objective truth. Rather, the question is how the customer talks about skills, and how the customer might want to talk about skills in the future. Get the talking started, and then parse it.

Entity-relationship data models are not terribly complicated. They force clear thinking about some very basic questions, and as such they are a huge advance over the haphazard design methods of yore. But they are limited in many ways. Perhaps most importantly, they are not hierarchical. That is, they do not provide a natural way to represent relationships between entities that generalize or specialize one another. Consider, for example, a data model with entities such as "person" and "employee". How to notate the obvious relationship between these two entities? One could define a relationship like "is-a", as in "an employee is a person". But that would be wildly ambiguous. Why not say "Joe is an employee"? Because those are two different meanings of "is-a", only the former of which belongs in an entity-relationship diagram. (Joe is not an entity; he is an entity instance. In other words, Joe is not a concept but a particular example of a concept. He would have his own row in a database.) Also, the entity-relationship scheme provides no natural way to represent what mathematicians call the "transitivity" of "is-a": if a vice-president is an employee and an employee is a person, it follows that a vice-president is a person. AI people who work in the area of "knowledge representation" have invested tremendous effort sorting out (what they call) the "epistemological status" of all of the various links one might wish to draw in an entity-relationship diagram: links representing causal connections, logical connections, part-whole relationships, and so on. Those who are interested in such things can see the citations in my chapter of "Technology and Privacy".

We need not get into those things for the assignment, but I do think it is useful to work through an example of (what the object-oriented framework calls) a class hierarchy. Having worked through your data model, therefore, you might give some thought to the natural abstractions that your model presents. To continue the previous example, if your data model includes entities called "person" and "employee", then obviously "employee" is a subclass of "person" (although police dogs are supposedly considered police department employees). Or to take a more abstract
example, "decision" might be a subclass of "action", which is a subclass of "event". This is where the metaphysics comes in. One might try to enumerate every subclass of "action" which occurs in the course of team-based work -- it sounds crazy, and it's definitely impossible, but it's a useful exercise anyway. Don't try to represent these class/subclass relationships in the entity-relationship model (it can be done, but you don't want to know how). Just start from your entities, call them each a class, fill in the missing subclasses and superclasses to complete the class hierarchy, and draw a simple diagram in which the classes are arrayed in a hierarchical tree-structure, or an outline.

As a reminder, don't try to do everything. Your data model should not be an encyclopedia or a dictionary. The point is to make design studies that focus in a sustained way on a limited set of issues in hopes of learning larger lessons. If your data model has fifteen or twenty entities then you're in good shape, and you can probably do a thoughtful design exercise with fewer. "Sustained" here means working the example over several times, hitting walls or noticing patterns each time, and then starting over. Next week we'll move onward from the data model to the uses that could be made of it. But it'll be useful to think ahead this week, making scenarios that might be spelled out in more detail next week. In fact, next week you may well have to backtrack, reworking part or all of your data model as the practicalities of data plumbing become clearer.
IS 240 -- Information Systems Analysis and Design

Assignment for week 8: Service design

The task of system design is to move from a concept to a specification that is sufficiently detailed that the technical people can take over and build it. Traditional design methodologies are cumbersome and linear, with little room for iteration or backtracking. In this course, by contrast, we want to emphasize that design is a discovery process, and that discovery takes place through iteration. Therefore we are dispensing with most of the cumbersome methodology and notation, and instead going for the underlying idea. We are also emphasizing several aspects of the design of information services that move to center stage in a world of cheap computing and ubiquitous networking, for example the way that one's information service fits into the patterns of daily life and cooperative work.

