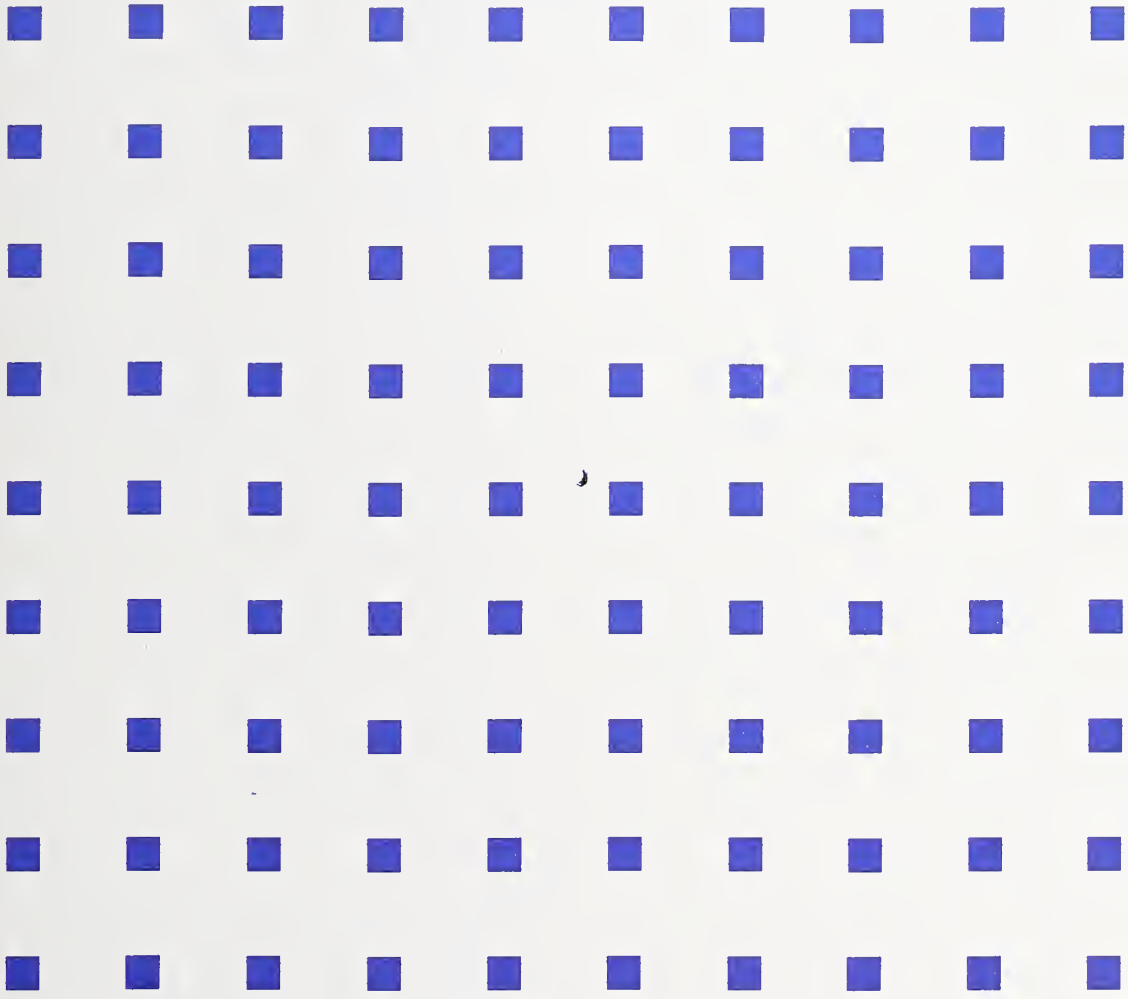
pecial Publication 500-237

Information Technology:

Usability Engineering: Industry-Government Collaboration for System Effectiveness and Efficiency

SYMPOSIUM TRANSCRIPTION

Laura L. Downey and
Sharon J. Laskowski, Editors



U.S. Department of Commerce
Technology Administration
National Institute of Standards and Technology

NIST

The National Institute of Standards and Technology was established in 1988 by Congress to "assist industry in the development of technology . . . needed to improve product quality, to modernize manufacturing processes, to ensure product reliability . . . and to facilitate rapid commercialization . . . of products based on new scientific discoveries."

NIST, originally founded as the National Bureau of Standards in 1901, works to strengthen U.S. industry's competitiveness; advance science and engineering; and improve public health, safety, and the environment. One of the agency's basic functions is to develop, maintain, and retain custody of the national standards of measurement, and provide the means and methods for comparing standards used in science, engineering, manufacturing, commerce, industry, and education with the standards adopted or recognized by the Federal Government.

As an agency of the U.S. Commerce Department's Technology Administration, NIST conducts basic and applied research in the physical sciences and engineering, and develops measurement techniques, test methods, standards, and related services. The Institute does generic and precompetitive work on new and advanced technologies. NIST's research facilities are located at Gaithersburg, MD 20899, and at Boulder, CO 80303. Major technical operating units and their principal activities are listed below. For more information contact the Publications and Program Inquiries Desk, 301-975-3058.

Office of the Director
- National Quality Program
- International and Academic Affairs

Technology Services
- Standards Services
- Technology Partnerships
- Measurement Services
- Technology Innovation
- Information Services

Advanced Technology Program
- Economic Assessment
- Information Technology and Applications
- Chemical and Biomedical Technology
- Materials and Manufacturing Technology
- Electronics and Photonics Technology

Manufacturing Extension Partnership Program
- Regional Programs
- National Programs
- Program Development

Electronics and Electrical Engineering Laboratory
- Microelectronics
- Law Enforcement Standards
- Electricity
- Semiconductor Electronics
- Electromagnetic Fields[1]
- Electromagnetic Technology[1]
- Optoelectronics[1]

Chemical Science and Technology Laboratory
- Biotechnology
- Physical and Chemical Properties[2]
- Analytical Chemistry
- Process Measurements
- Surface and Microanalysis Science

Physics Laboratory
- Electron and Optical Physics
- Atomic Physics
- Optical Technology
- Ionizing Radiation
- Time and Frequency[1]
- Quantum Physics[1]

Materials Science and Engineering Laboratory
- Intelligent Processing of Materials
- Ceramics
- Materials Reliability[1]
- Polymers
- Metallurgy
- NIST Center for Neutron Research

Manufacturing Engineering Laboratory
- Precision Engineering
- Automated Production Technology
- Intelligent Systems
- Fabrication Technology
- Manufacturing Systems Integration

Building and Fire Research Laboratory
- Structures
- Building Materials
- Building Environment
- Fire Safety Engineering
- Fire Science

Information Technology Laboratory
- Mathematical and Computational Sciences[2]
- Advanced Network Technologies
- Computer Security
- Information Access and User Interfaces
- High Performance Systems and Services
- Distributed Computing and Information Services
- Software Diagnostics and Conformance Testing

[1] At Boulder, CO 80303.
[2] Some elements at Boulder, CO.

NIST Special Publication 500-237

Information Technology:

SYMPOSIUM TRANSCRIPTION

Usability Engineering: Industry-Government Collaboration for System Effectiveness and Efficiency

held February 26, 1996, at the National Institute of Standards and Technology

Laura L. Downey and
Sharon J. Laskowski, Editors

Information Access and User Interfaces Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

July 1997

U.S. Department of Commerce
William M. Daley, Secretary

Technology Administration
Gary R. Bachula, Acting Under Secretary for Technology

National Institute of Standards and Technology
Robert E. Hebner, Acting Director

Reports on Information Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) stimulates U.S. economic growth and industrial competitiveness through technical leadership and collaborative research in critical infrastructure technology, including tests, test methods, reference data, and forward-looking standards, to advance the development and productive use of information technology. To overcome barriers to usability, scalability, interoperability, and security in information systems and networks, ITL programs focus on a broad range of networking, security, and advanced information technologies, as well as the mathematical, statistical and computational sciences. This Special Publication 500 series reports on ITL's research in tests and test methods for information technology, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology
Special Publication 500-237
Natl. Inst. Stand. Technol.
Spec. Publ. 500-237
261 pages (July 1997)
CODEN: NSPUE2

U.S. Government Printing Office
Washington: 1997

For sale by the Superintendent of Documents
U.S. Government Printing Office, Washington, DC 20402-9325

FOREWORD

On February 26, 1996, the National Institute of Standards and Technology (NIST) sponsored the symposium "Usability Engineering: Industry–Government Collaboration for System Effectiveness and Efficiency." The symposium brought together over 100 industry and government professionals to exchange information and strategies for achieving effectiveness, efficiency and satisfaction in computer–based government systems. It also provided a forum for raising awareness about usability engineering and its role in increasing productivity and decreasing costs.

The symposium was the first of its kind to address the need to incorporate usability engineering practices into the design and development of computer-based government systems. With the mandate and fiscal necessity of employing commercial, off-the-shelf (COTS) products with legacy data, the complexity of government systems is only increased. It is critical that government and industry implement the best practices now available to ensure usability. This is especially applicable to goverment systems because of the large number (and high cost) of custom computer systems and legacy system integration and updates.

This document provides a complete record of the workshop presentations in a conversational style based on the transcription of the symposium videotapes. Figures from the original proceedings were included and referenced to enhance the transcription of each speaker's presentation. For a brief overview of the symposium, including background information, attendee demographics, and feedback, see the symposium report in the SIGCHI Bulletin, October 1996, Volume 28, Number 4. Much positive feedback was received and a second symposium is planned for March 3, 1997.

As a note of information, this version (August 1997) of the transcription supersedes the preliminary version dated February 27, 1997. Also, any mention of specific products in either version of the symposium transcription is for informational purposes only and does not constitute any recommendation and/or endorsement by NIST.

Editors:
Laura L. Downey, NIST (Symposium Chair)
Dr. Sharon J. Laskowski, NIST (Program Committee)

(Any errors in the symposium record are a result of the transcription and editing done at NIST.)

TABLE OF CONTENTS

TABLE OF CONTENTS (CON'T)

AUTHOR INDEX

AUTHOR INDEX (CON'T)

EXECUTIVE OVERVIEW OF USABILITY ENGINEERING

USABILITY ENGINEERING: INDUSTRY-GOVERNMENT COLLABORATION FOR SYSTEM EFFECTIVENESS AND EFFICIENCY

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY SYMPOSIUM

10 January 1996

H. Rex Hartson, Ph.D.

Virginia Tech

Department of Computer Science

Blacksburg VA 24061

Internet: hartson@vt.edu

540/231-4857

Dr. H. Rex Hartson,
Professor of Computer Science, Virginia Tech
Executive Overview of Usability Engineering

Hartson: I'm really happy to be here. I see so many people that I know and have worked with in different ways. It is always enjoyable talking with people who want to talk more and listen about usability and have various discussions about it. Always something good comes out of that. Besides my current post at Virginia Tech, I also bring a background of engineering from industry. That's an important aspect of what we're talking about here in that usability engineering implies an engineering approach which does not mean blue star research, but time tested techniques that have a proven track record.

Usability engineering is a way to get usability into products and it's also a way to know usability has been achieved. This means being cost conscious. It involves ways to make it [products] good enough and not necessarily achieve perfection. There are a lot of different approaches and a lot of different activities and techniques in usability engineering as it exists out there today. In this overview we'll take little snippets of things that are typical—that you find out when you look into the ordinary usability engineering group and look at usability laboratories in various places in operation already. In a way these are little snippets from the book that Debby Hix and I wrote on developing user interfaces which we use in teaching short courses that last much longer than the 45 minutes I have here.

These days I think there is less and less need to motivate the need for usability engineering, but there is still a real need. A recent survey—and I don't know whether you know this, but this symposium really came out of a CHI '95 workshop—and at that workshop we talked about a survey that revealed that 94% of Federal employees use computers on the job. (Figure 1)

MOTIVATION FOR IMPROVING THE USER INTERFACE

- The need for usability
  - 94% of Federal workers use computers on the job
  - Half of those are new users (beginners)!
- The problem: hard-to-use, hard-to-understand user interfaces
- Computation vs. communication
- Functionality vs. usability

Figure 1

Almost everyone does and that in itself is not a surprising number. But, that same survey showed *that over half of those Federal employees were new at using at least one of the systems they used and so usability turns out to be a really important aspect of it*, but there's a problem. Many of those systems that Federal employees are using are hard to use, hard to understand. In other words, a lot of attention might have been paid to the software design for the computation, but were the users able to communicate with the software? That is the real question. The functionality might have been just great, but it's the usability that really gives you access to the functionality.

Many people think of usability, especially software engineering people who are used to doing things in formal ways, tend to think of usability as a kind of fluffy, vague kind of a thing, but in fact it really is very tangible. (Figure 2) It can be quantified. It can be measured. Usability is concerned with ease of understandability, speed of task performance, error rates during task performance, and even user satisfaction. These can all be measured and in fact that's what usability engineering is all about. More recently (and I really approve of this trend in usability) usability in the large also includes usefulness. That means not just talking about the user interface, but being concerned with the system as a whole—that is how much utility it brings to the user. It's easy to make a dumb-down system look simple, but then it really is not as useful as it could be. So we're really not talking about just usability. The new ISO standard defines usability to include efficiency, user satisfaction and effectiveness. Those of you who have worked with me know that I can't do anything without some cartoons. The idea of thinking of usability as user friendliness or just a pretty design is the kind of thing you might do if you were Dilbert. Dilbert says, "... but thanks to my leadership the new air traffic control system was designed on time and under budget. Of course I had to cut a few corners, the big radar looking thing is a wall clock, and most of the buttons are glued on." Someone comments, "Well, it looks like it might be dangerous." Dilbert responds, "Great. I finish early and what do I get? Feature creep!"

WHAT IS USABILITY?

- Usability can be quantified and tangible
  - * Ease of understandability
  - * Speed of user task performance
  - * User error rate
  - * Retention over time
  - * User satisfaction
- Usability-in-the-large includes usefulness

Figure 2

Usability would not have met so much resistance if people would realize that what usability really is about is meeting organizational goals. Now most organizations, whether they are Government organizations or otherwise, have goals that look something like this. (Figure 3) Increased user and organization productivity, both individuals and organizations want to be productive. Decreased user training costs, big budget for training and decreased errors. No one would argue with those goals. Yet for most of those 94% of the Federal workers who work with these new systems, the usable interface on those systems is the thing that will have a major impact on their job including meeting these productivity goals on the job. One mistake seen in some government institutions (especially the military) is that training is used as a substitute for usability and often with little success. It's easier to add on the end yet in fact it's a very costly substitute and it's one where the cost keeps on going. A much better alternative is to build in usability in the first place and that comes from usability engineering.

SUPPORTING ORGANIZATIONAL GOALS

- Goals of most organizations include
  - * Increased user and organization productivity
  - * Decreased user training costs
  - * Decreased user errors
- A ***usable interface*** has a major impact on achieving these goals
- Training is a costly substitute for usability
- Ensuring a usable interface comes from ***usablity engineering***

Figure 3

The evolution of good design requires a kind of attention to a couple of different things. (Figure 4) One is the product and one is the process, not really a new idea. The product in this case is the interface itself. We're focusing on that part of the system—the contents, the human factors, the guidelines, look and feel, etc. The process is how to develop that product. A significant cause of poor usability in the product stems from a lack of understanding of the development process. Let's be clear by what is meant by development because it is a loaded word that means different things to different people. Development in this context means user interface development, but broken down into two sub-components. (Figure 5)

One is the development of the user interaction component which is the idea of the user interface design, but independent of the software that would be used to implement it. The second sub-component is the the development of the user interface software which is a separate idea. These things help look at interface behavior. There is a task view and the user's view and then there is

the design of the software that extenuates the user interface design. The reason for separation is because the processes required by interaction development are separate and quite fundamentally

ENSURING USABILITY

- Evolution of a good design requires focus on at least two key elements:
  - * ***Product*** — the interface itself; content, human factors, look and feel, guidelines
  - * ***Process*** — techniques and methods for interface development
- A significant cause of poor usability in *product* is lack of understanding of proper development *process*

COMPONENTS IN THE PROCESS

Figure 4

USER INTERFACE DEVELOPMENT

Development of ***user interaction*** component	Development of ***user interface software*** component
* Interface behavior * View of user & interaction developer	* Code instantiating interaction * View of system & programmer

- User interaction development is separate from, and fundamentally different from, user interface software development
- Different skills and abilities
- ISSUE: Interaction design is an iteratively evolving design spec for the interface software

Figure 5

different from the kind of things used to develop software. Software development processes are fairly well known and understood and they've been around a lot longer than usability engineering has. It's hard to catch onto the idea that quite different skills and abilities are needed to develop the interaction part. In any case what's evolving out of this is that the two are fitting together nicely in an overall life cycle and development process and that the design that one comes up with for the user interaction now is starting to be thought of as the kind of design spec for the user interface software. The development activities for this part of the picture are what is addressed in usability engineering—rapid prototyping, formative evaluation and usability, etc. All of this should be done before starting work on the software or the interface and that's what prototyping helps us do. (Figure 6)

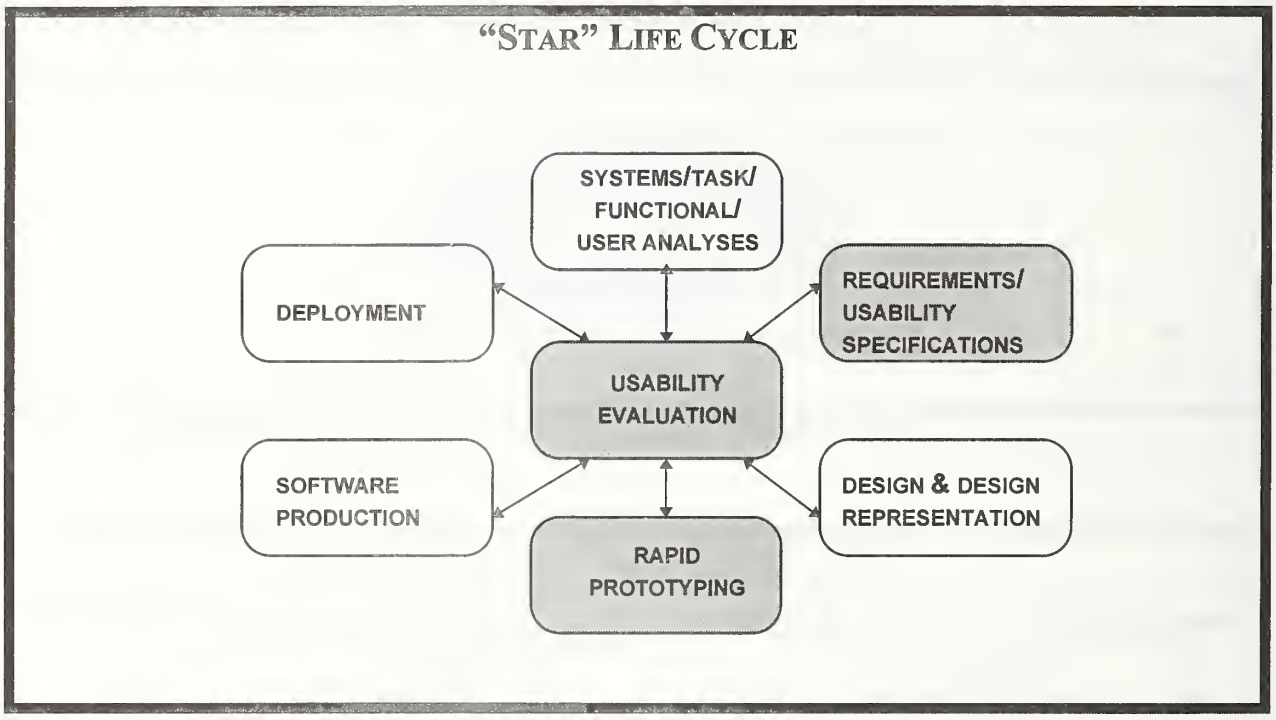


Figure 6

Figure 6 is a picture of the "Star Life Cycle" which Debby Hix and I talk about in our book. The idea for the life cycle was born during a consulting job that Debby Hix and I did some years ago. In the late 80's while consulting for a large three–letter vendor to be unnamed, we were studying the way that people develop systems, and thought this would help us somehow integrate usability issues into the bigger development process. We noticed that this was a shop where a top down waterfall method was mandated, but we also noticed that after people became comfortable with us and through informal interaction, that they did their development a lot of different ways—top down, bottom up, inside out, etc. But, when it came time to report what they did of course they reported that they did the whole thing exactly top down because that was

exactly what they were supposed to do. The main point that we learned from this experience was that development of the interface really couldn't successfully be done with an exclusively top down kind of a process. So we developed this idea of a life cycle that looks more like a star where the points indicate that one can start anywhere.

It is not necessary or practical to do one thing to completion before going to the next "phase." Phases don't exist in the same sense as they do in the waterfall method. Managers like the waterfall method because it provides an exact completion date for a phase. That date arrives so the phase is done. When that phased is signed off, all the problems are thrown over the wall to the next phase. The serious problems may start escalating out of control.

EVALUATION–CENTERED
USER INTERACTION LIFE CYCLE

- Star life cycle is for interaction design, not software
- Star life cycle is support for
  - * *Formative evaluation* — empirical testing with users as design evolves
  - * *Iterative refinement* — design, evaluate, modify
- *Rapid prototyping* is vehicle for formative evaluation

Figure 7

As stated before, the star life cycle is a life cycle for the interaction design. (Figure 7) This is not a software life cycle. It involves formative evaluation and rapid prototyping as the vehicle that provides life to the cycle. Iterative refinement is important to the design which is the thing that really leads to the first glimpse of the product. The first key issue is the product or in other words the user interaction component. (Figure 8)

There are guidelines and principles out there for the user interaction component. It turns out that for many years these things have been in the literature and people haven't paid too much attention to them. There's a nice set of guidelines for text based interfaces in the Smith and Mosier report, but that's a very old report and I don't expect it to be applied to the graphical user interface (GUI) world. For GUI's we now have style guides that in a sense have taken over, but the style guides are not enough. Use or application of a style guide does not guarantee usability. There is not as much attention in those style guides as there were in the old guidelines. Those principles and guidelines get at fundamental design issues that in fact would be design issues in almost any kind of interface, such as consistency, real world metaphors, making sure that we at least accommodate what we know so far about human cognitive limits like short term memory limits, how to compose messages, how to use color (which we really don't know too much about yet), screening layouts and that sort of thing. Additionally, these style guides contain certain general guidelines about how to use menus and buttons and other kinds of widgets.

That was the product now the process is something that we're still working on even though we already have quite a bit that's well established. (Figure 9) That's really the major emphasis here. There is a lot of research aimed at developing new techniques and methods, but because well-established techniques exist, it's not pie in the sky research. It's a process that's been proven to work. It's an iterative user-centered design done early with continual involvement of users and user representatives and lots of up front analysis—user analysis defining what your user classes

A Key Issue in Usability

- ***Developing the product*** — the user interaction component
- Well-established ***guidelines, principles*** exist to help *produce a more usable interface*
  - Consistency
  - Real–world metaphors
  - Short–term memory limits
  - Message composition
  - Use of color
  - Screen layout
  - User locus of control
  - Designing for user erros
  - Cognitive and physical affordance
  - Task thread continuity, etc...
- Well–established rules for use of ***interaction styles*** also exist
  - Menus
  - Buttons
  - Fields in a form, etc...

Figure 8

are, task analysis, functional analysis, usability specifications (which are quantitative usability goals that you can then test to find out whether you have met them), and of course rapid prototyping to support it all. Formative evaluation is the final step where you get to see the results of your labor. But, this usability engineering is not just user testing. A lot of people think of usability engineering as just what is done in the usability lab, but it's all the things done to build up to that point which enable people to take advantage of the process.

Well, what are these usability specifications? (Figure 10) I stole a line from the software engineering world to build my slides. *If you can't measure it you can't manage it,* and a lot of

what we're talking about here really is management. It's managing the process, and it's also managing the process as usual within budget and within schedule constraints. Usability specifications are very practical items. They are operationally defined. They are defined in terms of the real tasks that users perform—the representative tasks. These are called benchmark tasks when used to form base comparative measurements, i.e., user task performance time or error rates. Two important tasks to look at are representative tasks (examples of what users really do),

ANOTHER KEY ISSUE IN USABILITY

- ***Understanding the user interaction development process*** — the techniques and methods for producing a *usable* interface
- Well-established methods exist for user interaction development
  - * Iterative, user-centered design
  - * Very different from "waterfall" model
  - * Early and continual involvement of representative users (and users representatives) in the interaction development process
  - * User/task/functional analyses
  - * Usability specifications (metrics)
  - * Rapid prototyping of interaction design
  - * Formative evaluation of interaction design
- It's a complete process, not just user testing

Figure 9

and of course mission critical tasks. Mission critical tasks may not be representative in the sense that they are actually carried out, but when they are, it is important that usability supports the user when performing that task. We can measure objective measures like user performance, the time

QUANTIFYING USABILITY

- If you can't measure it, you can't manage it!
- Usability specifications — operationally defined, measurable goals, used as criteria for effective and efficient user interface
  - * Based on:
    - benchmark tasks
    - representative tasks
    - mission critical tasks
  - * User performance (e.g., time, errors)
  - * User satisfaction (e.g., questionnaire)

Figure 10

and errors, and we can measure subjective measures such as user satisfaction via questionnaires and the like. These usability specifications are then applied in the formative evaluation—testing in a usability lab with representative users or in other words, user based assessment of usability. (Figure 11) Testers tend to get as many different kinds of people as possible, but it is important to get representative users because they provide an awful lot of information. Also, if it's feasible, bring in the occasional HCI expert. Many of the organizations represented here today already have people who are usability specialists. Bring them in and have them help with the evaluation. It is possible to do your own evaluation in various settings, but the most comfortable place to do it is in a usability lab if available. What's the advantage of the usability lab in doing this kind of testing? Anybody? Control, right. OK, and what's the disadvantage? Control, right. One advantage of having the control is that all of the tasks are set up, the users are isolated and nothing interferes. However, that kind of control doesn't make it a very realistic environment for the task performance and so sometimes it's nice to get the measurements out in the field in the users' context.

FORMATIVE EVALUATION

- Evaluation - early and continual
- User-based assessment of usability
  - * Representative users
  - * HCI experts
- Various settings
  - * In usability laboratory
  - * In the field
- Two kinds of data
  - * Quantitative
  - * Qualitative
    - – Critical incident taking
    - – Verbal protocol taking

Figure 11

Testing produces two kinds of data—quantitative data (which we have already looked at) and the qualitative data which is very important in determining what the usability problems are and why there are problems. The qualitative data tend to tell us things about why users have trouble, where they really have trouble, whereas the quantitative data tells us that there is a problem and that we need to fix something in order to meet our usability goals. Two of the more common sources of qualitative data are critical incident taking and verbal protocol taking. Critical incident taking is simply watching for events or occurrences during the user's task performance that imply or indicate some kind of impact on usability. It might be good or it might be bad, but a place where they have trouble, hit a snag, or a place where they get held up—anything like that—might be considered a critical incident. And, usually underlying a critical incident is some kind of a usability problem. Also verbal protocol or the encouragement of talking out loud is very helpful to know what's going on in the mind of the user when they are just sitting there not knowing what to do. We want to know what they are thinking about it. Of course audio taping helps support the post talk gathering and analysis of this kind of data.

Analyze Data

- Determine if usability specifications are met
- ***If not:*** identify interface problems and solve in order of cost and effect on usability
- Cost/importance analysis — to determine which design changes will have greatest impact on usability
- ***Impact analysis*** — determining which problems most affect usability
- ***Cost/importance analysis*** — determining which problems, when solved, will give biggest improvement to usability for least cost
- ***Engineering:*** achievement of specifications, not perfection
- Provides essential management control for iterative process

Figure 12

The first thing to do after seeing the quantitative results is to decide if the usability specs were met. At that point, if they are it's usually good engineering judgment to declare a success for this version and go forward with planning the next one—even though lots of usability problems may still exist. Again it's an engineering activity to identify the interface problems to be solved. The important thing is to try and get them ordered or ranked in some way by both cost and severity or importance. There are ways to assign both cost and importance. The idea is to solve the ones that are the most important and the least costly first to get the best bang for the buck. This is critical because almost always resources run out before all the problems are solved. So again this is an engineering pursuit. We want to achieve the specifications. Sometimes resources are not even available to do that, but as Astrid Nielsen is fond of saying, "whatever you do is better than doing nothing." Remember we are not looking for perfection but minimization of problems. This point is about management control and is really important because in any of these iterative processes, the star life cycle and all of the iterative processes for this part of the development, to a manager mean what? Iteration means going around in circles, right? Managers don't like to talk about processes going around in circles. They don't have that phase to sign off on. They don't have a delivery date that they can say it's done.

So what we need now is another way to manage process, another criterion for when we can stop iterating. During the late 80's the Bureau of Land Management (BLM) was in the planning stages for a very large geographical information system. At that time they were talking half a billion dollar budget and ten calendar years for the schedule. It was a system that would be used by lots of people in the US Forest Service and the Bureau of Land Management across the whole country to manage natural resources in lots of different ways. We got a call from a man called David Rupert. I don't know if he's by any chance here in the audience. . . I lost track of David a

long time ago, but this was back in 1989 and I wrote him down here as a real usability hero because basically he was the one lone voice crying in the wilderness in BLM. (Figure 13)

A Usability Success Story

- The Bureau of Land Management (BLM)
- Very large geographical information system
- David Ruppert - real usability hero at BLM,
- c. 1989
  - * Fought for usability when it was unpopular
  - * Tired of old way of doing business
  - * Wanted to head off situation by including RFP requirements on the interface development *process*
- The outcome?

Figure 13

He was in charge of the user interface, but he took that as a serious mandate to also be in charge of usability, which is not really what his bosses had in mind at the time. No one else there, mainly software developers, really had an appreciation of what usability was or what it's importance was. Yet they were planning this enormous investment into this system that would be used by all kinds of users over a long period of time. So it was a huge investment. By the time we got there to help with consulting, they had already identified 33 different user classes and actually had defined classes from managers to people in the field. It is easy to imagine the different kinds of usability requirements that might come out of that. Well, he was fighting for usability at a time when it wasn't so popular, but he was tired of the old way of doing business. The whole project was getting to the point where we were writing RFP's and we were going to go out on the bricks and get people to respond and to send in bids. He was very concerned that the standard scenario was going to be played over yet one more time—that is somebody's going to win the bid and they are going to promise some usability, but they don't know what it means and neither do the people that are paying the bills. So they will go off and two years later come back with a system and it will not be usable.

So what happens now? Well, there's only one thing really to do and that is to invest more. They are the only people now who are capable of fixing it. The project gets into this kind of iteration, but it's an unplanned iteration that goes on until either time or resources run out. So what he wanted to do was to head off that situation by including something up front in the RFP that forced the hand of the people who were writing proposals to prove that they knew something about usability. We didn't want to build in any requirement for a specific usability technique because we knew that over the next ten years things would change in the usability area. However, we wanted (at least) the RFP to force people bidding on this to prove that they had in

place some sort of usability methods and would therefore know a little bit about actually producing this usability that was so easy to promise back in those days. It turned out to be a pretty good success in that at least that did get in the RFP and the vendors were very responsive. The vendors that did know about usability were relieved to see that, because they were tired of competing against other vendors who didn't know about it, and, of course, on a cost benefit basis, it's difficult to prove the benefit. Basically, they are talking about providing more and competing with people who are providing less, and it's difficult to do that cost effectively.

COST JUSTIFICATION OF USABILITY

Can we afford to include usability engineering in our system development process?

- The answer is that usability engineering does *not* add *overall* cost, for two reasons:
  - Usability doesn't add as much development cost as many people think
  - Usability saves many other costs
- Added costs is confined
  - Reality: Interaction development process must be iterative — *cannot* get it right the first time
  - But interaction development is *small* part of overall system development
- Rest of development — user interface software and other application software — is not necessarily iterative

Figure 14

People often ask me how did that all work out, and the answer is I don't really know entirely, because we lost track of those people. It was a consulting job and we didn't stay on it for a long time. But, interestingly enough, a few months ago I got a call at Virginia Tech from somebody in the Forestry Department. They said we just got a contract to develop an expert system for (I think) landscape architecture for making things look pretty after strip mining, cleaning up after some kind of a intrusion on the environment. They said we know how to do the things involved in getting production rules for landscape architecture and the forestry part, but there's this thing in here about a user interface and I said, "Gee, did that come from the BLM?" and they said, "Yeah, that's it." It turns out that it was the same thing which still had life—coming back at me many years after we were first involved. I would have to say that was a success. Somehow it worked, and I think it will be true more and more for vendors or contractors or those people on the government side who deal with contracting.

More and more contracts will be awarded based on whether people can demonstrate awareness and demonstrate a knowledge of usability engineering. Some contracts will be denied on that same basis. A recent *Roanoke Times* article (week of February 19, 1996) was about a system

being developed in the State of Virginia. It didn't give a lot of technical details, because it was a short newspaper article, but it was about a system that was being developed for the State of Virginia to streamline welfare applications. Apparently, the system was such a bust that the Virginia State Police are investigating the system developers. In fact, they were even looking into the possibility of criminal charges. Apparently, the system was that bad. The heading of the article in the *Roanoke Times* cited the main problem as, "usage was awkward, inefficient and time consuming." It is obvious that people were very concerned about paying money to develop that system and then receiving a system that was unusable. That's a tell-tale story of the times.

A lot of people ask about cost-justifying usability. Clare-Marie Karat will give details on this later, but let's look at it from a general perspective. (Figure 14) Some people may ask, "Can we afford to include usability engineering in our system development process?" The real question is, "Can we afford not to?" That may be trite, but it is true. I think the most important thing we can say about cost with regard to usability engineering is that it does not add to the overall cost. There are two reasons why. First, usability doesn't add as much to development costs as many people think, because it only really affects a small part of the overall development life cycle. There are a lot of other things that have to do with developing interactive systems that don't have to do with this. Secondly, usability saves many other costs, and much more than put into it by including usability.

COST JUSTIFICATION OF USABILITY

- Poor usability is costly; good usability is all *about saving cost*
  - * Costs of hardware and software vs. costs of "personware"
  - * Costs of development vs. costs of operation
    - – Development costs are mostly one time, while operational costs accrue for years
    - – Cost/benefit scope must be broad enough to include usage, as well as training, help desk support, etc.

Figure 15

So let's look at the first reason, essentially that the usability engineering cost is really confined to a small part of the development process. It is true that the interaction development process must be iterative. We cannot get it right the first time. We never will be able to because we're dealing with an infinite number of possibilities. We are working with human beings, and their behavior and their performance is not really predictable. However, the interaction development is a very small part of the overall system development. I like to think of it as sort of onion skin shells of development processes in the big overall life cycle. We have the interaction development together with the user interface software development which has it's own little life cycle. Then we have the overall software life cycle and the even bigger system development life cycles, systems engineering and all kinds of other things that people worry about including the hardware aspects. Usability engineering activities do not affect all these different parts of development,

just the interaction development. That's what this point is, and, of course, again in order to make that pay off work, you really have to work with prototypes. If we wait until the whole thing is put into software (as a number of groups still do), we don't manage to get that leverage. (Figure 15)

The second reason is about cost savings. I make the claim that poor usability itself is costly, and can we afford to have that? Good usability is all about saving costs. In the old days we worried mostly about the cost of hardware; then software became the main cost of producing systems, but now we have to look at the cost of personware and that's really the cost of usage and other people issues associated with the system. Think about the cost of development versus the cost of operation. Development costs accrue mostly at one time. If the system has a long life after that, operational costs, which of course include maintenance, accrue on a continuing basis. The problem with cost benefit analysis in this scenario is that most people don't pass the scope broadly enough to consider all of the costs and all of the benefits. In fact it's one of those areas (since we are a kind of enabling technology) where in fact the cost and the benefit can be quite far apart from each other. We have to think more globally if we are going to do useful cost benefit analysis. Think about usage and training. One of the problems in the scope problem is that very often one group pays for the development and another group pays for the usage, but in fact the government has one advantage over the rest of the world in that users are also the government's employees. Government software systems are not generally produced for the marketplace which places government in a position to make back some of the costs. (Figure 16)

COST JUSTIFICATION OF USABILITY

- What costs the most:
  - * Finding a design problem early or late in development process?
  - * User-based tasks that are quick, efficient, and accurate, or error-prone tasks that take more time?
  - * Confused users, or confident and competent users?

Figure 16

Here's the question. "Which of these things costs the most: finding a design problem early or late in the development process?" We've heard that one many times. The traditional rule, rule of ten, is if a problem is found that costs a dollar while still in the design phase, it might cost ten dollars after code is written for it, and it might cost a hundred dollars if found after it's been deployed in the field. Do we want user-based tasks that are quick, efficient and accurate or do we want user tasks that are going to take more time? That's really what we're talking about when including usability—that kind of trade off. Do we want confused users or do we want confident and competent users? That's the question of which costs the most. (Figure 17)

Cost Justification Of Usability

- Development savings from usability in process
  - * Save high software maintenance costs — the cost of trying to get it right after release
  - * Much is gained that can be re-used
  - * Can even save implementation costs
- Costs sidestepped in development often show up as higher costs to be paid in system usage
- Usage savings; even more significant if users are your employees!
  - * Save operational productivity costs
  - * Save user training costs
  - * Save costs of user errors
  - * Save costs of database errors
  - * Save costs of help desk and user support operations
  - * Save intangible costs of employee dissatisfaction

Figure 17

Usability gives added value to a product every day that it's used. First of all, we can actually get development cost savings from having usability in the process, especially if we include maintenance as part of the development cycle. High maintenance cost often results because of usability problems. We may think of them as sort of software problems, because the system has already gotten to the software stage by that point, but they really are usability problems that could have been prevented before this point.

At Bell Telephone, the use of a prototype with real users revealed that one large component of the functionality that was planned for the system was not wanted or needed by the users. This allowed them to eliminate further design and all of the associated software production. They would not have learned this if they hadn't done some prototyping with real users first. Cost, side–step and development often show up as higher costs to be paid later in system usage. This is especially true in the government. These usage savings are more important if our own employees are the ones who are the users. Usage savings include operational productivity costs, user training costs, cost of user errors and data base errors. In the government, there are a number of very large systems with very large data bases and lots of transactions going on in the data base area. Think about the cost of an error that slips through and later on it's discovered and has to be fixed. Think also about cost at the help desk, user support operations and even the intangible cost of user satisfaction or dissatisfaction. (Figure 18)

COST JUSTIFICATION: A SIMPLE EXAMPLE

- For a -large distributed system:

Users: 75,000

Average transactions/user a day: 20

Transactions/day: 1,500,000

User time per transaction: 5 - 20 minutes

Average time saved per transaction, due to improved usability: 30 seconds

Average fully-loaded hourly rate: $25.00

- Saved per year

= 75,000 users * 20 trans/user-day * .5 min/trans

230 days/yr * $25/hr * I hr/60 mins

= $71,875,000.00

- Other savings: user training, help desk
- Regardless of what usability engineering cost for this product, payback is enormous

Figure 18

Let me give you a simple hypothetical example from one of the places that we do short courses, the Social Security Administration (SSA). We have a number of people here from SSA and we're going to get a talk from Pat Stoos from the Social Security Administration later on. We just made up some of these numbers. They are actually probably conservative compared to some of the things seen in one of these large systems. We begin with 75,000 users spread all over the country and maybe let's just say only 20 transactions per user per day. That means about one and a half million transactions, and let's suppose that the time that it takes to do these transactions may vary from five to 20 minutes each involving, for example, getting user client records into the data base system. Suppose that we can only save due to improved usability and again I'm being conservative because we can only knock 30 seconds off that transaction on the average. If we take a rather low number like 25 dollars or a fully loaded hourly rate and if we multiply all that out per year, all those people and all those transactions per person and so on, we get a number that's kind of interesting, 71 million plus and that's just for this one little case, this one little 30 seconds we shaved off due to improved usability. We can understand from this at least on large systems that have lots of users that we really are talking about real economics here. Whatever the cost of usability engineering it's not going to even approach that figure that's for sure. But as Dennis Wixon said recently in one of his E-mail messages on the U Test mail server, wouldn't it be nice if we no longer had to even talk about justifying the costs of usability? (Figure 19)

COST JUSTIFICATION OF USABILITY

- But won't it be nice when we no longer have to justify "costs" of usability"?
- When have you heard anyone ask: Can we afford costs of designing data structures, implementing algorithms, doing quality assurance, etc.... ?! [Wixon, UTEST communication, 1991]

Figure 19

He asked in this mail message when have we heard anybody ask can we afford the cost of designing data structures or implementing algorithms or doing quality assurance or all the other things that we now have rolled as a sort of normal way of doing business. If we think back over the history of system development, people had to fight for all these ideas. People had to fight for software engineering for structuring programs as part of being competitive.

On a recent plane ride, I came across an isue of *Business Week.* It had on the front cover, "productivity to the rescue;" technology is transforming American economy into the most productive in the world and the result—higher living standards inevitable. The whole issue was devoted to why we are going to basically regain our place in the world marketplace and why things will be great from an economic view. I like that idea, but I am also leery about the idea of just throwing technology at problems, because I think a lot of us in the field know that that doesn't always solve the problems. The single most important factor on getting this promised productivity out of all that technology really is usability and that's where usability engineering comes in. (Figure 20)

So, I'm going to close with this one summary statement. Remember this from this morning's talk if you remember nothing else. Users will evaluate your interface that much is sure. If we can do it in an orderly and structured way so that we can take advantage of the results of the evaluation to make the system better, then that only makes good cost sense. Thank you.

Questions?

Female: You mentioned that we don't know that much about color.

