Multithreading + Qualitative Methods

CS160: User Interfaces

John Canny

Review

Event-Driven Interfaces

- Hierarchy of components or widgets
- Input events dispatched to components
- Components process events with callback methods

Model-View-Controller

- Break up a component into
  - **Model** of the data backing the widget(s)
  - **View** determining the look of the widget
  - **Controller** for handling input events
- Provides scalability and extensibility

This time

Multithreading – why and how

Ethnographic methods

Why Multithreading?

An interactive program always has at least one thread handling user interaction.

To keep the program interactive, user callback code must always return fast, i.e. 0.1 second or less.

A long callback routine blocks all user input, e.g.

- Getting help
- Aborting
- Doing something else while waiting
- This violates several usability heuristics (which ones)?

Many real computing tasks take too long:

- File I/O, database updates
- Big calculations
- http transactions

Need separate threads for these to keep the UI alive.

Why Multithreading?

Multiplayer games.

On the server, each player may asynchronously change the game board. Need threads to listen for these updates and update the database, and to propagate changes.

On the client, need at least one additional thread (usually two) to send and receive packets from the server.

Why Multithreading?

More examples

Multicore processors – threads are an easy way to use all the computing you have available.

Dynamic widgets:

- Clocks
- Progress indicators
- Mail inbox…

The production version of almost any CS160 project should be multithreaded, but ***NOT*** the prototypes developed for the course.

Why a lecture on multi-threading?

Its not that difficult to do the things that most CS160-style projects need to do, BUT

Without careful design and planning, multithreaded code is the fastest and most effective way to get into deep trouble…

Why a lecture on multi-threading?

Consider a simple 10-line, non-branching program. There is only one possible control flow.

Now suppose **every** line has a binary test. There are 1024 possible outcomes.

Now suppose the **non-branching** program is executed by two threads. There are 184,000 possible interleavings of the 20 instructions that will be executed.

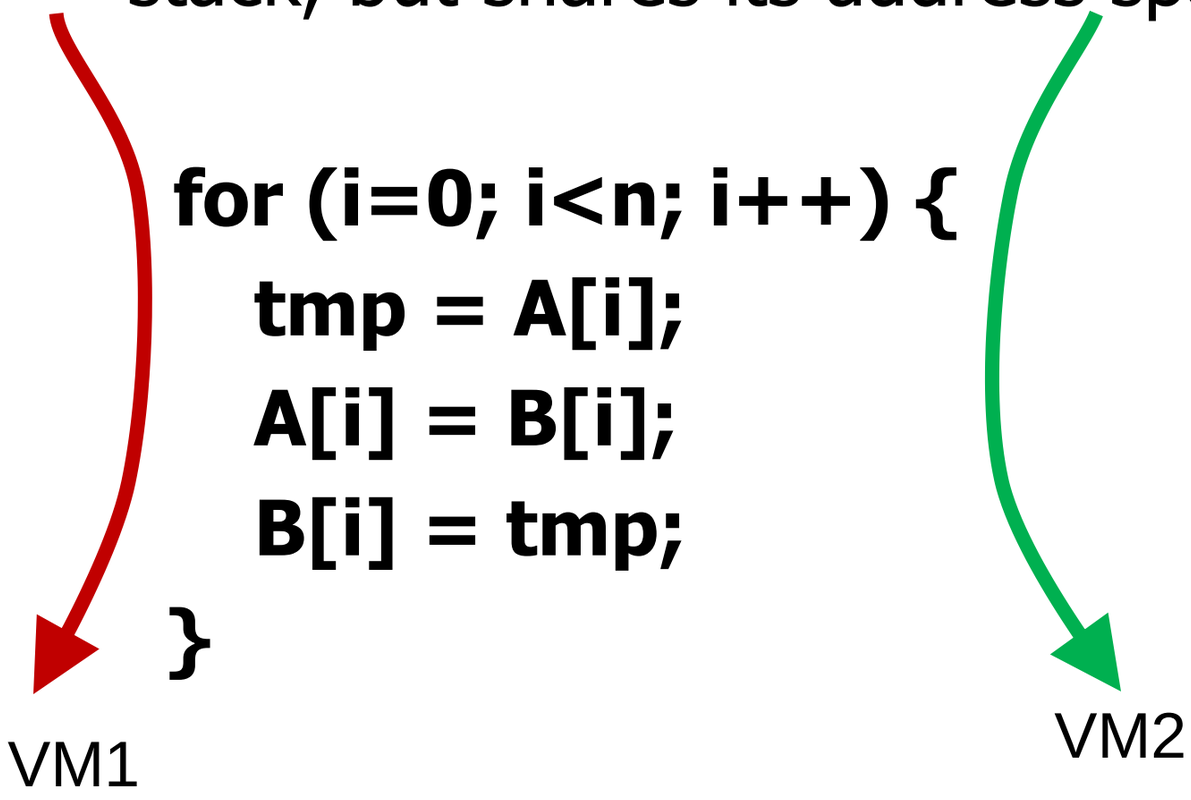
Far less attention is paid to multi-threaded programming than any other aspect of structured programming.

