Directions for Statistical Systems

John Chambers
Bell Labs, Lucent Technologies

March 22, 1999

Concepts, for Practical Software

Some conceptual hooks for a model of what makes computing with data work well:

1. *Language*, the ability to express computations.
2. *Objects*, the ability to deal with structure.
3. *Interfaces*, the ability to connect to other computations.

Useful Languages for Computing with Data

Users of interactive systems should, and will, become programmers. These user/programmers have two central needs:

- to tell the software what they want initially, quickly and clearly;
- but, then to gradually refine what they say to come closer to what they really need.

A slogan that sums this up:

To Turn Ideas into Software, Quickly and Faithfully

Ideas into Software, Quickly

The user becomes a programmer when the system doesn't *quite* do the right thing.

We'll slip more easily into programming if:

- we can capture what was just done, maybe modify it a little;
- objects and expressions are high-level;
- there's a consistent model behind the language.

Helpful Language Properties for Computing with Data

1. Easy to get started. *(when a user becomes a programmer)*

2. An expression does what it seems to do. *(avoids hidden side-effects)*

3. Reliable Syntax. *(simple, but not too simple)*

4. Progress by gradual refinement. *(classes, methods help)*

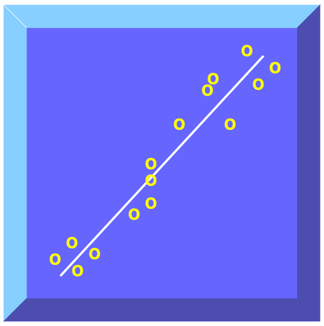
5. Everything is an Object. *(esp. the software)*

6. Flexible packaging. *(easy to organize, use)*

Which is the Model for the Future?

`lm(Fuel ~ Weight + Type, auto98)`

OR



GUI's and Languages

They should be allies, not competitors.

- Buttons, etc. get meaning from the *context*.
- Good languages needed to *define* and *modify* context.
- *One* language for both button and context?

Ideas into Software, Faithfully

How can languages encourage flexible, understandable software?

1. simple, and simply modified;
2. locality;
3. the language as objects.

No language gets full marks, perhaps none should.

Concepts, for Practical Software

Some conceptual hooks for a model of what makes computing with data work well:

1. *Language*, the ability to express computations.
2. *Objects*, the ability to deal with structure.
3. *Interfaces*, the ability to connect to other computations.

Structure in Data through Classes

Data are organized by classes, and classes are designed around:

1. natural basic classes (numeric, string, ...);
2. recursive definition (*attributes*);
3. relations to other classes (*extension*);
4. computations defined for them (*methods*).

And with one key principle:

Everything is an Object

Structure in Data (cont.)

Some comments:

- Similar concepts recur in many modern languages & systems (Java, Corba, S).
- Dynamic discovery of class definitions (*reflectance*, *metadata*) is a key tool.

Concepts, for Practical Software

Some conceptual hooks for a model of what makes computing with data work well:

1. *Language*, the ability to express computations.
2. *Objects*, the ability to deal with structure.
3. *Interfaces*, the ability to connect to other computations.

Interfaces between Systems

Users learn one language, send requests to many.

What helps make this work?

1. a good model for the requests;
2. hiding the remote communication;
3. avoiding the $n \times n$ connections among systems.

We handle 1 fairly well now; 2 and 3 will get better.

So, How are We Doing Now?