Hartson: OK, the question refers to the point I made about our not knowing all that much about the use of color in interfaces and the question is sort of what do we know and what do we not know. Well there are a number of guidelines out there that most designers are aware of not to use light blue on a semi-light blue background for text, for example, because it's difficult for humans to perceive blue. It's one of the colors that our rods and cones are not very well tuned to. A very small percentage of our color sensors are tuned to a pure blue, but of course there are things about color contrasts that have the same relation to the idea of contrast in a black and white design. It's the idea of not trying to read a stop sign with the sun behind it because it makes your eyes drop down so you can't read the sign and also there are rules about not using garish, bright day glow colors because although it might be fun at the beginning somebody sitting there for eight hours working in front of the terminal would certainly have eye fatigue

before the day is out. So there are some simple guidelines like that, but in fact we know really very little about the cognition and human perception of colors, especially of color and shape and other graphical devices like that for encoding data, using color as a way to convey meaning for example. Of course there are also problems in the fact that there are fairly large individual differences in the perception of color, especially among people who might have color deficiency in their vision. When we were walking up the hall we were looking for the auditorium, and I don't remember I guess it was Bill Hefley who was saying it's either in the green or the red and we were trying to figure out whether green or red came after blue or before orange, that sort of thing. It's difficult to encode meaning like ordinal meaning, which is what we needed to find the auditorium in a name of a color. There is one kind of problem.

SUMMARY

Users will evaluate an interface sooner or later . . .

Let's have them do it sooner, so that results of evaluation can help ensure usability of the interface.

Female: There's part eight of Article 9241 which is about color and the forthcoming ANSI standard on software user interfaces will have a section on color. That will be out in about a year or two.

Question: Why do we call it usability engineering, and what is its relationship with software engineering?

Hartson: OK. Well that's a very good question actually. The question was why do we call it Usability Engineering and what is the relationship with software engineering. The answer to the second one is about another half hour to an hour talk that we normally give in the short course,

but I'll try. Let me start with the first one. I think it's a good term because, and this is just my own personal opinion, the way I saw human computer interaction developing ten or fifteen years ago was first as a research topic which is not surprising and I saw a lot of people on the cognitive psychology side thinking that it's sort of like the seven blind men and the elephant thing. They were looking at human computer interaction from a cognitive view and thinking that's what it was about and then the software engineering people were looking at it from a software tools and user interface management systems and thinking about it just a way to produce code for the user interface. In fact neither the software people understood the human aspect nor the psychologists really understood anything about the constraints and problems in the software world which were significant. I think usability engineering is a term sort of helping to bring those two worlds together on a practical basis and de-emphasize the research which is still going on and will help us later on, but it's not what's going to help us immediately. The relationship with software engineering, well without some overheads to show that, basically I see some parallel embedded life cycles and the interaction development process is one that lives inside other bigger development life cycles and if you draw out the software engineering life cycle you go through systems analysis and domain design and software design, software momentation, casting and that sort of thing. If you use that same life cycle concept for the interaction development it's a loser because you wait until the end until you have the software done and then it's too late to do anything. So that's why rapid prototyping in this formative evaluation is a nice short cut to all of that up front. That's the closest I can get to that in the short time we have. I think it's time for the next speaker. Thank you for your good questions.

Downey: We're happy to have Bill Hefley here from the Software Engineering Institute at Carnegie Melon to give our keynote address on usability trends in Government. Bill.

Usability Trends in Government

Bill Hefley

Human-Computer Interaction Institute,
Software Engineering Institute, &
Dept. of Engineering & Public Policy
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Mr. William E. Hefley, Senior Member of Technical Staff, Software Engineering Institute, Carnegie Mellon University
Keynote: Usability Trends in Government

I'm going to start by breaking all the rules and come back to both of those questions. One of the points Rex didn't make about your question that I have trouble with is that there is a percentage of the male population that have color vision problems. Now I can take a test and tell you red or green so I don't have red/green color blindness which is the one that's most common amongst the gentleman. But, I can't look at the patterns. Have you ever seen the visual tests where they give you the patterns and you're supposed to say this is a helicopter or this is a tank or this is whatever? I can tell you the colors of every dot, but I can't tell you what the pictures are. So we look at things like that and then we look at our guidelines and it says, use red for alarms. Well which percentage of your population can't see that? So there are some things to be wary of. The other question about engineering I want to touch on as we get into the talk a little bit, but there's also some bit of where we're coming from and where we're going embodied in that.

I have the distinct pleasure of being up at Carnegie Melon where we have both the software engineering institute and the HCI institute, and since I am one of these people that straddles both fields, I'm associated with both institutions.

I came out of industry before going back to the University setting so I have built systems for everything from real–time shop clerk control where our average users were grade school and high school educated, to systems that were designed to be operated and then thrown away for only seven days on orbit where we have very highly trained skilled Ph.D. level users who are flying in the shuttle orbiter. This background gives us a glimpse into the tremendous range of people that we design for in different applications. I'm going to talk about usability trends, but we'll also look backwards at where we've been and how that shaped where we are and where we're going.

The topics for this talk are: heritage, commercial practices, why focus on people and usability, future trends and insights. Why do we care? Why do I want to focus on people and usability? We'll discuss some of the future trends and where we're going, and then I'll add some insights from my last 20 years of experience—at some of the things that I've seen in terms of what's really been effective for us.

Let's look at the heritage topic first. How many of you are in the product development cycle where you ship a new product to be on the shelves every Christmas or twice a year in the mass market? How many of you are in the world where you build systems and you deploy, and they tend to be there for a while? What are the contrasts between those two environments? In the first, we can rapidly make product improvements. This year's model is always better than last year's model or it has more features.

I hope I don't offend my host. We were laughing as we came in this morning looking at all of the things on the sign post for what NIST does and one of the comments was, "Oh, that's the

building over there where the VCR programming is done. You know, the rocket science is over there and VCR programming is there." VCR's are something we poke fun at, because they have usability problems galore, but they've changed over time and they've gotten smarter or they've gotten more features built into them. They are rapid turnaround. Some of my systems, just now in the last year or two or three, are going to operational testing, and I've been at the university now for nine years. So the things that I was responsible for (designing windowing systems) ten and twelve years ago are now just starting to go out to the field. That's a long life cycle. Heads are nodding. You know it's a slow world, but "we don't get no respect."

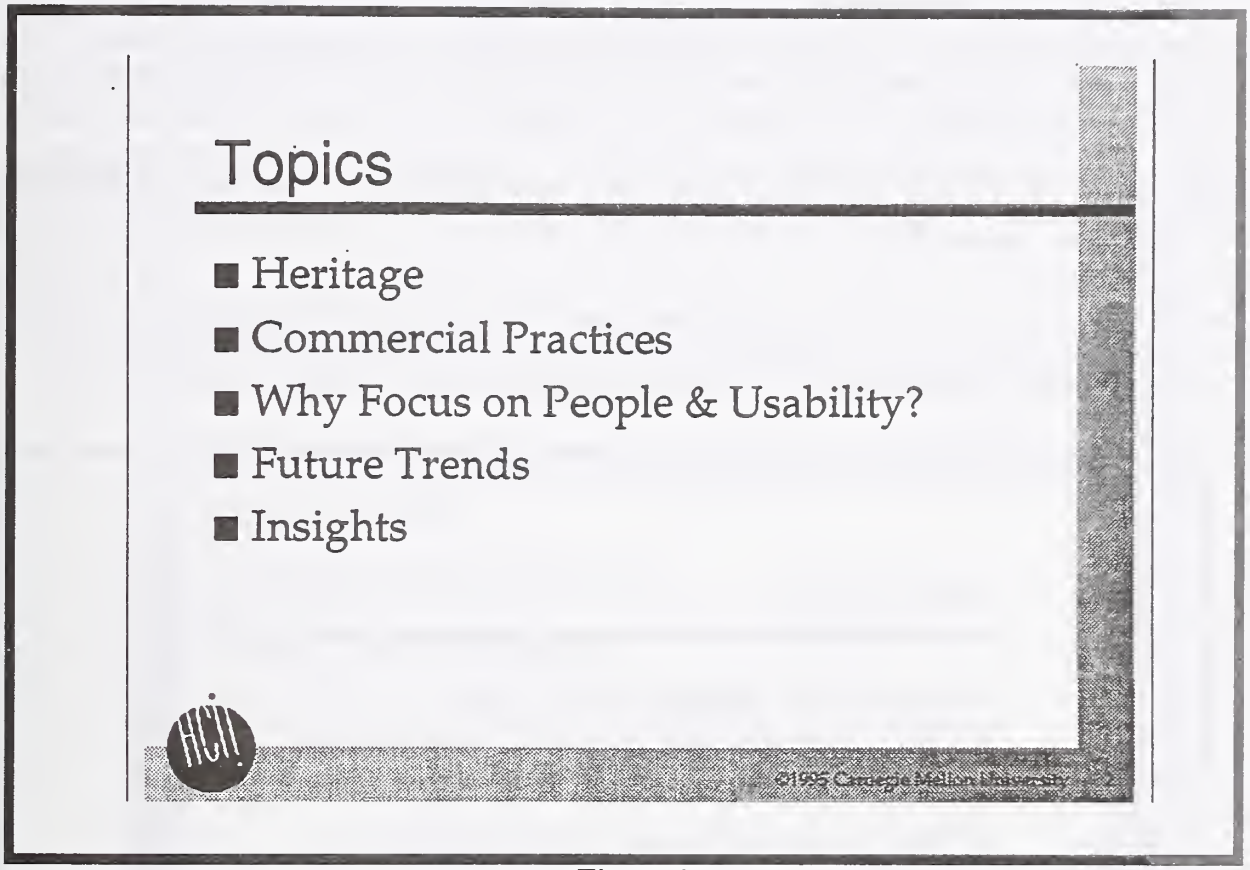


Figure 1

The classical human factors in engineering came out of the psychology world primarily, then got into user testing. This is where I want to come back to a previous question, testing after the fact. There's always the claim about too little too late. After we knew we had a problem we brought them in to fix it for us. Part of the issue with usability engineering is trying to communicate that there's some contribution that can be made up front as part of the engineering process. Some of us have been publishing papers and writing about that and some of the software world for several years have been pushing that there really is a contribution that can be made for interaction design. At the same time we're doing things like data base design and software design. Whose on first? Who are we buying for? Where are the cost trade offs accruing to? Are you being able

to design systems for our users? Is it for the developers? Or in some cases is it the procurer? Program managers get rewarded for delivering systems on time, not for delivering systems to get fielded.

I was teaching a course last week on Web site design and one of the students in the class said, "I went to a site the other day, and I really didn't like it, because [when] I got to the point where I could download a form, they told me not to use the form. [It was] printed right on it, "Don't use this form for real use." We decided that would be a good example to go to as an example for the class. So we went to the site. The site was www.irs.ustreas.gov and the first page is beautiful. They've had a graphics artist do that page, and it looks like a newspaper headline that we've never seen. It has colors; it has this; it has that. It has a ticker tape that scrolls across the bottom. We never got past that. It has these stories—that I'm not sure how they relate to any kind of real use for what you might want to get from that site—but when we finally found the menu, it was about six lines of text at the bottom and it was two or three or four options per line depending on how many would fit on a line. There might have been a bullet or a dash between the options to separate them, but the truth was they weren't readable without sitting there and scanning each item. They weren't easily separated. They weren't grouped in any way that made sense. We gave up and couldn't even get back to finding that form. So in this case, the procurer or the developer got their wiz bang, it looks good, it's wonderful and they got the beautiful site, but as users we were left out in the cold and couldn't get to it, couldn't find what we wanted.

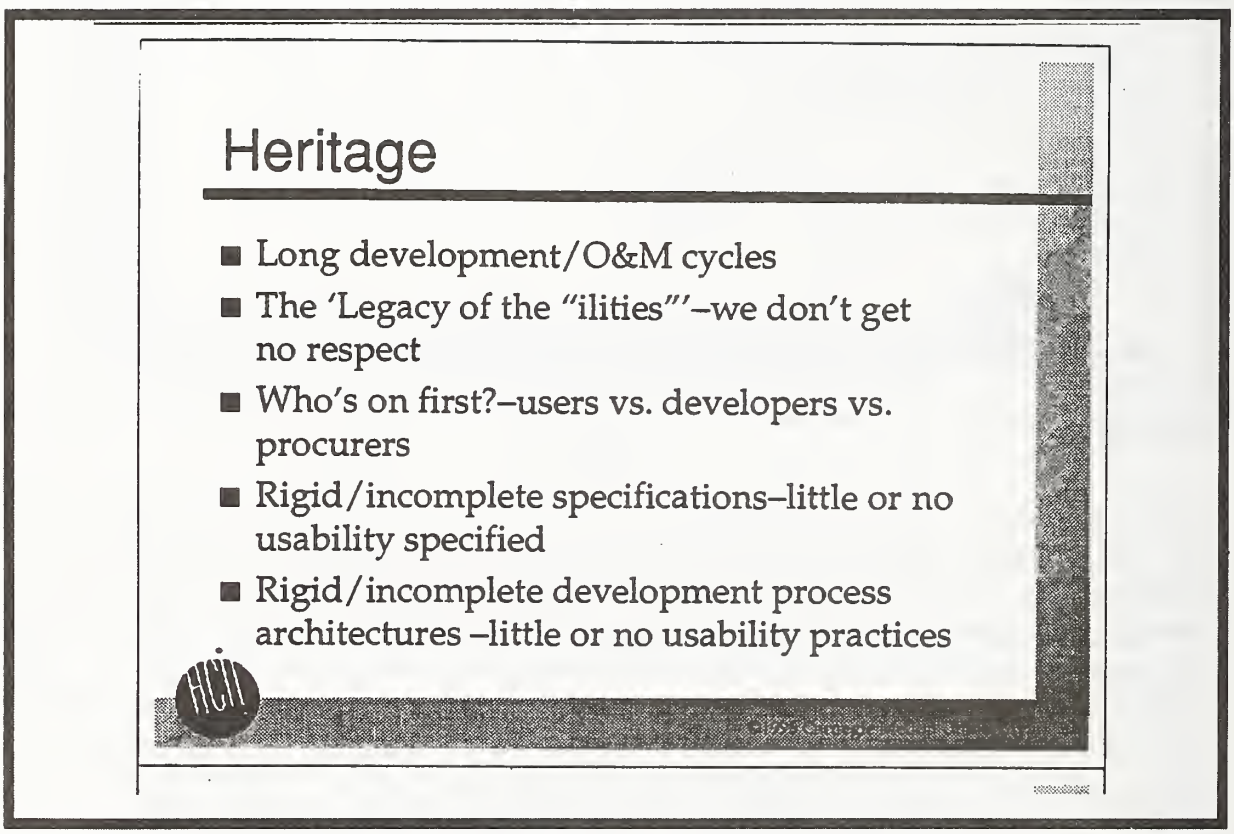


Figure 2

Next, let's look at complete specifications. I had a contract a few years a go that required me, as the lead for the user interface efforts, to write an interface control document for the user interface. I was pretty enlightened. I had to describe the physical, electrical, pneumatic and hydraulic interfaces to my users. Look around, how many of you have electrical interfaces? Any of you have pneumatic interfaces? I want to talk to you afterwards. This was an impossible task. Clearly, somebody had thought about usability and had built it into the contract, but not in a way that we could deliver. Not in a way that would guarantee that the system when deployed would work and be usable.

Another example, and this is the counterpoint of Rex's where he saw a good example. I was asked several years ago to review an RFP for the usability and HCI aspects. While reading through it, I came across a puzzling document. At this point, I had about 11 years in the field and here was this document—I just didn't know what it was. I checked all of my standard files,

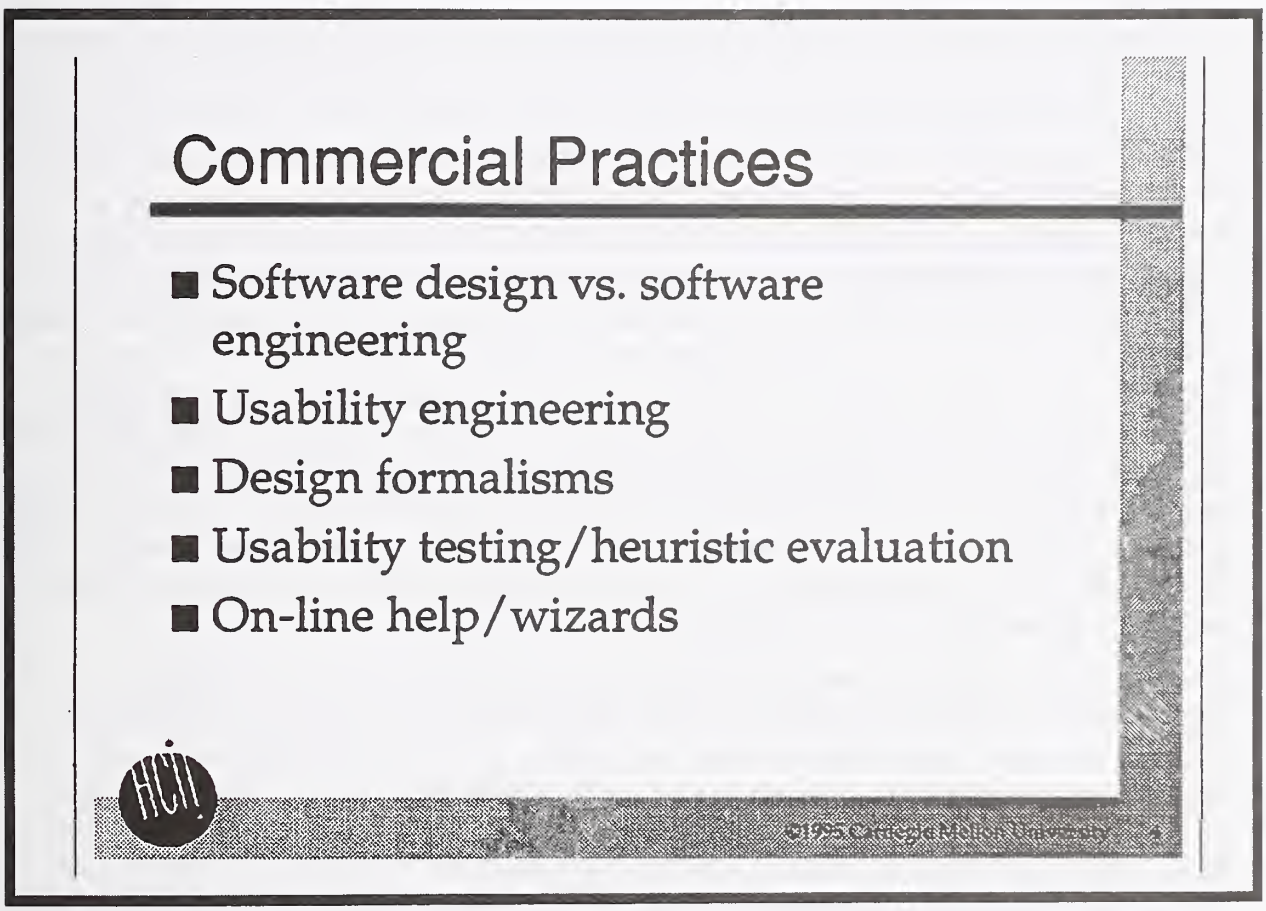


Figure 3

and I couldn't find it. I checked my group's standard files, and I couldn't find it, and I called the corporate library and said, "Can you find it for me?" About three days later, they called me back and said, "We'll have to get it from off site storage if you really want it. Are you sure you really want it?" I said, "Well, my customer is calling it out in the draft RFP that we're reviewing so

they must want it. We know usability is a concern to them." This is a system that had a very long life. The original system was probably fifteen years old, and they've gone through a number of mods and upgrades. They told us their last three upgrades were absolute disasters. Technically they worked. They met every functional requirement. They just couldn't use the thing. It just wasn't usable. Usability was a concern for them so I said, to the librarian, "I know that this is important. My customer is telling me usability is a concern. Please find that document for me." Well, they called me back again, "Are you sure you really want this? You know that document was superseded in 1969." Here they were fifteen, sixteen, seventeen years later still calling out an old specification that was out of date. It was in their boilerplate, and they were carrying it forward, because they knew they wanted something like usability, but then they were telling me that they didn't know what was wrong with their process, and they didn't know what was happening, but the systems they were buying just weren't usable. They just didn't understand the problem.

Tied into the art/process architecture issue is what process were they expecting their contractors to use? What process did they have for insuring that usability was part of their practices? They either didn't have one, or what they were expecting they didn't understand what they were getting, i.e., pneumatic interfaces, hydraulic interfaces, etc. So this is some of where we come from and some of the issues that have driven us to where we are today.

What's happening in the commercial world? Let's start with two buzzwords. There is this buzzword about software engineering—building the system right. Lately there's been a lot of talk in the commercial world about software design—building the right system. Allan Cooper who was the gentleman who built Visual Basic (or the father of Visual Basic) has a book out that talks about software design. It's really about usability engineering. It's about designing usable systems. What's happening there? I'll talk about that trend in a few minutes. One of the things he's driving at is there are more and more systems that are really being used by real people. Think back to your first computer. Mine had the extended memory, I had 16K. It sat in a little air conditioned room on a raised floor. It had less computer power than probably is in my watch today. So as the technologies change we push things out from those arcane temples to where we all have that kind of computer power on our desks.

More people are using the systems and that leads us to this discussion about usability engineering. We really are seeing cases where organizations are taking and trying to apply usability concepts. Many are kin to what Rex was talking about this morning as part of their standard way of engineering and designing systems. Design formulas is another area where we're seeing some wins. There was some work done at Nynex just in the last few years where they actually built a model of the proposed system before they built the real system—looking at the number of transactions they had much akin to what Rex showed and it was something on the order of eight seconds per transaction or fifteen seconds per transaction or something on that order. When we multiply that out by the number of directory systems or operator assistant calls that are made, the impact to them for one design versus another was something on the order of 8 million dollars in actual labor costs per year.

So did their design have any impact? Sure it did, but the impacts weren't accrued to the developers. They were accrued after the operating units. With one trade off in the design of the interface they were able to save quite a bit of money. Usability testing and the evaluations that Rex talked about have been a tremendous amount of work in the last few years on things like on-line help. Partially it's that Band-Aid. Partially, it's that thing that helps us, but also the other thing that's happening is it's forcing us to start looking at what are the user tasks? What do they need to know up front? There is some synergy between the on-line help work and some of the design work that's going on.

Why do we care? Why should we focus on people and usability? There are three things that I want to touch on as trends happening across the community today. The first is demographics. We have touched on that a bit. The second is just general awareness and also a second order effect—something that's masked, but we're starting to see a tremendous impact of.

We've already mentioned the number of people that have computing resources available, but didn't have in the past. Also besides the IRS site, here's another example. The State of Washington has developed something called WIN, the Washington Information Network. This is a kiosk-based system. They are delivering services out to constituents. When I met with the director of IT recently up there in the State Capitol of Olympia, they were actually able to show things like the number of forms filled in, number of jobs where constituents have been matched to jobs and have been employed as a result of using the kiosk. Instead of having to walk into an employment office or a deployment office, they are serving over 200 thousand users a year, and

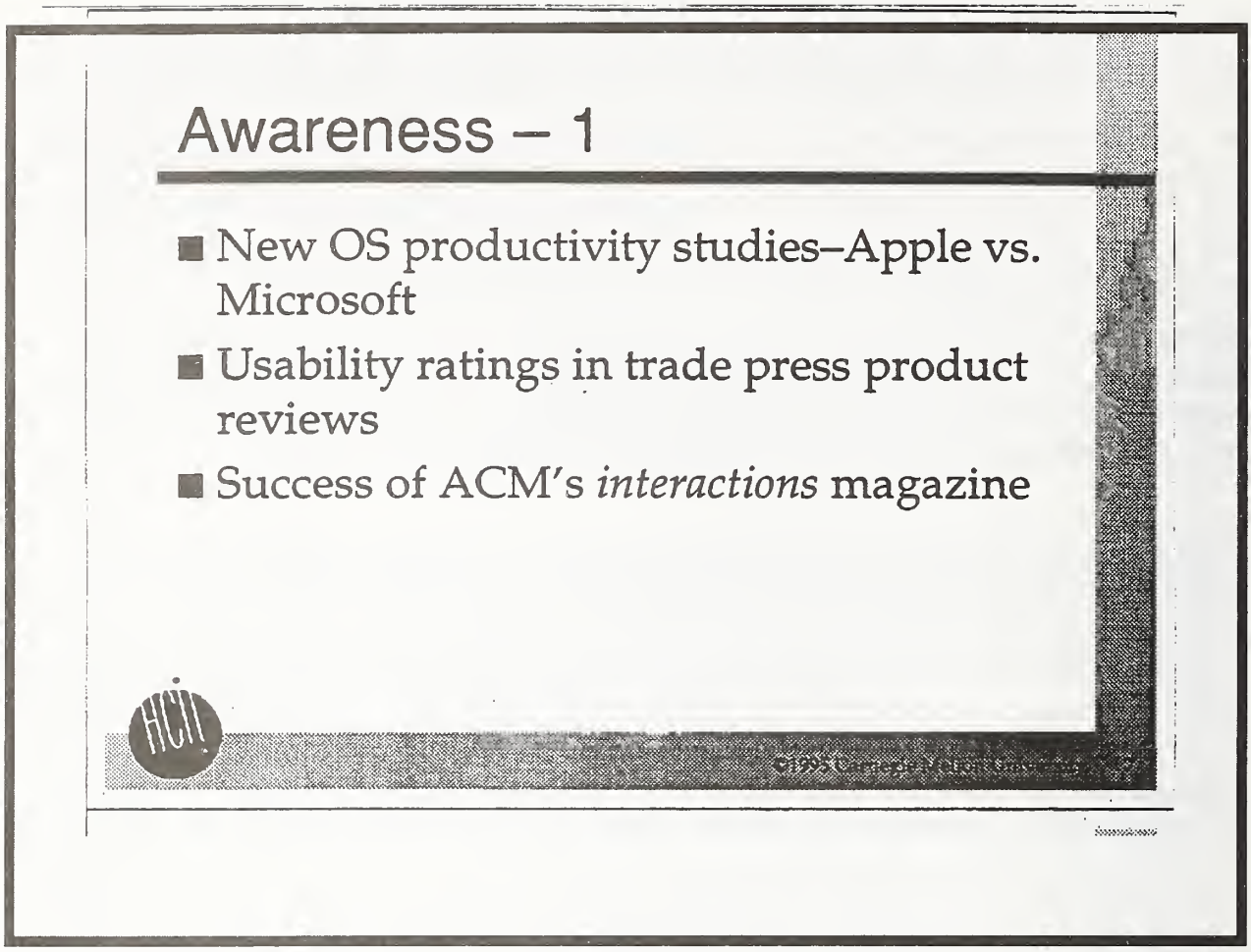


Figure 4

these are everyday, regular people. We're not designing for those Ph.D.–level folks that are going to fly in the Orbiter. We're designing for Joe Average, Jane Average that comes in off the street, walks into the shopping mall and is using this kiosk touch screen right there wherever it is that they happen to be. They are delivering services cost effectively. This is another way of looking at that cost savings issue, because that person using the kiosk is not using a live person in an office somewhere. They are not tying up an individual delivering service in an office where there is the cost for that person, the cost of the facility, the cost of having to be there, the cost of having the lights on, the heat on, etc. So we're seeing that the audience is changing. The people that are using the systems are changing—94% of Federal users, as opposed to what used to be a few of us who could touch the computers. Is the world aware of this?

How many of you saw this month's PC World magazine? There was an article in there about Win 95. Win 95 is the big issue for a lot of people right now. Part of their evaluation is they've looked at productivity. They looked at ease of use. They looked at how easy it was to perform functions—how easy it was to get helper information from the Mac and from the Apple. Some

of the trade publications now rate ease of use as part of their product reviews. The ACM has launched a magazine in the last two years called *Interactions* and it's focused on the usability engineering, the design, the user interface design community.

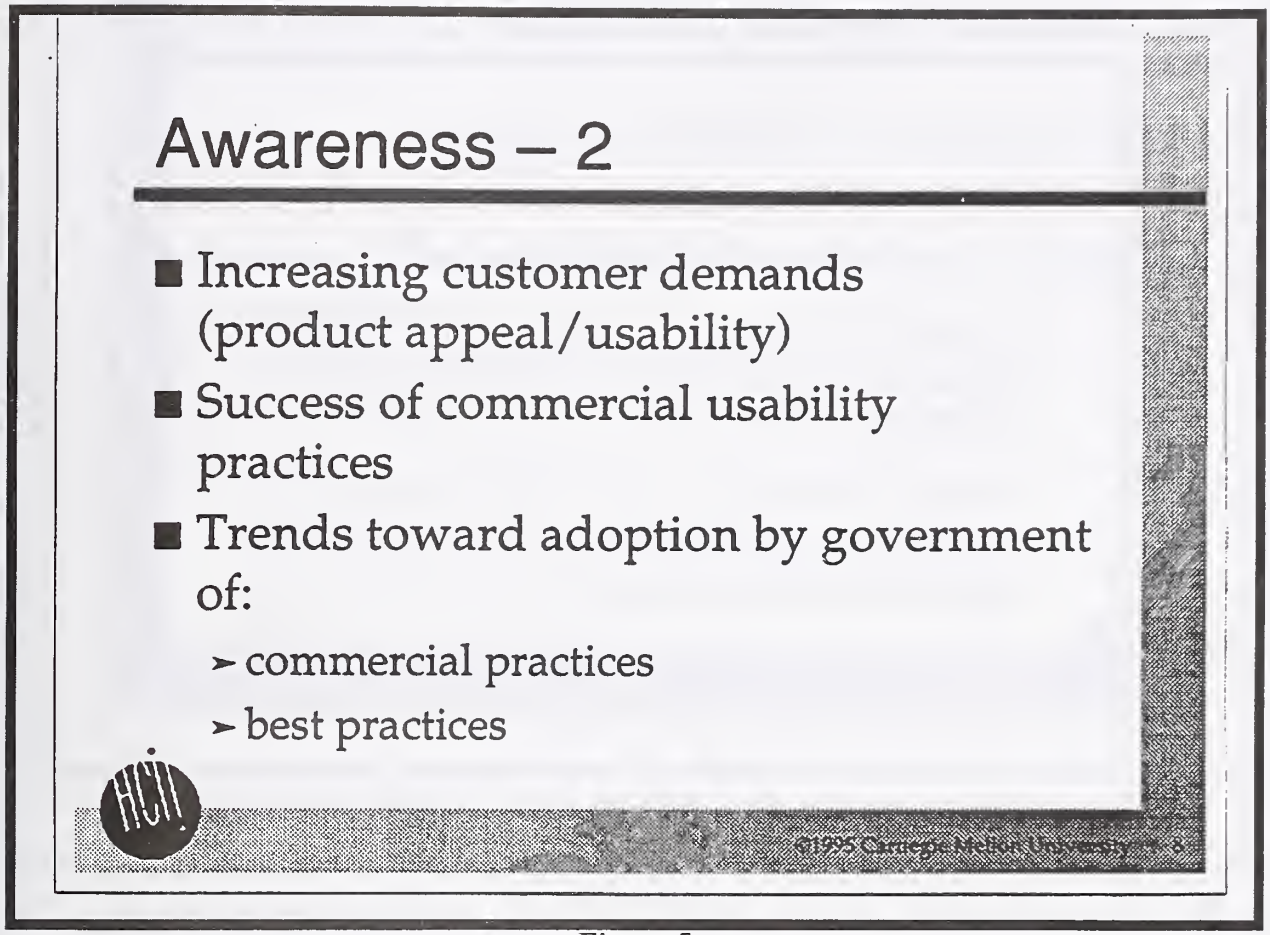


Figure 5

That magazine has been very successful surprising some of the people that were involved in the activity, because there was a demand for it. There was interest in it and it met all of it's goals in the first two years. So there's a growing awareness, but along with that our customers want more.

They want product appeal. They want the usability. They want something that works. If a person can't find those tax forms will he or she go back there again—if the product doesn't let them do what they need to do? The number one concern on the Web right now, the biggest user concern, is they can't find information. If people can't find information are they going to go back to those sites? What kind of a problem is that? Is that a technical problem? No the bandwidth is there. Some may be skeptical, but we have a connection, the connection works. It's not the connection falling apart; it's the usability, the search engines, the technology that's there for finding the information, presenting it in a way that can be used. That's the primary breakdown people are seeing.

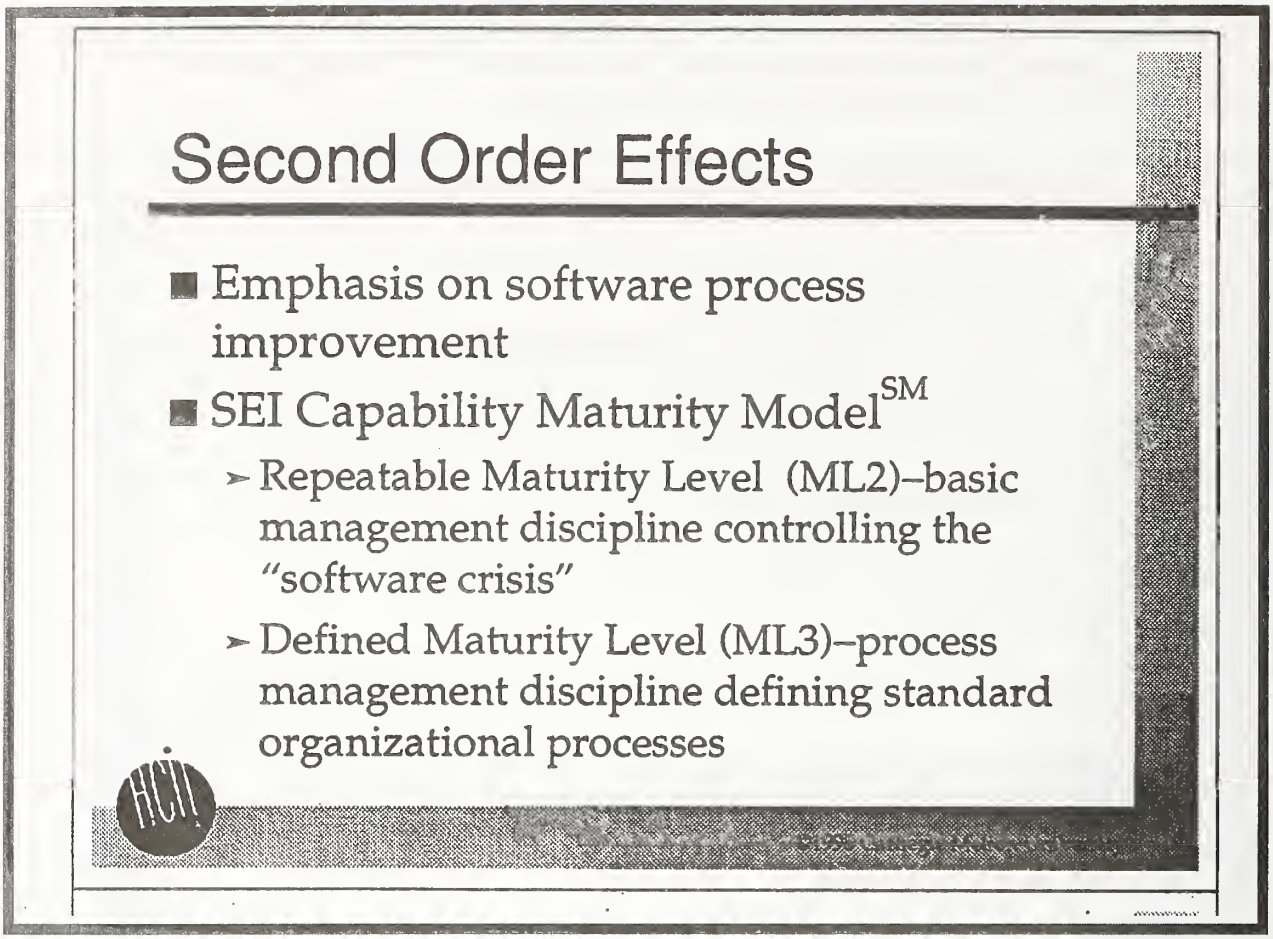


Figure 6

People are hearing about successes of the commercial practices. They look at the Nynex example. They look at some of the other examples where commercial practices are having impact, and we can do that too. We can do faster, better, deliver more useful products. That's what the Washington Information people did. They delivered a system that was really useful to the constituent, to the end user. The other factor that's really making an impact, especially in the military community in the last year or two, is this trend towards adoption of best practices and adoption of commercial practices. We've moved away from the MIL standards. So the things that are currently being used in practice and industry today are gaining more acceptance in terms of use by Government agencies and their contractors.

The second order of fact, this is something I also want to mention. How many of you are aware of this thing that was developed at the SEI called the Capability Maturity Model? A fair number of you. It's a set of benchmark best practices for getting software management under control. Level II, it's a five level model where level I is "y'all come." Maturity Level II is basic management discipline in place. We have project plans; we track to them; we do configuration control; we do quality assurance. We get our basic management discipline in place, and it addresses what for years has been considered the software crisis. Of course we know that study

after study after study told us that the software crisis wasn't a software problem, it was a management problem. We weren't managing our software activities. When an organization moves to level III , it is focusing on defined organization processes. It has a standard engineering

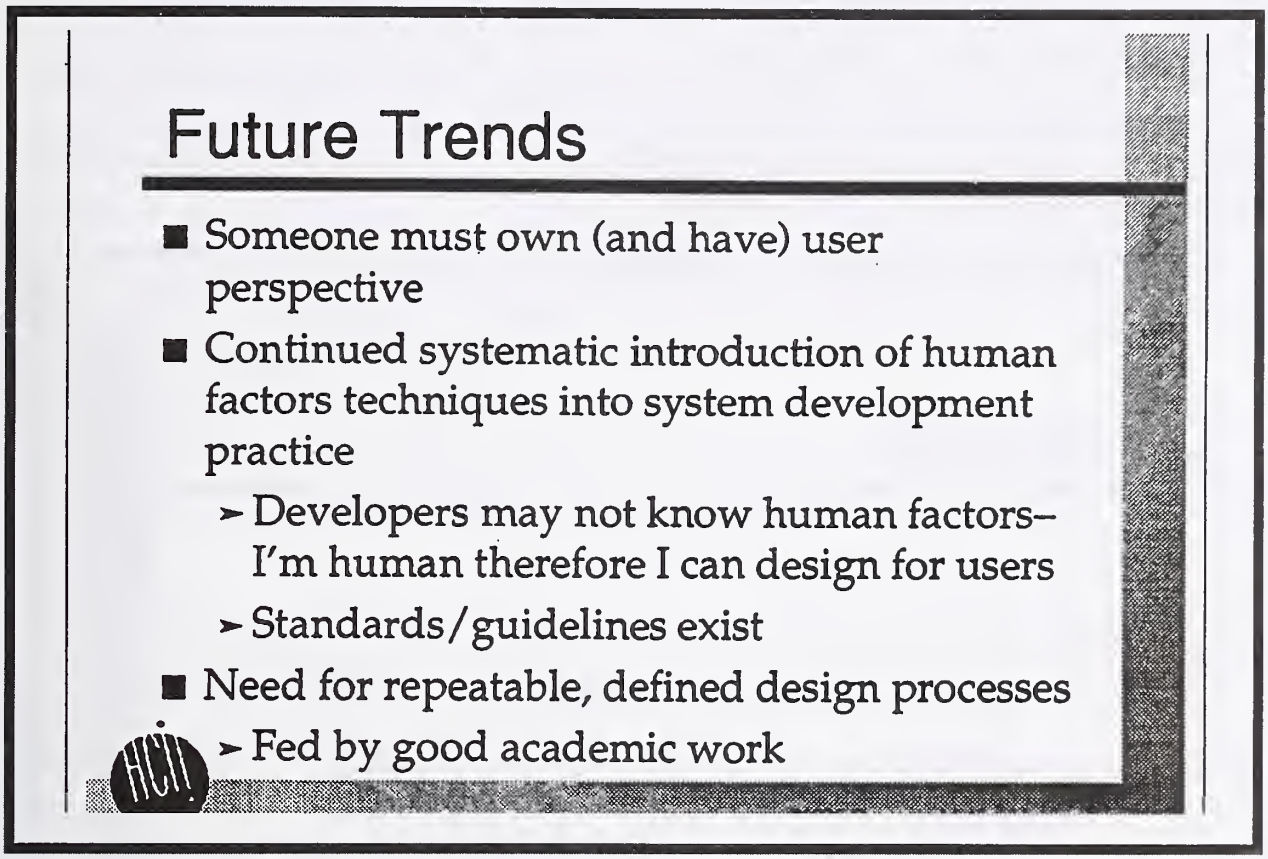


Figure 7

process across the organization. One of the things that we're starting to see as the second order effect is as organizations start getting Level II and they get the basic discipline in place, they are now starting to move from building the system right to being able to focus on building the right system.

Remember the distinction made before about software engineering versus software design, building the right system? As we're starting to get our processes defined and as we're starting to look at what we really have to do to build our systems, we're starting to find opportunities for defining our usability engineering processes to dovetail with our software engineering processes to be part of our fine process for building systems within an organization.

One organization that I'm familiar with now has an effort within their user-centered design group to set a similar set of standards. The CMM doesn't address usability engineering, but they are trying to within their own organization, build a set of benchmark practices that say these are the things our organization ought to be doing to implement user–centered design across the

organization. They are working with their software process improvement people to rule that out as part of their standard internal assessment. So when they look at programs, or they look at divisions, or they look at product groups, they are going to be looking not only at the software process, but also their usability processes and know where they are and then start to set guidelines and goals for improvement across the corporation. Now they are not primarily in the government market. They are in the market primarily where they have to ship product, and they have to sell product, and are very concerned about focusing not just on getting past Level II, but getting their defined processes and making sure they define the right set of processes. This lets them deliver a product that their customers want, that appeals to their customers, that meets their customer's needs, that will sell in the marketplace.