Threads vs. Processes

A **Process** is an autonomous **virtual machine** with its own stack and memory address space.

An identical program can run as two processes without interference in memory (except for memory-mapped I/O).

A **thread** is a **partial virtual machine**. Each thread has its own stack, but shares its address space with other threads.



Thread Safety

Code is **thread safe** if it can be called from multiple threads without interaction between them.

A simple C++ function or method works just fine:

```
int fact (int n) {
   int i, p;
   for (i=1,p=1; i<=n; i++)
      p*=i;
   return p;
}
```

Separate ints i,p are created **on the stack** each time the function is called. Each thread has its own copy.

Thread un-Safety

But what happens if we change the variables to static (shared)?

```
int fact (int n) {
   static int i, p;
   for (i=1,p=1; i<=n; i++)
   p*=i;
   return p;
}
```

Thread Safety and OOP

- Object-Oriented Programming makes heavy use of class instances, and these are often heap-allocated. Heap storage is shared between threads, and so class instances can be seen and manipulated by all threads.
- You need to take extra care when writing OOP multi-threaded code for this reason.
- Share references to class instances between threads only when absolutely necessary (i.e. when they are the communication channel between the threads).
- Only one thread should be responsible for allocating and de-allocating classes of a given type (failure to do this is the cause of the Windows Vista “Explorer Helper Object” crashes).

Thread Communication

The point of using threads is to allow **fast communication through shared memory.** Everything can't be thread safe, or the threads could never cooperate.

Threads can communicate through various shared objects such as:

- Synchronized queues
- General monitors
- Databases

In increasing order of shared state..

Let's start with the simplest method…

Synchronized Queues

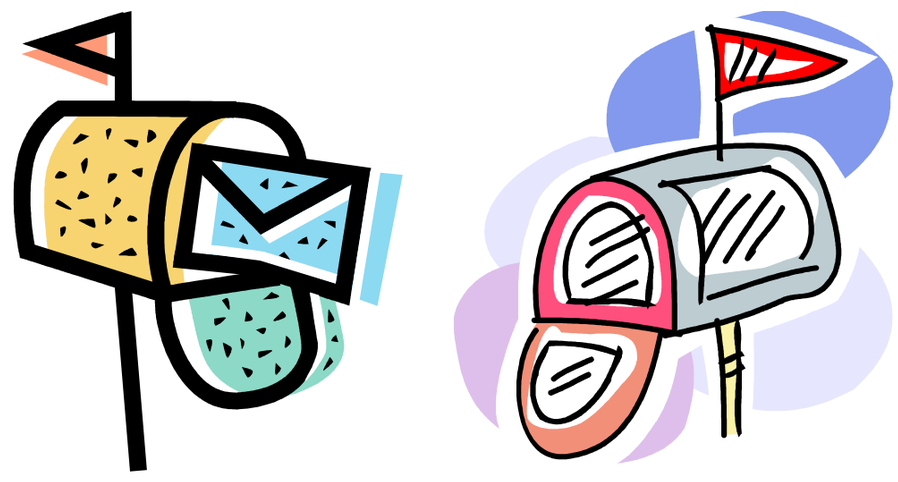
A synchronized queue allows one thread to insert objects into a queue and another to read them.

If the only shared state between threads is through the queue, programming is much easier.

The two threads work asynchronously – the queue is called "synchronized" because data is kept consistent, i.e. the consumer thread cannot access an object being inserted until the producer finishes.