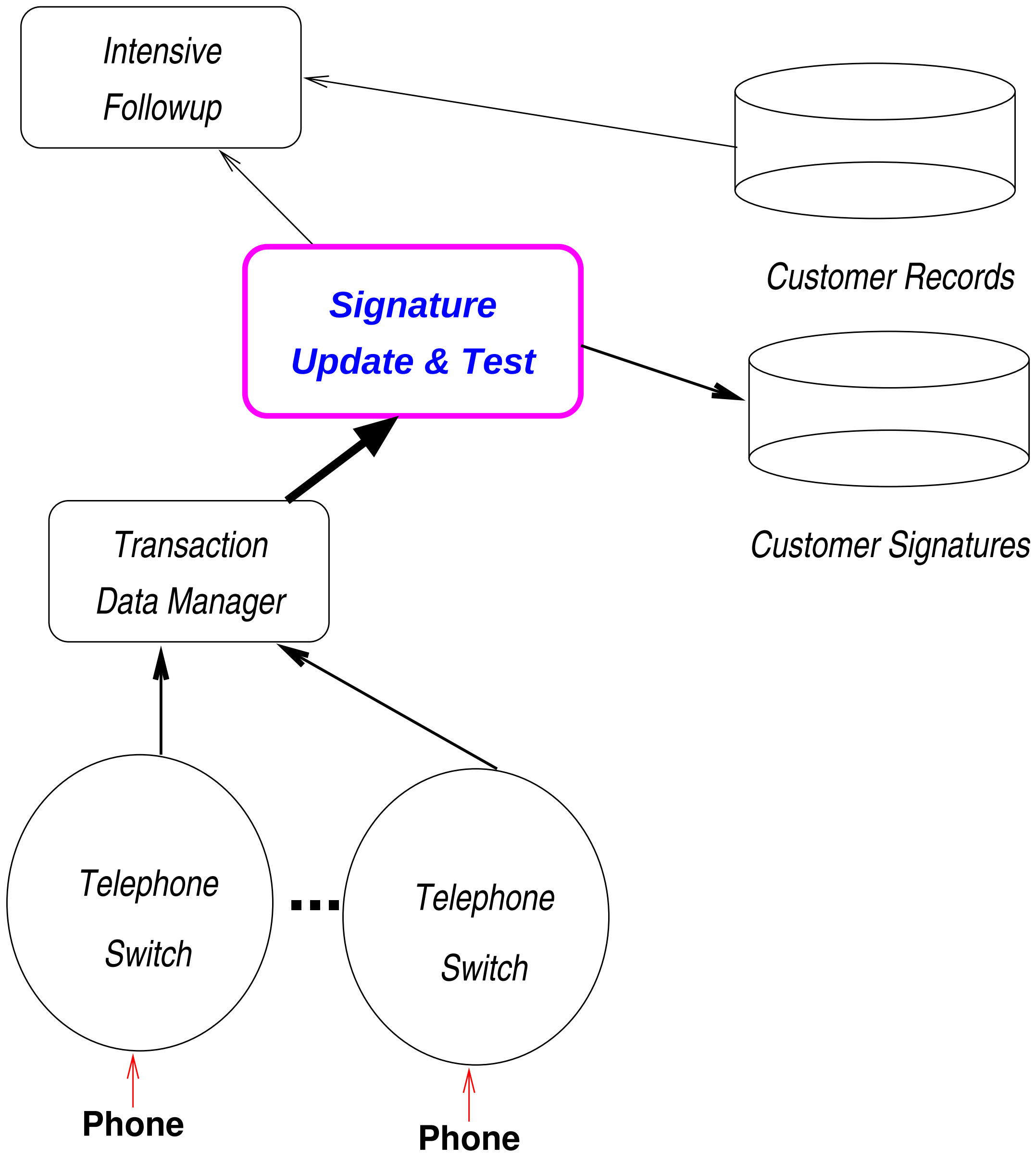
- Language- and Object-based computing has made a big difference. Consider

  ```
  lm(Fuel ~ Weight + Type, auto98)
  ```

- Interfaces are helpful, but more needs to be done.

- But the world has changed at least as fast as we have!

A Project: Telephone Call Data



Statistical Analysis for Fraud Indication

- build “signature” of caller’s behavior;
- *quickly* summarize new calls;
- results flow into the process (followup, databases);
- *not* mechanical decisions, but human followup.

What Makes for Success?

Good ideas and hard work, of course. Also:

- Beyond the classic role.
- The *process*, not just the data.
- Meeting computing challenges.

How could the computing environment have helped more?

A Challenge for the Statistics Profession

We can make a greater impact in our scientific and other interactions, but we must widen the *scope* of our involvement.

Lots of challenges for statistics generally, but what can computing do to help?