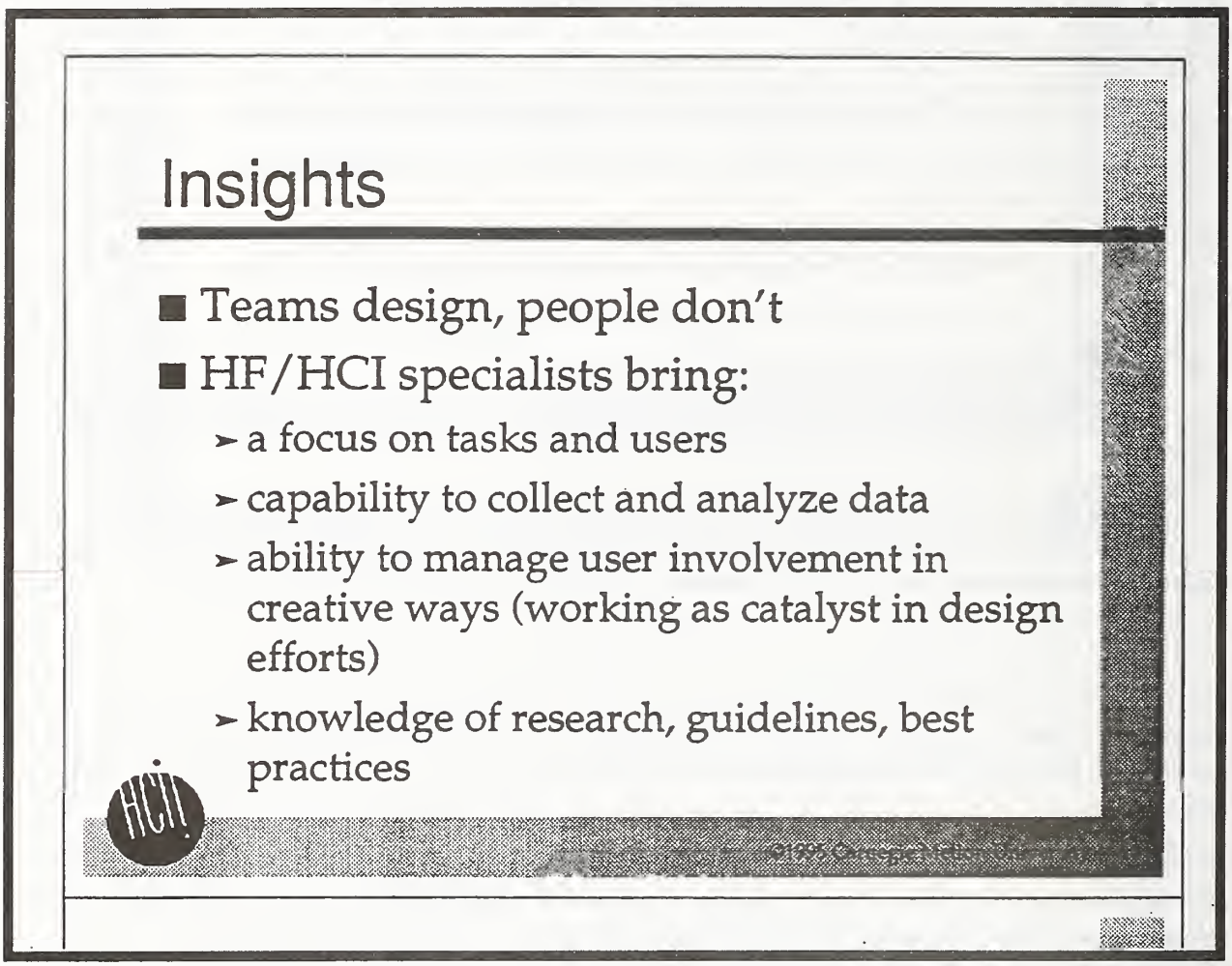


Figure 8

This is a second order effect. It's something that's been masked because so much of what we've seen is the crisis as we develop systems. Rex didn't tell you the corollary of the waterfall model. The corollary of the waterfall model is 99% of the time when we come to that last sign off and we sign off that we're really done, we're really only 90% there. It's that last percent, that last

10% that takes the next 90% of the effort. The software process improvement work has been trying to focus on changing that, getting us to the point where we have that basic discipline in place and then define our processes. What we're seeing over and over is that once we get that chaff out of the way organizations are starting to focus on what's really meaningful to them. Some of the things that are meaningful are delivering the right product. Whether it's a product that ships and sells in the marketplace or a product that supports their constituents that meets the needs of the real user community.

What are some of the future trends? What are some emerging trends? We touched on it earlier. Somebody has to own the user perspective. It doesn't matter if it's a human factors person, if it's a software person, if it's a product marketeer, if it's a product manager. Somebody has to own that user perspective. Somebody has to step up and be the champion. Somebody has to make sure it happens. Whether it's part of your defined process and maybe it's your process owner that's making sure that happens. Maybe it's your product manager, maybe it's the project manager. Maybe it's somebody like this chap at BLM who just stood up and said, "I've had enough." Somebody has to be watching out for that user perspective, making sure that we get the right system and that it is usable. This means continued introduction and adoption of some things that are already known that are already tried, some usability engineering techniques, some of the formal methods into our standard practice. Developers are human so they design for users, right?

There is some truth to that, but quite often developers don't know much about the human factors issues and some of the other things. In fact, one of the things that was really humbling for me is that I was originally trained as a programmer, and I got into designing user interfaces which at that time were screen designs. I said I really don't know enough about what I was doing, so I went back to school, and I took a degree in psychology, and what I realized after taking a degree in psychology was that I still didn't know a whole lot about designing screens, but I really understood a lot about why I didn't know. We've seen and Elizabeth Buie will talk later today that there are some standards, there are some guidelines, there are some things that we can apply and can help us develop systems that actually will meet our user's needs that can decrease some of those error times and cut down on the help desk.

Now again that comes back to an issue about to whom do the benefits accrue? There are a series of companies right now that are making very good money answering help desk calls for Windows 95. Now I'm not picking on Microsoft, don't take me to task for that, but there is some market benefit there, because they are making money on those calls. As we continue to actually show that these techniques can be provided up front and used in our systems, we're starting to see the impact of the benefits. That leads us to the point where we were talking about getting to repeatable defined design processes that are not just the software processes that also include the things that we do for our usability engineering. Some of that continues to need to be fed by research, and this is a point that is quite often lost in the sense of "let's do basic research." Sometimes the basic research that we need is focused on methodologies. Nigel Bevan and I were chatting earlier today about work methodology and that's work that needs to go on. How can we

best take some of those lessons learned from the basic research and tie into our methodologies and tie into processes that can be used?

One of the other things that we found is team design. How many of you design systems by yourself? How many of you work as part of a larger team? We were talking about designing Web sites and I said, "What are some of the kinds of skills that you might need?" The class started making a list: somebody who has systems admin. skills to run a server, somebody who can program because of back end interfaces to the IT systems, somebody that knows something about designing the interface and the pages and the content. Oh, yeah, content; we need somebody who can write. We're going to put up some images so we need an artist—an industrial designer or a graphics artist or somebody who knows how to design interaction. We want to engage people. We want to know something about how to bring them back, captivate them. That's a team of people.

There are very few of us in this room who probably have at an expert level all of those skills. So we really are focused on teams that design and that's part of this getting away from the "bringing in too late" to "coming in early." Here's a success story—the pneumatic, electrical, hydraulic physical interface story. Every bit of information you need to know to understand is that this was custom hardware, so we were having to build our own windowing environment. We couldn't go off the shelf and buy a windowing system. We turned that around to our customer and said what we really propose is a three fold ICD, interface control document. We'll give it to you in three installments, and this is usability on the installment plan. The first volume was based on the up front analysis and the task. The up front, not the detail analysis, but the workload analysis and the systems analysis and the allocation of tasks between the users and the system. The first volume will be basically our windowing system design so it's the spiritual guide, because we knew enough about the task to be able to identify what kinds of things we had to have in the physical environment. So the keyboard, the track ball, the windowing system, those sorts of things, were spec'ed out first based on early front end analysis of what the users would be doing. By the way, this was taking a job that used to be done by a crew of five down to a single operator so there was some interesting information presentation issues we had to deal with there to be able to keep this user informed and not overloaded. We designed the windowing environment first. That was Volume I. We shipped it and it went to press. That also allowed the hardware folks to get on with saying things like these labels go on these keys and we can now spec the hardware and get it built.

The second volume was a detailed task analysis. We chose an old human factors technique called operational sequence diagrams, i.e., this goes out of the system into the operator's head or into the operator's eyes or it's on the screen. This is what goes into the system from the operator's actions. It was a very simple representational technique, but we didn't worry about which screen element. What we worried about was the information content. By the time we finished the second volume we knew exactly how the system would be operated. We knew what the information flows were and how they would fit back with the software design because the software folks were now almost ready to begin their design.

Third volume, Chinese menu problem, take one from column A and one from column B. In the second volume it mapped the specific windowing element in the first volume. The third volume was the complete illustrated system manual.

Now to the windowing environment that was the coding manual for the actual code implementation part of the system. Did it work? Yes. They spent less time fussing with getting the interface right, making sure they had the right information, making sure that they didn't omit information that was critical to the users that the system was actually able to be built. They put the system on display once and the feedback we had was that people walking in off the street at the open house were able to use it. That was quite a compliment because our audience was Master's level four to eight years professional experience, journeymen level in their field, and yet we knew that wasn't the environment our users would actually be in. So what did we bring to that party? We brought a focus on the task for the users. We knew that the spec was wrong. We were designing for an audience that didn't have as much skill as they told us it would have. We really spent a lot of time focusing on the task.

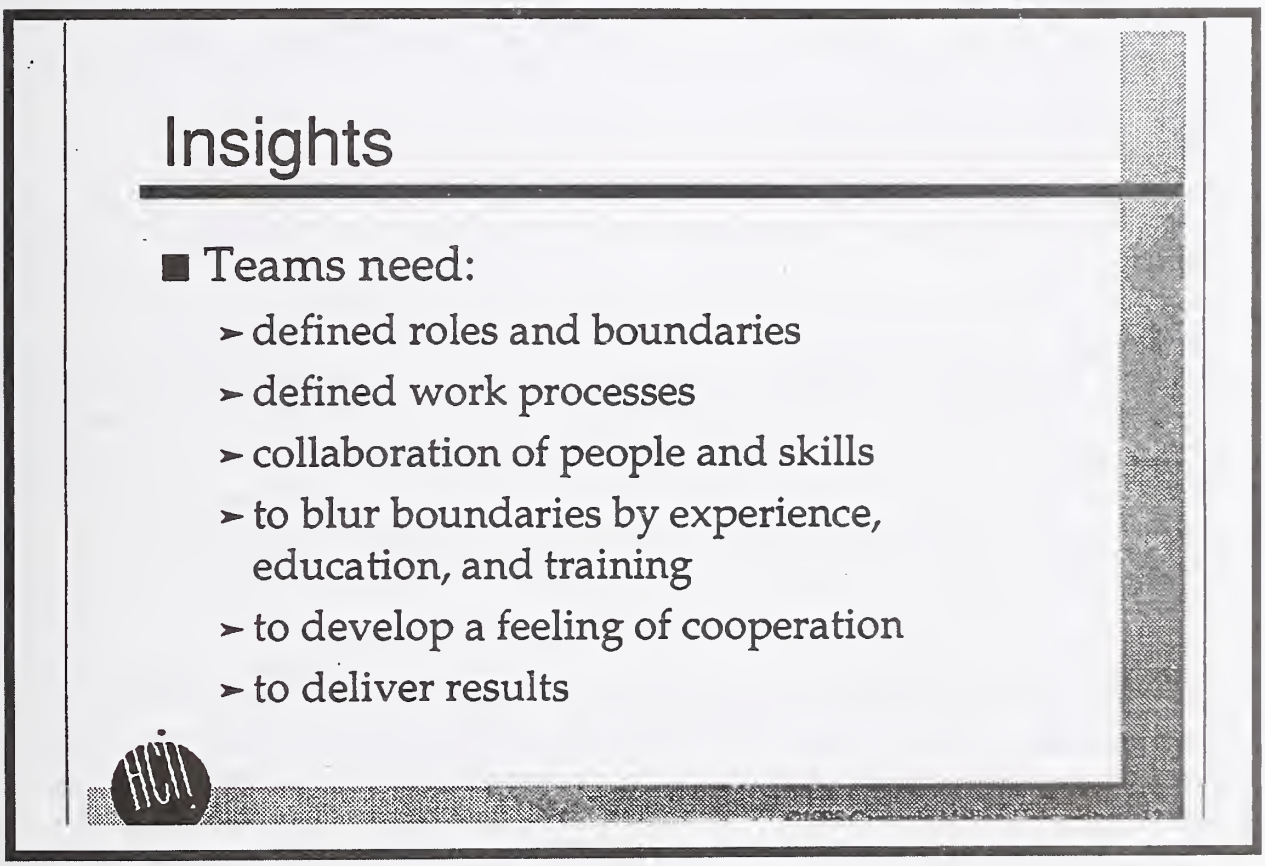


Figure 9

We brought a capability to collect and analyze data, whether from usability testing or from prototyping to manage user involvement, focus groups, formative evaluations, usability testing, prototype evaluations, design walk throughs. We had an interesting team there. We had an

industrial designer (coming back to this issue about teams.) We had an industrial designer, a journeymen level human factors engineer, a software engineer who was doing our prototyping, an Ed. D. with cognitive task analysis background and a Ph.D. cognitive level psychologist. Then there was me the jack of all trades just kind of trying to keep this merry band moving in the right direction. The other thing that this group of folks brought was some knowledge about the research guidelines and best practices. We were able to really change the direction that program went. It wasn't a single interface. The document that was delivered up that talked about physical, electrical and hydraulic interfaces. So we can really build on the teams as well as the technical specialties, but what does that lead us to?

It leads us to some things that we need to really pay attention to as we use and apply these engineering skills in developing real systems. Remember the second order effect that talked about defined processes? Here is the corollary to that: not only do we need to worry about those processes, but also how we can contribute? What are the boundaries? More and more software folks as they get into doing user interfaces start worrying about doing it right, but don't want to. In some cases they admit they don't know and in some cases they don't want to admit they don't know how to do it best. There are some real ego issues involved here that are not just engineering issues. That leads us to this issue about collaborating about people and skills. We really need to be part of the engineering team. We need to be part of the design team. I mentioned that team that we put together, that was a catch up situation, but we had to put together a really diverse team to make that system what it had to be. We blurred some of our boundaries there by mixing experience and by education. We spent a fair amount of time doing informal training of our software engineers. There was a lot of give and take. Those documents weren't produced in a vacuum. What did that give us? They felt like we weren't just that group doing it to them. We were actually collaborating with the developers and the system was delivered. It actually met its goals and met its usability objectives. Those are some of the points that we found really have to be focused on in applying usability engineering. It's especially going to be more and more important as we build systems that are reaching a wider audience, an audience that has more diverse backgrounds than what we used to design for, systems that have to do more. As we look at how to reduce costs in either service delivery, reduce costs in development, reduce costs in our back–end support, those are all issues that we're concerned with today.

We're really trying to focus on addressing some of these issues on out into the next generations of systems. Not to just apply usability engineering, but have it be part of our normal way of doing business. Having the techniques be accepted as part of our system development practices, as part of our defined design processes. It's going to another, I don't want to say generation, because that's scary. People think about years in that, but it's going to take another cycle of success for things to circulate out. There's been some work on technology transition that looks at these things and unfortunately the research shows us that it takes somewhere between four to eight years to get something really well established within a corporation or a large organization. For something that's a radical change like structured programming, things like that take on the order of between 12 and 20 years to gain widespread acceptance, but we're starting to see some various forces come together.

We talked about usability engineering as a focus on the users and tasks from an HCI perspective. If we go across to the software engineering community, or the object-oriented design community, we see them starting to focus on users. The only way we really know how to design the right objects is to have use cases. Larry Constantine wrote an article in *Interactions* last year talking about use cases. Use cases are really user scenarios. Now that the software community is starting to talk about this, we really need that. Here's an opportunity for some synergy where both communities, both disciplines try to focus on developing the same sorts of things to deliver the right system. I'm open to questions, attacks. Oh, now the hands went up on that one. Yes, sir.

Male: What about the differences in learning styles and applying that to interface design?

Hefley: Yes, there's been a fair amount of work looking at learning styles and how to apply that to interface design. A lot of that hasn't gotten beyond the academic work into real usable sideline kinds of things. We know there are some people that are more visually oriented. And there are some good tasks that can be used, field independence tests, the ones that have primarily been used in academic research to get at that. But, how can we translate that into design practices? It's a leap of faith that a lot of people haven't made yet. You're nodding your head. Grab this mic if you want to jump in, because this is something that I think is going to have an impact for us as we start really realizing that we are designing for a much more diverse audience for many of our systems then we ever did originally. We used to think about designing for the office, and we knew that most of our people were coming with this skill level or this level of education and we could just do a design for that. Designing for the mass audience is a different audience.

Audience Question: Training will tell you.

Hefley: Training indeed will tell you, and that's where we see so much more emphasis today on the on-line help and the Wizards. I know a lot of the work that Jared Spool and those folks have been doing have been focused on Wizards in recent months, primarily because it's a way of capturing some of that task information and putting it in a way that can be formatted and delivered with the product.

Audience Follow–up: Yes, I think that one of the biggest challenges if you were to lay out challenges for where are we going from here would be designing for all these individual differences. Studies have shown that various kinds of individual differences in terms of aptitudes, age, background and culture and so on account for more differences in user performance than design in training and things like that. Unfortunately, they are also very difficult to design for. We can't make a completely different design for every user and even if we could then they would not be able to communicate with themselves about the system. That's one of the problems of high degrees of customization. Users make their system look so different from every one else's that they can't even talk with someone else about it. So maybe the idea of lots and lots of different views and different user performances that are all rolled into a single integrated view of the design. That I think is the direction to go in. I think the direction not to go, and this is personal opinion, is the idea of adaptive interfaces. I'm not talking about adaptable interfaces because that's what customization is, but adaptive interfaces where the interface sees patterns in the user's behavior and the user's abilities and so on and makes changes

arbitrarily to sort of match those. Sometimes that works, but even when it matches the user's needs it comes as a surprise and violates the sort of main guideline of the user being in control. So it's better to offer all these possible customizations, but let the user control it.

Hefley: When you talk about designing for diverse audiences one of the other things that people are starting to pay attention to depending on the system's ability, is the great diversity of age. It's not just an issue of the teenagers have been exposed to computers and will use them where an older population has not. It's also there are some definite physical differences in terms of visual accuracy, in terms of how some of the different age groups will interact with the systems. So that's another concern there and that ties back to your point about learning, is how can we make it non-threatening to walk up and use something. I know people who will not use an ATM because it's just too threatening. I'd rather wait till Monday to go to the bank where I can use a teller thank you.

Thomas J. Watson Research Center
PO Box 218
Yorktown Heights, NY 10598

Cost-Benefit Analysis of Usability Engineering

Clare-Marie Karat, Ph.D.
e-mail: ckarat@watson.ibm.com

January 10, 1996

Dr. Clare-Marie Karat, IBM TJ Watson Research Center
Cost-Benefit Analysis of Usability Engineering

Let me tell you briefly how I got into this field. Before working for IBM, I worked in state government in Texas. I was basically presented with the problem of being able for our department to do a whole lot more with a whole lot less, and my strategy in this was for us to re-engineer our service delivery process, in other words, to automate it and deliver that in the marketplace or in our client locations.

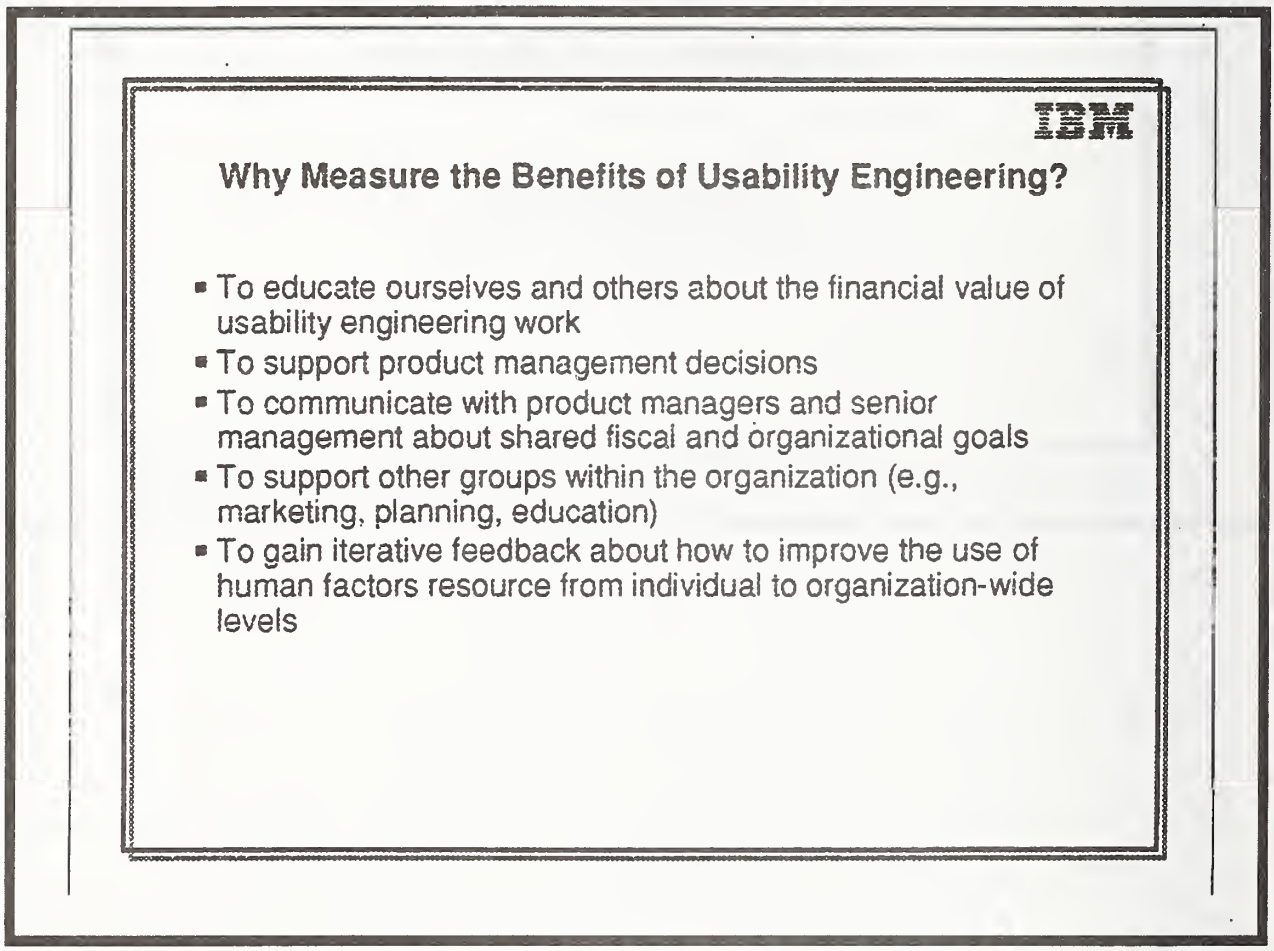


Figure 1

This was an incredible education for me and I learned there the value of re-engineering as well as the usability engineering component. And they go hand in hand. The re-engineering piece is a macro level design and then the user interface design is the micro level—actually a task-to-task specific way to complete the work. In order to get the biggest bang for the buck, we can't do one without the other.

Let's begin with a cost benefit. There's a book that came out a couple of years ago. It's called "Cost Justifying Usability". It's edited by Deborah Mayhew and Randolph Bias, published by

Academic Press and I have two chapters in that book. If you'd like more information about how to actually do these calculations, please refer to that. There are a whole host of wonderful chapters on different topics in there as well. When I joined IBM, the first project that I worked on had already begun. I basically met the product development manager and he said to me, "Sure you can join my project. I'm not going to give you any money, you can't change a schedule, but let's have you come on board and see what kind of value–add you are and that you can provide from this field." I took that challenge, and it worked out rather well.

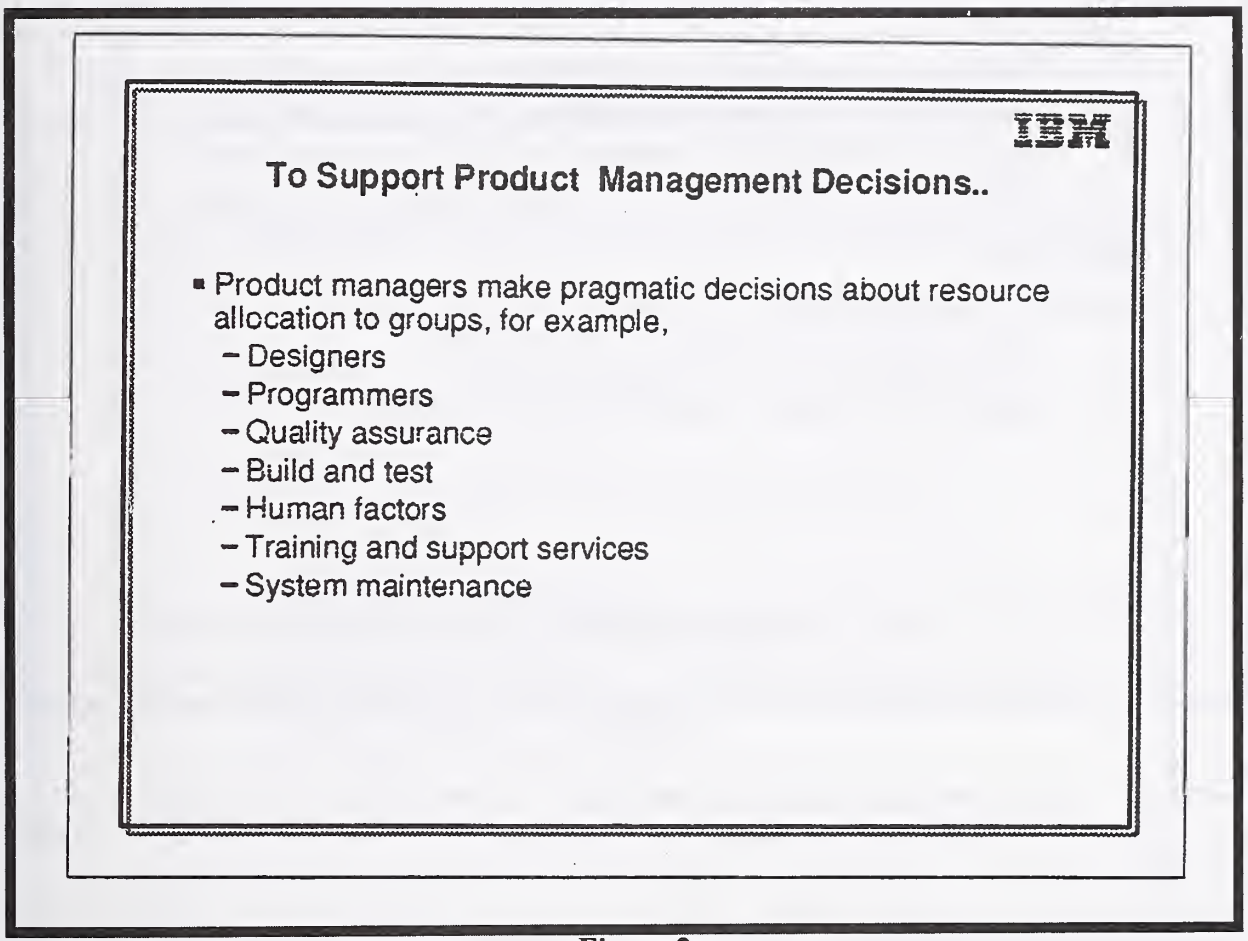


Figure 2

The key reason for doing cost-benefit analysis is to be able to communicate the value of the work that we do. How can we do that if we don't really measure it and manage those measures? The second piece is about being able to support development decisions. This one is pretty far reaching.

Doing negotiations sessions about the role of usability in product development, I learned that I was going to be working with a number of other groups. They generally had been collecting information and were able to communicate for sometime about what it was they would be able to produce if given X amount of money. In order to be able to even enter the game, it was necessary to come up with a methodology and a way of presenting the work in usability. Basically, I tied it to the things that were of the most importance to the product manager, time,

money and resource. Once a project has started (including this cost benefit analysis), it's very important to work closely with other areas in the organization.

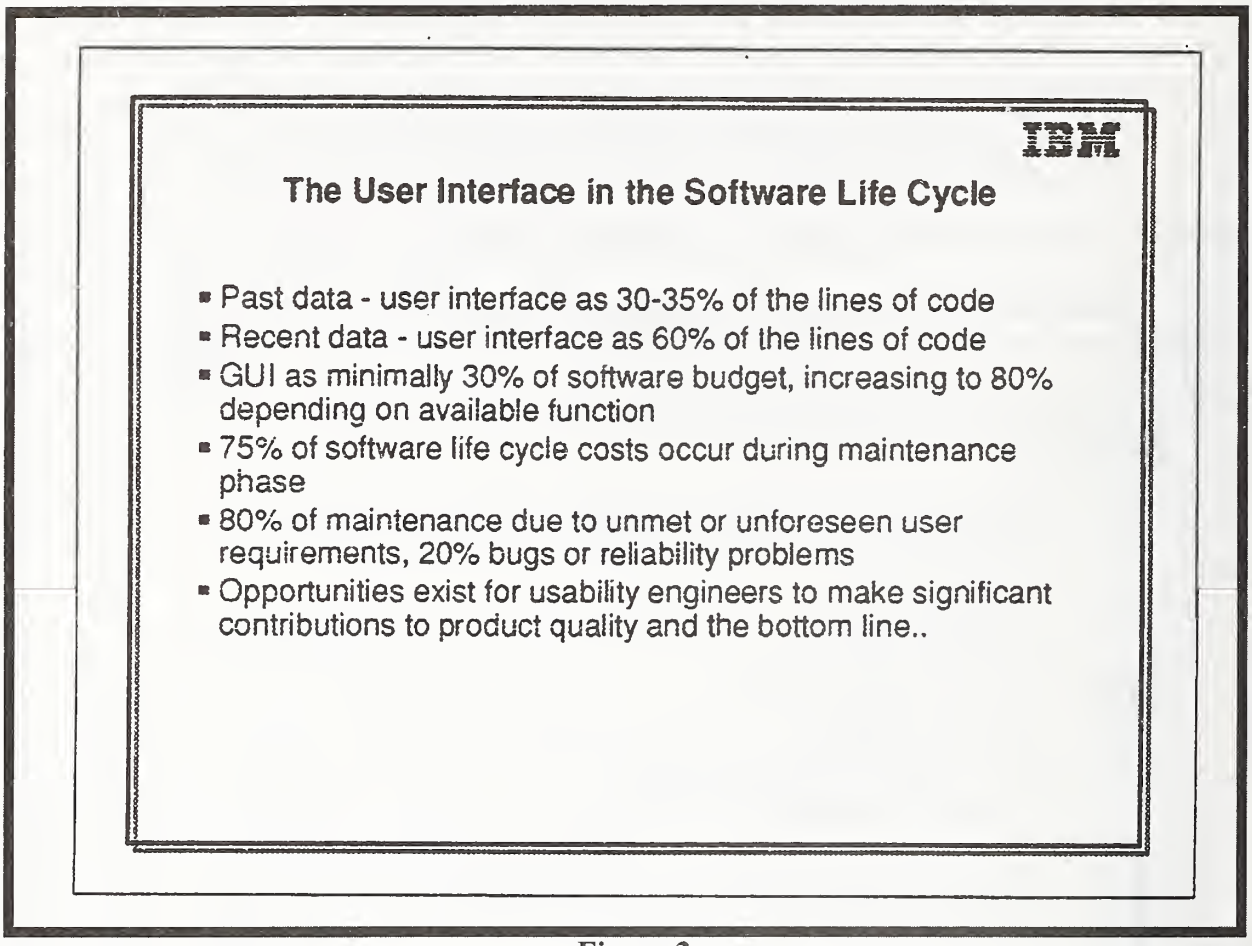


Figure 3

There are some real benefits in terms of cost savings there as well. Here is one example. In terms of working with, for example, education, in setting out usability objectives and conducting the field and laboratory evaluations and design sessions to handle the problems that arise, we may be able to handle the bulk of them, (severity 1), but maybe there are some 2's that because of time constraints and money, we just can't get to. Those can be turned over to the training groups. There are tradeoffs. We all live in the real world but with a little closer coordination we can do a much better job. We found that we've been able to reduce significantly our training costs because of that. In terms of communicating with senior managers (as a couple of people have already mentioned), we generally manage our organization in terms of goals. To be able to actually show how our work contributes to those goals is a pretty effective way of communicating. This is just another piece of the power that we get from doing this kind of work.

This last bullet (in Figure 3) is more for ourselves, the bulk of the people in the room who have categorized themselves as usability practitioners. If we start collecting information about the cost of the work that we're doing in usability and the associated benefits, we'll be able to make better judgments over time about what particular method to use in a particular situation. Tradeoffs that come up frequently occur in terms of whether to do field or laboratory testing,

whether to do a walk-through or a usability test. These are things to think about depending upon the type of project. Once we have this kind of information, it is easier to make a better decision and help guide other decisions in our organizations.

One of the things about working on cost benefit analysis is that we need to take a large view of the product cycle, really the life cycle. If we look at some of the statistics on Figure 4, we'll see that the graphical user interface has become a large chunk of the work on any one project. An even larger point is the fact that the bulk of the costs come in maintenance. A speaker earlier today talked about this distinction—that somebody would build it and somebody maintains it. A number of years ago in terms of our organization as well as our performance measures, we changed the way that works within IBM. A particular manager is responsible not only for

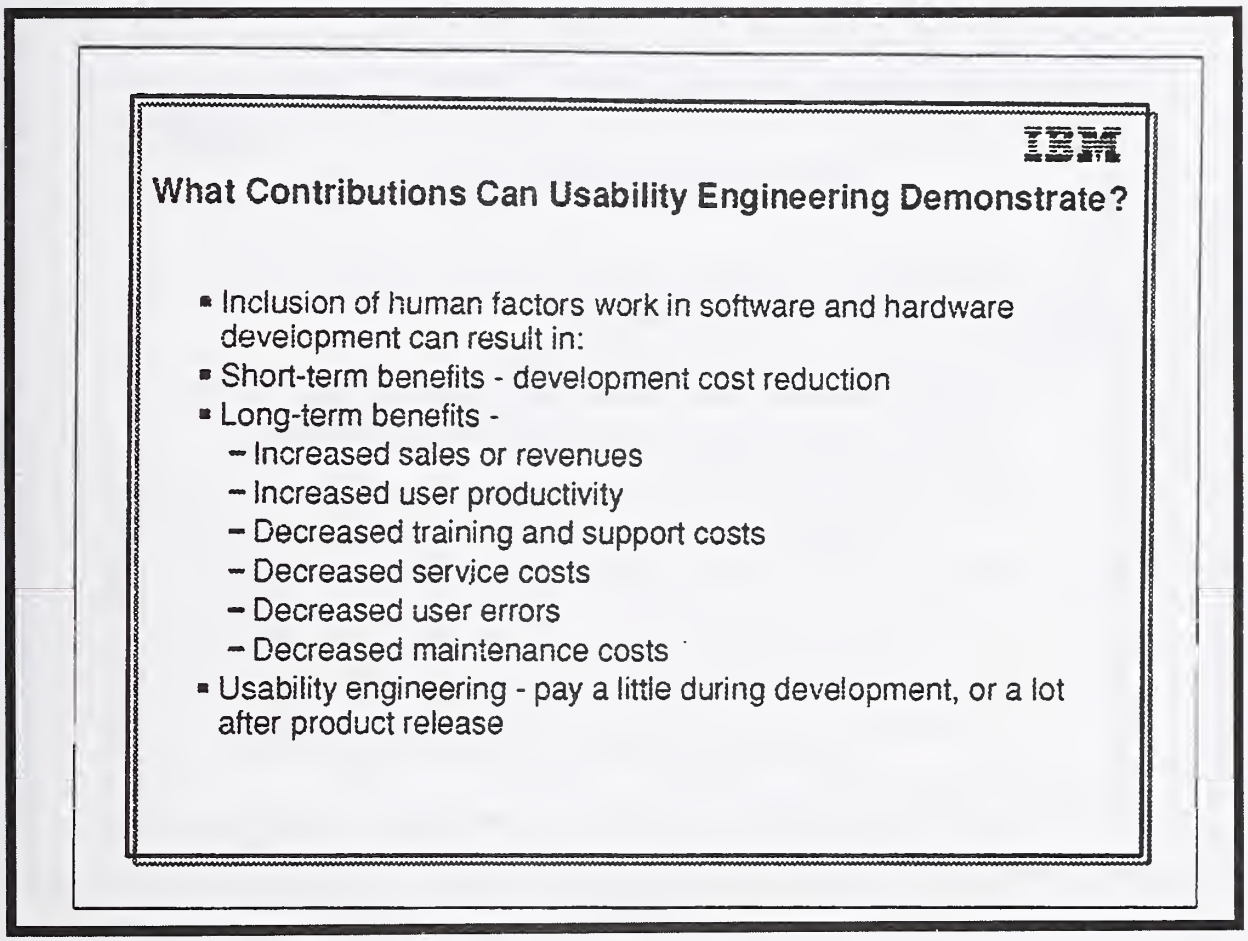


Figure 4

development but for all the operational costs, maintenance costs. That gives one a very different view. If we take a little bit of time, invest a little bit of money now and identify these problems, we're going to be able to save ten-fold what we could down the road. This is real impact. The fifth bullet on Figure 4 is exciting in terms of the opportunity that it provides. The bulk of these maintenance requests are really unmet user needs. This is something that through user-centered design, we can really hope to address at least half of these during development. That gives us quite a competitive edge.

The contributions of usability engineers are both short term and long term. The short term benefits involve actually decreasing the amount of money it costs to do a development project. This is based on experience. If we have usability involved from the start, it is actually part of the critical path of development. If it's part of a critical path, part of development, as we go forward, we are doing participatory design sessions with users—doing user-centered design. Using early low fidelity prototypes, we are are getting the requirements refined. We know that we're on track with the user interface and that we're going to have a highly satisfied customer at the end. My experience in projects where we haven't had it as a part of the critical path is that close to the end (Bill mentioned you get to that 90% point) this host of problems comes up. In order to fix all of those the project exceeds the original guideline.

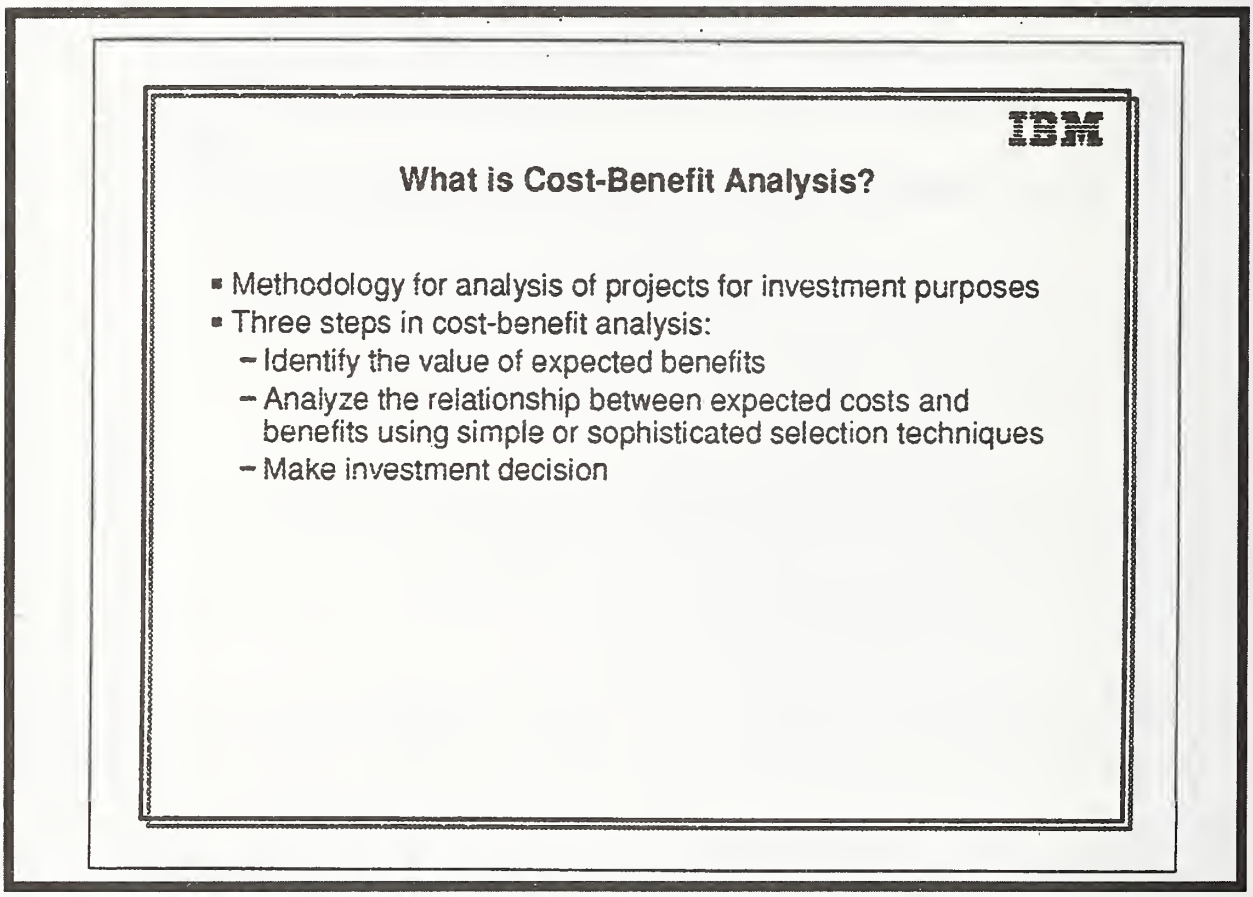


Figure 5

Think of it in terms of the actual cost versus budget and cost for a project. If we look at those actual costs, we see that in a project that has usability and one that doesn't, it's far less expensive to go ahead and build quality in. That's really what we are doing here. That's the argument that a project manager can take forward to management. Long term benefits for external marketplace products are the revenues. If it's a product for internal use then we get the incredible benefits of increased productivity. Even if it's a product for the marketplace though, that data on increased productivity can be used in your marketing brochures. That's an example of a difference between whether it's a direct or indirect benefit. There are a host of other benefits: training and support

costs, service costs which can be a huge, huge piece. There are a lot of service contracts and the profit margins on them is very small. Think about the relative cost for any particular service call and the fact that with usability in a product, those costs can be cut down by say half, maybe 2/3. This allows us to bid a much more competitive service contract and keep up profitability.