Accordingly, we asked you last week to sketch an information service that you might like to design to support collaborative work activities. We focused particularly on the data model because that's a concrete place to begin. But your work from last week is only a first iteration, and soon you will probably throw it out and start over, or at least your design will evolve considerably. Because design is a matter of reconciling diverse constraints, this week we want you to iterate the design of your information service by taking a wide variety of factors into account. This assignment will explain the factors, and we hope that in confronting them you'll be compelled to iterate your design. Work out the details of each factor in a rough way, and then see if any problems arise. And see whether you've been inspired to reconceptualize your design to make it more interesting or useful. If all goes well, the result will feel more like a discovery than a preconceived idea. You will need a consciously chosen group process that reconciles divergent thinking -- throwing out lots of ideas without promising that they make sense -- and convergent thinking -- structured work that selects the most promising options and works out their details in an exploratory way. This cycle works best if you have a lot of paper in front of you, and lots of things to write with. The resulting materials will help document your work process.

Here, then, are the factors that your information service design should take into account. Each factor comes with a list of questions. Your answers to these questions do not need to be overly precise at this point. The important thing is to have thought about them, and to have made conscious choices. You should also go for answers to the questions that are interesting, that is, that explore a potentially novel and useful region of the design space. Your presentation does not need to work its way mechanically through all of these factors. Instead, you should find the conceptual center of your design -- the unifying theme that makes sense of the design on every level -- and organize your presentation around that. Beware of excessive complexity: a good design expresses a clear concept in an easily grasped way. If your design feels like a utility belt with dozens of disparate tools then you're not doing it right. Your job is not to support every aspect of the activity. Rather, your job is to use the iterative process of design to explore a single aspect, preferably an interesting one. We have gone to great lengths in the past several weeks to

http://polaris.gseis.ucla.edu/pagre/240/week8.html
develop a good vocabulary for such things; now is the time to convert that vocabulary into design.

Roles. We have banned the word "user" in this course because the roles that people play in the vicinity of any interesting information service are more differentiated than that. Potential roles include author, editor, manager, system administrator, seller, buyer, customer service representative, and the ongoing role of the system designer. In other words, your roles should sound more like job titles -- or social roles anyway -- than the generic "user". Because your information service is going to support some type of team work, "team member" is probably an important role. Do the team members all have the same role, or are their roles differentiated? Does the same team member take up different roles at different times? If so then your data model will want to make that distinction. Does your service tend to rigidify the roles, so that team members are stuck more firmly in their roles than they had been? This might be a bad idea, or it might not. Remember that we are talking about the technology of 2010, when ubiquitous networking will make it possible to invent new roles or reinvent old roles through the mediation of the network. Who would the team like to be able to reach out to? What kinds of ongoing to loosely coupled relationships would it like to be able to maintain? Perhaps the service keeps track of everyone's network of professional relationships?

Geography. Where is everyone? All in one place? Mobile? Does the service provide different modes of work depending on whether people are together or apart or whatever? If the people occupying the various roles can be spread all over the map, where should the data be stored? A great big centralized database can be convenient for administrative purposes, but it can also be slow for people who have to open a connection over an unpredictable network every time they need any item of information. You may not have the numbers you need to specify in a rational way how the data is distributed, but just thinking and mapping the geography of the activity is a good way to start on the service architecture.

Access. Who gets access to what information? Perhaps everybody who occupies a certain role gets access to certain types of information, or perhaps the service needs to differentiate among specific individuals. Concerns might include security, privacy, theft, and trade secrets. Think about a service that supports and structures a relationship with a competitor, a customer, or a subordinate. Think about the potential consequences of information getting into the wrong hands. For example, partial work products can be interpreted as evidence of bad work practices, and brainstorming notes can seem foolish out of context. Think, too, about the ways that people in various roles get access to the data. Do they subscribe to it, that is, have it continually updated on a display on their wrist or a diagram drawn on a large bit-mapped flat-panel display on their office wall? Do they have a way of specifying what data they are subscribed to in this fashion and how it is presented? Do they enter a traditional boolean search query? Do they pass it around to one another, for example in links in an electronic message? Do they call in for it on a phone or other device? Does everyone have their own particular window on the data?