Computational Challenges

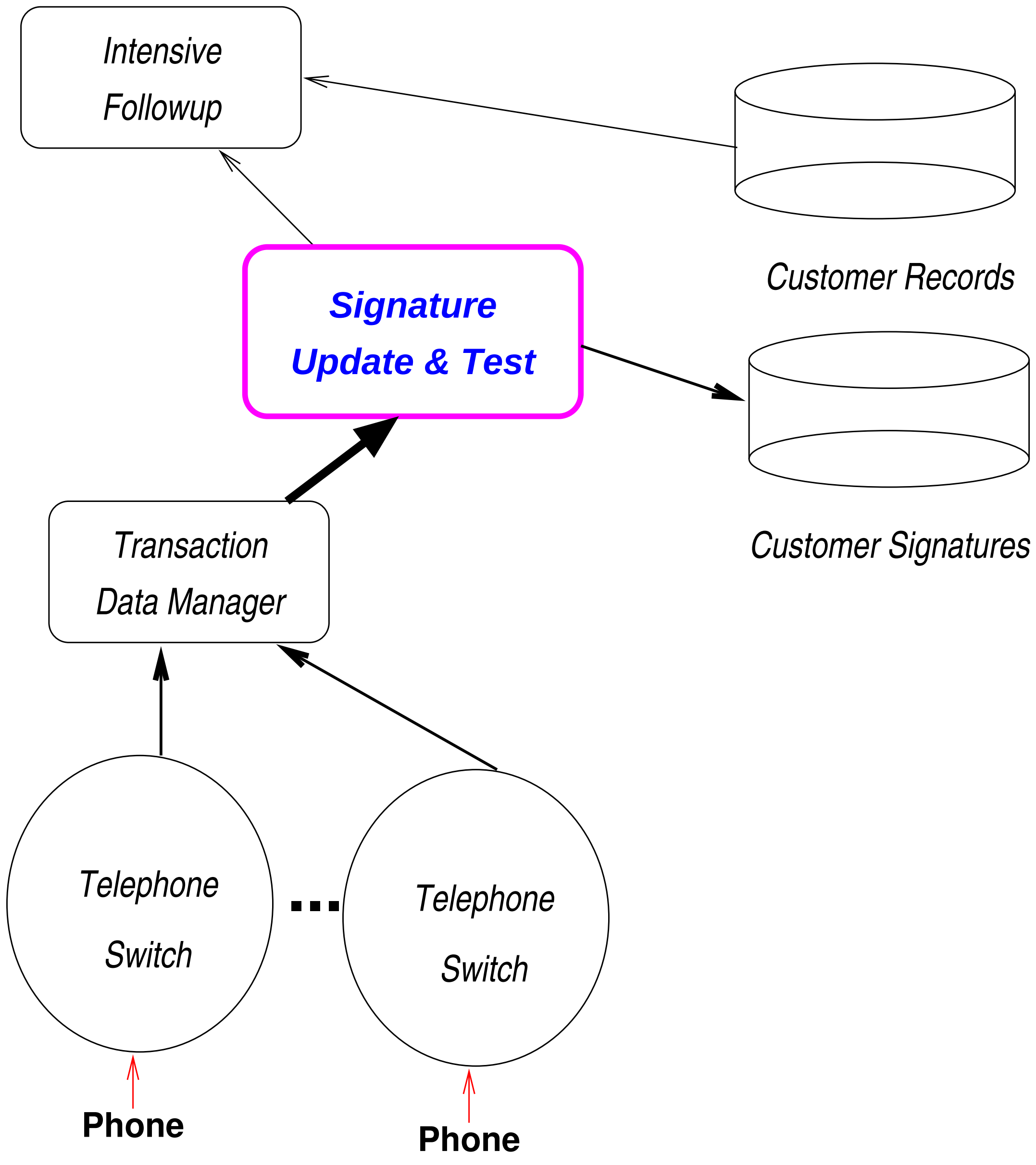
1. They are mostly *not* traditional statistical computing.