There are a few key points in constructing a cost-benefit analysis. The first is to identify the expected benefits. This means actual expected benefits and cost that is looking at the life cycle of the product as it will affect your organization. This includes all of the people who will need to touch it, use it, support it, and also the whole work flow of your business and how those products are going to touch it. There are two different techniques for then analyzing the relationship between the cost and the benefit. There's a host of them that are based on the time value of money and those take a little bit longer to work through. We will look at some examples that really talk about pay-back periods and cost benefit ratios. That's the second classification. Most of our managers have to weigh the decision to invest in one of a number of projects and there's only so much money, once the information is available about the relationship between the two, we can help to build the decision towards funding a particular project that we are interested in.

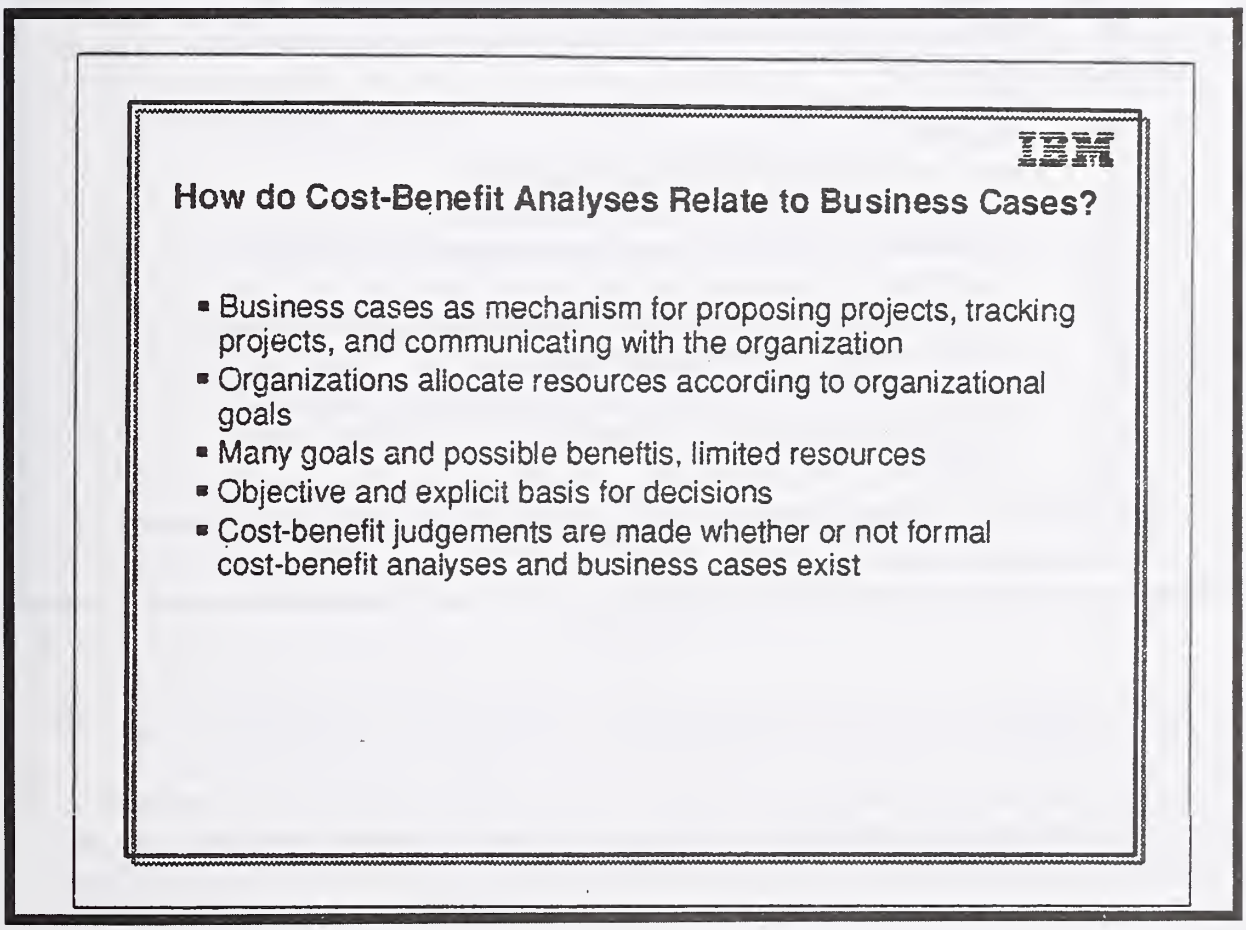


Figure 6

Business cases are a way to communicate the status of a project at any particular time in its development. One of the key points is that people are going to make a decision and make a

judgment about the value of usability whether they have quantitative or qualitative data to back it up or not. So, it really behooves us as practitioners to collect that information and make it available, not only for our own position but also to understand what the optimal amount of coverage is for the organization, i.e., should we have a centralized team? Should we decentralize in certain areas? Does it make sense to have a lab on site? Can we cost justify that in terms of the amount of work? These kinds of decisions can start being made once some cost benefit information has been gathered.

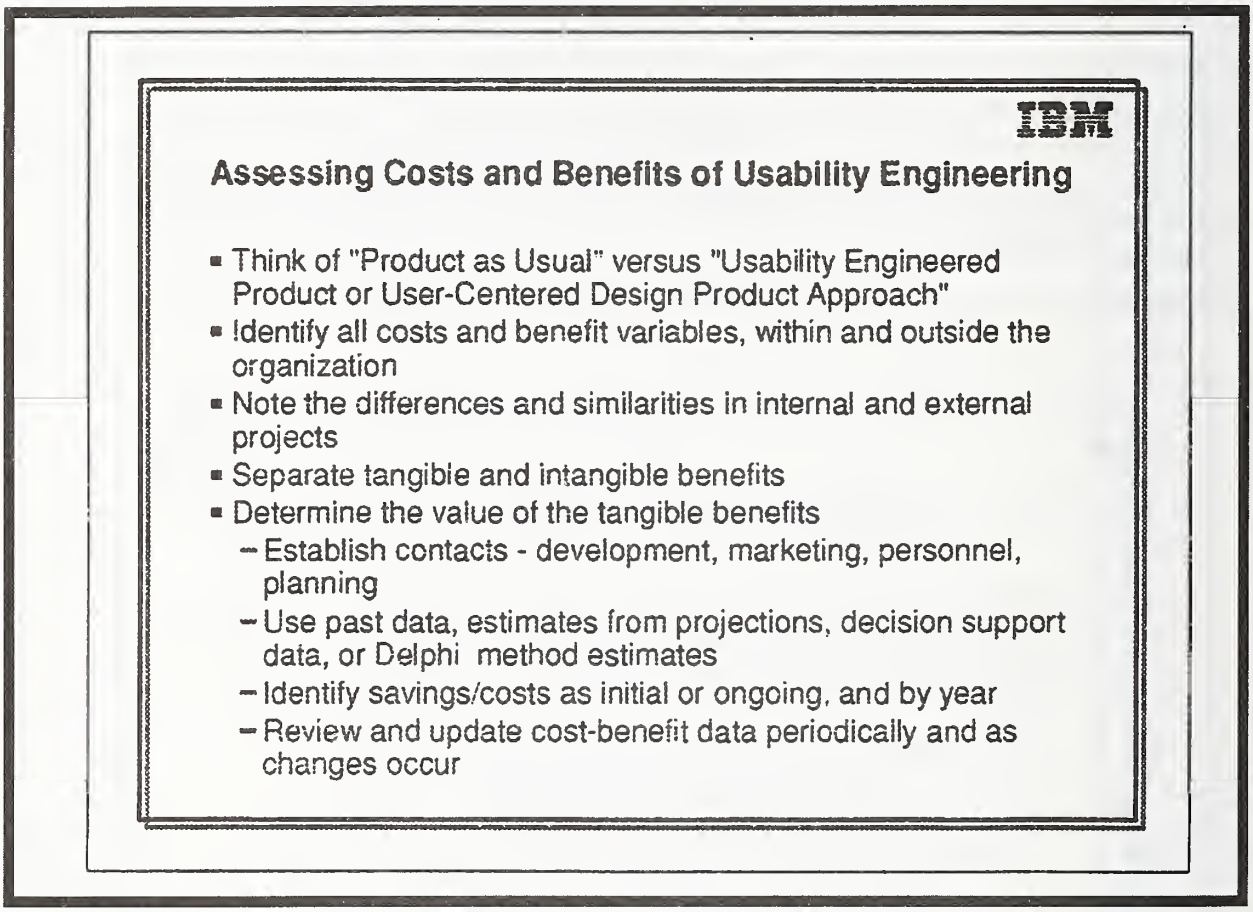


Figure 7

At the heart of the analysis we really want to look at how the delta we are offering differs from the way the project proposal would be without the usability activities included.

Think of a proposal as it would be with and without this work. For those where usability is already a critical part of the development process, you might be thinking about starting to use new techniques. Not everyone is using participatory design sessions to design user interface. Maybe some haven't tried using field tests rather than laboratory tests or usability walk throughs on site.

Think about using different techniques to get different types of benefits and use this type of analysis to understand the value. This is accomplished by identifying the cost and benefit variables both within the organization and outside. This goes back to really understanding the

business work flow to complete the task and where this product is going to touch different parts of that process. Recall differences between internal and external effects. The increase in productivity, in the internal context, was one example of how (developing a product for use in your own organization), is a direct benefit. That means that your group can complete more transactions with the same amount of personnel.

It depends on what the real issues are in your organization, but the idea is to really understand all the variables and understand whether the effect will be direct or indirect. In quantifying these benefits, I've gone out and talked to people in different areas. When I needed personnel costs, I went to personnel and asked them, "Tell me what does this mean to be fully burdened? What do I need to include?" When I needed planning data, I went to the planners. Cultivate contacts like this yourself. They may have some project data from a past project. Maybe it's close enough that you can make an estimate. Over time, you'll get better and better at estimating.

I've also on occasion used the Delphi method. This means getting a group of individuals who work anonymously with each other to come up with a group estimate of what a particular cost should be. That's another technique. It is critical to understand whether the savings and costs identified are one-year costs or will be annualized. Last, be prepared. The world changes. I don't know a single time when I've started a project and what I had planned actually occurred as planned. Things happen in the world. Something will happen. A new requirement will come up. You'll have to build in this piece of function that you hadn't even been prepared to do before. So you need to think, go back, revisit your plan, and make adjustments.

OK, I'd like to talk to you about a success story. This is a project I worked on at IBM a few years ago. The first thing I did in this project was meet with the team to learn what this application was about. It was a security application that we used in combination with our customer data bases, so it was pretty important. I learned from them what their ideas were about the design of the application as it was then, and their usability objectives. They basically thought that without any training we could take our group of users, some 25,000 people, and have them learn to use the product, error free, on their second or third attempt. That's great. That's wonderful that we can specify that usability objective. We then took a look at that interface they had sketched out, and I built a set of screens with it (charts), went out into the field, into the business situation where these users were and I acted like I was a computer and gave them a path. I gave them the first chart and said, "Here take this pen. Write on there what you would do if you were sitting at your computer." They did, and then based on the information received, I would flip the charts to give them the next screen.

I had an assistant with me who was doing some time on task timings and recording some other data. At the end of one of these tasks, I'd stop and take a few notes myself. What I learned from this field test was that the initial design had some problems. The developers felt that the majority of these users would be able to learn this task the second or third time, and what I learned was that only 20% of the users could complete the task at all, and of those, they had a lot of trouble with it. I went back to the team with that information. I basically had identified four large problems and it was illuminating to them. The assistant that I had taken with me was one of the section leads on the development team and it was one of those amazing experiences for this person to see up close and personal the difficulty a user could have with the system. So I had a little buy-in there already when I went back to the team meeting.

What this opened up was a discussion on putting in a help desk. We're not going to add any training materials but we'll have a help desk. That was a solution, but who's going to staff this and who's going to pay for it? They estimated it would be at least a few people for a few months to handle this help desk and nobody had money for it. It wasn't planned. It wasn't in the budget. So I suggested what we could do is redesign the user interface. They thought that was a great idea. We took the four major problems and of those, three of them we could solve. We did the redesign, we put it back into this low-fidelity prototype and the fourth one was one that happened to be a navigational problem and we couldn't find a low cost way to solve it. There was one solution we knew about but it was going to be so costly that the project manager said no. Go back and do another field test. If we see that problem again, we'll face it then. I went back and

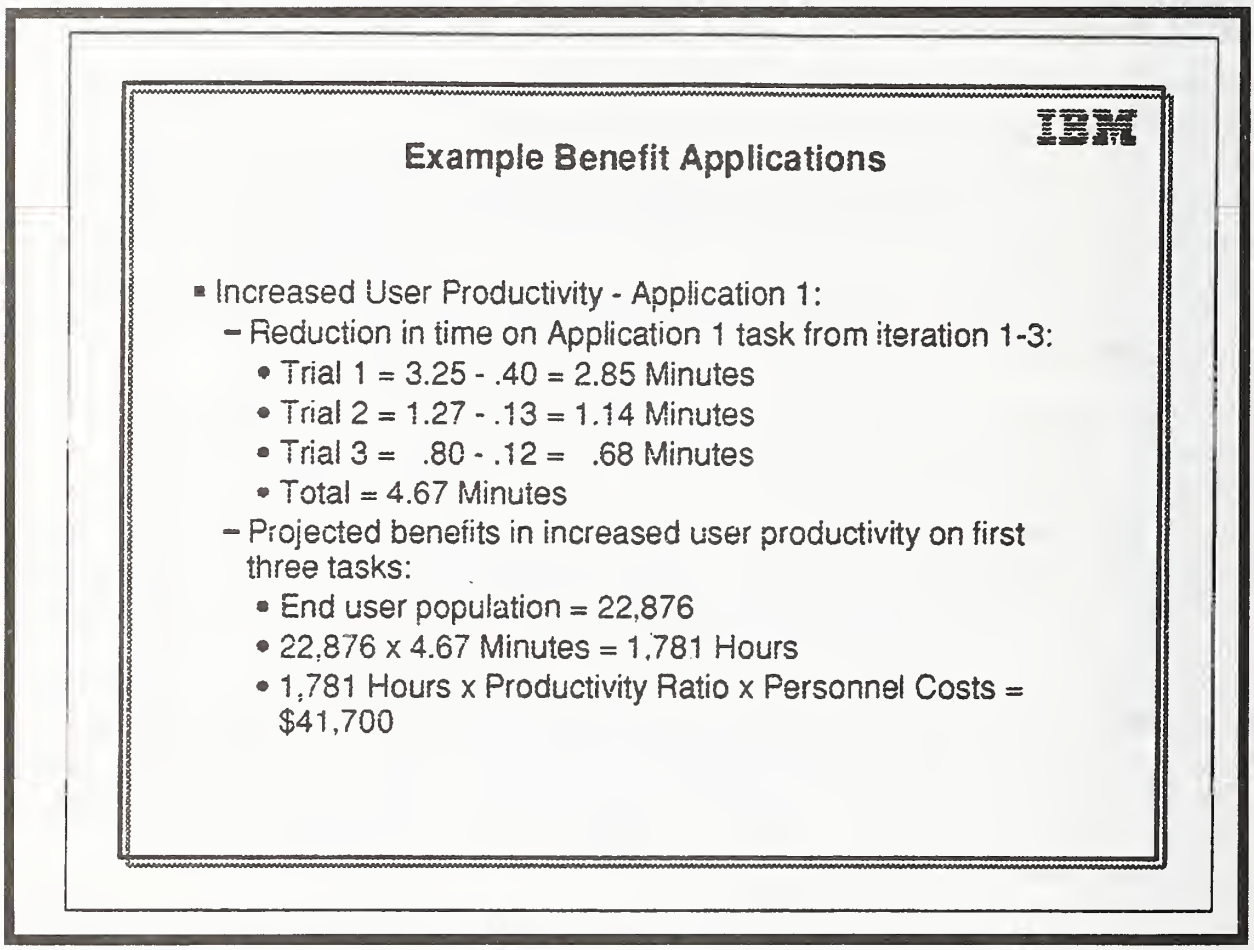


Figure 8

did another iteration of testing and what you see in Figure 8 is the time on task. This is the first field test that I did and it's basically a user's first attempt, second attempt and third attempt at doing a task. The goal for this application is to have somebody get through this security system in about six seconds. As the data shows we were taking over 3 minutes in the beginning with the first user interface plus the fact that only 20% of the people could actually do the task. On the second iteration, we saw an improvement. We brought the times down, but we ran right up against that navigational problem that we had identified the first time around. So we went back.

This time, the team had really begun to buy into the idea of usability work in this user-centered design and one of the fellows on the team figured out a beautiful way to finesse this issue. He was one of our gurus who had been around who knew the systems very, very well, and had been there years and years.

The programmers up to this point (while we've been doing this user interface design work), had been doing all their other under-the-covers programming. At this point we said go ahead, code the user interface. Trial 3 was our test of the system on a test machine. This was with the actual code that we are going to roll out. And we were quite pleased. By this point we had people bumping up against what was expert use on the system. This is the performance on the system by one of the programmers who knew the system back and forth. He knew how to go in and make it cruise. So, we were able to get the users to meet the usability objective. By the second or third time we actually had 95% of the users doing this error free and within the kind of time that we had set. We rolled it out on time, under budget and with very high user satisfaction with very,

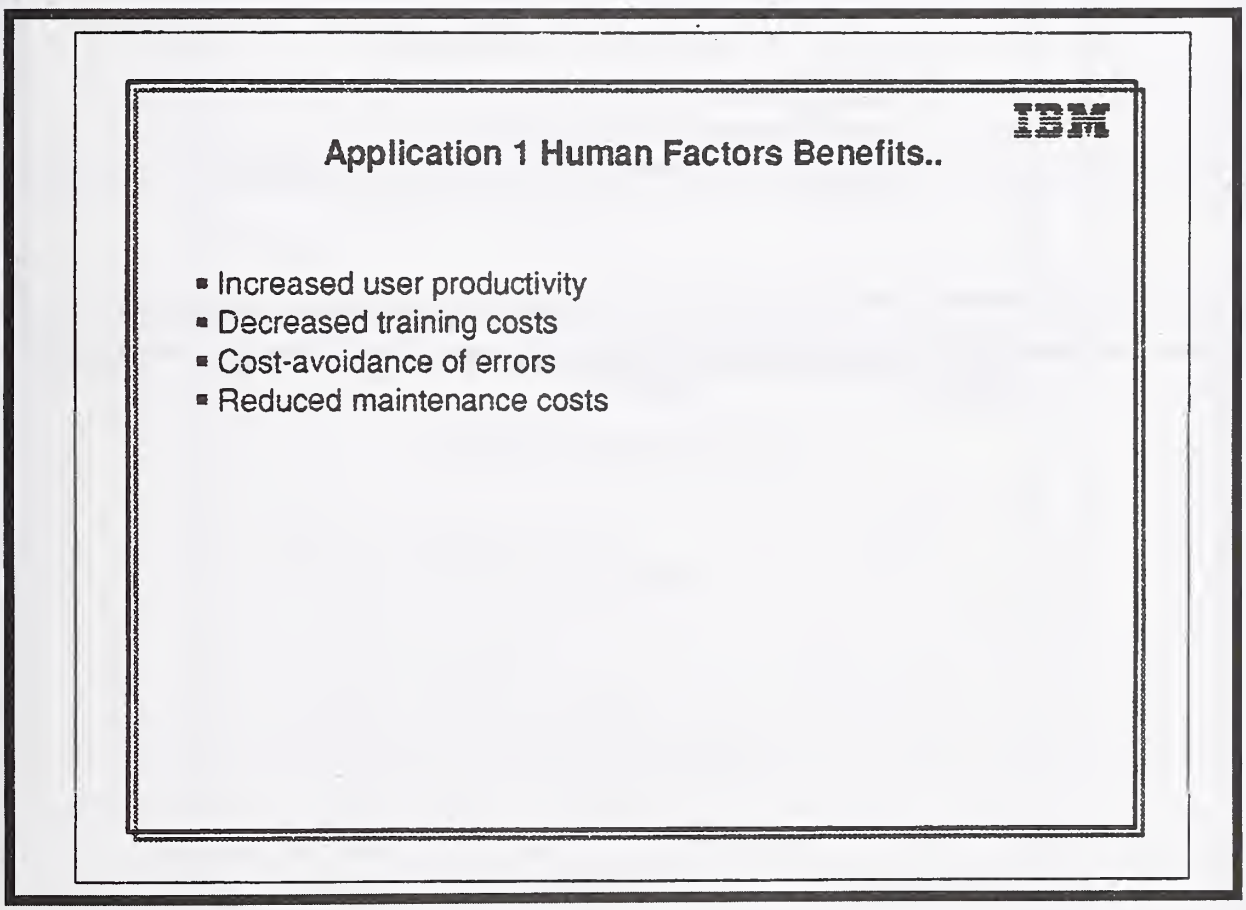


Figure 9

very little improvements or modifications to the system after it was released. As a result, all of us received bonuses. Now I really had the team with me. This was the first experience and this is only one data point. This has been published before, and I offer it as an example for you to

take to your own groups, to your own management. In terms of the value of usability, I basically took the difference from this top line to the third line all the way through and quantified that as a financial measure to say this was the value of usability.

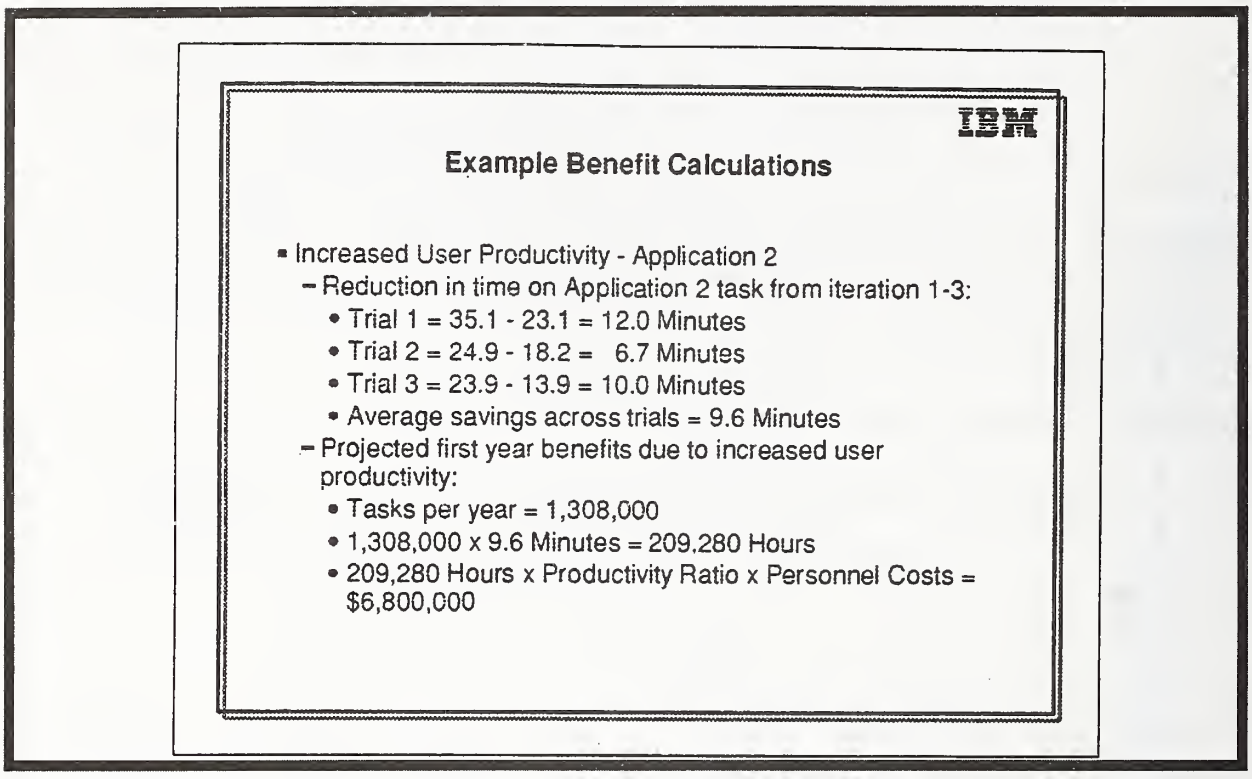


Figure 10

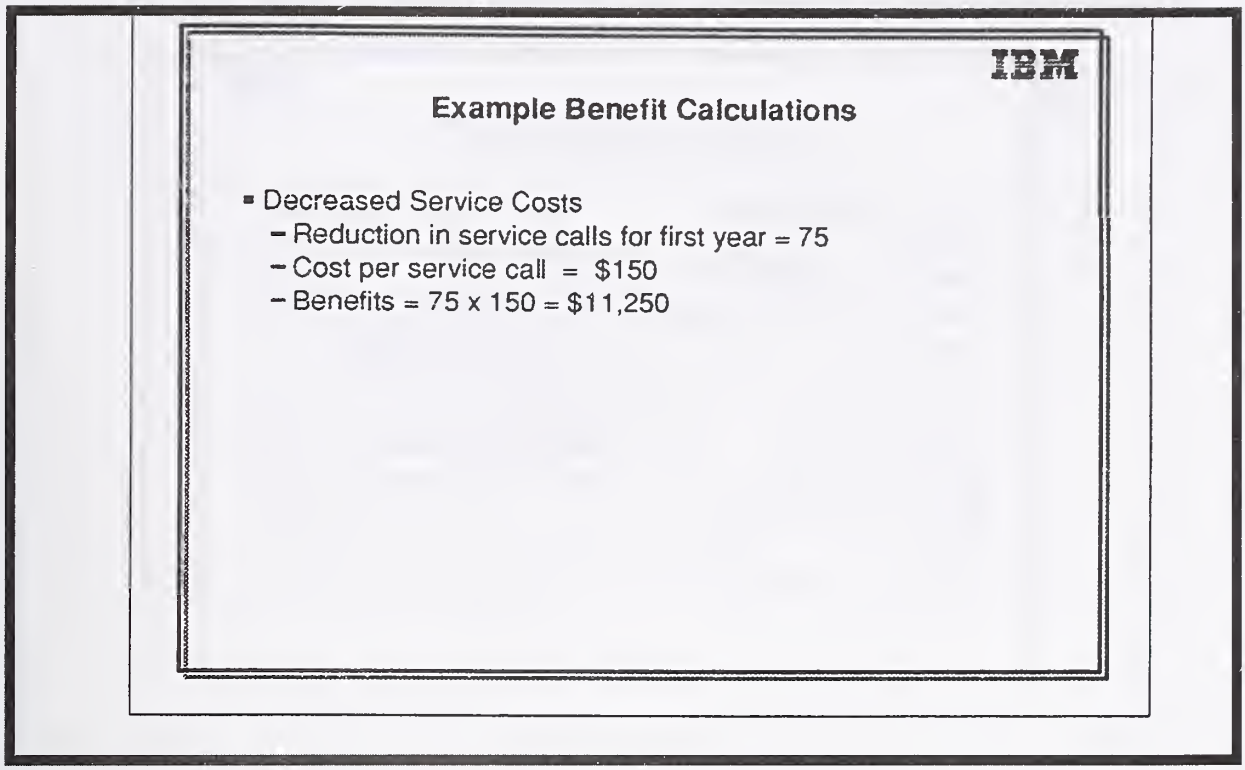


Figure 11

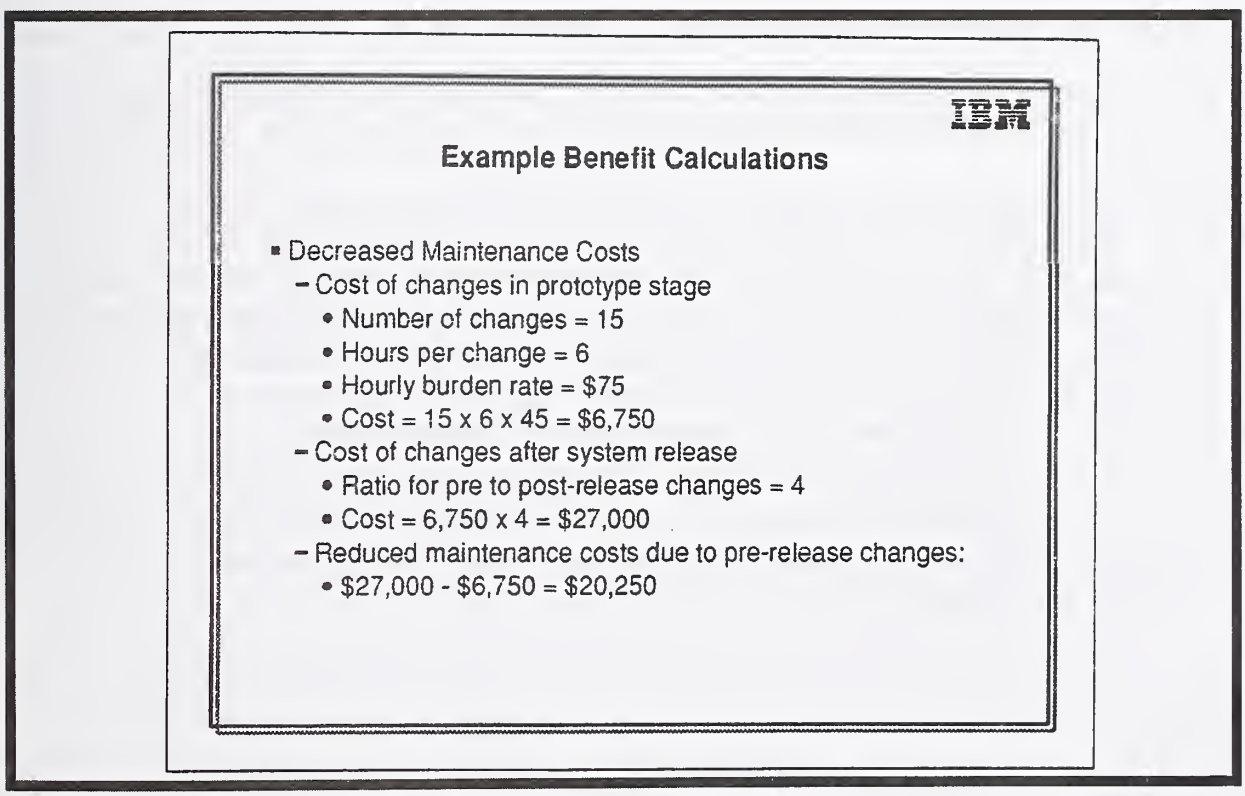


Figure 12

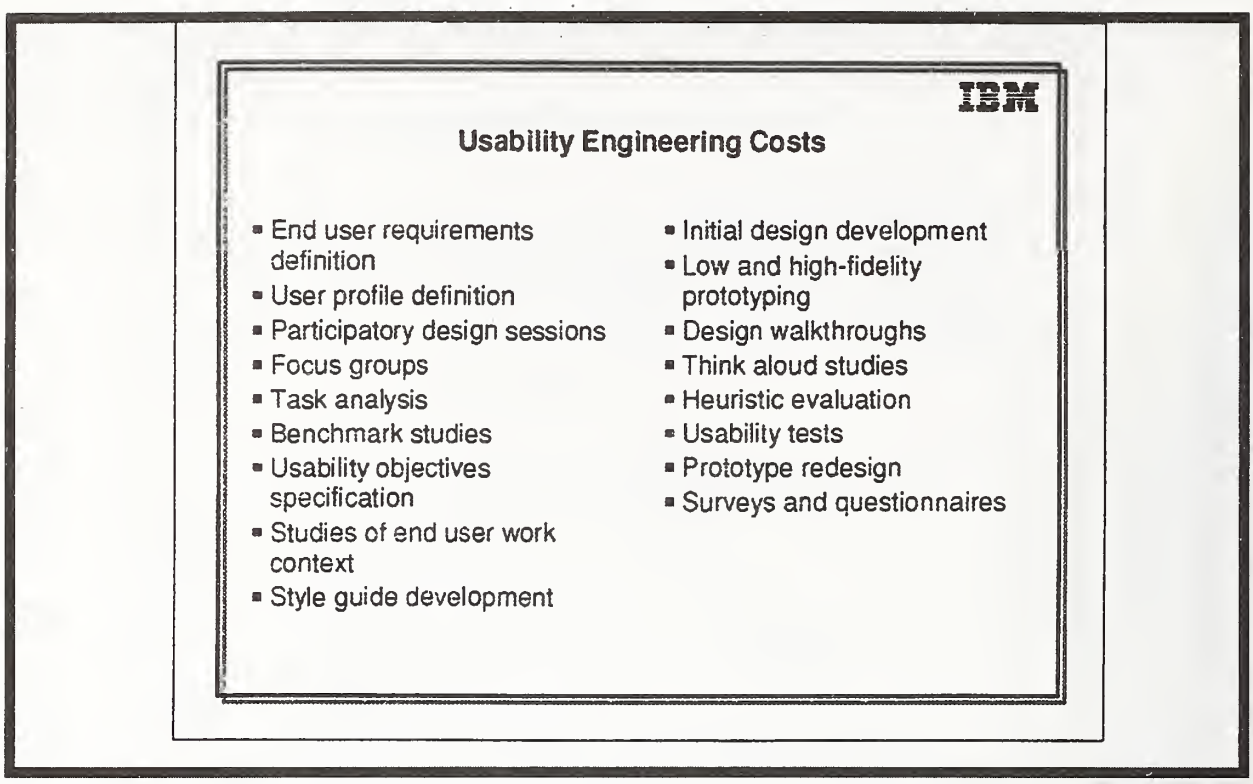


Figure 13

Audience Question: How about the ones that couldn't do the task at all? Did you quantify that?

Karat: At the end we had something like 98% of the users who were working error free. Everyone could complete the task. Some of the users would make an error on their first or second attempt.

Male: Did you quantify those people that could just not do the job without errors?

Karat: Yes, in fact, that was part of the first team meeting. This is our customer data base. These people are going to be using this system while they are on the phone with customers, this has to work. There are real customer issues if it doesn't work and if they go in and make a mistake and have to recover from that error, from that mistake with the customer. It just compounds, so yes, we had an estimate for that as well. That's pretty high.

So here's the value of the usability work that you saw in that diagram and it's basically looking for the value of that increase in productivity across the first three attempts. This was a system that the users would use about 12 times a day. So I took a very conservative estimate of the value of the work. I came up with this $42,000 figure.

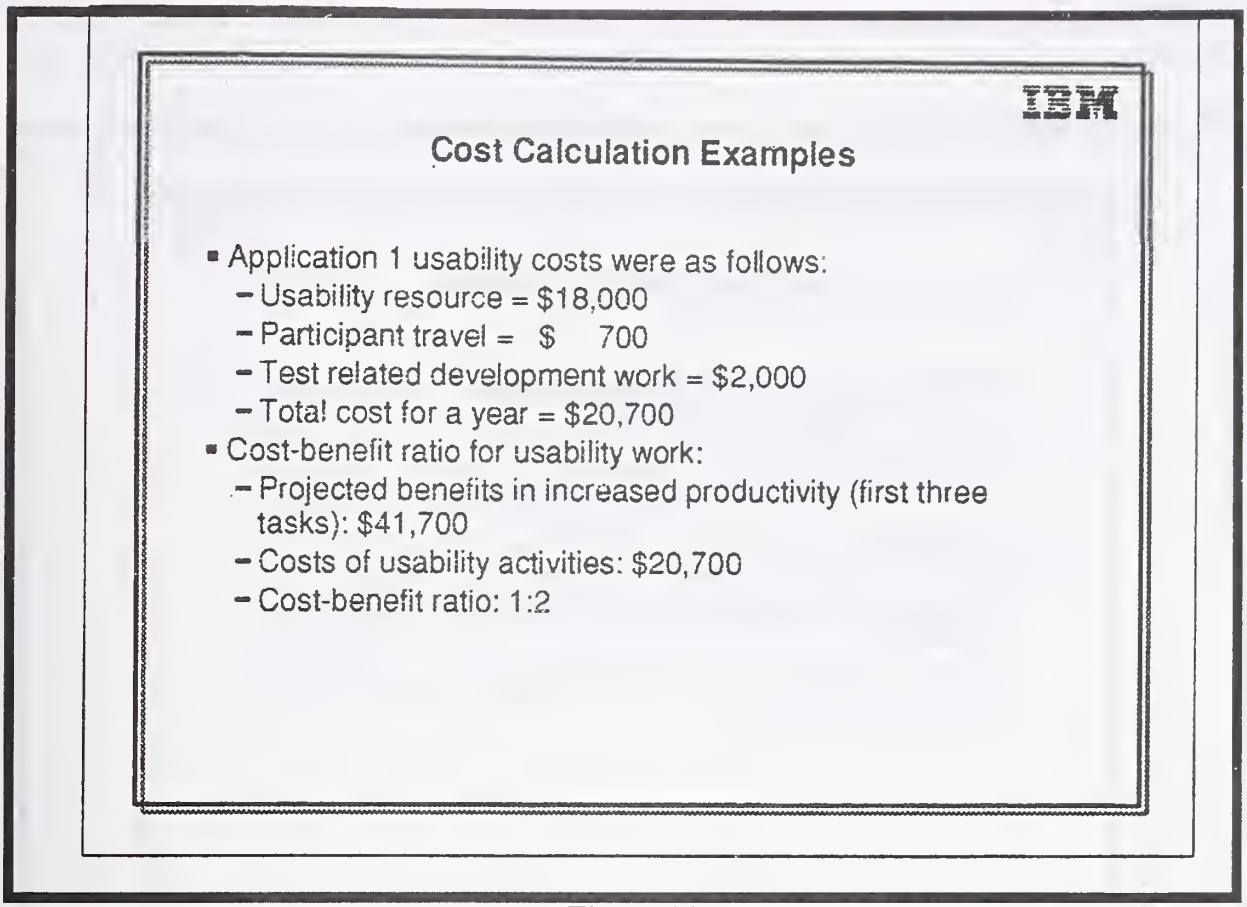


Figure 14

We'll come back to this in a minute once you see the costs and the next chart gets to this gentleman's question about what was the cost of the errors?

In that team meeting where we really looked at what our options were, we looked at the issue of increased productivity, what it was going to take to set up a help line and have some kind of documentation that we gave to everyone. As it was, the system we rolled out was "walk up and use". The only thing they saw was what was on the screen. That was it and they were able to easily make the change. We were then able to see the cost avoidance of the errors and the fact that this work was happening on the phone, real time with our customers and this huge reduction in maintenance costs. Based on that, I started getting some momentum. Just like Bill was saying, you start from the grass roots and then build momentum. The next system we tried this on was the much larger one.

By the way, that first system was built in about 7 months. It was a rather small development effort, something around a million dollars. This is a multi-million dollar system that we built over the course of two years and in the terms of the value of usability, the key measure for us on this one again was increased productivity. This is a transaction system where we really needed to be able to have a lot more throughput, and this is the value of usability for the first year the system was rolled out.

This system has been in place a number of years now so it's a hefty payback. Let me go to the costs now.

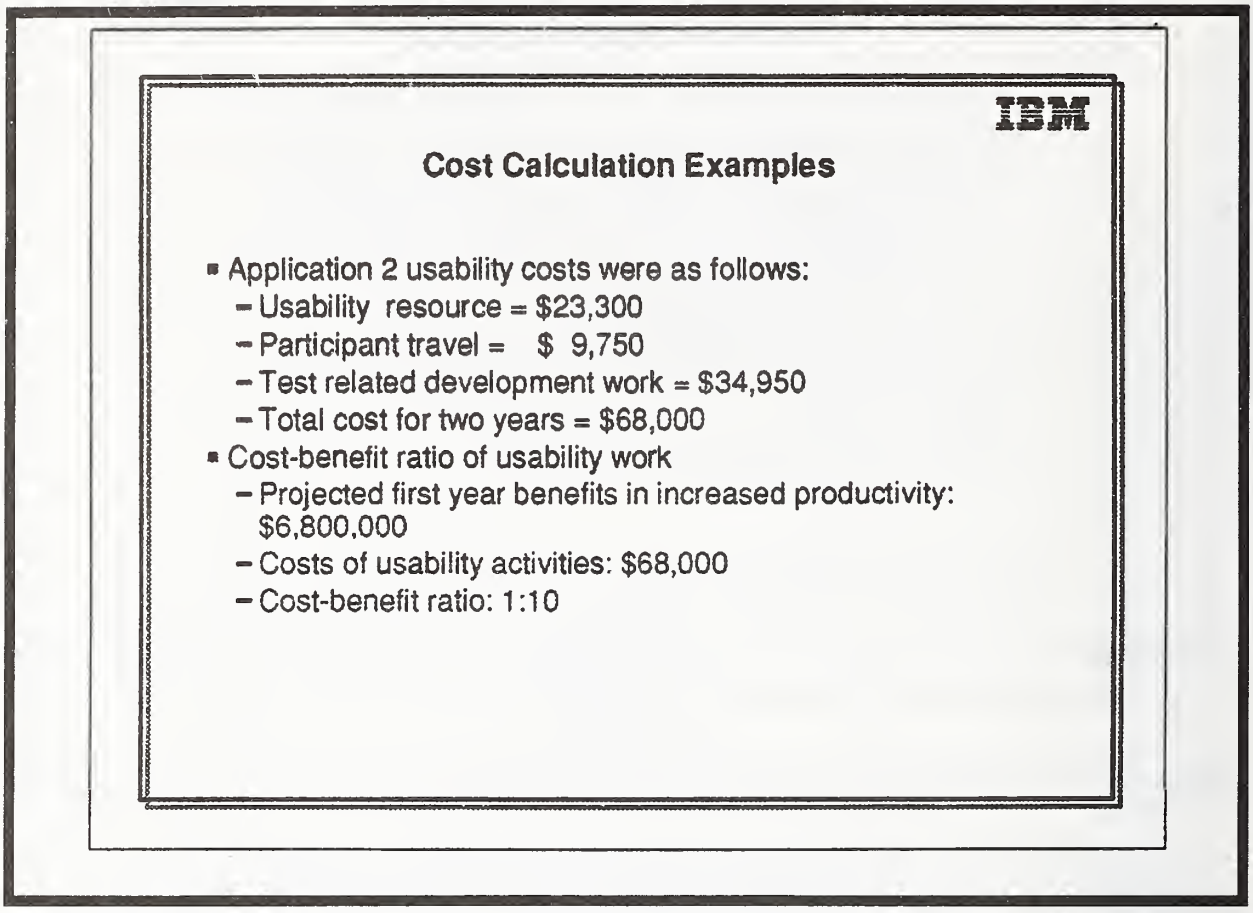


Figure 15

A couple of people have mentioned different types of activities that go on, and, again, I think this is tailored to the particular situation you're in, the type of usability goals you have. Remember the question from the woman over here who was asking about is there a handbook or some type of guide that can step you through this step by step? I think one of the reasons there is not is that each particular situation is unique and you really have to go in and look at what the user is doing, the work environment they are doing it in, their task. What is my goal here? What do I have to achieve and then with that you turn around and look at this tool kit of methods and techniques and decide what to do. A number of us have written chapters that give a general overview of usability engineering methods such as: it's very good to start with low fidelity prototypes and then go to coding interface. But in general it's very nice to have usability walk–throughs from the beginning, heuristic evaluations and usability testing milestones.