Capture. You've decided what kinds of data go into your database, but who creates this data? Are
the occupants of a particular role responsible for typing it in? Is it captured by an environmental sensor? By specialized devices that talk to one another over a (wireline or wireless) network? Is the data produced as a by-product of computer-mediated work activities whose deliverables are themselves part of the service's data? Does everyone have incentives to create the necessary data? What happens if the data is not created in a timely fashion, or if it gets out of date? Will data entry be an ongoing, complicated burden? Will the person who creates the entry be required to make a lot of gray-area judgement calls about the way that things (e.g., diseases, concepts, arguments, qualities, materials, tasks) should be classified? Can the data be used to control people, watch over them, or otherwise generate consequences for their lives? Can the data be used for additional purposes once the service is under way? These questions can raise real difficulties for a draft design, and send you back to reconceptualize it.

Structure. What data should be structured, and what data should not be? A traditional e-mail message, for example, includes some semi-structured data -- the header with its repertoire of distinct fields with fixed meanings and constrained formats -- and some unstructured data -- the body of the message, along with the contents of the Subject field and some other items. An eBay auction item likewise includes both structured and unstructured data. Most documents are this way, and the "document" perspective provides a valuable contrast to the "data" perspective. Of course, even the body of an e-mail message is "data" in a narrow sense: it consists of a sequence of characters from a standardized character set. Structured data can be processed in more and better ways than unstructured data, but it also imposes more constraints on whoever or whatever creates and captures it.

Actions. What actions do the various role occupants need to take? That is, what commands do they issue to the service? An eBay customer, for example, might take actions such as putting a product up for auction, placing a bid in someone else's auction, and leaving "feedback" about another customer. Likewise, everything that appears on a command menu (such as long the top edge of the Macintosh screen) is an action. Some actions are aimed at retrieving information; others constitute institutional acts like ordering a product or committing to a contract; others are part of an interaction with someone else. Just describe the activities involved in using the service and then look for the verbs whose subjects are people. Don't forget "boundary" actions such as signing on, logging off, creating an account, undoing an action, reporting an error, or destroying a record that's not needed any more.

History. Does the service maintain a history of past actions? Why? If so you will need a data model for that history. Who gets access to the history? And so on.

Interfaces. What existing services does your service exchange data with? One type of interface is the one between your service and the hardware or software platform(s) it is built upon. A database can be an interface when more than one service uses it; for example, if your service draws information from a database whose contents are generated by another service, that's an interface. Your service will also have an interface, called a device driver, to any electronic device that it interacts with -- a sensor, an actuator, any kind of input or output device, a handheld or portable
device, a device that is built into a car, a cell phone, etc. Every interface will have its own data model, which will be a subset of the data model for your service as a whole.

Processes. How does the data flow? Perform a rough sketch of a hierarchical decomposition of the system, just enough to make its overall structure clear. You absolutely needn't expand your functional decomposition down to the level of individual machine instructions such as adding, storing, comparing, and printing. Understand which processes get commands (and possibly data) from people in specific roles. Understand which processes can create new records in which databases, and which processes can change old records. Understand which processes can read records from which databases. Remember that a data flow must always pass through a process: there is no such thing as a data flow directly from one database to another, or from a database to a person, or a person to a database. Give your processes common-sense names, not ones that sound technical.

Versions. List the functionality that your service provides, and ask whether the service comes in different versions. It is common, for example, to have an underpowered free version that builds up a critical mass of customers who might want to buy a feature-filled advanced version. Also, get a rough sense of the order in which features -- that is, which databases, which actions, support for which devices -- should be implemented. It is more important to get a prototype into people's hands than to spend years building a monster that nobody might want.