Suitable examples:

- Stock ticker
- sport score updates
- live vote tallying,…



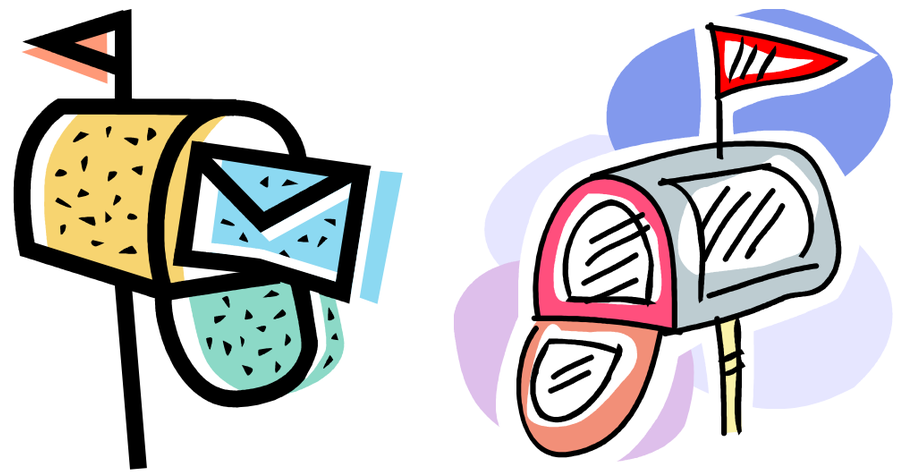
Synchronized Queues

Examples of synchronized queue APIs:

- .NET MessageQueue objects
- Thread-specific message queues in the Windows GUI API
- Java Message Service (JMS)

And similar primitives exist between processes:

- HTTP post/get
- Unix Sockets
- CORBA asynchronous messaging



Synchronized Queues

Advantages:

- Code is basically sequential in each thread. Much easier to develop and debug.
- Reusable queue libraries do all the hard work.

Disadvantages:

- Inefficient if too much communication:
  - Serialization and de-serialization of data objects
- May be necessary to duplicate data structures on each end to support application methods
  - If this happens, you should be using real shared objects or proxies

Synchronized Data Structures

To implement a data structure that several threads can access, you normally have to protect certain lines of code.

```
Class myQueue {
  int [] A;
  int head, tail;
  …
  static void enqueue(int x) {
     A[head] = x;
     head++;
     if (head >= qsize)
        head = 0;
  }
```

What happens if thread interrupted here?

Or here?

Synchronized Data Structures

C# allows us to protect a method completely. Only one thread can execute it at a time.

```
…
[MethodImpl(MethodImplOptions.Synchronized)]
static void enqueue(int x) {
    A[head] = x;
    head++;
    if (head >= qsize)
       head = 0;
}
```

But what if we have many queues running at the same time?

Monitors

Locking code sections is often overkill. We only need to lock access to a specific object. We can use a ***monitor*** to do that

```
Class myQueue {
   ...
   object myLockObject = new Object();
   static void enqueue(int x) {
      lock(myLockObject) {
         A[head] = x;
         head++;
         if (head >= qsize)
            head = 0;
      }
   }
   static int dequeue(int x) {
      lock(myLockObject) {
```

Create a monitor local to this class instance

Only one thread can execute this code with this lock object, i.e. on this class instance.

Dequeue method should use the same lock object

Monitors

With a monitor we can allow concurrent access to different data objects. Only when there is contention over the same object will one thread need to wait for another.

With large, complex data objects (e.g. large data tables), we may want monitors which are more fine-grained than the object itself. E.g. a monitor for each row of the table.

Databases

Databases are almost universal in large, client-server applications including multi-player games.

Not only do they provide flexible, large-scale data storage, but they have excellent support for concurrent access.

All DB operations are synchronized, and DBs can be locked at a variety of levels including DB, table, and at the row level.

This time

Multithreading – why and how

Ethnographic methods

Ethnography in HCI

Ethnography (from Greek roots for people + writing) is an approach studying human behavior.

It emphasizes observation in context (field work), cultural factors and detailed description without appeal to a prior theory.

Ethnography is a ***qualitative method*** – it produces written descriptions, not data.

It provides information for designers to use in the early, exploratory stages of product design. Before a product is ever conceived, it allows designers to understand their target users, their wants, needs, desires, values etc.

Ethnography

Ethnography has its roots in anthropology, especially the work of anthropologists like Margaret Mead.

Mead studied native cultures by going to live with them and learning their daily practices.

She used photographs to chronicle these practices.

She also asked village children to create drawings for her. These provided a window into the children's thought processes.

Why Ethnography?

In the early days of HCI, there was much interest in psychology and cognitive science, the "quantitative" social sciences.