In terms of a real kind of step by step method you could use every time, I don't think that really fits our field, at least not right now. Here's the tool kit of some of the activities that can go on that might be necessary to be able to meet the business needs that you have in a particular

situation. There are of course different types of costs associated with each of those. Let's go back to that first example of the 7 months development effort.

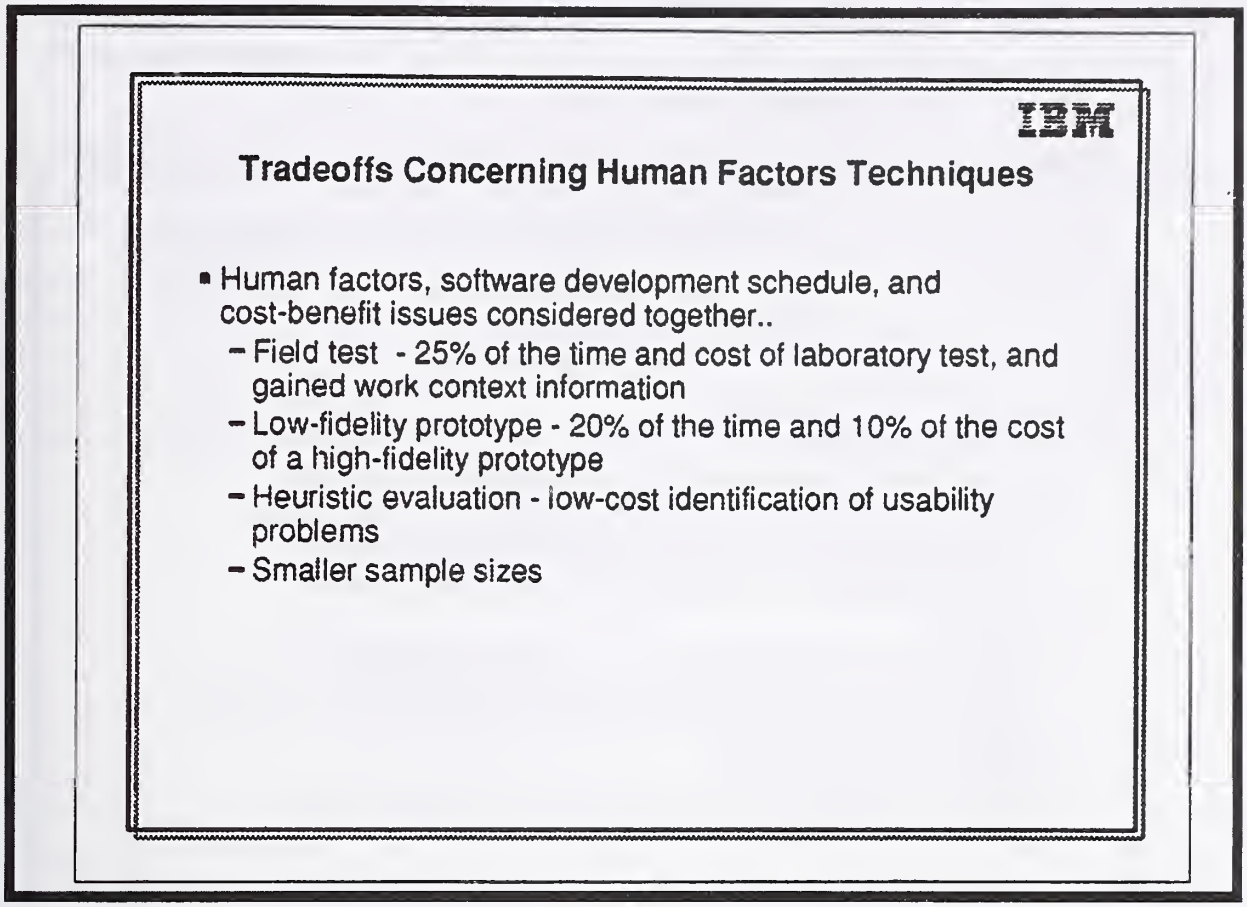


Figure 16

I worked on this project part time. I had responsibility for a couple of other things. What you see here basically is the breakout for my time from participant travel costs as I was going locally working with these different people, buying them lunch and stuff so that they would let me work with them and it was just a small bit of development work. This was for doing the final testing with the real code on the test machine so the product manager actually did give me a little bit of support there towards the end for a total cost of about $20,000. Remember the benefit just for the first three times the users used the system was about $42,000 so even with this limited amount of involvement I was getting a one to two pay back. For some situations where it would probably be more typical in your arenas with much larger systems, this other one (Figure 15) is probably more appropriate to look at.

This was the transaction system that was built for this effort. There's my usability resource again. (Again, I was part-time, working on a few projects at the same time.) Here I brought people in from different parts of the country. This is a system that's used throughout the United States. I wanted to make sure I really had a good representative sample. For this system (Figure 15) as well as on the first one, I made these low fidelity prototypes with charts. I pretended to be

the computer. For this one we actually thought it was worth the money and the time to build a live prototype so we had some programmer time to build that. Cost of the work was $68,000. Remember this was first year only savings, almost $7 million or so for a pay back of about one to one hundred. And I would venture that this is more the kind of ratio that you would be able to achieve in your areas. (Figure 16) For the practitioners in the room, here's some of the cost data that I collected while I was doing these couple of studies.

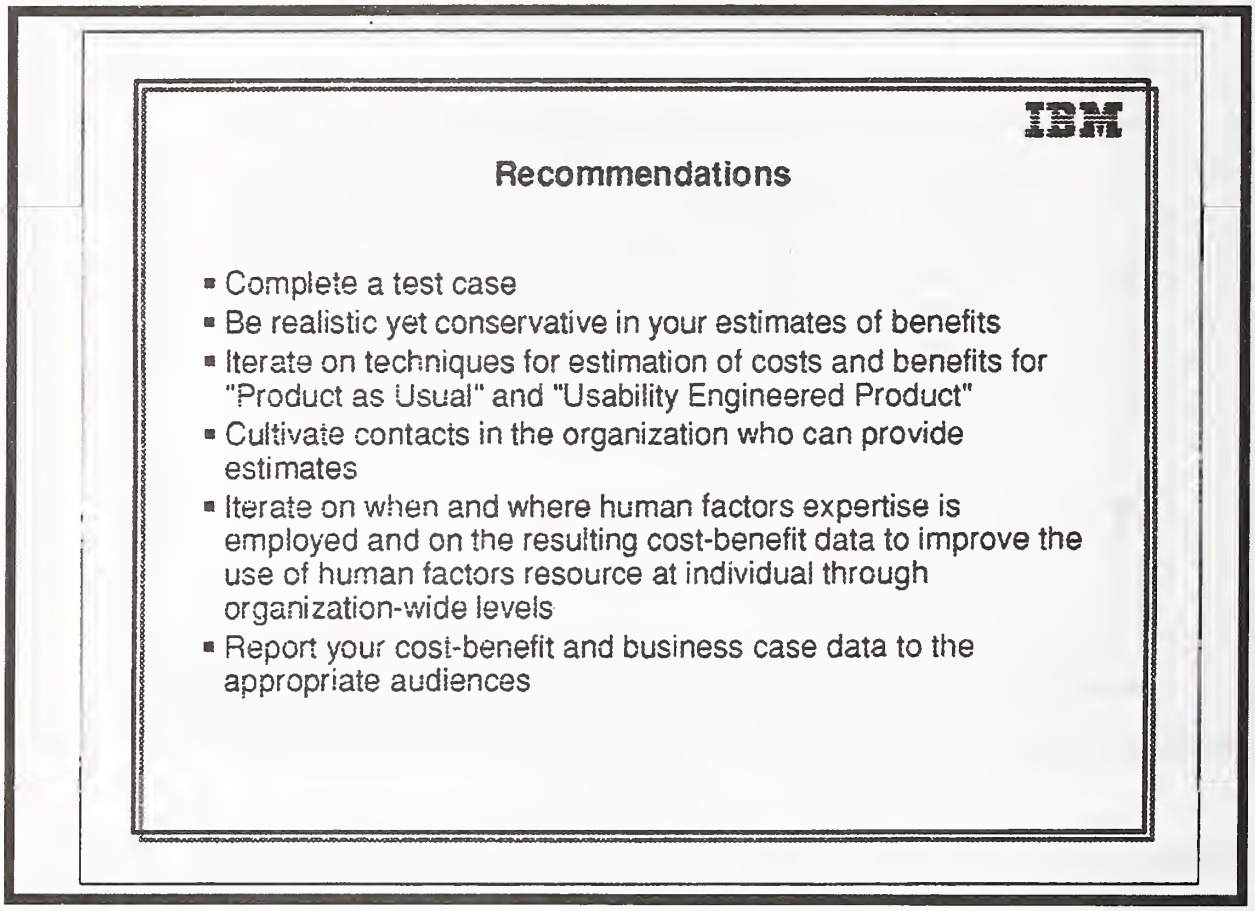


Figure 17

You know for me as one person to go into the field to where these people were working I was able to do that much faster at much lower cost and I urge you to consider doing that yourselves. Again with the prototype and again it depends on how complex the system is you're trying to represent to the user, but in that first system I could do that easily, I could do it quickly, cheaply, and get a lot of benefit from it. This is just an item to let you think about using other techniques like heuristic evaluation which involve the use of usability experts to review your prototype rather than having your representative users actually reviewing it and then to think about your sample sizes. In the last few years I've heard some wild things. People bringing their sample sizes down to 2 for a test. I think that's a little bit small but there's a lot to be gained by talking to a few more users and what I generally like to say to people is think about the probability of a serious error in your system. You know what is the probability that that might occur with any particular user? If it's something like 10%, give yourself enough of a chance during your testing

to have that error pop up. There was one incident that I had on one project where I ran 10 users in one iteration and it was the 10th user who not only hit the problem but then was able to describe the solution to me and that really stayed with me. I keep my sample sizes around 6 or 8 for an iteration but I know there are a lot of companies that are working with about 2, so you might just keep that in mind.

And to finish up, I really advocate that you start collecting this usability data and that as project managers and managers making decisions in this area, that you request it, in fact demand it. There no reason not to. You're asking for trouble if you don't have usability as far as the critical path and you don't have the associated quantitative data with it.

I'd urge you to be conservative in your estimates, again build that network so you can make better estimates and over time share information with them. Build your own expertise and those of your peers in the organization that you can work with about what techniques to use, where you get the best return. I know a lot of this is classified information, but where you can share it with people, de–identify it so that we can all build our methodologies in this area. Thank you.

Making Sense of HCI/GUI Standards and Guidelines

Elizabeth Buie

Computer Sciences Corporation
System Sciences Division
Laurel, Maryland

Ms. Elizabeth Buie, Computer Sciences Corporation
Making Sense of HCI Standards and Guidelines

I'd like to just say a word or two first about what Clare-Marie just said.

In government systems a lot of times the cost of errors is not in user time but in human lives if you think about air traffic control or military systems. So, we also need to factor that in and about government systems becoming competitive even in areas that you would not think. I work in areas, spacecraft flight operations for example, and NASA has been losing business to universities for building spacecraft control centers and you would not expect that so we're having to become more streamlined and usability can help a lot in that area.

OK, here's an agenda I'm not going to go over.

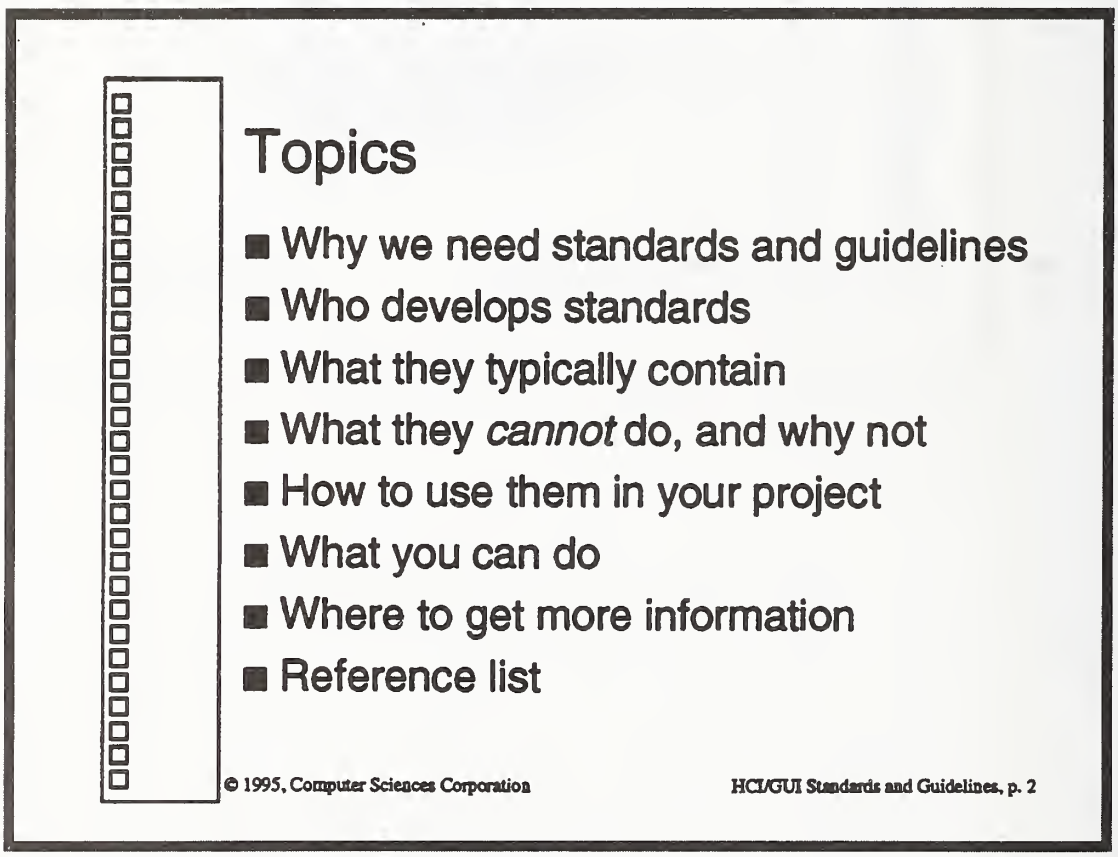


Figure 1

I'm just going to get right into it because we're pressed for time. Why do we need standards for human computer interaction, interfaces? To standardize the look and feel throughout an application or project or across applications or projects if you've got a lot of users that are going to be using more than one. Standards embody a large body of human factors research and that's practice in HCI. Standards and guidelines can help streamline design by making look and feel decisions in advance.

They can also help (in some cases they are required to comply with the law, for example, in many parts of Europe) with contractual obligations—a lot of military contracts have specific requirements for complying with certain standards or guidelines.

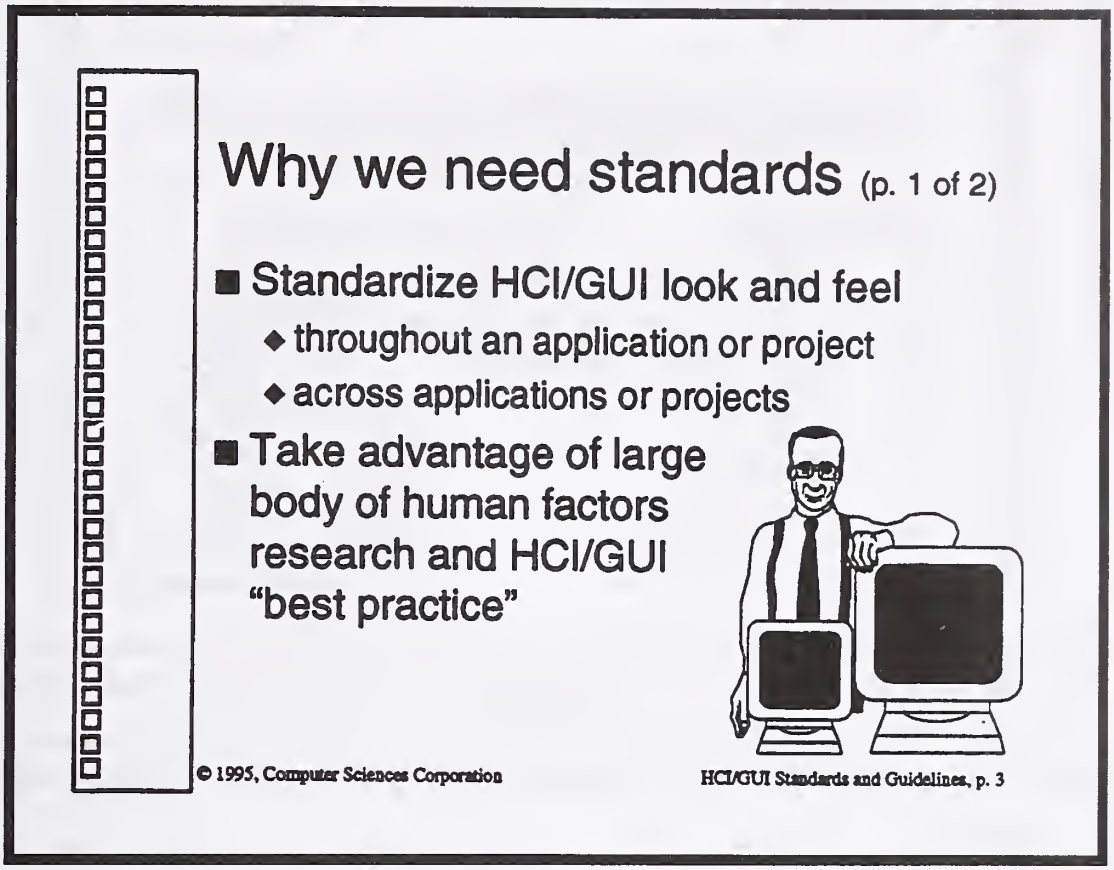


Figure 2

What kinds of groups or organizations develop standards? Well, the biggest one is the International Organization for Standardization and they've got a number HCI standards, the biggest known one is 9241 Ergonomic Requirements for Office Work with Visual Display Terminals, all this is in your handout. There are also national organizations. ANSI has had one on hardware for awhile, ergonomics and work station hardware. We're in the process of developing one for ergonomics of software user interfaces. That should be out in about a year for review. There are other national bodies, DIN is the German organization. There's the British Standards Institute. There are military and government organizations. DOD as a whole, MIL standards I think come out of the Army. There's NASA. There's the Air Force. A variety of government organizations have developed standards that apply in scope to their own sorts of projects. There's even one that has come out of the Air Force for spacecraft control user

Why we need standards (p. 2 of 2)

- Streamline HCI/GUI design by making look-and-feel decisions in advance
- Comply
  - With law (e.g., in Europe)
  - With contractual obligations (e.g., many US military contracts)

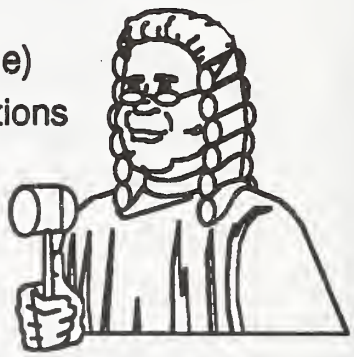


Figure 3

Who develops standards

(p. 1 of 2)

- International organizations (e.g., ISO)
- National organizations (e.g., ANSI, DIN)
- Military/Government organizations (e.g., DoD, MIL, NASA)
- Commerce/Industry (e.g., Macintosh, CUA, Motif)

Figure 4

Who develops standards
(p. 2 of 2)

- Independents (a company or person)
- Projects (tailoring them from other sources)

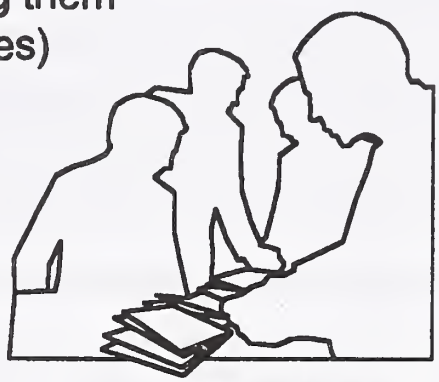


Figure 5

What standards typically contain

- Requirements and/or recommendations for HCI/GUI design features
- Rationales/principles
- Examples
- Exceptions
- References/sources

Figure 6

interfaces. There's commerce and industry, Macintosh, IBM, Microsoft, Common User Access, Motif. These have varying levels of human factors input into them but at least the standardization is a goal.

Independent companies or organizations may develop standards or guidelines. There's a company called Corporate Computing somewhere in the midwest and they have a set of guidelines mostly for windows systems and they also have a tool that allows you to tailor the guidelines to your own project. It allows you to look them up when you're developing them and then individual projects will develop guidelines or standards for their own projects, tailoring them from other sources.

Typical content of a standard or guidelines document will be requirements and/or recommendations for design features. They may contain rationales or principles why these are good ideas, and also examples of how they would be, how they would show up in the design itself, exceptions where they would not apply and references and sources for people who want to find more information about where they came from usually. Now, what can standards not do?

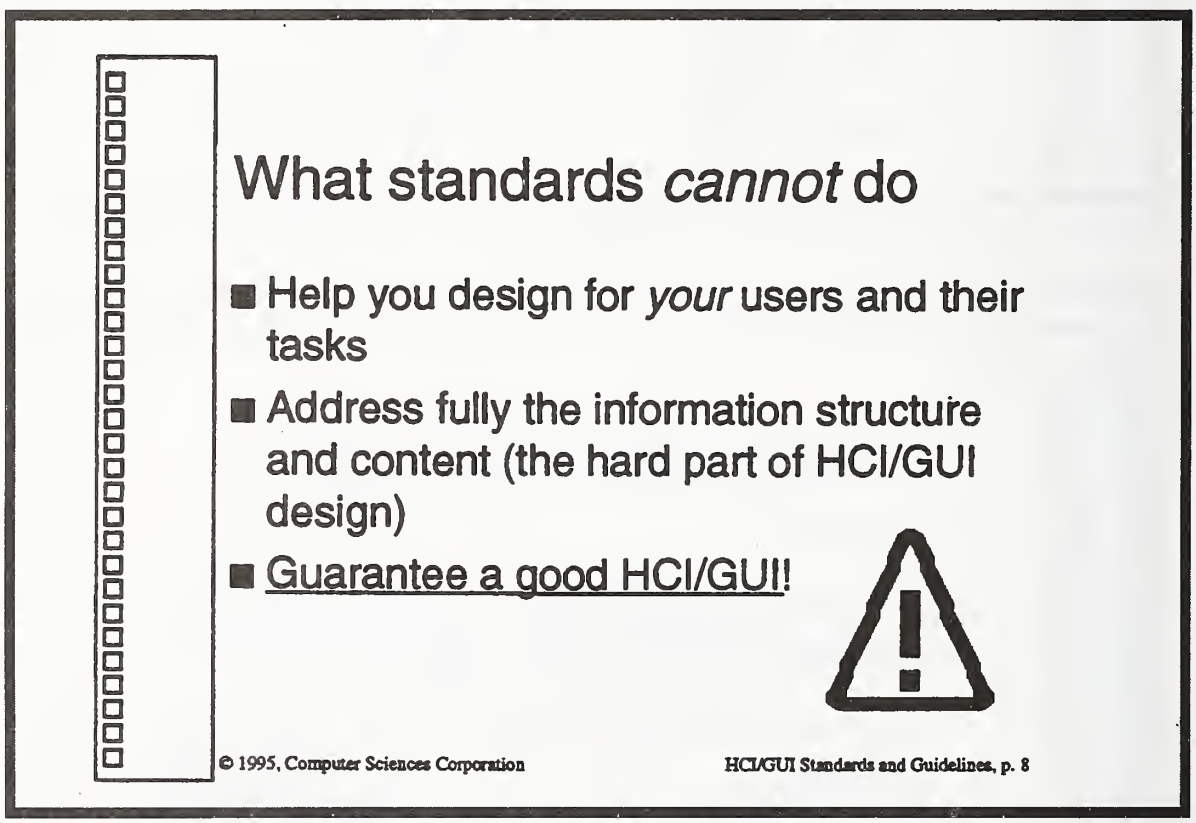


Figure 7

There's a big problem with standards in that they have a tendency to provide a false sense of security to people who don't know a lot about usability. A lot of people, especially software engineers, system engineers, will say, "Just give me a good set of standards and that's all I need." It is not. It doesn't help you design for your users and their tasks. It doesn't address fully the information of structure and content which is really the hard part of design. It addresses

primarily the look and feel and the information structure and content is more related to your users and their tasks than the look and feel is so it kind of takes care of the surface level. Jared Spool of User Interface Engineering has some choice words about standards, but as a member of a standards developing organization I have forgotten or repressed them but I appreciate that perspective. A lot of people object to standards because they say they constrain creativity and to some extent that's true. But, on the other hand, in my view, the real important part of creativity is not so much in the look and feel as in the matching between the design and the users and their tasks.

Standards applied appropriately can actually help free you up to be creative where it counts. And they cannot guarantee you a good design. You still need a process. Why not?

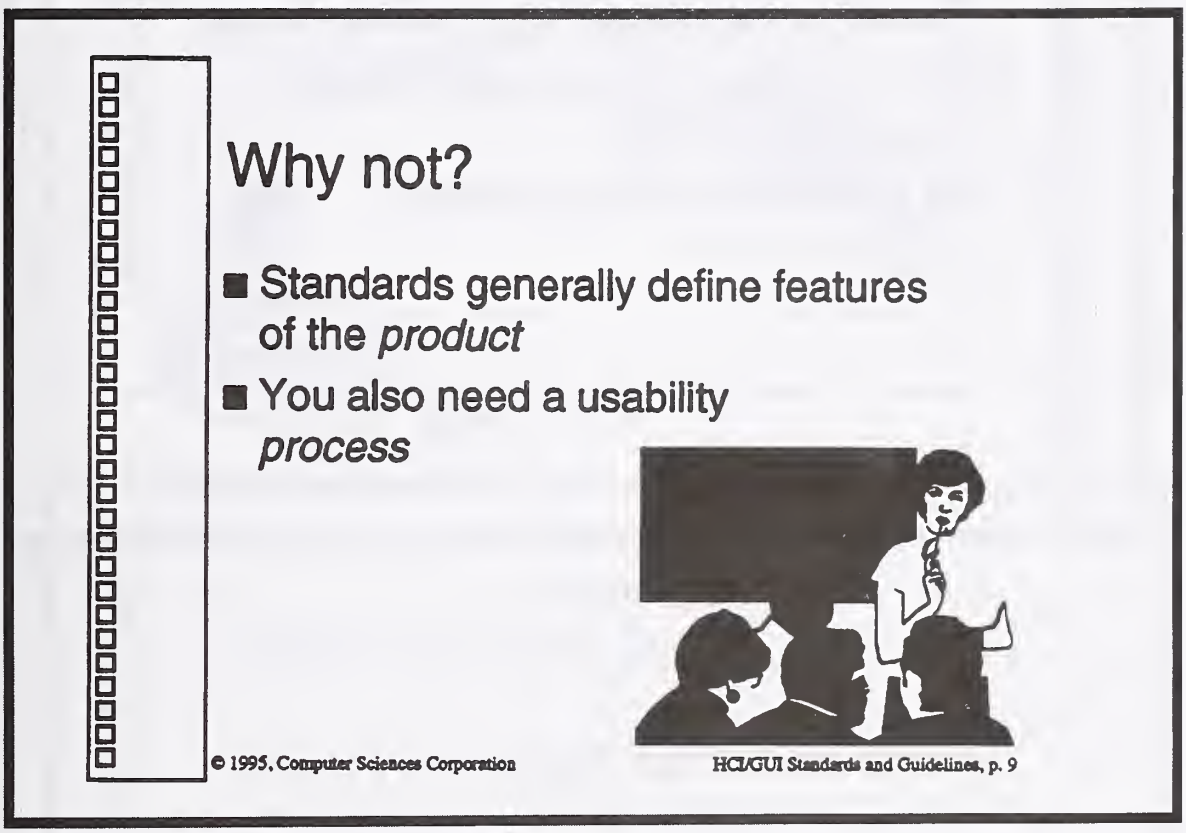


Figure 8

Because they typically describe features of the product and as Rex has pointed out, there's this whole process thing that we need to deal with as well. There is an effort in ISO which is being edited by Tom Stewart of CDSI to define a standard for User–Centered Design of Interactive Systems. But, it will be a while before that's out at a level that most of us will be able to use it. But again, that's something that will need to be tailored to any individual project. Now how can you use standards in your project?

First select the standards or guidelines that you're going to use, ISO, ANSI, military, whatever. Create a project standard by determining how they will be applied in your environment and then

apply them while you design the interface. (Figure 8) Now through that it may come up in usability testing or even implementation, sometimes software implementation reveals issues that

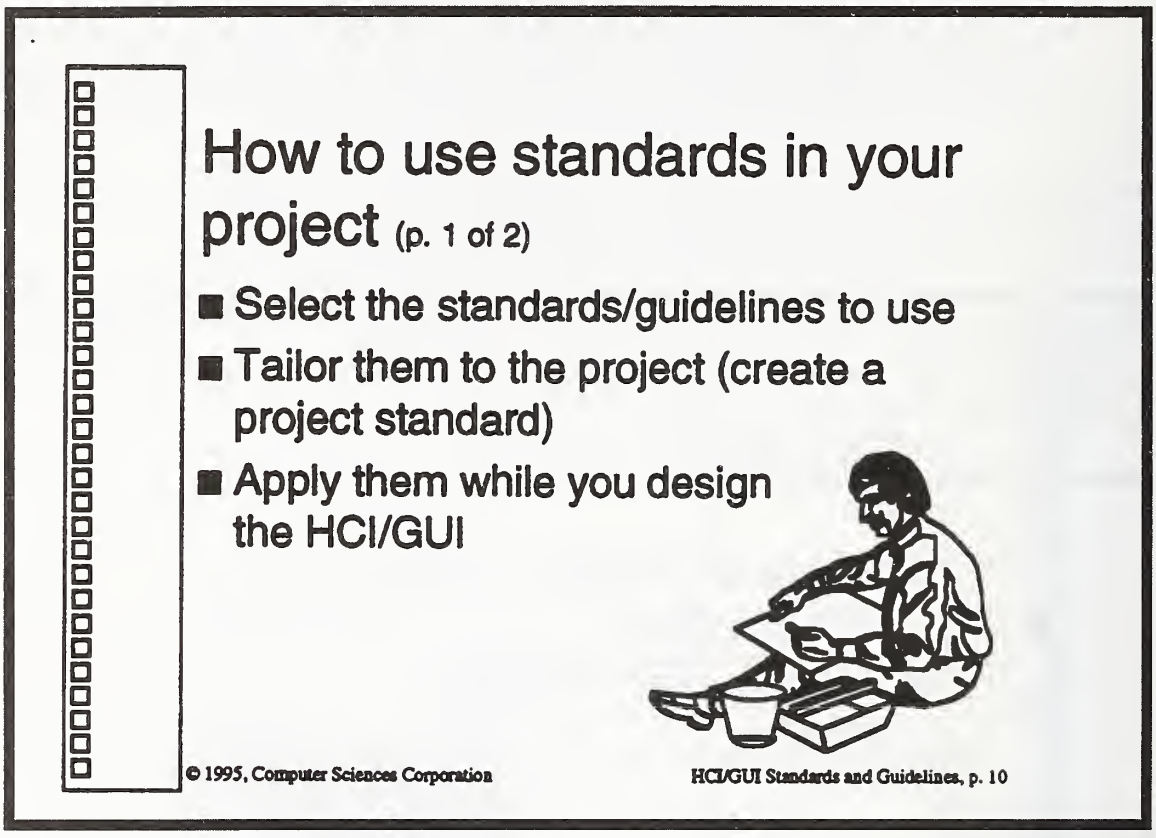


Figure 9

How to use standards in your project (p. 2 of 2)

- Refine the project standard as needed
- Inspect the HCI/GUI design and implementation to verify compliance

Figure 10

What you can do

- Bug your management to incorporate HCI/GUI standards into your organization
- Encourage procurement officials to specify HCI standards in the RFP
- Keep up with developments in the field; keep yourself informed

Figure 11

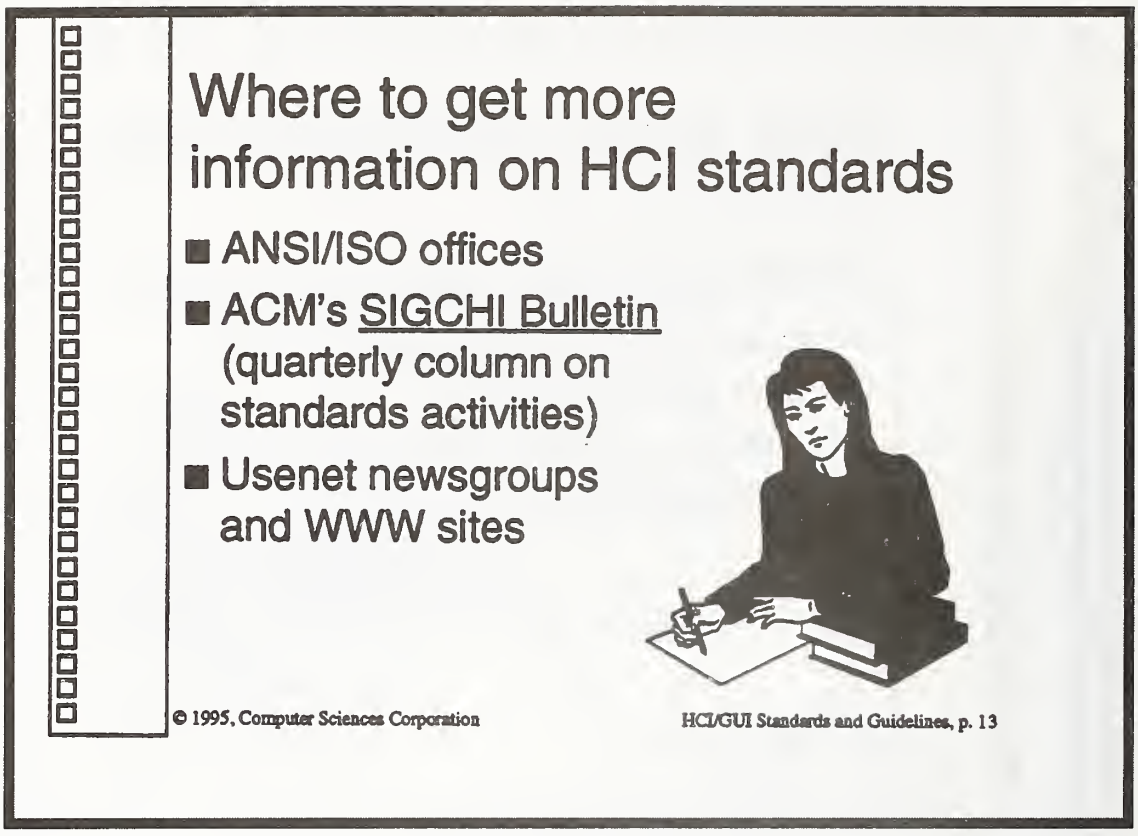


Figure 12

design hadn't thought about and so you need to go back and revise your interface design based on input from the software engineering. Usability testing may reveal that your application of the standard in your project wasn't quite what you needed so you may refine the project standards, and then as you do, your heuristic evaluation or final refinement, final verification of the design, you want to inspect the design and implementation of the design to make sure that you actually have complied with the standards that you've developed. (Figure 9)

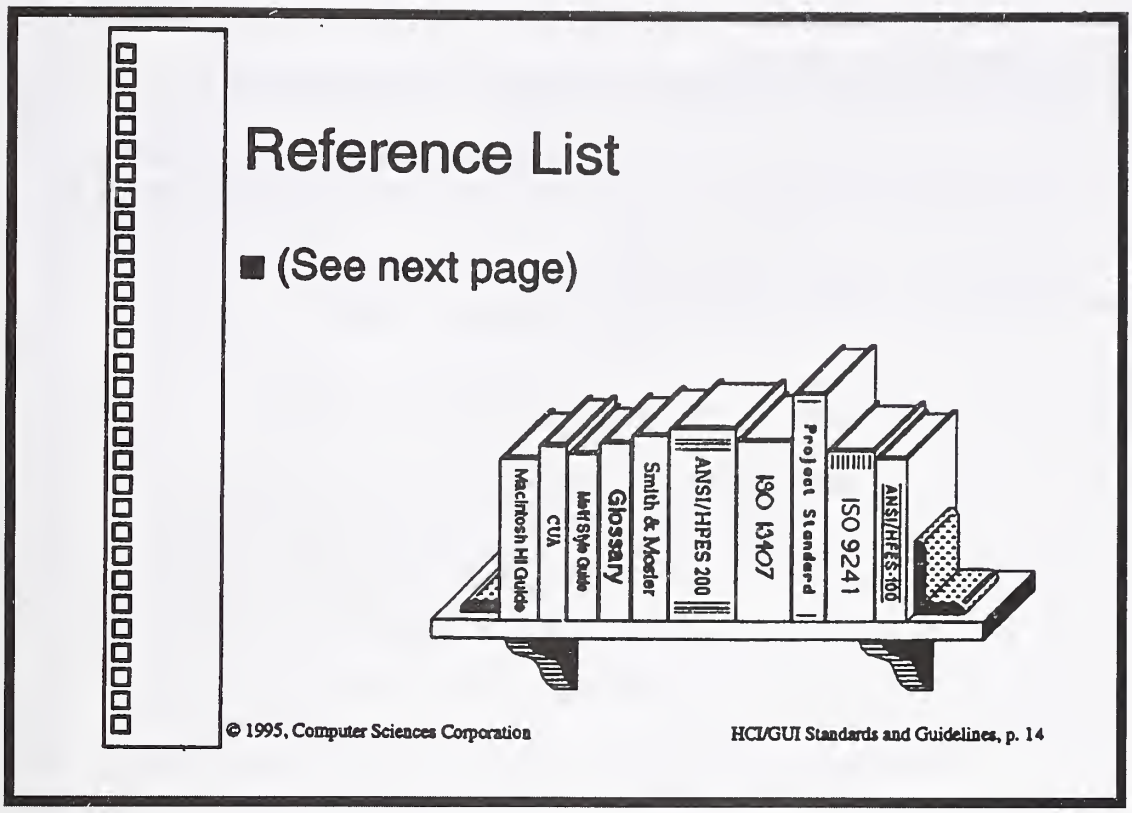


Figure 13

What can you do to incorporate standards into your organization? Bug your management. And this next suggestion is very important, you can encourage procurement officials to specify HCI standards in the RFP. Contractors are much more amenable to proposing something if it's in the RFP and the only way that we're going to get any of these process or product requirements reliably into government systems is to have the procurement people say up front that they want them so the encouragement of including these things in the RFP is in my view the most important of these three things that you can do. Of course, keep up with developments in the field, that's also essential. Where you can get more information on standards, ANSI or ISO offices? There's a quarterly column in the ACM's SIGCHI Bulletin on standards activities and what organization is doing what, what the status of the various standards, documents are, and there are Usenet news groups and Worldwide Web sites. Usenet news groups are primarily about human factors, and every once in a while, there will be a discussion on standards on that and SIGCHI has a Web site.

There are five or six at least sort of general HCI Web sites that occasionally have pointers to standards information. For the reference list, see the proceedings. There's a new one that is not in the list by Wanda Smith on ISO and ANSI standards. Does anybody know who the publisher is? I don't remember that.

Audience Question: The question was in industry what's the most common set of standards that people base their usability criteria on?

Buie: I would have to say it's the commercial standards. For windows systems it would be CUA and for Macintosh it would be the Apple Human Interface Guide and as I said the amount of human factors that went into those is variable and the depth in which those documents deal with recommendations I find for example the Motif Style Guide to be kind of skimpy, but Bellcore has a set of guidelines for Motif applications that goes beyond the Motif Style Guide and introduces a fair amount of human factors into it and is a good addendum to that.

Resources for HCI/GUI Standards and Guidelines*

Books

Apple Computer (1994?). *Macintosh human interface guidelines.* Cambridge MA: Addison Wesley.

Brown, C.M. (1988). *Human-computer interface design guidelines.* Norwood NJ.

Downtown, A. (1991). *Engineering the human-computer interface.* New York: McGraw-Hill.

Hix, D., & Hartson, H.R. (1993). *Developing user interfaces: Ensuring usability through product and process.* New York: John Wiley and Sons.

Karat, J. (ed.). (1991). *Taking software design seriously: Practical techniques for human-computer interaction design.* New York: Academic Press.

Mayhew, D. (1992). *Principles and guidelines in software user interface design.* Englewood Cliffs NJ: Prentice-Hall.

Nielsen, J. (1993). *Usability engineering.* New York: Academic Press.

Open Software Foundation (1993). *OSF/Motif Style Guide* (Rev. 1.2). Englewood Cliffs NJ: Prentice-Hall.

Tognazzini, B. (1991). *Tog on interface.* Cambridge MA: Addison-Wesley.