Other issues in information service design. Back up and think about your service as a service. What other issues arise? Will it work economically? Will enough people use it to pay back the cost of development? How can you make it more widely useful? Return to your notes from the first class where we listed a batch of issues on the board.
Assignment for week 9: Information design

Information design concerns the visual display of structured information. It derives not from the technology world but from graphic design: people who design the visual appearance and organization of things like brochures, signage, museum displays, and expository illustrations for newspapers and textbooks. We are emphasizing information design in this course because we believe that it is the future. In particular, the ergonomically oriented field of user interface design is stuck in the single windows/menus/mouse paradigm, and the more general field of interaction design does not have adequate ideas about the organization of the information that one interacts with. We do believe that these fields have their place, but that the best first step is to learn from the graphic designers about the visual display of information before revisiting the question of how to interact with it. So in asking you to conceptualize the information design of your service, we are not asking you to figure out what commands go in what menus. Rather, we are asking you to figure out whether you want to use menus, or whether you want to draw on some other information design metaphor. More generally, our focus is not on interaction but on the organizing principles of the things one interacts with.

Last week you worked out some of the internal structure of your information service. Your assignment this week is to prototype the physical form and information design of your service. The purpose of this document is not to explain what information design -- the readings and lectures will do that. Rather, this document will present a conceptual framework for understanding the full scope of the problem that information design solves. Along the way we will also discuss issues of physical form, although that is a secondary focus.

The first priority of information design for electronic information services is that the whole design, inside and outside, should be unified by a single conception: a clear and powerful metaphor or symbol or ideology that provides guidance to every aspect of the design. The most famous example of such a unified conception is the idea of individual liberation that informed the design of the Apple Macintosh. The Macintosh was design in depth: not only was every aspect of the hardware, operating system, packaging, and company communications informed by the liberation ideology, but the designers also wrote a manual for the "look and feel" of Macintosh applications that were developed by third parties. Another, closely related design principle of the Macintosh was a clear mapping between the internal workings of the machine and its outward behavior. Entities of the data model became types of record structures, and they also became elements of the interface. Visual elements (e.g., icons) and commands (e.g., menu items, behavior of the mouse) mapped cleanly onto the internal workings of the software. The increased power of computer processors and the increased diversity of input and output devices and communications mechanisms have greatly multiplied the design space for interactions between people and machines, but these design principles still hold. Of course, it may be impossible to achieve the idea of a unified conception that drives every aspect of a design. But it is important to try, and
every aspect of a design that diverges from the unifying conception is likely to be problematic in some way.

As the design space widens, it helps to have a conceptual framework with which to map the full range of potential information designs. Among the dimensions to consider are these:


* Time. How does the image change over time? Does it change in response to actions that people take, and if so which ones? We are accustomed to the personal computer model and the "user illusion", according to which almost all changes take place in response to specific commands by the person who is sitting at the keyboard. But in a distributed system that interconnects many people who occupy several differentiated roles, the design space is obviously much larger.

* Narrative. If the information participates in telling a story, explaining something, simulating something, or showing how something works, then all of the categories of narrative apply. A narrative has a beginning and end, a world with rules, an internal logic, characters and places, events and scenes, and so on. It also has instructed seeing: not just the "showing" aspect of the story (facts, events, actions, and other concrete specifics) but the "telling" (what it meant, what was really happening, the pattern, what the character was thinking, what aspect of the events are to be significant, and so on). Even when the information service or the people who are using it are not clearly telling a story, narrative concepts can be heuristically helpful in articulating the temporal dimension of its relationship to the people who use it.

* Action. Systems analysis produces, among other things, a repertoire of actions that people in various roles need to be able to take by using the service. Examples of actions might be bidding on a product, signing off on an invoice, deleting a message, reclassifying a document, promoting someone to a new job title, or closing a case. Notice that some of the actions are defined purely in terms of the service's workings (e.g., deleting a message), while others (bidding, signing off, reclassifying, promoting, closing) are also institutionally significant -- that is, they change the world and not just the machine. The visual display of the service might make clear which actions are possible at a given juncture. It might provide feedback about which action has been taken. It might ask for reconfirmation. It might suggest actions. Although our focus here is primarily on the visual display of actions (e.g., with icons that can be clicked), one might also accomplish these purposes using sound (e.g., recognizing or speaking words) or touch (e.g., the "clicking" sensation of a key), and you are welcome to mix the various modalities if that's the direction the design process takes you. Observe that the visual representation of action is usually closely related to the visual display of everything else; this is particularly evident in the case of "direct manipulation" interfaces such as the dragging and dropping of icons in the Macintosh.