These areas provided insights into low-level user behavior, Fitts' law, MHP, memory principles etc.

But there was a dearth of methods to help understanding of users at high level: their needs, wants, values etc.

Ethnography does not provide such theories, but instead it can help designers **understand and even think like their users.**

For early stage design, that is all that is needed.

Ethnography is now one of the most widely used behavioral methods.

Why Ethnography?

An ethnographic study is a powerful assessment of users' needs: A crucial goal of an ethnographic study is to gain the capacity to view a system through the eyes of the user.

It uncovers the true nature of the system user's job: Users often performs tasks and communicates in ways that are outside of their official job description.

The ethnographer can play the role of the end-user: For example, the ethnographer can act as the end-user in participatory design when "real" end-users are difficult to procure.

The open-ended and unbiased nature of ethnography allows for discovery: The atheoretical nature of ethnography can often yield unexpected revelations about how a system is used.

Drawbacks

Time: The open-ended nature of ethnography requires a great deal of time, as does the analysis of data.

Presentation: The data (especially field notes) may be hard for designers to consume. On the flip side though, distillation of these notes can be an invaluable source for designers.

Scale and generality: To achieve the needed "depth" of observation, ethnographers must concentrate on a few subjects. Its never clear how general the results might be, or what phenomena may not have been observed at all.

Forms of Ethnography

Contextual Inquiry: is a specific type of ethnographic method which follows the master/apprentice model. Two others are:

Observational study: is an approach which attempts to minimize the impact of the observation on users' behavior (c.f. Star Trek's prime directive).

Participant Observation: The ethnographer participates in subjects' normal practices in order to better understand them. (what actually happens in Star Trek episodes).

In practice "ethnography" is an open, long-term process which contrasts with a task-focused application of contextual inquiry.

Examples

Intel's "People and Practices" group. Anthrolopologists who manage strategic thinking (and field work) for Intel's four market areas:

- Office computing
- Home computing
- Mobile computing
- Health care

Digital Home

Group managed by Genevieve Bell

Projects:

- Inside Asia – technology use (esp. mobile) in Asia
- Ethnographic studies of technology in Europe.
- Auspicious computing
- Religion and technology

Inside Asia

Asian countries have very different social practices.

- In Korea, phone use was much more reflective of social networks and hierarchies. The phone is a **social network maintainer**, not a convenience.

- In Australia, there were stronger norms for technology use in public spaces – don’t use laptops in cafes.

Inside Asia

- Widespread use of computers (at home) for education. Not just for schoolwork, but for distance education. The reasons differed: China, family and upward mobility, Korea, as an aspiration (Confucianism), India, as a responsibility (for certain castes).

- Use of messaging to sustain family networks.

- Gossip and “Jaiku” style mobile messaging.

- Integration of technology with religious practices

Concurrent Ethnography

An ethnography is pursued concurrently with, and linked to, the design of a system.

An ethnographer works with a small group of users, and meets/advocates for the users with the design team.

Ethnomethodology

A form of ethnography concerned with users' methods – how they accomplish their activities and especially how they cooperate.

A key tenet of this approach is that practices be **accountable**, that users have mental models and can give accounts of what they are doing.

We took a stab at this in your contextual interviews.

Ethnomethodology gives a more detailed recipe for doing it.

Evaluative Ethnography

Study of users experience with a prototype after it is built.

This approach fights with many of the principles of ethnography – the narrow tasks and goals hinders the ethnographers vision of all the surrounding context: culture, related tasks, needs, values, etc.

Rapid Ethnography

A true ethnographic field study will often be beyond a companies means. Rapid ethnography may be the answer. It involves

Narrow the focus: to those activities most relevant to the application to be designed. Use key informants to guide you.

Use multiple interactive observation techniques to increase the likelihood of discovering *exceptional* and useful behavior.

Use collaborative and computerized iterative data analysis methods.

Focus

Establish a general area of interest, and a set of questions to be answered by the field work. E.g.

In a hospital study, focus on medications. How are doses recorded and how are this info passed between workers?

In an air traffic control system, what manual activities needed automation, what did not, and what activities could be preserved in the new system.

Key Informants

Guides are users with a good knowledge of the activities to be studied, and the social network of users who perform them. They should also understand social tensions and can help avoid observation in unwelcome settings.

Liminal informants are users near the periphery of the social network. They typically have a fresh perspective on the group's social dynamics, but enough contacts to navigate inside the group.