Process Standards (HCI/GUI Design, Evaluation, and Verification)

Standard	Title	Contents	Source
ISO 9241 (Part 11)	Ergonomic Requirements for Office Work with Visual Display Terminals	Guidance on Usability	American National Standards Institute 11 W. 42nd St., 13th Floor New York NY 10036 Tel: +1.212.642.4900 Fax: +1.212.398.0023
MIL-STD-1801	User-Computer Interface	(various requirements for verification)	Wright-Patterson AFB ATTN: ASD/ENES Wright-Patterson AFB, OH 45433
ISO/NP/13407 (under development)	Human-Centred Design for Interactive Systems	Assessing the benefits of human-centred design Interactive system development processes and human-centred design Planning the human-centred design process Human-centred design activities Managing and documenting human-centred design	(not yet publicly available)

* This material is adapted from a NASA/Goddard Space Flight Center newsletter and is in the public domain.

Figure 14

Software Standards and Guidelines (HCI/GUI Appearance and Behavior)

Standard	Title	Contents	Source
ISO 9241 (Parts 10-17)	Ergonomic Requirements for Office Work with Visual Display Terminals	Dialogue principles Guidance on usability Presentation of information User guidance Menu dialogues Command dialogues Direct manipulation dialogues Form filling dialogues	American National Standards Institute 11 W. 42nd St., 13th Floor New York NY 10036 Tel: +1.212.642.4900
ANSI/HFES-200 (under development; canvass draft due in late 1996)	Software User Interfaces	Same as ISO 9241, Parts 10-17, plus Color Accessibility for people with disabilities Voice and sound I/O	Human Factors and Ergonomics Society PO Box 1369 Santa Monica CA 90406 Tel: +1.310.394.1811
ESD-TR-86-278	Guidelines for Designing User Interface Software	Data entry Data display Sequence control User guidance Data protection Data transmission	Electronic Systems Division Air Force Systems Command Hanscom AFB MA 01731
NASA-STD-3000 Vol. 1, Sec. 9.6	Man-Systems Integration Standard	(various)	Johnson Space Center ATTN: SP34 NASA Houston TX 77058
MIL-STD-1472D (Sec. 5.15)	Human Engineering Design Criteria for Military Systems, Equipment and Facilities	Data entry Data display Interactive control Feedback Prompts Defaults Error management	Commander US Army Missile Command ATTN: AMSMI-EDS Redstone Arsenal AL 35898
DoD-HDBK-761	Human Engineering Guidelines for MIS	Dialogue Display Languages Input Devices Screens Printers Workstations	US Army (see above)
MIL-STD-1801	Human-Computer Interface	(various)	Wright-Patterson AFB ATTN: ASD/ENES Wright-Pat. AFB, OH 45433
BSR-AIAA R-023A-1995	Recommended Practice for Human-Computer Interfaces for Space System Operations	Data Entry Data Display Sequence Control User Guidance Data Transmission Data Protection	American Institute for Aeronautics & Astronautics 9 Jay Gould Ct, PO Box 253 Waldorf MD 20604 Tel: 1.800.682.2422 Fax: +1.301.843.0159
P1201.2 This failed two ballots and was withdrawn. It is here because other documents may reference it.	Graphical User Interface Drivability	Keyboards Pointing Devices Menus Controls Windows User guidance Common user actions	IEEE Standards Office 445 Hoes Lane PO Box 1331 Piscataway NJ 08855 Tel: 1.800.678.4333 Fax: +1.908.562.1571

Figure 15

Software Standards and Guidelines (HCI Implementation in Software)

Standard	Title	Contents	Source
P1201.1	POSIX	(various)	IEEE Standards Office 445 Hoes Lane PO Box 1331 Piscataway NJ 08855 Tel: 1.800.678.4333 Fax: +1.908.562.1571
P1295.1	Modular Toolkit Environment	(various)	IEEE (see above)

Hardware Standards and Guidelines

Standard	Title	Contents	Source
ANSI/HFES-100-1988 (currently under revision)	Human Factors Engineering of Visual Display Terminal Workstations	Working Environment Visual display design Workstation design Keyboard design Measurement techniques	Human Factors and Ergonomics Society PO Box 1369 Santa Monica CA 90406 Tel: +1.310.394.1811 Fax: +1.310.394.2410
ISO 9241, Parts 3-9	Ergonomic Requirements For Office Work with Visual Display Terminals	Visual display requirements Keyboard requirements Workplace requirements Environmental requirements Display requirements with reflections Requirements for displayed colours Requirements for non-keyboard input devices	American National Standards Institute 11 W. 42nd St., 13th Floor New York NY 10036 Tel: +1.212.642.4900 Fax: +1.212.398.0023
MIL-STD-1472D (selected parts)	Human Engineering Design Criteria for Military Systems, Equipment and Facilities	(various)	Commander US Army Missile Command ATTN: AMSMI-EDS Redstone Arsenal AL 35898
NASA-STD-3000 (selected parts)	Man-Systems Integration Standard	(various)	Johnson Space Center ATTN: SP34 NASA Houston TX 77058

Figure 16

A Success Story at the Social Security Administration
Usability in the Federal Government

Patricia Stoos
Director, Planning and Evaluation Staff
Office of Systems Design and Development

Ms. Pat Stoos, OSDD at Social Security Administration
A Usability Success Story at the Social Security Administration

As you can see from my outline for today's presentation (Figure 1), I want to start with a brief introduction to the Social Security Administration.

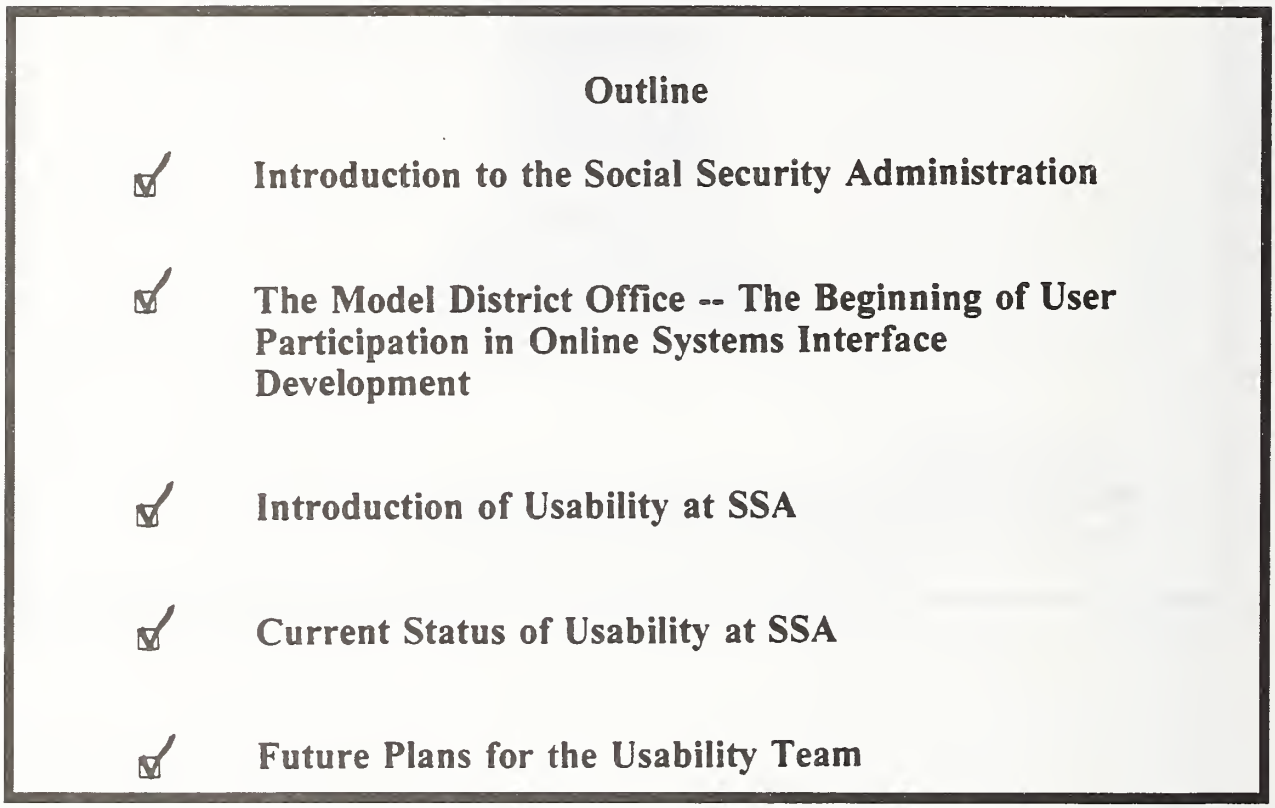


Figure 1

I think when you see the numbers and Rex already started talking about that, it becomes a very mammoth undertaking. I want to also talk about the Model District Office. It was the beginning of user participation in on-line systems interface development at the Social Security Administration and then an introduction to usability at SSA, our current status and our future plans. Most people when they think about Social Security think about the retirement, survivor, and disability programs.

That's what you hear about in terms of did you get your social security check? Usually that's what people talk about, but the charter for Social Security includes administering the Supplemental Security Income Program which is a needs-based program for aged, blind and disabled individuals. (Figure 2)

Social Security Administration
Charter

- **Administer Retirement, Survivor, Disability Health Insurance Program Under the Social Security Act**

- **Administer the Supplemental Security Income Program for the Aged, Blind and Disabled**

Figure 2

Highlights of SSA's Workloads

Social Security Numbers	
Cards Issued in FY 1994	16 Million
Wage Postings	
Number per Year	221 Million
Retirement, Survivors Disability Insurance	
Number of Applications in FY 1994	5.0 Million
Number of Beneficiaries	43.0 Million
Benefits Paid in FY 1994	$313.8 Billion
Supplemental Security Income	
Number of Applications in FY 1994	2.2 Million
Number of Recipients	5.8 Million
Benefits Paid in FY 1994	$24.8 Billion
Changes to Benefit Records	
Number of Changes per Year	89 Million

Figure 3

Figure 3 gives an idea of the numbers. I don't want to go through each one of them but you can see some of the numbers on our major workloads—issuing Social Security numbers, posting wages, taking claims for retirement, survivors, disability insurance, supplemental security income. We're talking about millions of events and billions of dollars. It's a lot of money and just for example the last one, 89 million times a year we make some kind of change to a record, someone's changed their name or they've changed their address, they changed some sort of data that influences how we calculate their benefits. That's a lot of transactions. And as Rex has talked about we actually have 16 million on-line transactions per day, 16 million and as you all know Social Security, just like the rest of government is downsizing, streamlining. We're not getting additional staff to deal with additional workloads. This is what SSA's operation organization looks like. (Figure 4)

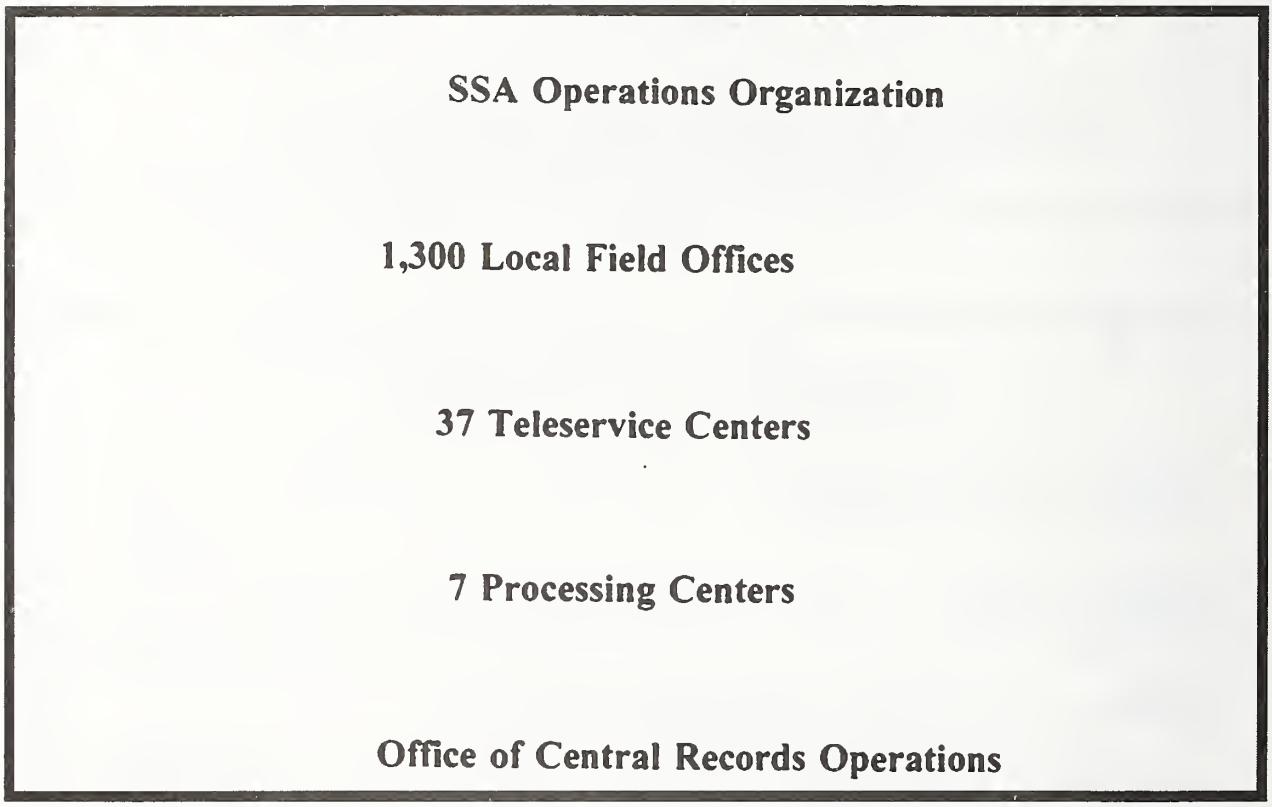


Figure 4

We have 1,300 field offices. Those are the walk-in offices in shopping centers. Many times people go there to get their Social Security card, to file for claims. We have 37 teleservice centers. When you call the 800 number for Social Security that's where you're connected and they can take some simple transactions. They can set up an appointment for you or they can answer general Social Security questions. We then have 7 processing centers and those centers are involved with processing our most complicated claims, those are the claims that the automated systems do not currently handle and those processing centers in addition are supporting the 800 number. They have some spike units that also take 800 number calls and in the Office of Central Records Operations, our earnings data are maintained. And of course earnings are important in Social Security. What you paid in is used to calculate your benefits.

The start of user participation in on-line systems interface development began with the establishment of the Model District Office and that occurred in the early 1980's. (Figure 5)

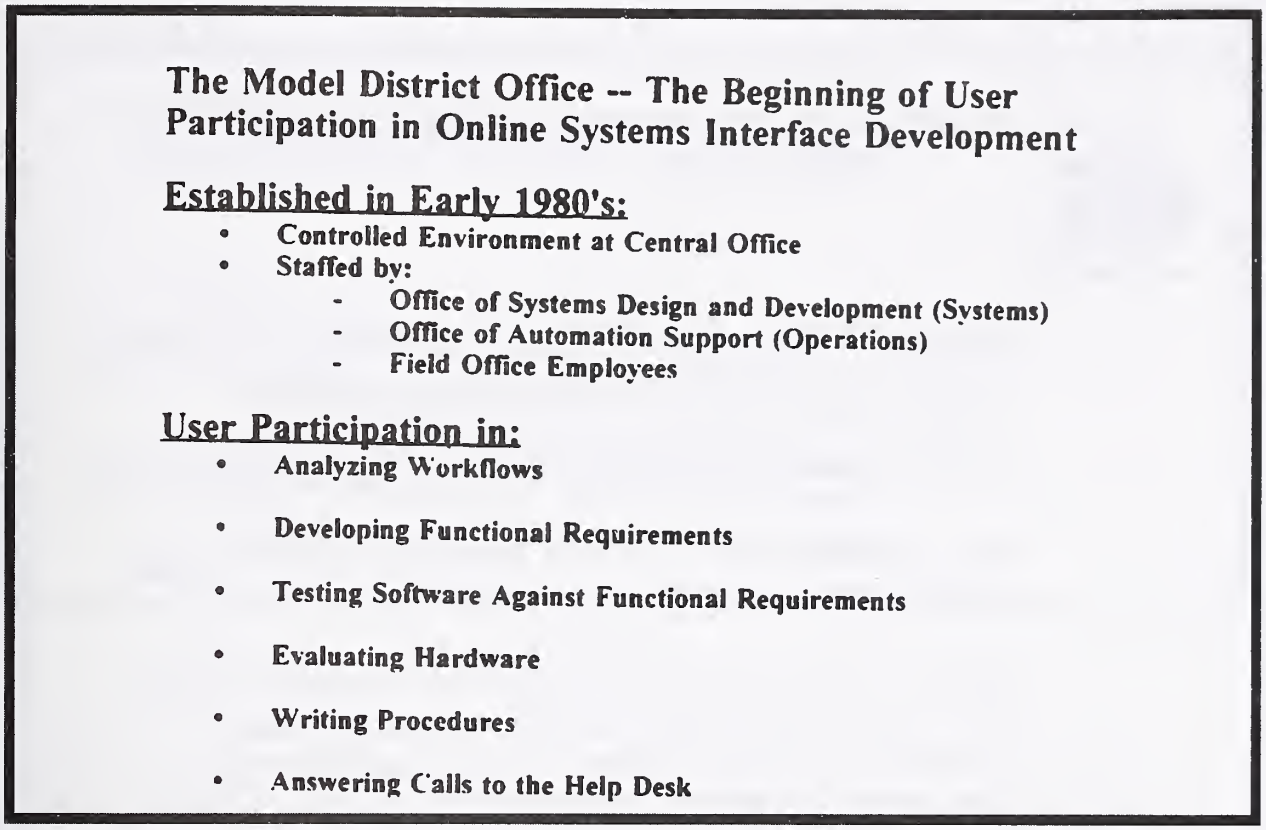


Figure 5

It was a controlled environment at the Social Security Headquarters and it was staffed by personnel from systems and from operations and we brought in field office employees to help us. This was our environment to help us test workflows, hardware and software, to help determine the functional requirements and to test that software against those functional requirements. We also had the users involved in writing procedures and in answering those calls to the Help Desk which of course we continued to receive. When you look at our budget Social Security spends about a million in travel and per diem costs for those field employees that we bring into the central office for activities like what we do in the Model District Office. That's a lot of money and one of the driving factors to incorporate usability at SSA was to maximize the return on their investment for those field office employees staffing a Model District Office. (Figure 6)

It's a large investment and we needed to get pay back for that. We also need to ensure the software meets the needs of the users. We had a lot of success in meeting the functional requirements but when the software was released it was not always embraced. We need to define and test the interface early enough so that changes can be incorporated and I'll give you an example of one of our experiences in that regard. And, of course, the benefits we talked about this morning. We need to increase user productivity, improve accuracy, reduce software development and maintenance costs, reduce user training costs, reduce the number of pages for printed documentation. Many of our field offices have a very small window of opportunity for

training. Generally it's maybe the first half hour of the day before the office actually opens for the public, but once those doors are opened, there is not an opportunity to take people off the floor and put them in a room to conduct training.

Driving Factors to Incorporate Usability at SSA

- **Need to Maximize the Return on Investment for Field Employees Staffing the Model District Office**
- **Need to Ensure Software Meets the Needs of the Users**
- **Need to Define and Test the Interface Early Enough So Changes Can Still Be Incorporated**
- **Need to: Increase User Productivity; Improve User Accuracy; Reduce Software Development and Maintenance Costs; Reduce User Training Costs; Reduce the Number of Pages of Printed Documentation**

Figure 6

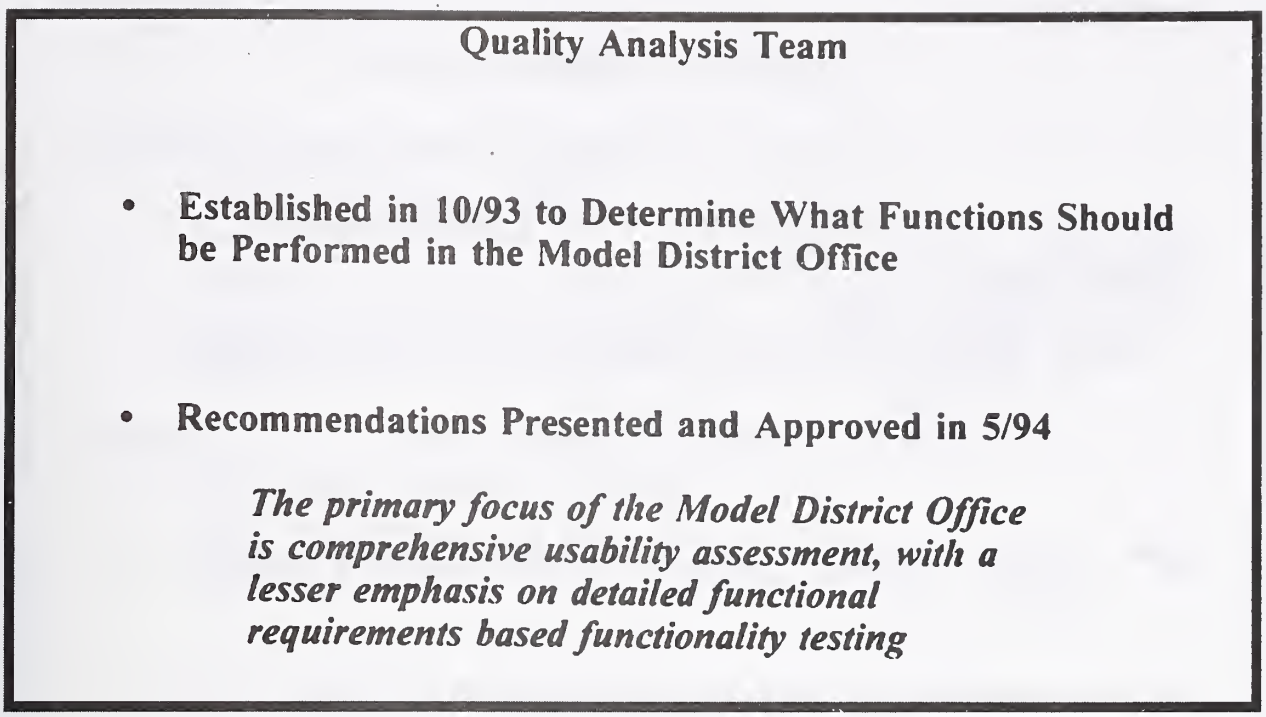


Figure 7

So we started with establishing a Quality Analysis Team (Figure 7) in October of 1993 to determine what functions should be performed in the Model District Office and that team came back with their recommendation which were presented and approved in May of '94. They said the primary focus of the Model District Office is a comprehensive usability assessment with lesser emphasis on detailed functional requirements-based functionality testing. The next step we did was to establish an Intercomponent Usability Team and we had representatives from system component and the operations component.

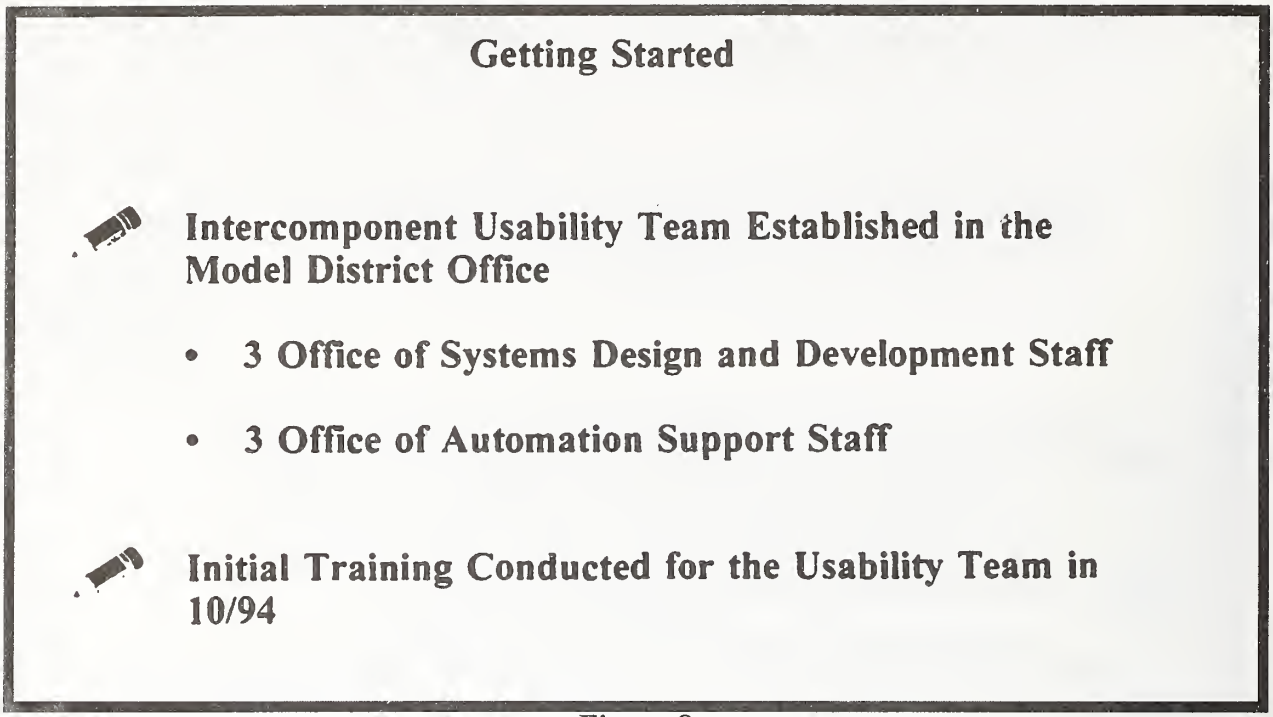


Figure 8

We asked that team to develop a plan on how we were going to introduce usability into software development at Social Security. So they began their initial training in October of '94 and were promptly given their first assignment. Now I know that you've heard things that you want to do when you first get involved in usability and I'll bet these are not the things you think of.

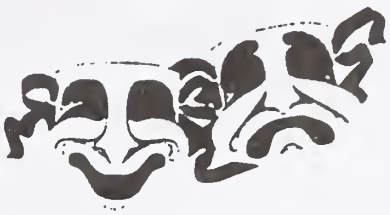
We were directed to evaluate a major software release. It was legislatively driven. The timeline could not be changed. Congress wasn't going to go back and say, "OK, Social Security, you can have some more time." The software was already in validation but my boss said, we're going to hear about it so I want to hear now as opposed to waiting a couple of months. Another piece of information about the application is that we had a working piece of software that performed that same function but it performed it for a different group of users and they had different naming conventions. They had a different workflow. They had different requirements, but we had that basis software and we said, "We can do this. We can make these changes and make it available nationwide." Well, like the comedy and drama, the little masks on here; it really seemed to us like the best of times and the worst of times.

It was a wonderful time in terms of our learning experience and finding out ways we could improve the software, but it was the worst of times in terms of our relationship with that project team. They did not receive us warmly. They did not want to hear about things that we found and we went through some pretty stressful meetings with them to show them, to explain to them the things that we found. So our first finding is not a surprise. This is not the way we recommend you start in usability but that's what happened to us so we can't change history. And early involvement is essential to the success. We had a very difficult time being thrown in at the back end of a project and of course there's a need to recognize the different user needs. You can't just take one workflow, one operation procedure in one place and say that's going to apply

First Experience with Usability

Directed by Associate Commissioner to Evaluate:

- **Major Software Release**
- **Legislatively Driven**
- **Software Already in Validation**
- **Targeted for a New Group of Users**



Findings:

- **How Not to Start in Usability**
- **Early Involvement is Essential for Success**
- **Need to Recognize Needs of Different Users**

... **Very Dedicated Project Team Incorporated Improvements in a Subsequent Maintenance Release That Has Been Well Received**

Figure 9

somewhere else. It didn't work for us. But on the happy side, in the end we did identify a lot of useful information for the project team and we did convince them to implement a subsequent maintenance release which had a very quick turnaround. That maintenance release has been very well received and we consider that to be a big success even though it was painful along the way. What is the current status of usability at SSA? I like to think that we've "recovered" from that first experience. We've sort of convinced people that we don't want to do it that way but that there is a lot of benefit from introducing usability early on in the life cycle and throughout the life cycle. We've completed six training sessions so there are about 120 people at Social Security who have been trained in the basic understanding of usability. We've conducted a number of evaluations and are continuing more. Here is one particular example to sort of give you a balance with my earlier example. We had a mainframe system, it was a small system but it was something that everyone would hit every time they tried to get between applications. We went

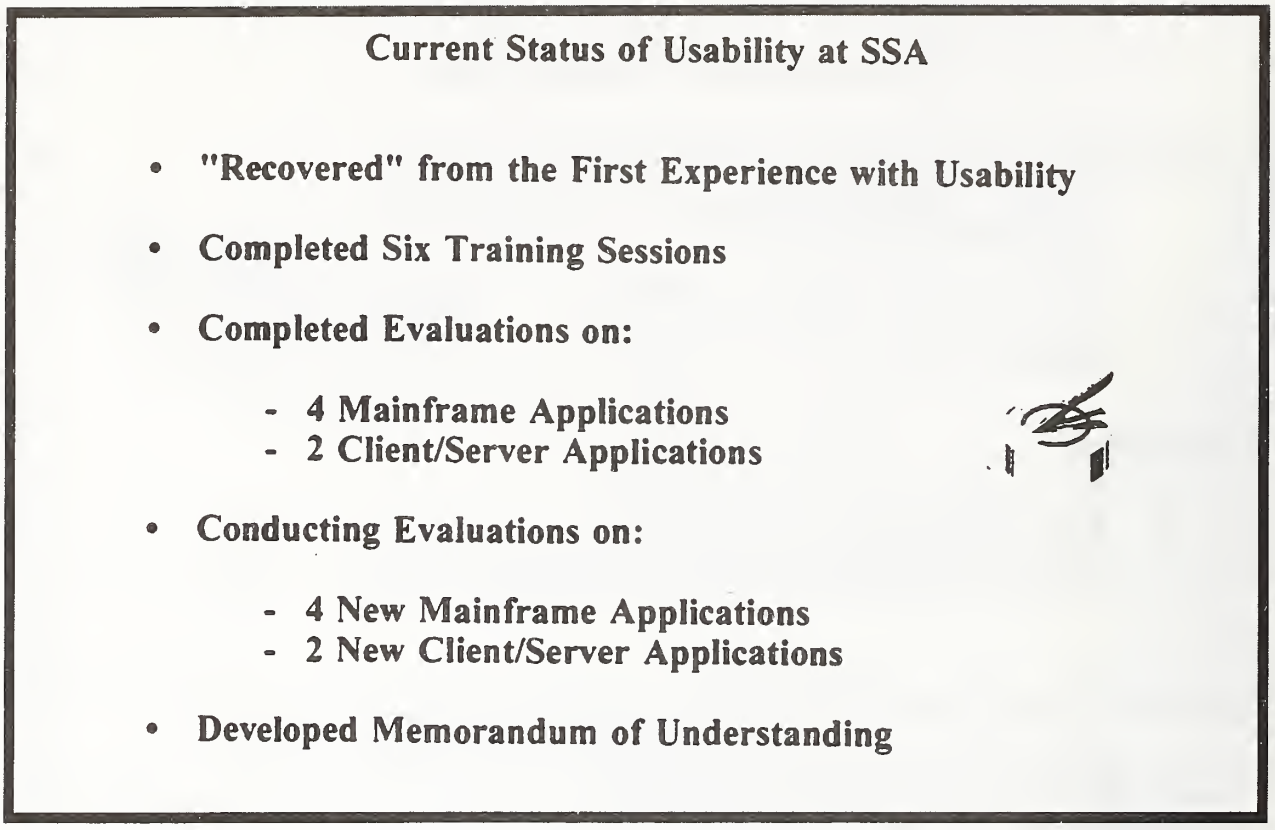


Figure 10

through three iterations with the programming staff and made some improvements each time. The first time we had users look at the system, they threw their hands up in the air and said I don't understand what I'm supposed to do. How do I get to where I need to be and in the end we've implemented the system in pilot in mid January, so it's been there about six weeks and we've gotten very positive feedback from the pilot. In fact, we have no recommendations for improvement in this particular piece of software which I think is pretty phenomenal. We're also in the process of developing a Memorandum of Understanding which hopefully will be signed next week between the operations component and the system's component laying out our rules and responsibilities, laying out the process for usability and our future plans.

We're working on some specifications for a usability lab. We have a facility, a Model District Office where we continue to do our usability work but we don't have a lot of the automated support that we'd like to have so we're in the process of doing that. Of course we have some very tight budget times right now and Social Security is one of the agencies that does not have an appropriation yet, so we don't have a lot of money available, but we're putting together everything we can in the meantime. We're working on putting together some style guides and some processes for including the developers in determining the style guide standards. We're continuing training and working with informal methods of meeting with project teams and with staff meetings with our senior managers. We're also looking at some remote testing. We have a group of employees in Auburn, Washington, who have assistive devices to help them use our

software and instead of the expense it would take to bring in that expertise to bring in that equipment, we're lookin are also developing a Test Case Library and perfecting our data analysis

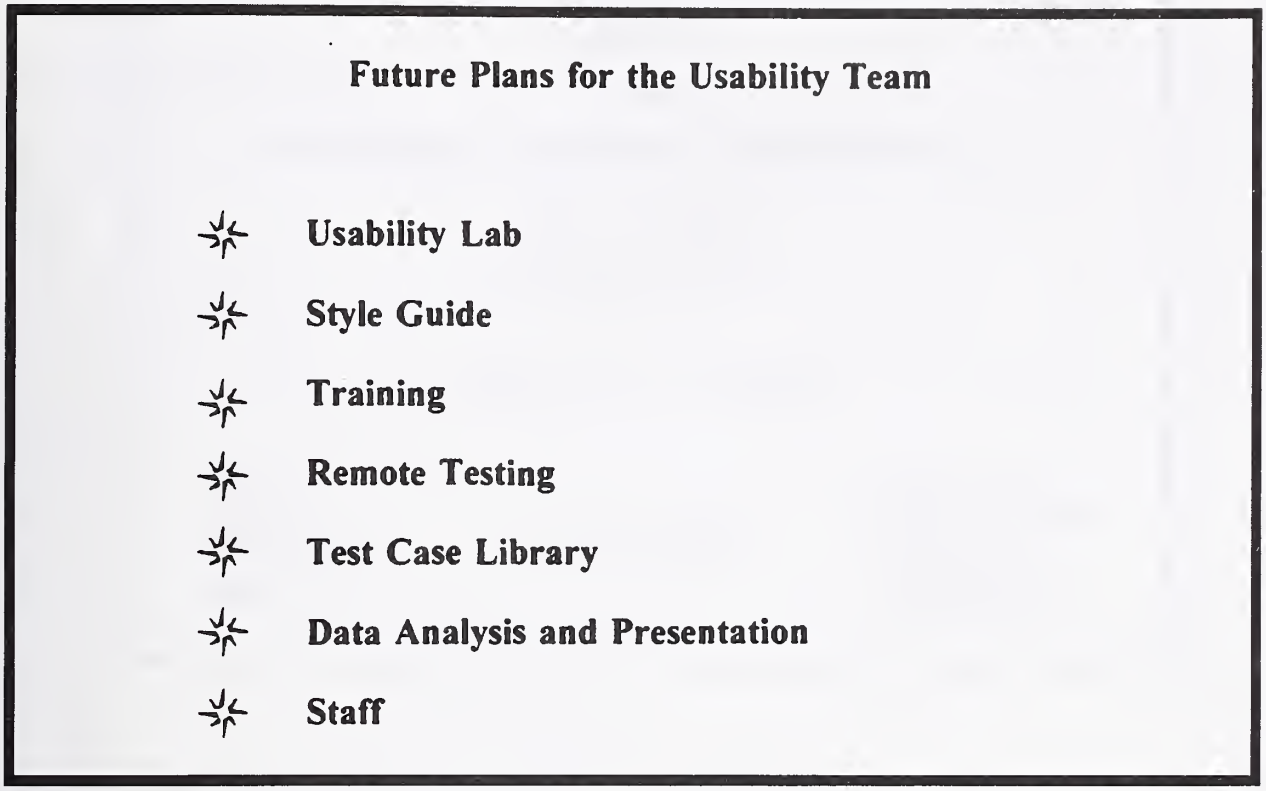


Figure 11

and presentation techniques. I think that one of the things we found with our earlier experiences is actually providing the data to the team and to our managers to convince them that the changes were needed. We're continuing with staff development. I'm looking forward to bringing in some additional staff although right now of course we have a hiring freeze so I'm not able to do that, but I do have a commitment from my boss that in the future we will be able to increase staff in the usability area. That's where we are at Social Security.

Focusing on Usability at the Federal Intelligent Document Understanding Laboratory: A Success Story

Theresa A. O'Connell

PRC Inc.
1500 PRC Drive
MS 5S3
McLean, VA 22102
703-556-1082 (PRC)
oconnell_teri@prc.com

Version: 29 November, 1995
Control No. FIDUL-960009

Federal IDU Laboratory
Tysons International Plaza II
1921 Gallows Road, Suite 800
Vienna, VA 22182
703,827-2220 (FIDUL)
toconnell@idulab.gov
Federal IDU Laboratory

Ms. Teri O'Connell, PRC
Focusing on Usability at the Federal Intelligent Document Understanding Laboratory

What I am going to be talking about today is how the Federal IDU Lab came to focus on usability testing. There are two parts to the Federal IDU Lab. There's a research section and then there's also a testing and evaluation part. I'm part of the testing and evaluation section. The FIDUL mission is to test and evaluate what we call "Intelligent Document Understanding Technologies".

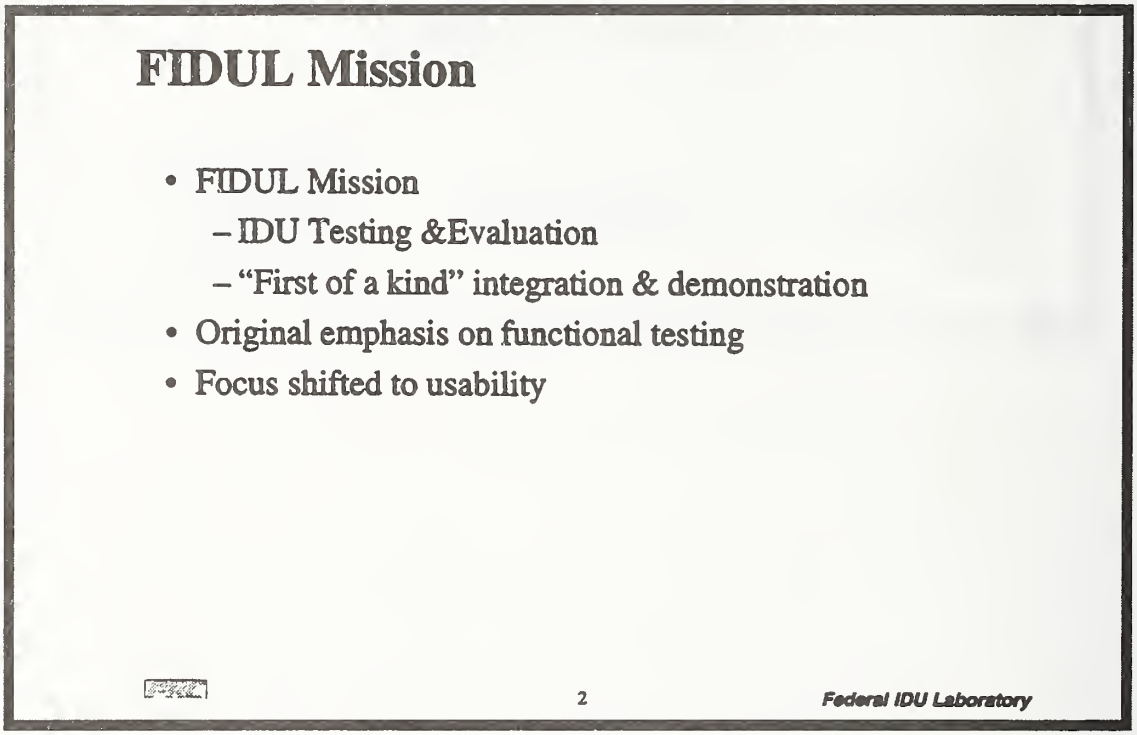


Figure 1

These include things such as (at this point) optical character recognition, machine translation and document detection. Most of the products that we look at have something to do with foreign languages so we come across a variety of writing systems—Chinese traditional and simplified, Arabic, and Cyrillic. We also come across some unusual operating systems such as Chinese Windows. We see products that are at the very beginning of their development cycle and then we also have people walk in the door with products that are completely shrink wrapped. We test and we evaluate to help the government make decisions on continuing the research or perhaps on buying these products for use in government offices and then later there is an expectation these products will also go out beyond the government. When the original RFP went out for the testing and evaluation part at the FIDUL, there was no mention of usability in it. The original emphasis was on functional testing, on performance testing, that type of thing. However, this focus shifted to usability.

So, what I'm going to be talking about today is a double success story. First, I'll tell you how we successfully participated in this shift of focus and how the FIDUL is successfully practicing usability, testing and evaluation. We'll talk about how I first had to sell the idea to PRC management and then how it fell on very fertile ears. We were very lucky in that the government sponsors knew all about usability testing. But, when they began to buy into having this be the focus, we had that same message that we've already heard about earlier today, sure do it, but no extra money. Do it in the existing budget. Do it within the existing facility, and by the way, make sure you stick to the original objective of the lab, and, of course, we want this done according to the scientific method.