* Linguistics. Human language is the starting point of systems analysis, and in most information
design human language is selected and associated with other meaning elements. (Even when no words appear on display screens, you should be aware of the language that people will use in thinking and talking about your service.) This language should be consistent in both the words you choose and the meanings you assign them; it should be appropriate to the people who are using the service; it should avoid trampling on other meanings that the same words might have in contexts that the people already know about; and it should be aligned in every possible way with the unifying conception of the design. It should be easy to configure the service to use a different language or dialect. Linguistic elements can be wildly misinterpreted, and they should always be tested with real members of the relevant communities -- especially ones who are less familiar with services such as yours -- to see what misinterpretations your chosen words might inadvertently invite.

* Design language. A "design language" is not a set of words of natural language. Rather, it is a set of design elements and design conventions that are used consistently in a range of contexts. It is crucial to have a design language any time that more than one visual display of information is supposed to operate as a "family". For example, all Macintosh applications are visibly and unmistakeably *Macintosh* applications. Every page generated by a given Web site should likewise employ shared conventions. The signs accompanying a museum display should be come in visually distinct categories (text explaining whole a room as opposed to an individual picture) while also clearly being part of the overall information design of the show. The elements in a design language might include symbols (e.g., icons, logos, navigational devices), typefaces (e.g., Times Roman for everything, italics for captions, small type for legal boilerplate), colors (e.g., background, pull quotes, highlights, selected items, urgency), and vocabulary items. The design conventions might include the layout of information on the screen (e.g., commands on the left, site map at the top, conventions from paper genres like spreadsheets), the set of actions that are available and how one performs them (e.g., drag and drop, "OK" button in the lower right corner), the use of titles and summaries (e.g., automatically generated text abstracts set off in side-bars), and the spatial relationships among the design elements (e.g., text is always on the button rather than next to it).

* Structure. Most information designs will include both structured and unstructured elements of a document, and each poses its own design challenge. For structured information, the challenge is to make the structure visible, while also making visible the lesson or point that can be taken home from the structured information as a whole. A tabular display of data, for example, does not make trends as visible as a graph does. But at least structured information occupies a predictable amount of real estate on a visual display. Unstructured information such as free text can be less predictable, and you may need a style manual for writers that dictates the size of each textual element such as a caption. Images are predictable in their spatial demands, but it can be very difficult to direct someone's attention within an image. Actions over structured information can be more complex than those defined over unstructured information.

* Reference. Every visual display is part of history, and it cannot help referring to culture, tradition, genre, and style. Everybody who uses your service will already have mastered a
staggering number and variety of visual codes, whether from technical diagrams, films, graphic design, cartoons, or computer interfaces, and the visual displays that your service generates will inevitably be interpreted with reference to those codes. If your service is going to remind your community of a textbook, then you must either embrace that reference or revise your design to stop inviting it. This principle applies on every level: from individual design elements, to conventions of spatial layout, to ways of telling a story, to ways of taking an action, to literary genres. If your service resembles the Macintosh interface then you will have little choice but to uphold the conventions of that interface. If your service looks like a comic book or a Russian propaganda poster or a television screen or a Rand McNally road map, then you have to anticipate that interpretation and take a stand toward it.

* Embedding. Your visual display does not exist in a vacuum. It is a physical object that stands in some physical and practical relationship to everything else that is going on around it. Think, for example, of a display that is a sign that people walk past, or a display that is sensitive to the devices that might be present, or to environmental conditions. People might look at the display from a variety of distances or angles, or they may notice changes in the display from the corner of their eye. Lighting conditions may not be reliable. Glare may be a problem.