Interactive Observations

Use multiple observers to make the most of the time in the field. Not everyone agrees on what they saw, especially ethnographers.

Focus on peaks of activity so that time in the field yields a maximum amount of useful data.

Elicitation: prompt users for their impressions.

Participant Observation: get involved with user activities.

Collaborative Data Analysis

Field data analysis almost always takes longer than data gathering.

Video and text data are the most challenging. But recently some text analysis tools (Folioviews and AskSam) have become available.

Cognitive mapping is a technique to distill and make sense of a large amount of qualitative data:

- Causal models
- Influence diagrams
- Concept maps

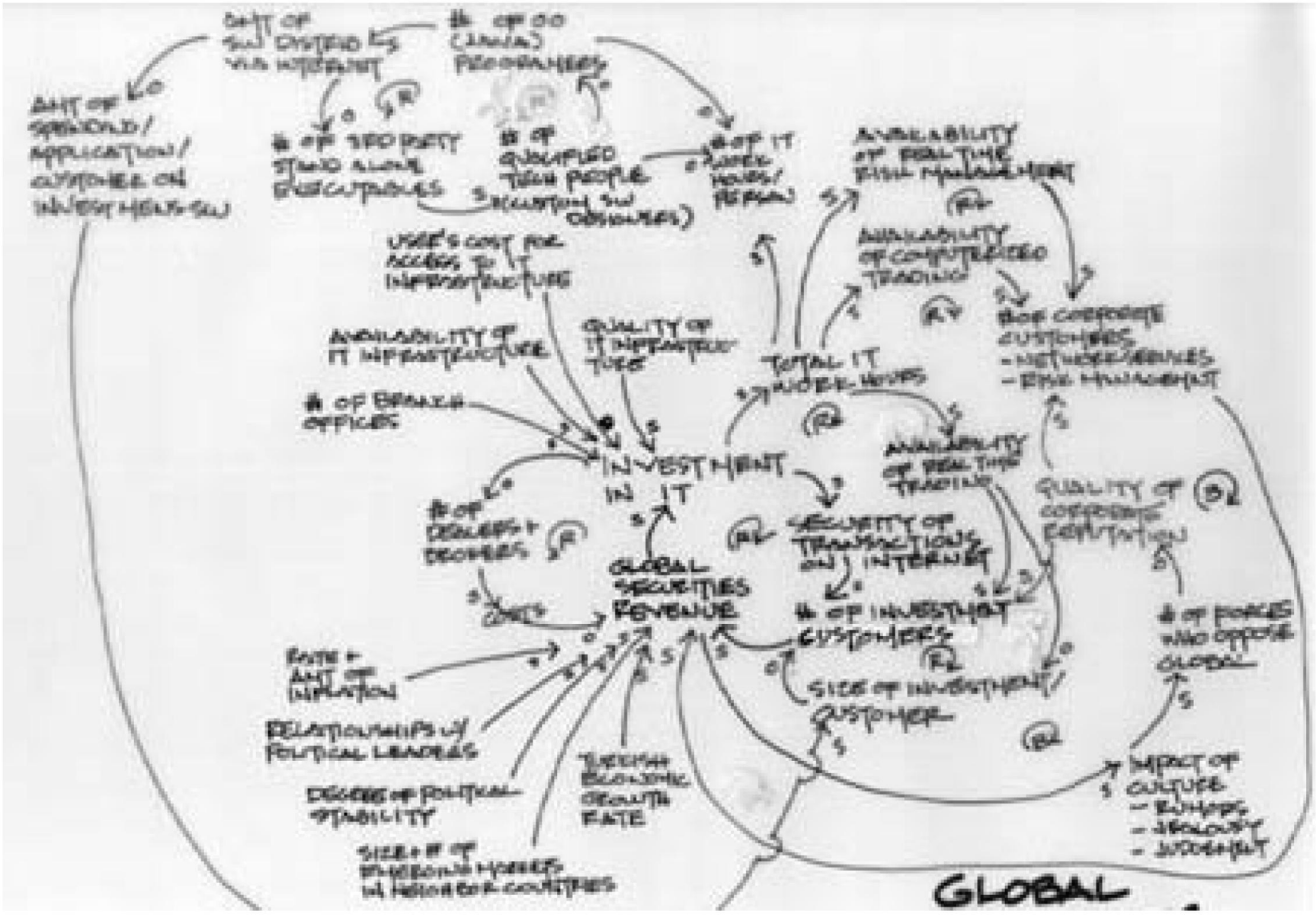
Example

"Thinking spaces" – an exploration of collaborative knowledge work and use of the internet.