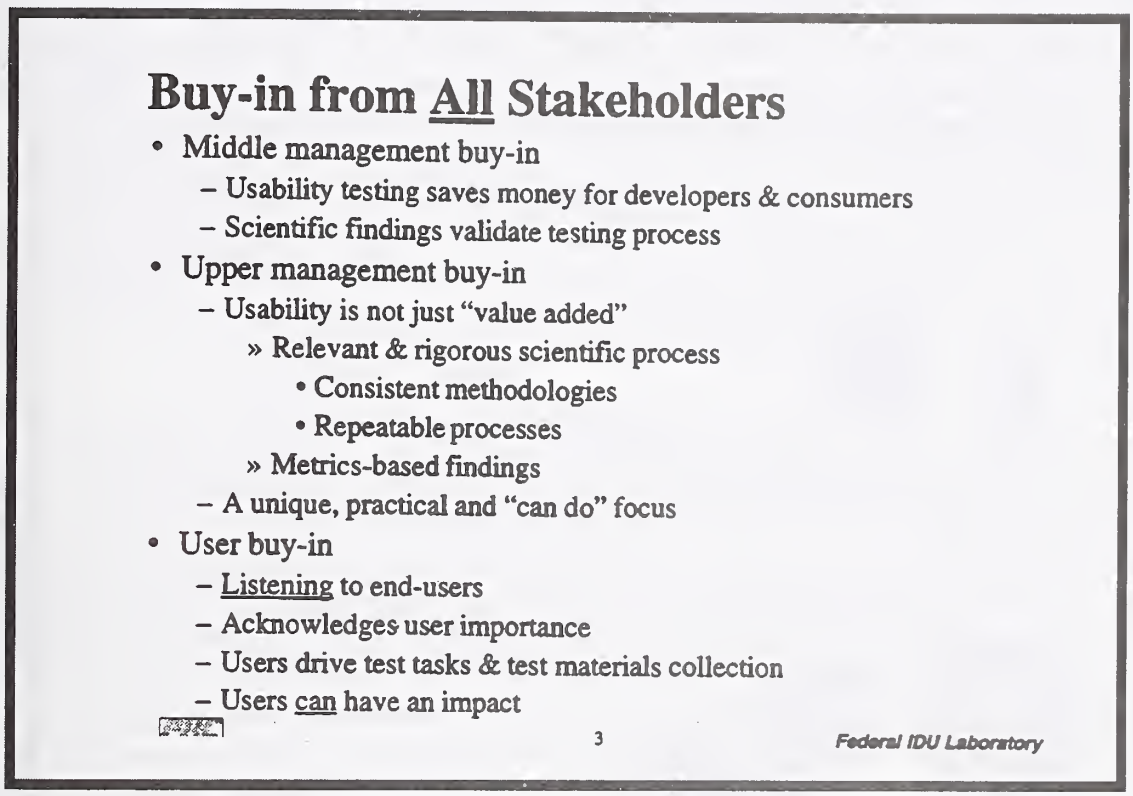


Figure 2

Now when I talk about buy-in from all stakeholders here remember this is a double success story so Figure 3 shows more than our strategy, it also shows the benefits that we are actually experiencing right now. The FIDUL came into existence only 6 months ago, so these things have been happening very, very fast for us, as we work with developers and consumers. The first thing I had to do at PRC was to get that middle management buy-in. And you notice this is a rather strange way. I go from middle management to upper management to end users. I did that on purpose because this in real life was the way that I saw it, having to sell the idea of usability in real life is the way I see us having to solicit user involvement. We always go first to middle management. They get the OK from upper management and then finally we get to talk to those end users. I had to convince the PRC management that usability was a good idea. I had to talk to the people who were going to be managing this project and I also had to talk to other people who were on the team. I found myself talking to software engineers and to computational

linguists who have a very definite idea of what usability testing was. It was smoke in mirrors. It was a waste of their good time, and we really shouldn't even be talking about this, but OK, we will politely listen to you.

I started by doing briefings and honest to goodness I really did have a slide in my first briefing that said, usability testing ---- or science to start there. And the answer to that was of course science. I found that by pushing this as a scientific method with scientific findings, that would actually validate the testing process, I was able to open their ears. When it came to upper management, I had to show this was more than added value, because you have to remember we had already won the contract, and they already had a very good idea of what they were going to give the government. We were going to give the government what we had talked about in our winning proposal. We had not talked about usability so before they would even let me talk to the government about usability, I had to prove to them that this was something that we needed. I had to demonstrate a relevant and rigorous scientific process. I had to show them that my methodologies were consistent. I had a notion in mind of things that could be repeated over and over so that I could get consistent results that end users and that the developers and that the customers would be able to depend on understanding and knowing what kind of things I would be talking to them about. Most of all that these findings would be metrics-based. I had to be very "can do." I had to show them we can do this without having to buy lots of equipment. This is a benefit. This is not a burden.

Finally, we got to the point in each of our projects when we were talking to users, getting the end users to participate. There was an automatic buy-in from end users. There never has been, so this was something that we had to work on and we worked on it first by listening to them. With every single project that we do, we start by going to their offices. We start by sitting down with them at their work stations, talking about their needs, their expectations, not only for the products that we are testing, but overall, their hopes for the future. So we started by listening, this means doing the requirements analysis based on what we hear from our end users. Well, this sometimes gets us back to middle management where they tell us, "There's no such thing as an end user. You're coming here, you're talking about a system where you start with a piece of paper in Chinese and you end up with something in English. Nobody has that. There is no end user for that."

So we tell middle management, "Well, you know the way that we got the idea for this system was when we were testing Chinese OCRs, they told us, 'you know the ultimate beneficiary of this product is not somebody who speaks Chinese. It's not even somebody that normally does optical character recognition. It's an analyst that speaks English,'" So, obviously we could see that there was a need here for machine translation for something to help these things get translated a little bit faster. We were able to go in and sit down and we talked to middle managers and they say, "Well, you know so and so over there needs to have Chinese to English translation. Go talk to him." We have identified a potential end user. This is a very big step for us because if we need to start with end users, we have to identify them even when the product that we're going to test has never existed, when those people have never heard of it, when they've never even dreamed of it.

We found that we have had wonderful success doing this, and as the people at the FIDUL started to really buy into usability, they wanted to have these users places where they never were before.

For example, we had actually brought end users to a government factory acceptance test. We've been able to bring them in and have them talk to the developers and explain to them things that we in our reports and with our charts and all of our reports could not convince them of. For example, we had to develop our optical character recognition system. This system translates, so we brought in our end users and we tried to tell them when they see "translate" they are not going to understand that we're talking about translating from a TIF image to a text image. These are people who think of translating as from language to another. So I told them this, but the head of the project would not listen to me. I brought end user number one in. "What does translate mean to you?" "Well, that's when you start in Chinese and you go to English." He still didn't believe it. I brought in end user number two. "What does translate mean to you?" Well we did it, by two end users. We didn't have to go back and bring in more end users, so there is a benefit there to them.

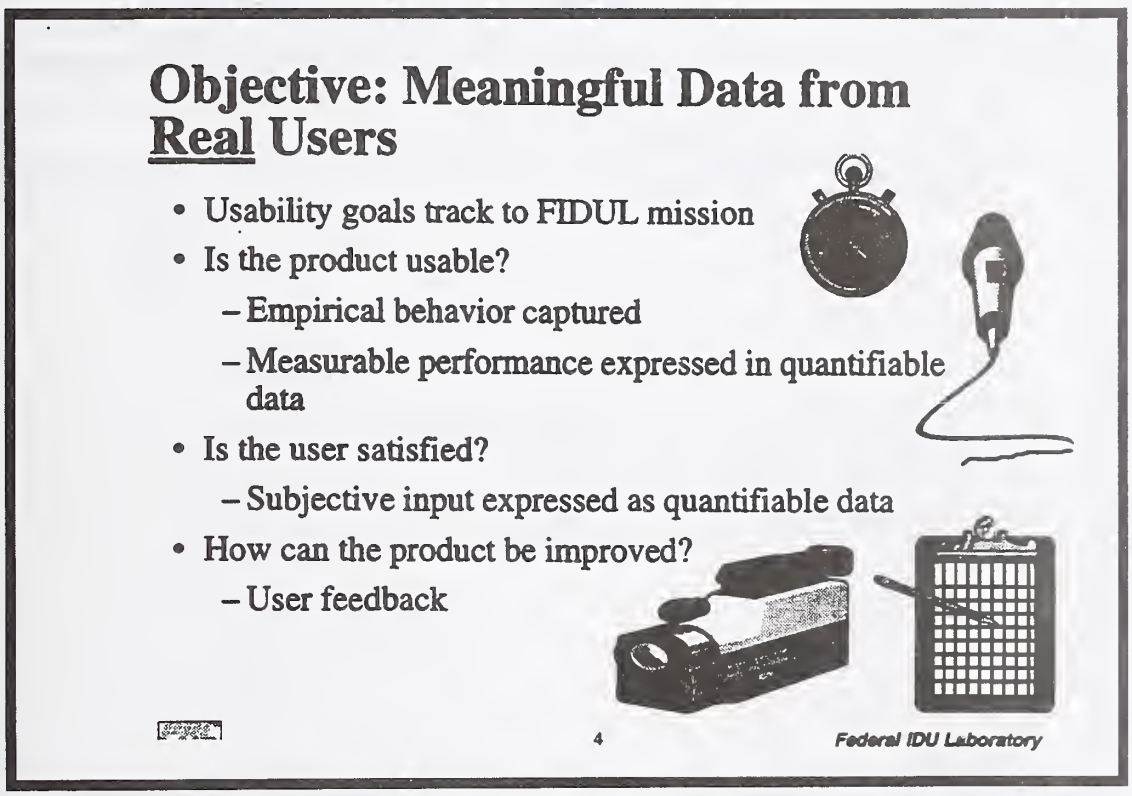


Figure 3

We did something that had never happened to these end users before, we acknowledged their importance in the development process. We acknowledged their importance in the decision made by people who never had any contact with them as to what products they are going to be using in their work place. We showed them that they can drive not only that final decision but they can drive things earlier on. When we do our test plans, our scenarios are directly driven by user input. They absolutely reflect what the users do. We had a very interesting situation by letting the end users drive the test process. We also used materials taken from their own work places. Going back to that Chinese Optical Character Recognition System, they reported wonderful, wonderful accuracy rates. When we brought in our end users with their own

materials, those accuracy rates plummeted. It turned out no OCR system in the world can accommodate every character there is in Chinese. It's simply impossible, not even Microsoft, Chinese Microsoft Windows can accommodate every single character. The characters that they had chosen were not the characters that came up over and over again in the work place, in the materials that the end users really needed to use so we were able to show that to them. They thought they had a solution in place. They said well the users can build a dictionary. They can build a dictionary that holds up to 20 characters. Our end users kind of went like this, "20 characters. Don't even start to talk to us until that dictionary can accommodate a minimum of 50 combinations of two characters each. That's what we need." We were able to show them this will take care of this low accuracy scores and the end users could see they would have an impact. Also, there's another reward that our end users get, we send a nice letter to their managers later thanking them for this time you gave us with your users. It was worth it, and they did a very good job. This letter goes into their personnel files. So what is our objective?

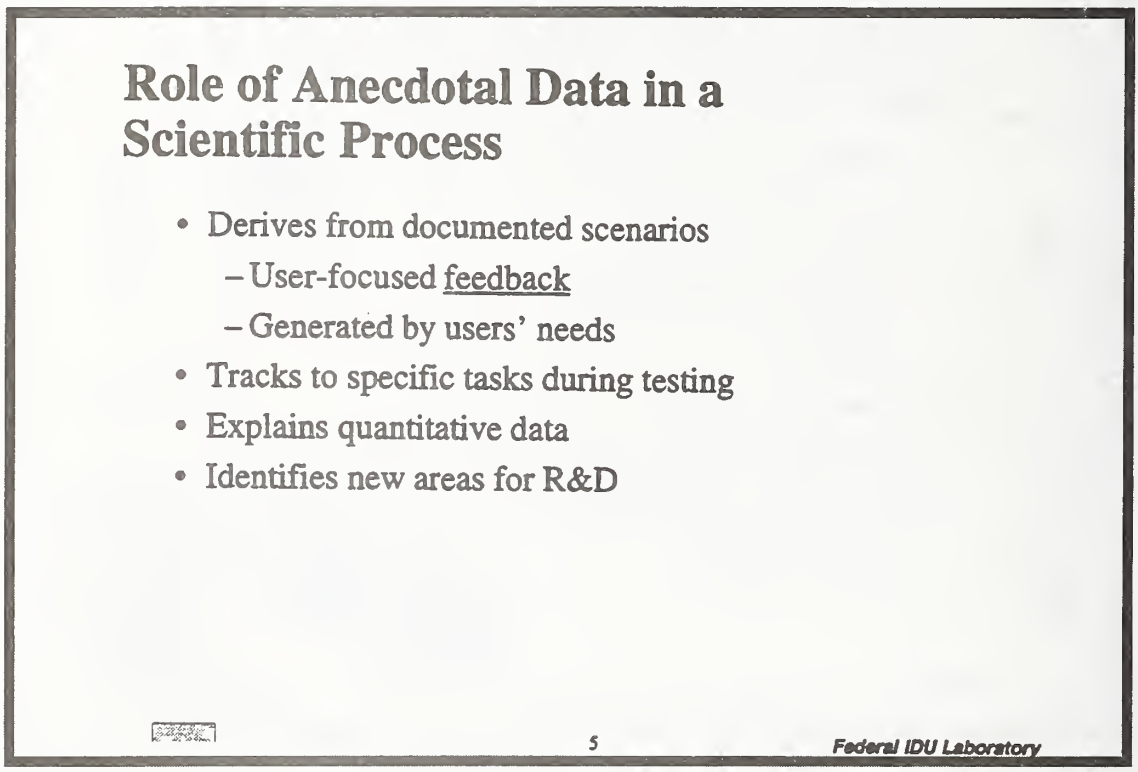


Figure 4

Obviously, it has to track back to that final mission that we talked about earlier. Now when the developers or when the people who tried to sell this product to the government come to the FIDUL, they say "Look, this product goes, and does this," and we come back with two very painful questions. We see how nice it goes, but can the user make it go? And number two, is the user satisfied with how he or she can make it go? In the end, we can show how these things can be improved based on user feedback. Now I have these pictures of this equipment around here.

Our very first test we did with a borrowed hand held timer and a borrowed tape recorder. In the end after that test, our sponsors did go out and buy us all sorts of equipment. They really did see

the value of what we were doing so by starting off small and doing it with just a few very simple tools we are slowly but surely working our way up to a real usability facility. We do not yet have a place because when the original plans were done, there were no plans for a usability lab but

there is a lot of talk about eventually having a usability lab. So we used a scientific process and we obtained quantitative data. So what is this anecdotal data that I'm talking about here?

It's what people call qualitative data. I found that developers don't like to hear that word "quality." It's very judgmental. This is what's wrong with your product, but then everybody loves a good story, because when I first started talking about anecdotal data the reaction of the developers was very interesting. We had been showing all the charts and all of the tables, all the things that you usually do and I said to them, "Now, would you like me to share some anecdotal data with you, ways that you can make these scores go up?" So it was like when Merrill Lynch talks; the entire attitude changed. They leaned forward and they listened, and something magic happened where we had a rather confrontational situation—this is your bad score, this is your good score, this is your bad score—rather head to head confrontational. Suddenly we were

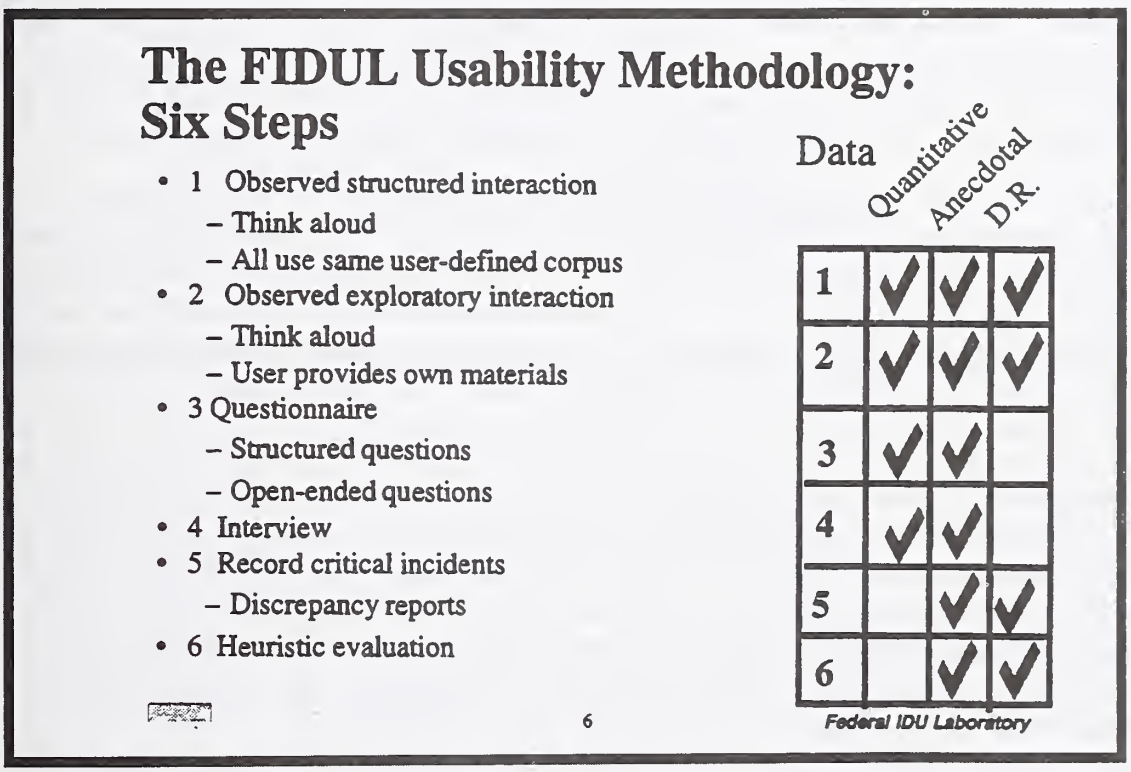


Figure 5

partners. I was sharing with them ideas from real end users, experiences of real end users on how they could make their products better. Now this anecdotal data tracks back to specific tasks during testing. It's collected while the users think aloud; it's collected in user questionnaires. It's used to explain the quantitative data, but it also does something else that is very near and dear to the hearts of our developers. Instead of looking at things as "this is what this product is lacking,"

they are able to go back to their sponsor, the government sponsor afterwards and say, "Look, your end users, your government users say, 'This product needs this.'"

This was not in anything we originally talked about. This is a new area for R&D and they have demonstrated a documented basis to go in and request that more R&D be done, and actually I think it's the other way. I have seen the people at the FIDUL say, "Ah, look, this did not show up any place in the product when we first talked to the developers but the end users are asking for it. Developers, are you interested in this R&D?" Well, I think you can guess what the answer to that is on the part of the developers: yes, they are. So we end up with this methodology. These are things that all of you do but the thing about this is that this is a consistent and a reliable methodology. These are the bare bones of our repeatable process, a definite place to start.

We need this because when we started in June, we had no usability testers. Then we had one, now we have two. I am very confident that very soon we'll have four. We started out with absolutely no advisors. Now we are building this Advisory Board. We have people, good people like Rex Hartson and Astrid Nielsen who are both here speaking to us today. We are building our team so we need a structure that we can point to and say this is what we do. The aim was to be comprehensive but of course some tests don't use all of these steps, some tests repeat them. For example, when we're doing field testing, we do more than one questionnaire. When we take this method and use it, for example, in a test where we actually bring people into the lab to drive a product or when we go out to their place, an end user can really get through this in less than one day, in five hours which is a very attractive thing to middle management.

We start with a structured interaction and think aloud. During the structured interaction, everyone uses the same corpus, the same set of test materials. This corpus is developed starting with materials that come from the work place. We collect quantitative data on things such as error rates. Also the time that it takes to initiate a task and I say initiate a task because we have to be very careful that what we're measuring is not the power of these exotic operating systems. We have other people in the lab who do that functional end performance testing. Then we go on to what we call observed exploratory interaction. This is simply test driving where we sit back and don't interact with the end users as much while they think aloud and test the material, their own material that they brought in addition to our test corpus which is completely and entirely from their own individual work places trying to get the system to do things that they want it to do. The questionnaire has those structured questions which will generate quantitative data and open ended questions which we use to collect what we're calling our anecdotal data. This is followed up by an interview. An interview is very important for us because so many of the products that we use are foreign language products, so many of our end users are not native speakers of English. Some of them feel more comfortable speaking on a one to one basis than they do in writing down answers. We do everything we can to elicit as much information as possible from the end users.

Through all of these steps we record critical incidents. Critical incidents are anything that impacts the way the user interacts with the system. We write these up in what we call discrepancy reports and then we have a Board that meets and we rank these problems so we can go back to the developers or to the people who are selling us the product and say look, this is the most important thing that you have to fix and this is based on end user feedback. Also we do a heuristic evaluation of the interface. This is based on comments from the end user and on our on

expertise as usability testers. So the bonus is that by focusing on the end user, we get a methodology we can use for a variety of technologies, a variety of writing systems, totally independent of any platform and which meets the final goals. Our pay-off is that we've met the goals plus we get a meaningful assessment of the product maturity and its suitability in the work place.

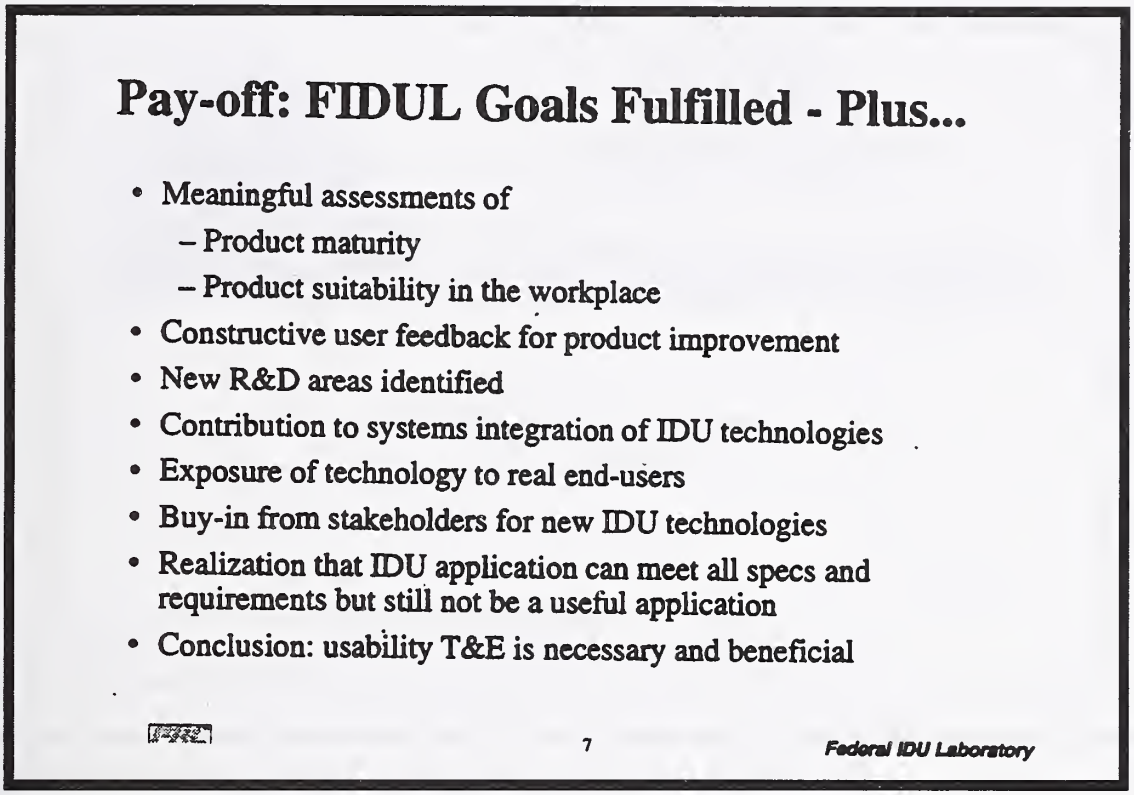


Figure 6

We have constructive user feedback for product improvement. We've identified those new areas for research and development. We've also been able to contribute to systems integration of IDU technologies. We've come up with ideas based on end user feedback for systems that start with paper and end up with document retrieval systems and incorporate machine translation. We get buy-in from stakeholders for technologies they had never heard of before, stakeholders being not only those end users but their managers. All of this grows out of the realization that these applications can meet all the specs and requirements, but we demonstrated that they can do that but still not be useful. The conclusion is that this testing is necessary, it's beneficial. It's a worthy focus for our testing and evaluation and we ended up with a situation where we have a Federal lab which is using usability criteria as a basis for making decisions on continuing research or purchasing products.

Vitro
a Tracor company

TIGERS

Tactical Information GUI Engineering
& Requirements Specification

**A Successful Usability Engineering Methodology:
Teaming Industry and Government in the Development
of the NATO Seasparrow User Interface**

Daniel Wallace & John Dawson
Vitro Corporation

Developed under US Navy contracts:
N00024-91-C-5133 & N00024-94-C-5402 (NATO Seasparrow)
N60921-93-D-A142 (Advanced Integrated Electronic Warfare System - AIEWS)

1/10/96

Dr. Daniel Wallace, Naval Surface Warfare Center
TIGERS - A Successful Usability Engineering Methodology

This program is called TIGERS (Tactical Information GUI Engineering & Requirements Specification). The importance of each element of the specification was chosen specifically because we are dealing with tactical information systems. We developed this usability engineering process for combat systems. We have engineered the graphical user interfaces and specified the requirements specifically and objectively so we have a means of tracing the requirements from start to finish to ensure that all of those are successfully integrated into the end product. TIGERS was developed at Vitro in collaboration with John Dawson, a systems engineer and Barbara Rostosky from NATO Sea Sparrow Project Office. I cannot emphasize enough the importance of having a good government sponsor who will force the contractor developing the application to implement the recommendations coming from a human factors engineer.

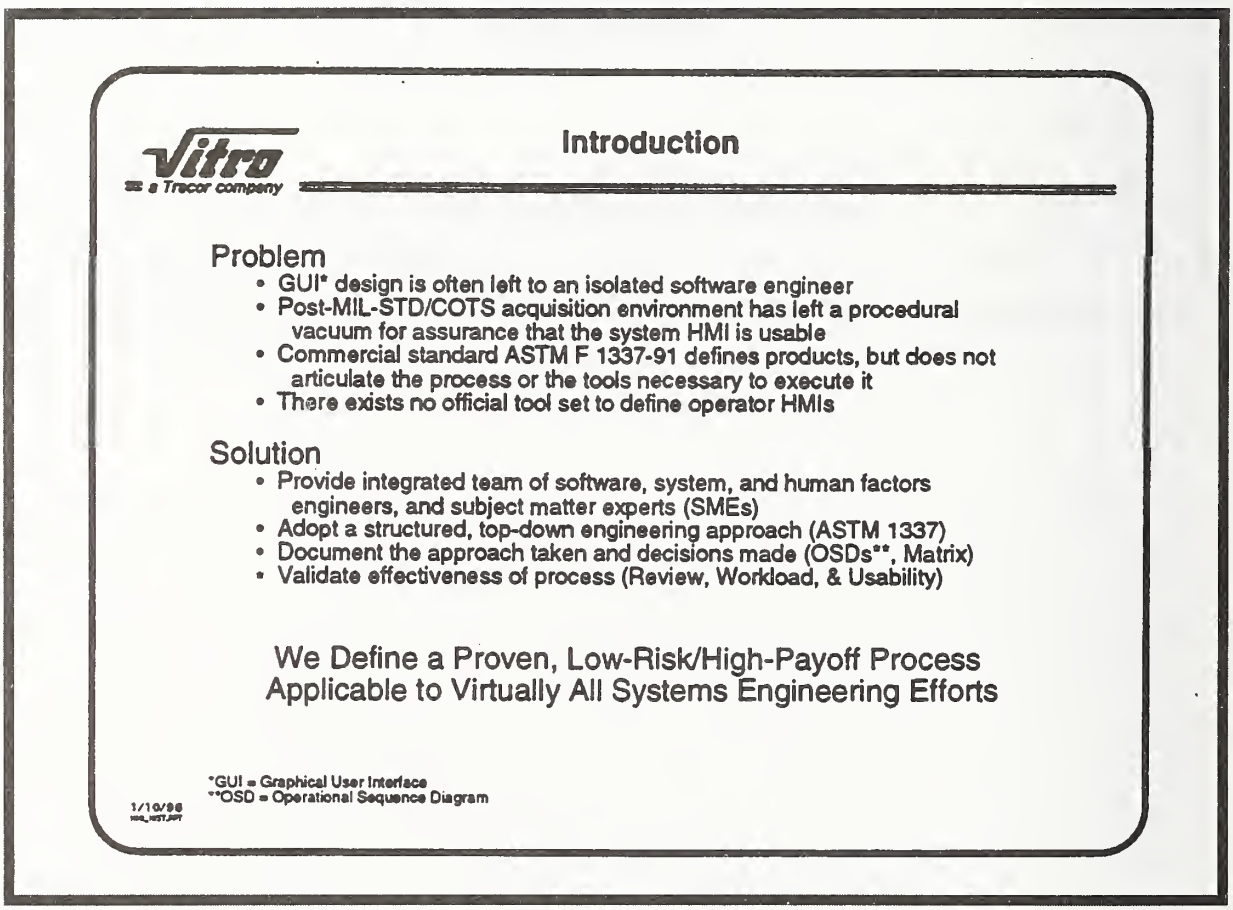


Figure 1

Often the development of graphical user interfaces is done by an isolated software engineer. (Figure 1) He knows how to code the system to accomplish a particular task but he is

systematically different from the end user. We have also found that the post MIL-Standard environment has left a bit of a procedural vacuum. There was a commercial standard mentioned earlier in this symposium, ASTM F 1337-91, which defines products that are important in a user interface development process. The products, however, do not necessarily address the process or the tools. Our solution to this was to provide an integrated team of software engineers, system engineers, and human factors engineers, as well as a core group of about three subject matter experts who we worked with closely from requirements all the way through usability testing.

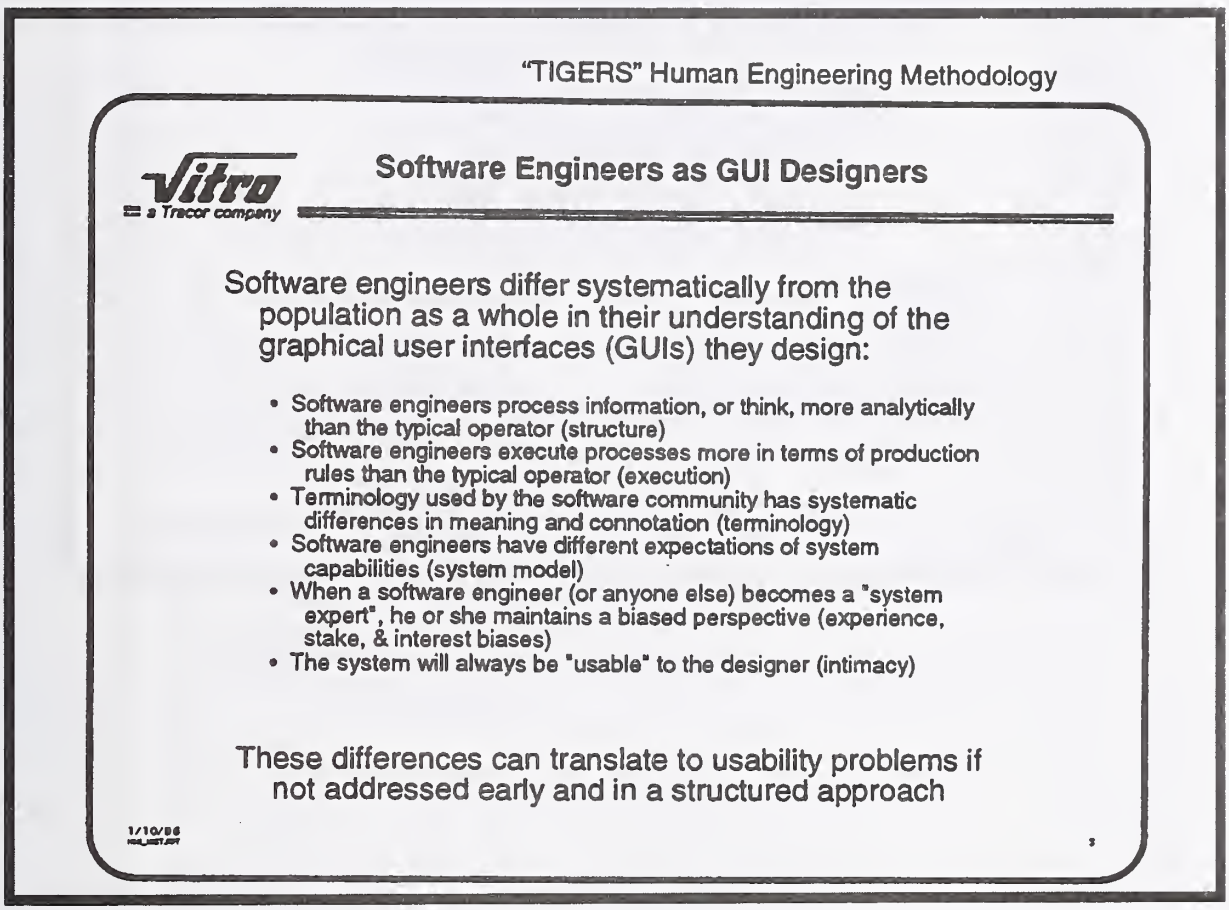


Figure 2

This figure (Figure 2) describes how software engineers differ from software users. These differences occur in how they structure a problem, how they execute their system model, their intimacy with the system, etc. All determine how they design the interface and are all fundamentally at odds with the way the end user is going to be interacting with the system.

Another issue is the acquisition environment. (Figure 3) We have increasing pressure to use commercial off-the-shelf (COTS) technologies and products. We have strong impetus to reduce staffing while at the same time, enhancing the capabilities of these systems.

The system we are using as an example is the NATO Sea Sparrow Surface Missile System. It is a self-defense missile system currently in the fleet that is basically a Navy version of the Patriot self-defense missile used in the Gulf War. The system is being redesigned to move from a

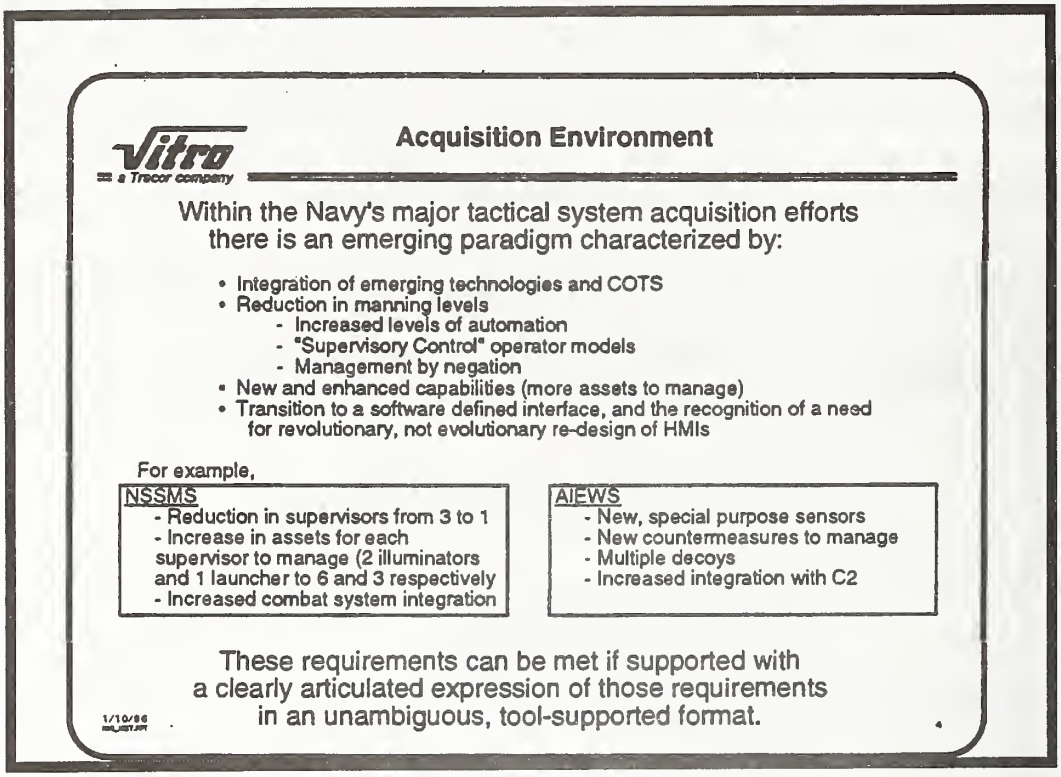


Figure 3

Vitro
= a Tracor company

ASTM F 1337-91

ASTM F 1337-91 identifies the following "Human Engineering Activities" that shall be part of the human engineering program:

- Operational Requirements
- Mission Requirements Analysis
- System Requirements Analysis
- Function Definition
- Function Allocation
- Equipment Selection
- Human Engineering System Analysis Report (HESAR types I & II)
- Gross Task Analysis
- Critical Task Analysis
- Human Engineering Design
- Studies, Experiments, Laboratory Testing (element level)
- Test and Evaluation (system level)

The process defined here supports in whole or in part each of the activities identified above

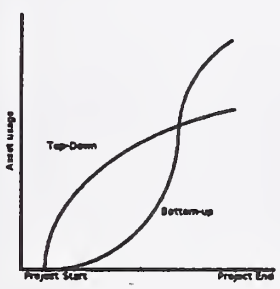
1/10/96

5

Figure 4

Vitro
= a Tracor company

Top-Down Engineering Approach



Although there is greater initial usage in assets, in the long run, a top-down approach:

- Is more cost effective
- Has less risk
- Is more adaptable
- Reduces system testing
- Is more integrated

1/10/96

* From: Elements of Systems Analysis, Gore & Stubbe, 1983

6

Figure 5

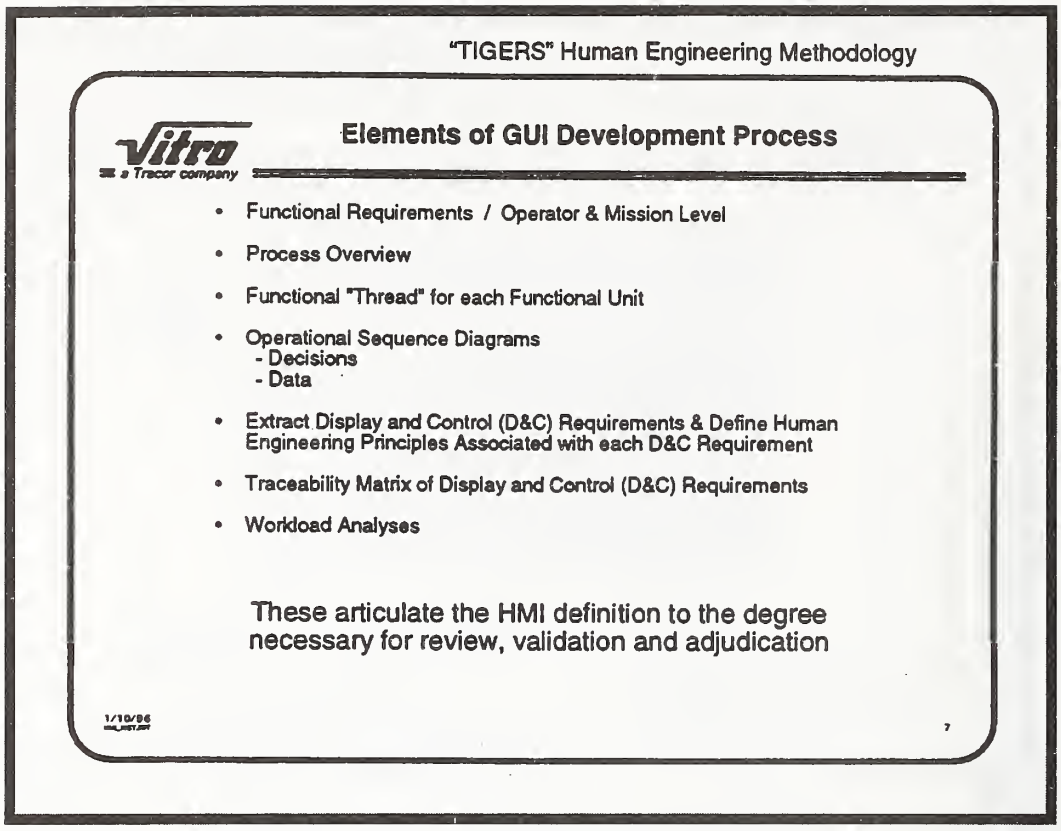


Figure 6

dedicated console with push button indicators and a rather clunky interface to a dual color work station with a graphical user interface. ASTM 1337 (Figure 4) articulates some of the activities and products that should be developed and to support this effort have been implemented in our TIGERS process.

First, we are very much committed to a top-down approach (Figure 5), engineering approach. A user interface like any other software needs to be designed in a disciplined engineering fashion—more science than art although clearly there is an art aspect to it. Note that the problem with a top-down approach, however, is that it is heavily loaded initially for investment but funding schedules tend to follow the bottom-up (see chart) curve.