* Attention. Closely related is the structure of the individuals' attention. What else is going on when people are looking at the display? Is the display calling out for attention or competing for attention with other visual things? Is the individual focused on the display for the whole time that the service is telling some story, or does the service have no guarantee about the times that people might arrive or leave? Is the individual's relationship to the display mainly exploratory or routine or what? Are multiple individuals engaged in joint attention, for example with one person explaining something to another person by directing their attention within the display? Will the individual's attention be directed by changes in the display that are caused by someone else at a remote location, for example in a distributed game or dialogue? Will particular design elements within the display be inviting attention? And so on.

* Function. What function does display play in the ongoing activity? The ongoing activity will probably be structured by an institution, a set of customs, a relationship among people, some rules, and so on, and the display will take on most of its significance within this context. A menu, for example, serves a very stable function in the rituals of the restaurant.

We are also asking you to design and mock up the physical form of your design. This could include a keyboard on a deskop, a flat screen display on a wall, a device that resembles a wristwatch or cell phone or palm pilot or credit card, something that is sewn into clothing or mounted on eyeglasses, installed in a car, carried on a belt or in a purse or backpack, or whatever you can imagine being built with the technology of ten years from now. You are most welcome to presuppose the processing capacity and communications capabilities of 2010, and to assume the kind of spontaneous wireless interaction among devices that we discussed in week 5. As with the information design of your service, the physical form should also express the overall conception of the design as a whole. It can do this in an infinite variety of ways, including its shape,
markings, colors, and plugs, or in the way that a person holds it, interacts with it, notices it, shares it with others, hides it, discards it, and so on. Its cost may be significant. And just as the
information design inevitably refers to a vast history, so the physical form will inevitably refer to
a vast history of industrial design. Furthermore, "information design" and "physical form"
overlap, inasmuch as the object itself conveys information whether it changes outwardly or not. Information design applies to simple LED's just as much as it applies to wall-sized bit-mapped
displays, and it also applies to symbols and icons on the physical device, such as the ones that
label the connectors, or that sit on the edge of the screen to show where the touch sensitive areas
are. All need to be designed in a coherent way.

We are concerned particularly with information design that supports shared work, and so it may
help to reflect on a number of examples of such design:

* "Local design" -- the design conventions evolved within a group, such as notation and
vocabulary, whether wholly original or inflections or variations or additions upon conventional
designs.

* Being aware of one another, like those video space interfaces that include "postage stamp" video
of each other's office so you can see who is in.

* Planning and coordinating collaborative work, such as PERT charts -- in a networked world,
those same displays can probably be continually updated to provide a running display of project
status.

* Information about individuals, such as contact information, skill set, resume, professional
network, whereabouts, current actions, history of actions, etc.

* Materials that individuals in the group prepare for one another, for example materials to review
before a meeting.

* Hand gestures and other symbols that synchronously accompany interaction.

* Any representations the group members may be working together jointly to produce, and a wide
range of annotations and suggestions etc about drafts of those deliverables.

* Agendas and minutes of meetings, and representations of parliamentary-type procedures like
voting, deciding who speaks next, moving along to the next topic, etc.

* All sorts of speech acts -- even though these might be done with unstructured English in a face-
to-face context, when done remotely or asynchronously they might require structure or design etc.

* Instructions (e.g., judge to jury, manager to subordinates, manufacturer to repair person) that a
team might jointly use to organize its work together.
* To-do lists, lists of action items, and documentation of the work.

* Progress reports, evaluation and feedback, queries to others outside the group, etc.

* Representations of group work processes that are meant as advertising for the group and its work, or reports on the work in a journal.

There are probably many more categories of information design to support joint work, but that's a sampling anyway.

That brings us, finally, to the process you'll follow in your own information design work, and to the results you will show to the class. Like every other sort of design, information design is iterative. Do not get overly detailed in your first several iterations. It is much more important to have a clear concept, and to explore the various dimensions of its consequences, than to spell out all of the details, especially at the beginning. Just develop enough of the consequences of your concept to understand the problems and opportunities that arise from it, and then reconceptualize. Rather than dwelling on one design concept, develop a number of them and explore them all, picking the one that seems to be going somewhere. Remember, once again, that your information design concept needs to be aligned with the unifying concept of the whole design. You may find that the unifying concept that you developed last week does not suggest a compelling information design, and that a new information design concept suggests a reconceptualization of the whole function and structure of the service. This is a good thing, and you should embrace it.