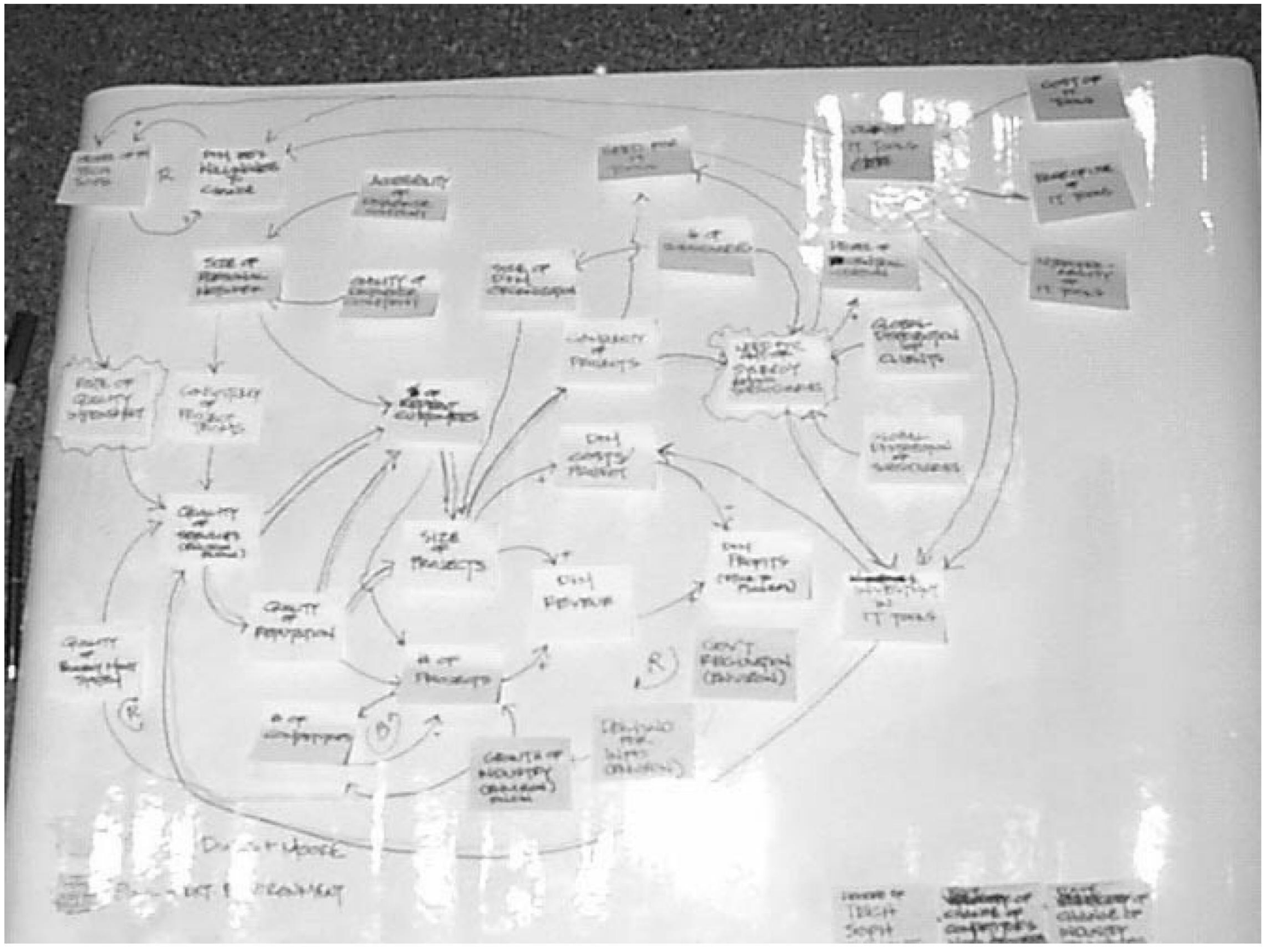
Key informants were managers responsible for Internet activity, and web masters.

Interactive observation: Users encouraged to create personal timelines of their internet use.

Causal Model 1



Causal Model 2



Summary

Multithreading

- Thread safety
- Shared data: queues, objects, DBs

Ethnographic methods

- Principles – culture, breadth, openness
- Rapid ethnography