2. Data mangement, user interfaces, diversity are crucial.

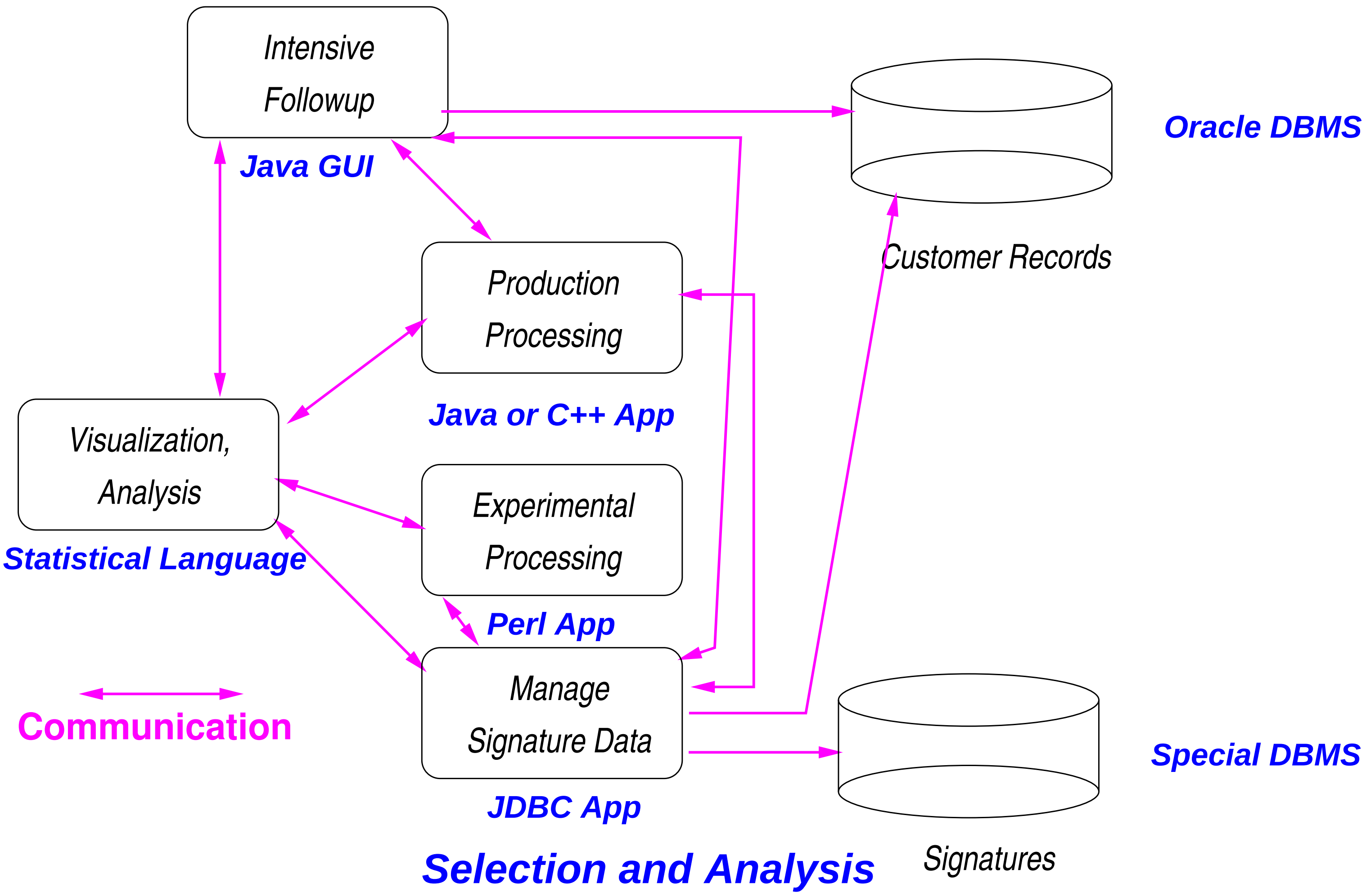
3. Skilled human time is the scarcest resource.

Let's imagine a fresh attack on examples like ours, with a "wish-list" for computing tools.

Telephone Call Data



Telephone Call Data



Distributed Computing with Data

Goals:

1. connect software for statistical, database, graphical interface, ...
2. the user can still have a single view;
3. systems that can grow and adapt.

New Programming Concepts

1. transparent communication of requests;
2. define capabilities across language & system;
3. program with synchronization, events, user interfaces;
4. dynamic, self-describing programming tools.

New languages and architectures can help (Java, Corba and others).

A New Project (Ω)

The goals include:

1. module definitions for what we want;
2. packages implementing the modules;
3. interactive language(s) for programming and end use.

The project will produce open source, and encourage broad participation.

Java and/or its virtual machine, plus tools such as Corba, are likely to be part of the technology.

See `www.omegahat.org` for where we are. Join in our mailing lists, *contribute* ideas and software if possible.

Summary

Concepts:

some deep, long-lasting ideas underly what we do in computing;

Challenges:

to expand our scope and diversify our tools;

and Fun:

exciting new directions are available to us.