In this assignment we are not emphasizing "user" participation. What you are developing is an early mock-up that illustrates a concept, not a serious candidate for a implementation as a finished system. You might think of your presentation to the class as a presentation to a community of practice that might use your service. Experience shows that such presentations work best when they are relatively low-tech. Slick PowerPoint demonstrations or carefully worked-out interfaces communicate your intention to implement just exactly what you designed. By contrast, mockups with cardboard and colored pens are nonthreatening, and because they connote a low level of effort they communicate a willingness to iterate, or even to throw things out and start over if necessary.
Assignment for week 10: Service architecture.

The main order of business this week is iterating your design, getting all of the pieces to make sense and work together. But we'll also take up a final substantive issue: the rudiments of information service architecture. So far, the information service you've been designing is just a couple of diagrams and some sketches for an interface, with little sense of how any of the stuff is implemented. Writing the actual code is someone else's job, at least according to the traditional division of labor. But as a designer, you must decide some deep structural questions: how the service is organized into modules, what platforms they're all built on, and what protocols they use to communicate with. We have addressed such questions before, back in week 4 when we looked at some of the platforms that are available as standards in the real world -- TCP/IP and WAP, for example -- and in week 5 when we explored what it means to build a service on top of a service layer -- the hypothetical Suitcase Layer. This week we will consider such matters more systematically.

It will help to read the sections in Messerschmitt on the client-server model, the peer-to-peer model, layered services, middleware, network protocols, and transaction processing. I will not repeat this material, but I will explain bits and pieces as part of the assignment.

Here is an intuitive way to understand the issue. Your service will probably include several distinct devices -- such as desktop and handheld computers -- that the various parties use to interact with one another. The question is, how is the computing activity divided among the various devices? One option is to have a central computer -- a server -- that does all the hard work. The server, which is operated by some central authority somewhere out on the Internet, stores all the data and performs all of the hard computing effort. Then each individual has a device -- a client -- that does little work besides displaying images on the screen and relaying commands back to the server. The client-server model can be contrasted with the peer-to-peer model, which has no server and indeed no asymmetrical relationship among the devices. Each device stores its own data, and the devices communicate with one another over the network directly when they need to interact. Both the client-server model and the peer-to-peer model provide coherent ways to divide the computational effort among several machines. And many other options are possible. Napster, for example, has a small server that tells the clients about one another, but most of the action happens in a peer-to-peer mode of data flows among the individuals' machines.

The great success of the Web has accustomed us to thinking in client-server terms. And indeed, the rational designer today hesitates before building an information service on any platform except the Web. But that will surely change as service models multiply, and so you should look at the service architecture problem in the more general way that people looked at it before the Web: what is the best way to distribute the data and computational effort among the various machines that are involved in your service?
You may wonder how this question might be answered. Surely the world is full of advanced methodologies for deriving optimal answers to it. Not so. You can read a book like Whitten and Bentley, and they will provide an elaborate design methodology that doesn't constrain you that much. Some answers are better than others, but if you comprehend the question then you are qualified to evaluate the answers.

More formally, you are starting with two diagrams: a data model and a dataflow diagram. Every entity in your data model corresponds to a relational database on some computer. Every process in your dataflow diagram corresponds to a piece of software code running on some computer. If a process requires information from a certain database, then it will be convenient for the process and the database to be located on the same computer. But it won't be obligatory: if a process on computer A needs data from a database on computer B, it can always retrieve it over the network. But this can be slow. What is the designer to do?

Much of this complexity can be suppressed with middleware. For example, a distributed database service layer might be designed to hide the boundaries between the different computers, so that every database looks like it is immediately ready-to-hand whenever any process on any computer wants to read from it or write to it. (The assignment for week 5 mentioned distributed object management; this is a distributed database for object-oriented data, as opposed to the relational model of data that we are using.) The distributed database middleware layer might replicate the various databases on several machines, using subtle protocols to keep the replicated copies consistent, or it might simply send a request over the network whenever a process on one machine requires data on another machine.

We are *not* going to assume the existence of a distributed database in this assignment. A distributed database is excessive generality for our purposes, and it fails to illuminate the architecture issues. Instead, we are going to face these issues squarely. So the question is: which databases do you want to store on which computers, and which processes do you want to implement on which computers? Even though you don't have a general-purpose distributed database, you will certainly have a middleware layer that allows any process to interact with a database on a different machine. (You don't have to worry about the mechanics of this interaction. It involves simple packets sent over the Internet.) So you have a wide range of choices, and the only real consideration is efficiency: it's a lot slower for a process to interact with a database on a machine that's 1000 miles away than for a process to interact with a database on the same machine. So the rules of thumb for design are obvious: try to group processes together with the databases they will use, and try to put processes and data close to the people who most want to use them. You can easily divide a database across several machines, for example with the records pertaining to each individual customer being located on the particular machine that the customer users. (Dividing databases in this way puts an extra burden on the middleware layer that interacts with remote databases, but don't worry about that.) You can also place duplicate copies of the same process on several different machines if that's useful. But don't try to replicate the same data on different machines; that's too hard.
If this all seems abstract, think of it this way. Ignore your data model for the moment. Look at your dataflow diagram, and draw lines through it that partition the processes and databases into different regions. Each region is a different machine. If you draw a big central region that contains most of the databases and processes then you're probably imagining a client-server architecture. If you draw a moderate-sized region around each individual person and his or her own local data and processes then you're probably imagining a peer-to-peer architecture. Your regions can overlap, indicating that you are dividing up some of the databases between different machines and duplicating some of the processes accordingly. This makes more sense when you see it drawn on the whiteboard, and I'll do just that in class.

Dividing the labor among different computers is still an abstract approach to service architecture. Along the way, you will have to make some more concrete decisions. First of all, what platforms is your service built on? This will include hardware platforms such as IBM-PC compatible personal computers, the Palm Pilot, WAP-enabled cell phones, and big mainframes operating as servers. You are welcome to posit new hardware platforms, much the way we did when designing for the Suitcase Layer in week 5. Remember that we are assuming the technological capabilities of ten years from now. Other platforms will be implemented in software, such as the World Wide Web, the Internet Protocol, and the Secure Sockets Layer. If you want to posit new software platforms then that's probably fine, but check with us first. Last week you invested much effort designing the visual appearance and physical form of people's interactions with your service, and now it is time to specify the guts of the devices that the people interact with.
IS 240 -- Information Systems Analysis and Design

Final presentation assignment.

This is the assignment for the final presentations for IS 240. We will hold the final presentations on Tuesday, June 13th from 5pm through 7pm in the usual GSE&IS classroom. Each team's presentation should last for 15 minutes.

The assignment is to show us your project as a design study, that is, in terms of what has been learned by exploring the design space. That includes the various levels of the service (data, interface, architecture, etc), the way the service is embedded in the community that uses it, the unifying conception of the service, and so on. The presentation need not be encyclopedic, and even though we do want you to use technical terms and notations correctly, the emphasis is not on right answers but on what's been learned from the design study.

You can use any genre of presentation you like: corporate PowerPoint, infomercial, academic talk, Greek tragedy, whatever will convey the ideas. The "what has been learned" can be conveyed through explicit bullet points, or you can let the design itself do more of the talking. The presentation itself should be designed, just as any presentation should be, but we do not require that the presentation be slick or highly produced, or that it employ labor-intensive graphics etc. None of that follows automatically from the real meaning of "design".

We will use reasonably bright lights, as we did last Tuesday evening. We will video tape all of the presentations, and we will put the videotapes on the Internet. You are welcome to edit your own group's tape before we put the tapes online. You are also welcome to run through your presentation a second or third time after everyone else is done, if you're not happy with the presentation or the tape.