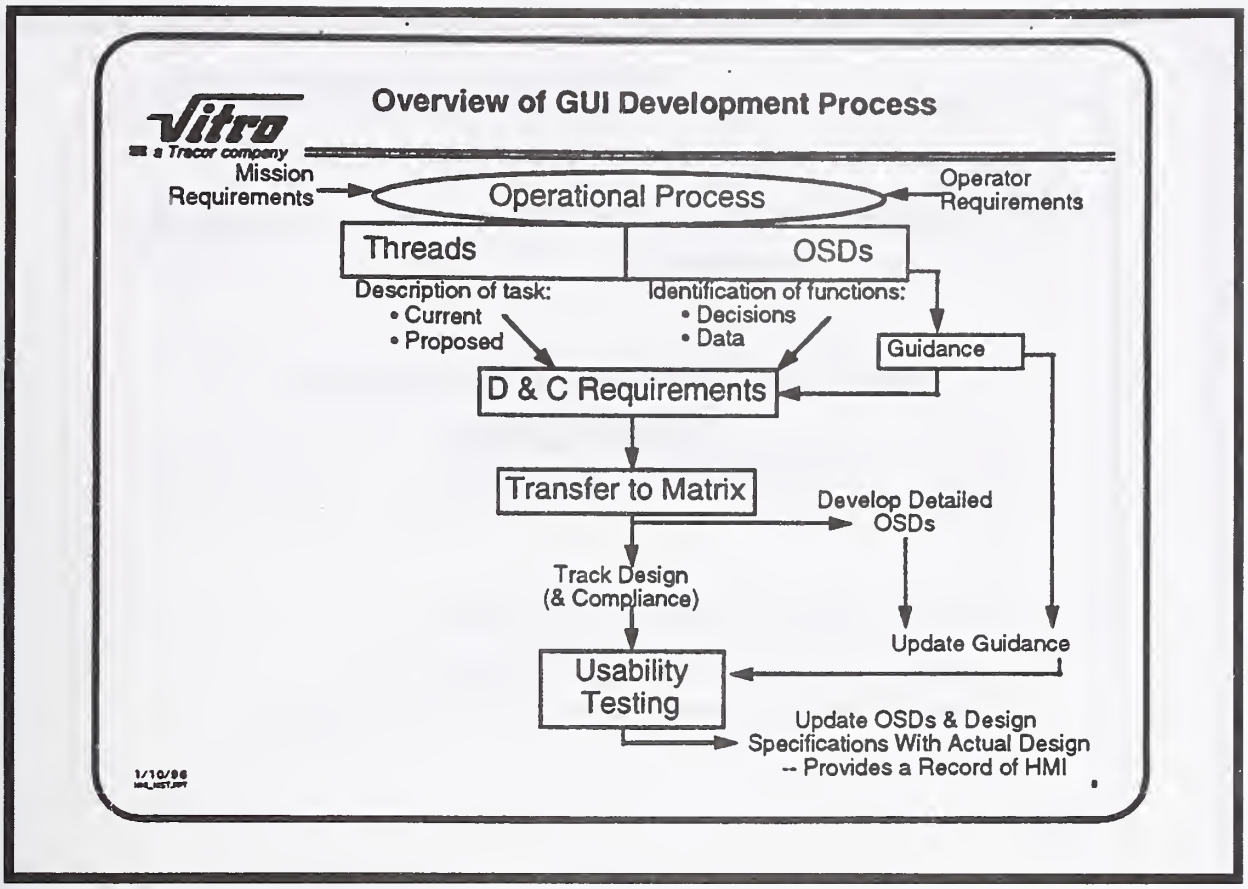


Figure 7

Some of the fundamental elements that we deal with in our process and that we applied to Navy Sea Sparrow are outlined in this chart (Figure 6). We started very early with the functional requirements from the project office level. That is, asking the question, "What do we expect the system to do?" The project office vision was rather simplistic: no reduction in functionality that we currently have. As engineers, we had to articulate the functional requirements by bringing in subject matter experts and asking how to translate this into a preliminary specification or an "A Spec." The subject matter experts were asked about expected functions and what the utility of each function was. Next we articulated an overview of the process, at a very high level, just on two sheets of paper. We took this to the fleet and reviewed it, to validate that these were the anticipated functions. As soon as we got validation at that level, we were able to create a context or a functional thread for all of the tasks that we anticipated for the operator. Once again, this was reviewed by the fleet which modified it and identified the tool that we needed to actually develop operational sequence diagrams. From those, we generated articulated the decisions that the operator had to make and the data required to make those decisions. We applied guidelines from the human factors literature to create the display and control requirements to facilitate the execution of these tasks by the operator. These were placed in a traceability matrix and workload analyses were performed.

The overview chart (Figure 7) shows the GUI development in relationship to the requirements development process.

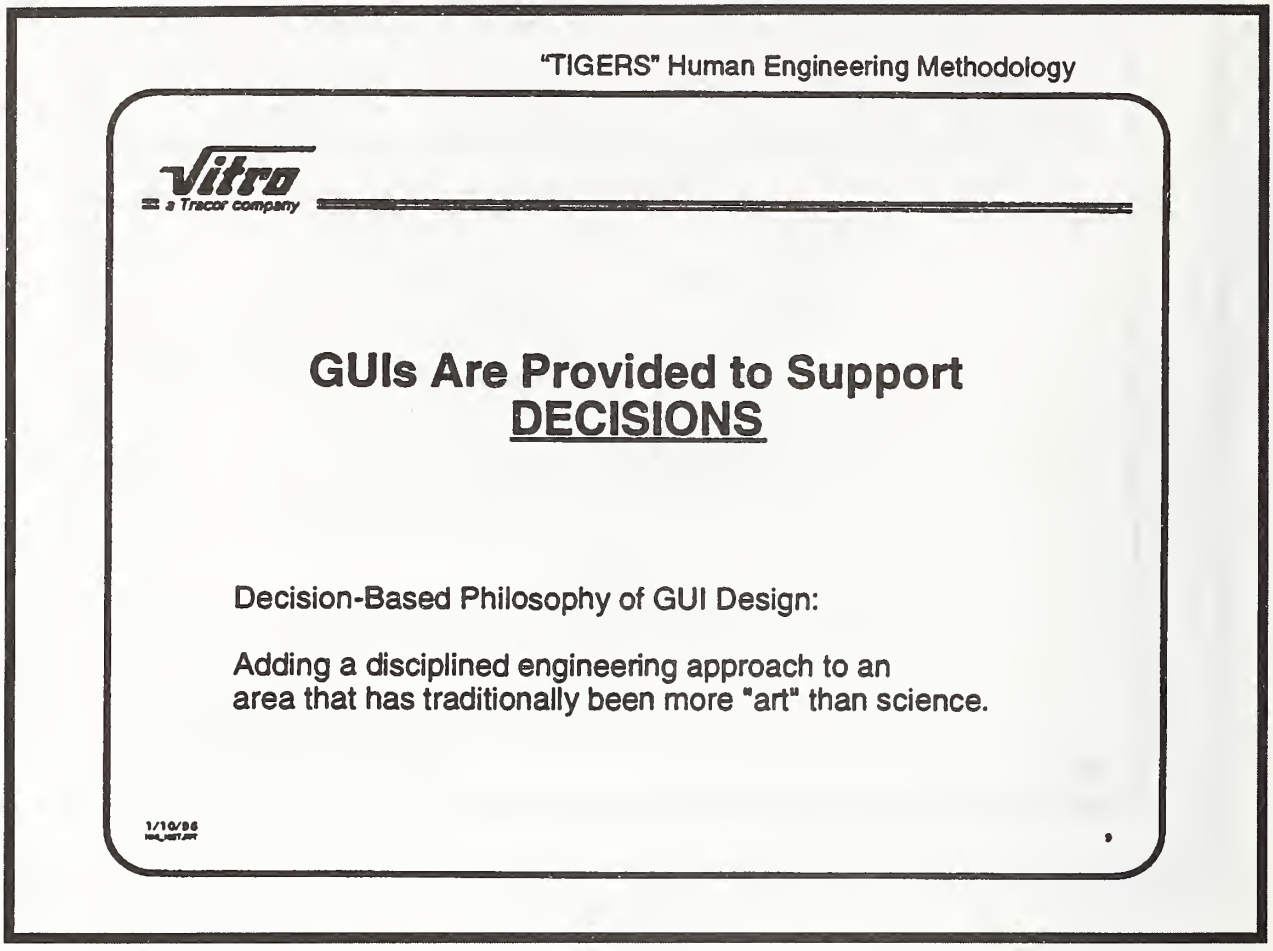


Figure 8

The philosophy driving all of this is to recognize the operator's role as the design is developed. This is especially important in this case for a real time combat system. The interface is there to facilitate the making of decisions. (Figure 8)

The operational sequence diagrams (OSDs) are made up of task units (Figure 9) where we identify the decision or function to be performed, articulating explicitly what the cue was to the operator that something needed to be done. This is not necessarily an external event. It may be an internal event, but we identified all events explicitly. We did this by once again going back to our subject matter experts and asking what information they look for to make that decision. The format they expect that information to be in, and then generated the control requirements that were necessary.

Here (Figure 10) we have the interaction or the actions that the operator is performing with the interface itself. The task, the data requirements and response all go into the format options.

We can fold this information back into our OSD's now (Figure 11) with the actual display elements that are going to be used by the operator. This is a very useful tool, because once the OSD's are developed we can take them to the fleet again for feedback are being made and what information is being sought.

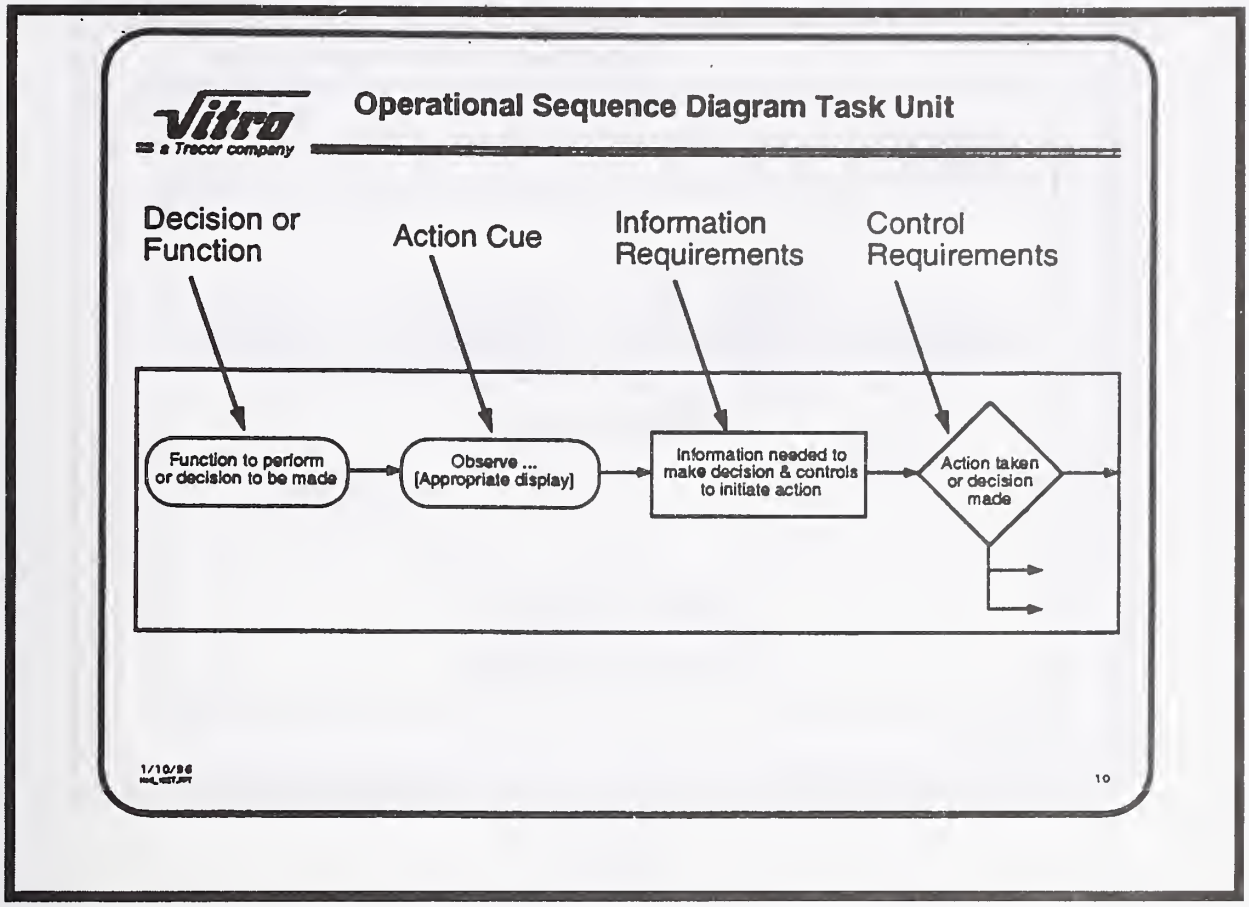


Figure 9

We were able to get validation from the fleet (Figure 12) about the tasking and display.

The workload analyses are described in the next set of charts (Figures 13–17). It is based on the use of a tool that assesses the tasking interactions as well as usability testing with representative operators. The analysis also includes a determination of task suitability for automation or human control.

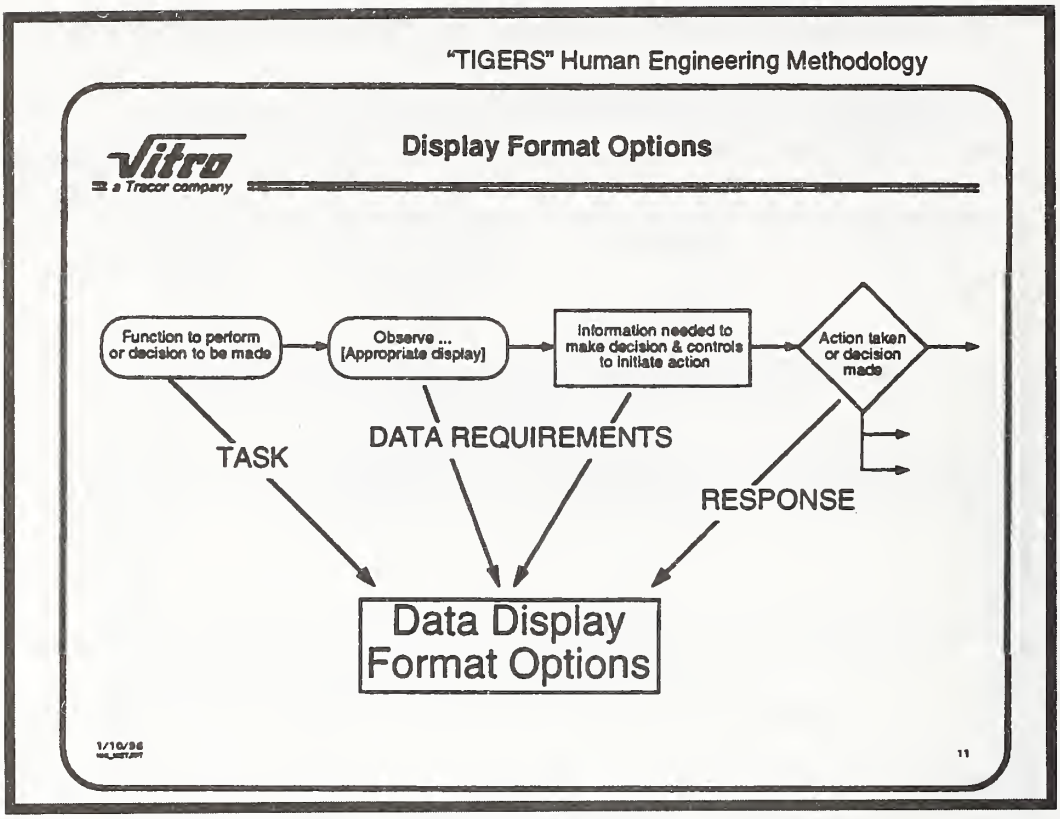

"TIGERS" Human Engineering Methodology
Vitro
a Tracor company
Display Format Options
Function to perform or decision to be made
Observe ... [Appropriate display]
Information needed to make decision & controls to initiate action
Action taken or decision made
TASK
DATA REQUIREMENTS
RESPONSE
Data Display Format Options
1/10/96
11

Figure 10

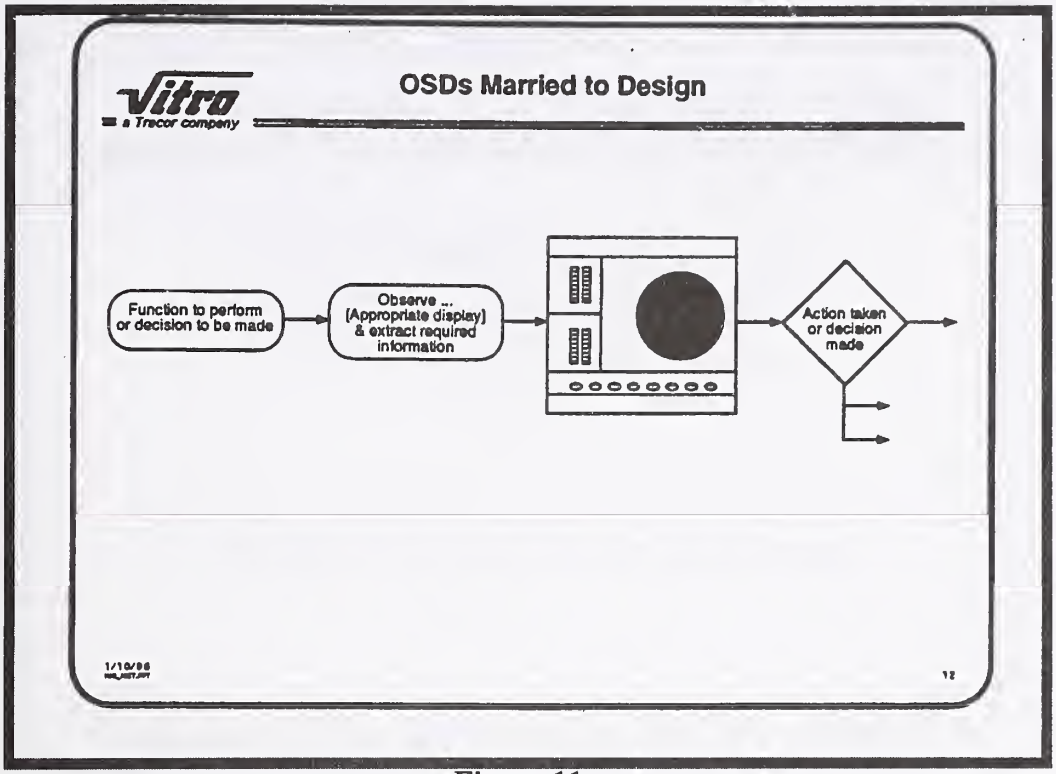

Vitro
a Trecor company
OSDs Married to Design
Function to perform or decision to be made
Observe ... (Appropriate display) & extract required information
Action taken or decision made

Figure 11

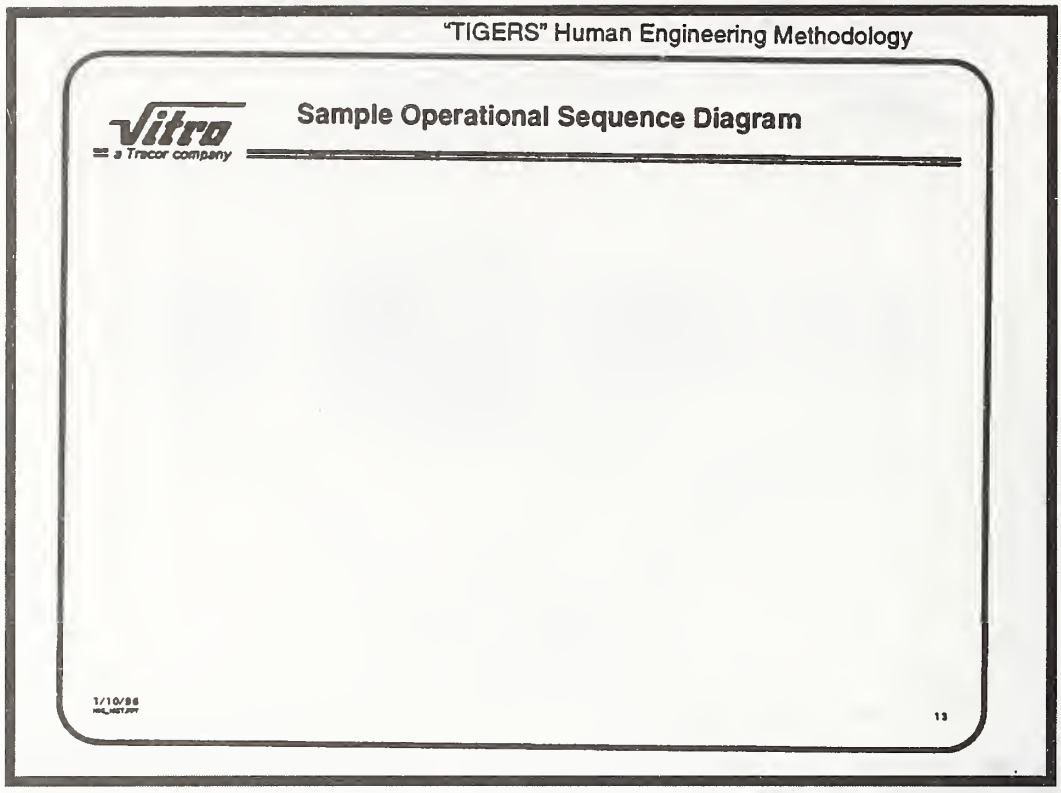


Figure 12

(intentionally blank)

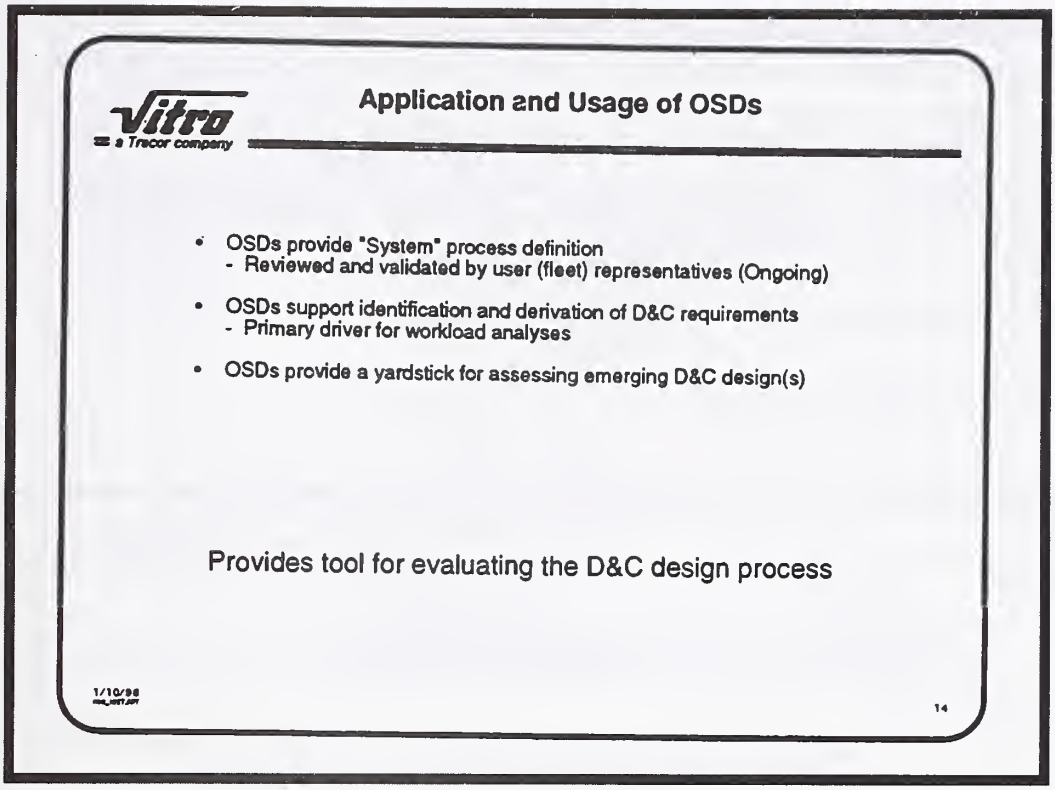


Figure 13

The traceability matrix (Figure 18) is an important part of this process because it forces us to articulate explicitly what the requirements are, what drove the requirements, where they have been allocated in the interface, how they are utilized by the operator, whether they are a transient function, on call, or full time. The matrix also includes the principles associated with implementation and any software trouble reports or software requirement specification references.

(Sample Traceability Matrix is blank in proceeding per speaker.)

Here is an example of a traceability matrix that we used for NATO Sea Sparrow. What is important is being able to trace every single requirement, show where in the interface has been implemented and how it is going to be used. A programmer can then take this matrix and pictures that we use in design guidelines, and actually code an interface that has already been, in part, validated by the users by going through this process.

With respect to usability testing (Figure 19), after we had done this iterative process using pieces of paper and mock ups, we, just a few months ago, took the prototype system to the fleet with representative operators. We used 10 operators with representative scenarios. We collected these measures.

The approach in the interface was accepted or validated by the fleet and we were able to get a good collection of specific recommendations that we could then take back to Raytheon who was coding the interface. Here are some sample recommendations. (Figure 20) Also, we taped the sessions to show to some of the developers. Note that we had some of the Raytheon representatives there at the usability test with us and they became some of our greatest champions as we were going through the negotiations of the final design specification and it was very effective.

We have a matrix showing how our process addressed each of the ASTM requirements. (Figure 21)

In summary, see figures 22 and 23.

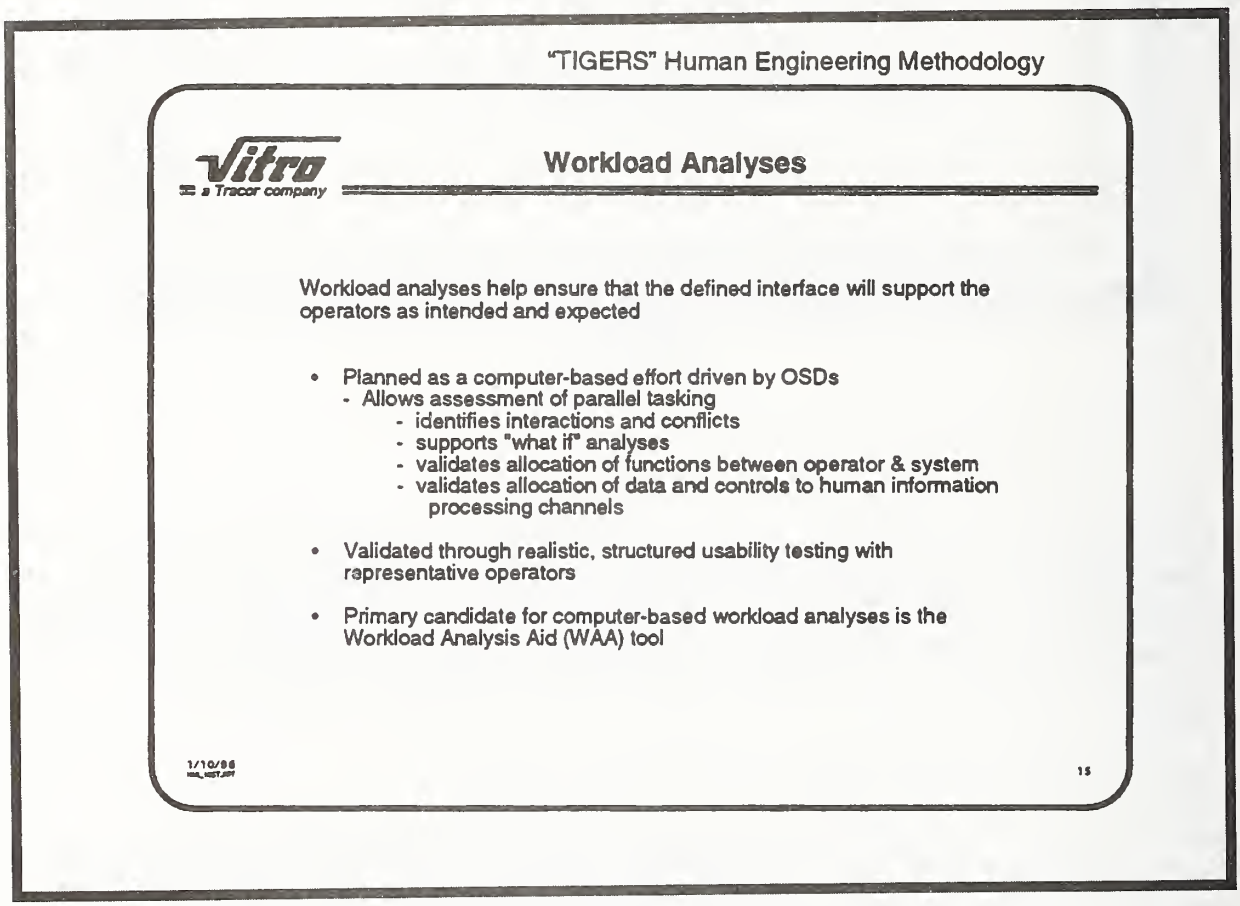


Figure 14

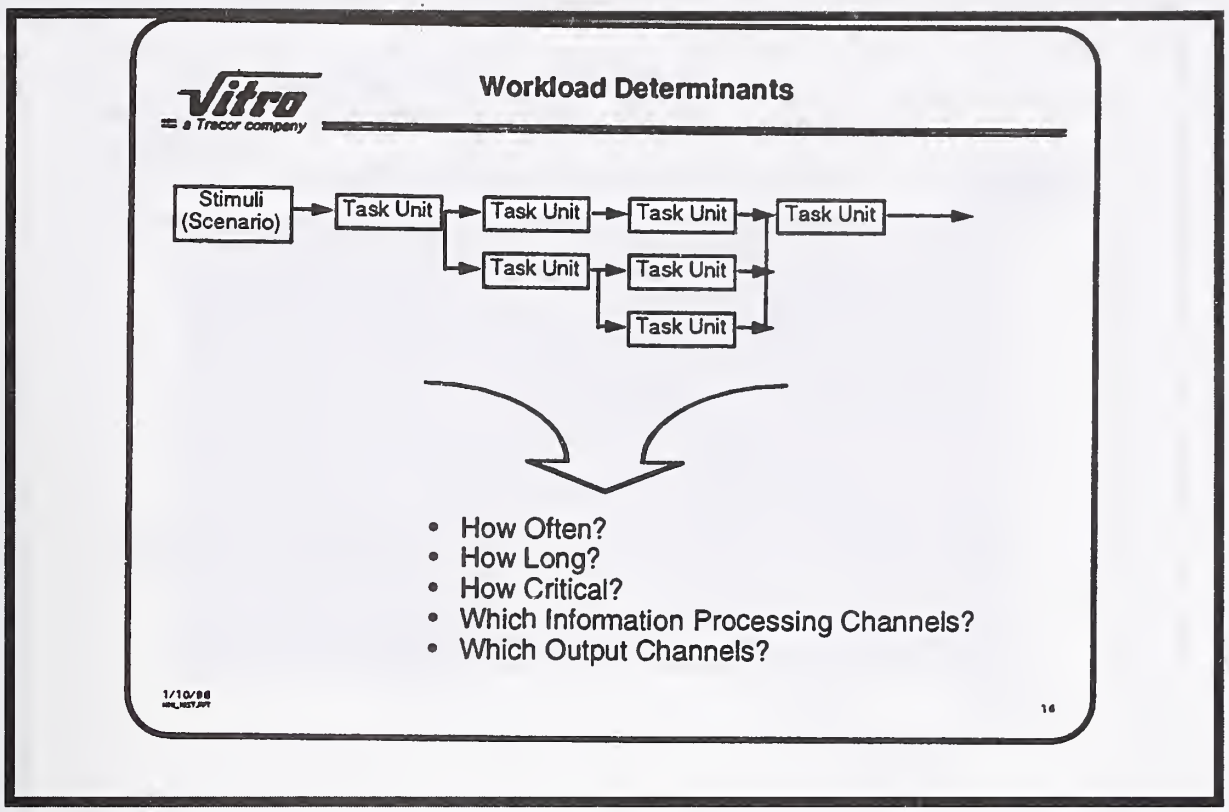
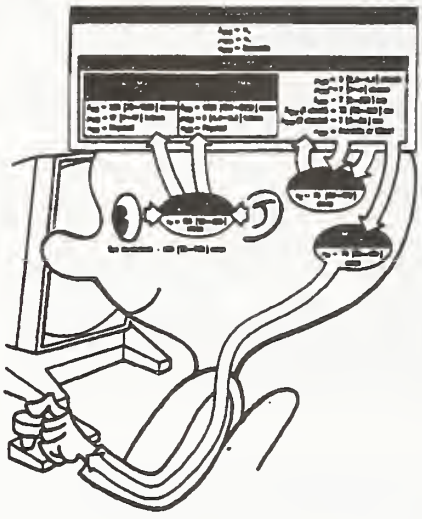

Vitro
a Tracor company
Workload Determinants
Stimuli (Scenario)
Task Unit
Task Unit
Task Unit
Task Unit
Task Unit
Task Unit
Task Unit
• How Often?
• How Long?
• How Critical?
• Which Information Processing Channels?
• Which Output Channels?
1/10/96
14

Figure 15

Vitro
a Tracor company

Human Information Processing Model



Processing Channels

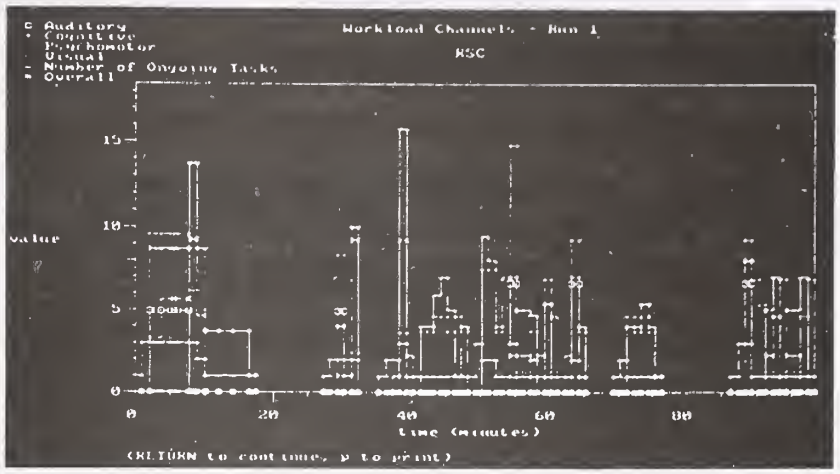
- Cognitive
- Visual
- Auditory
- Motor

1/10/96

17

Figure 16

Actual WAA Channel Loadings for NSSMS



1/10/96

18

Figure 17

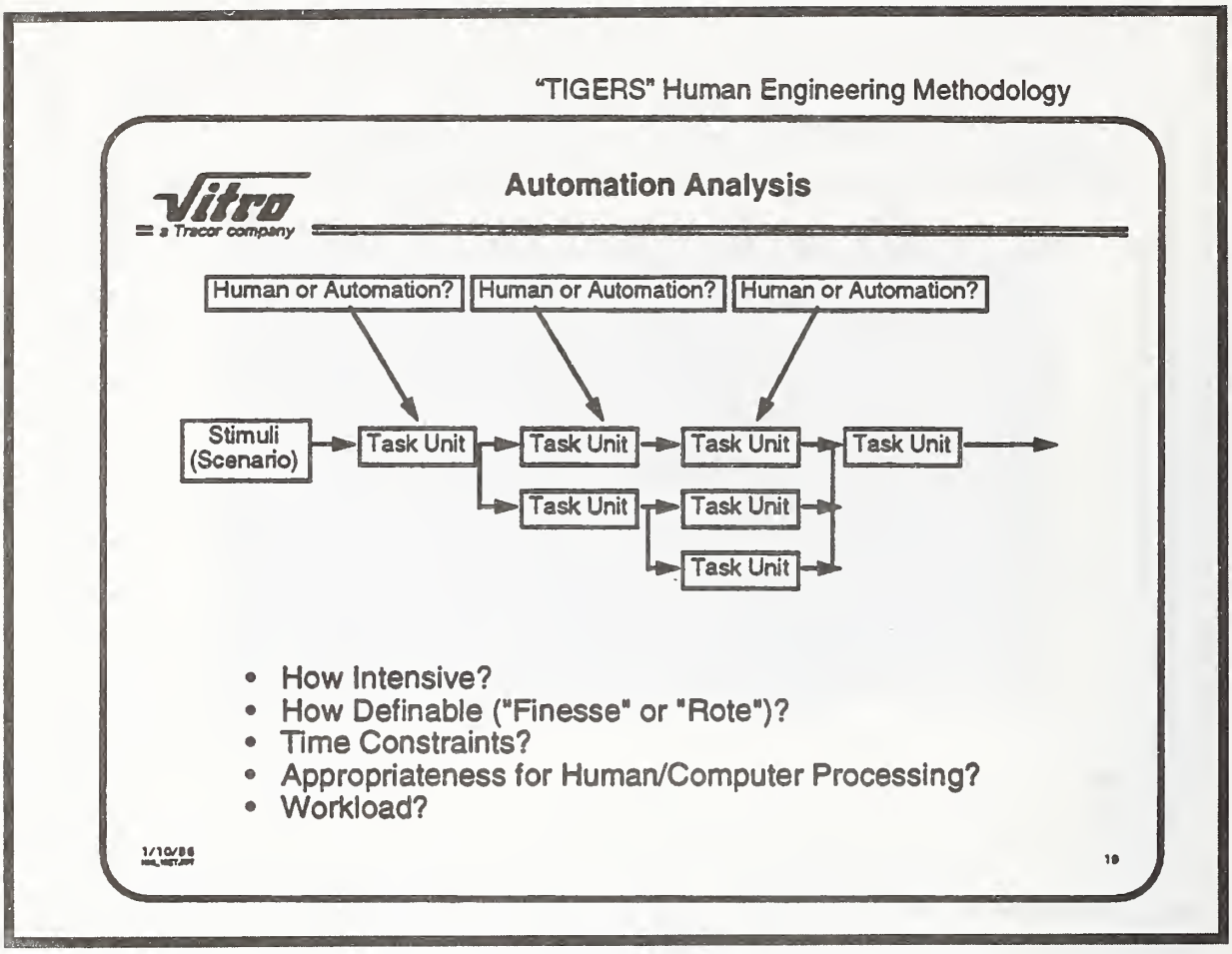

"TIGERS" Human Engineering Methodology
Vitro
a Tracor company
Automation Analysis
Human or Automation?
Human or Automation?
Human or Automation?
Stimuli (Scenario)
Task Unit
Task Unit
Task Unit
Task Unit
Task Unit
Task Unit
Task Unit
• How Intensive?
• How Definable ("Finesse" or "Rote")?
• Time Constraints?
• Appropriateness for Human/Computer Processing?
• Workload?
1/10/86
18

Figure 18

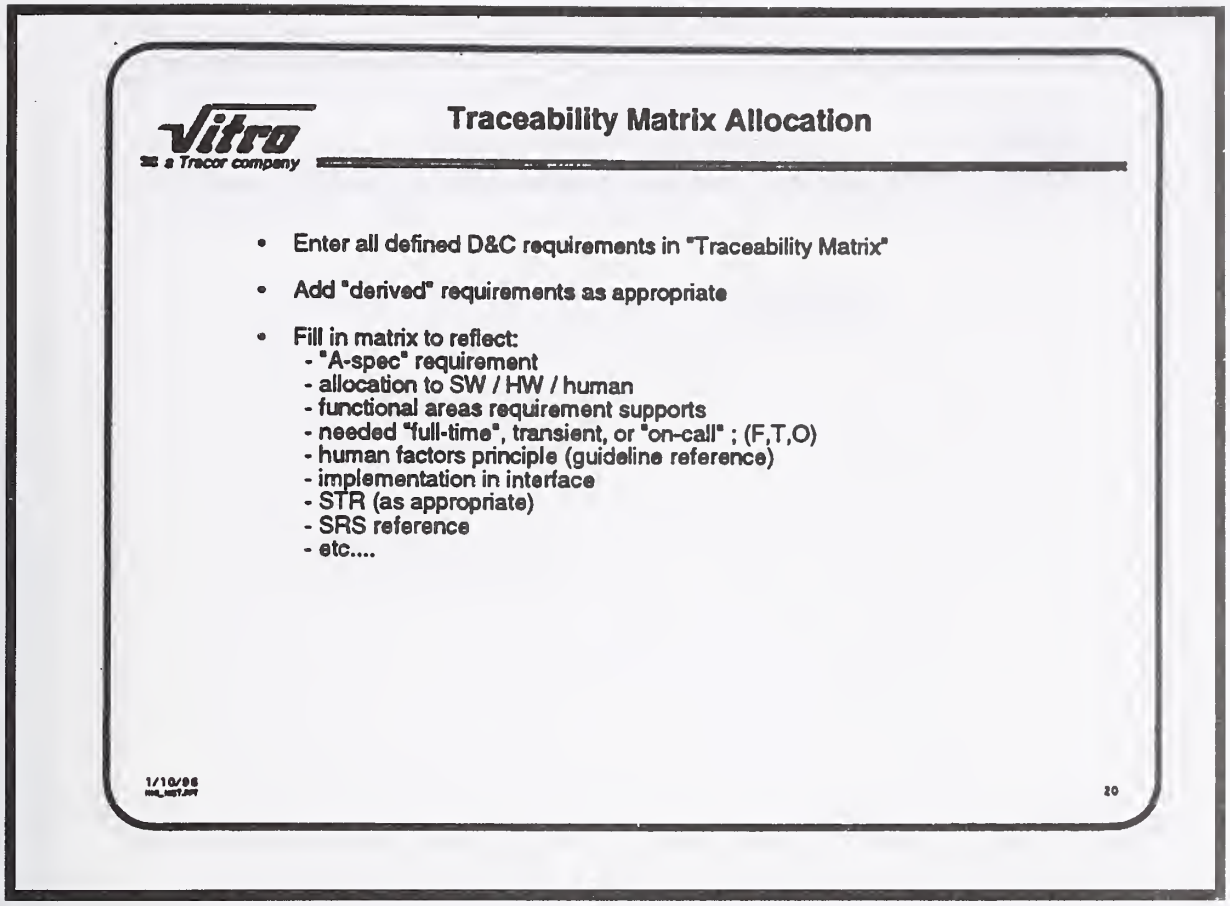

Vitro
a Tracor company
Traceability Matrix Allocation
• Enter all defined D&C requirements in "Traceability Matrix"
• Add "derived" requirements as appropriate
• Fill in matrix to reflect:
- "A-spec" requirement
- allocation to SW / HW / human
- functional areas requirement supports
- needed "full-time", transient, or "on-call" ; (F,T,O)
- human factors principle (guideline reference)
- implementation in interface
- STR (as appropriate)
- SRS reference
- etc....
1/10/96
20

Figure 19

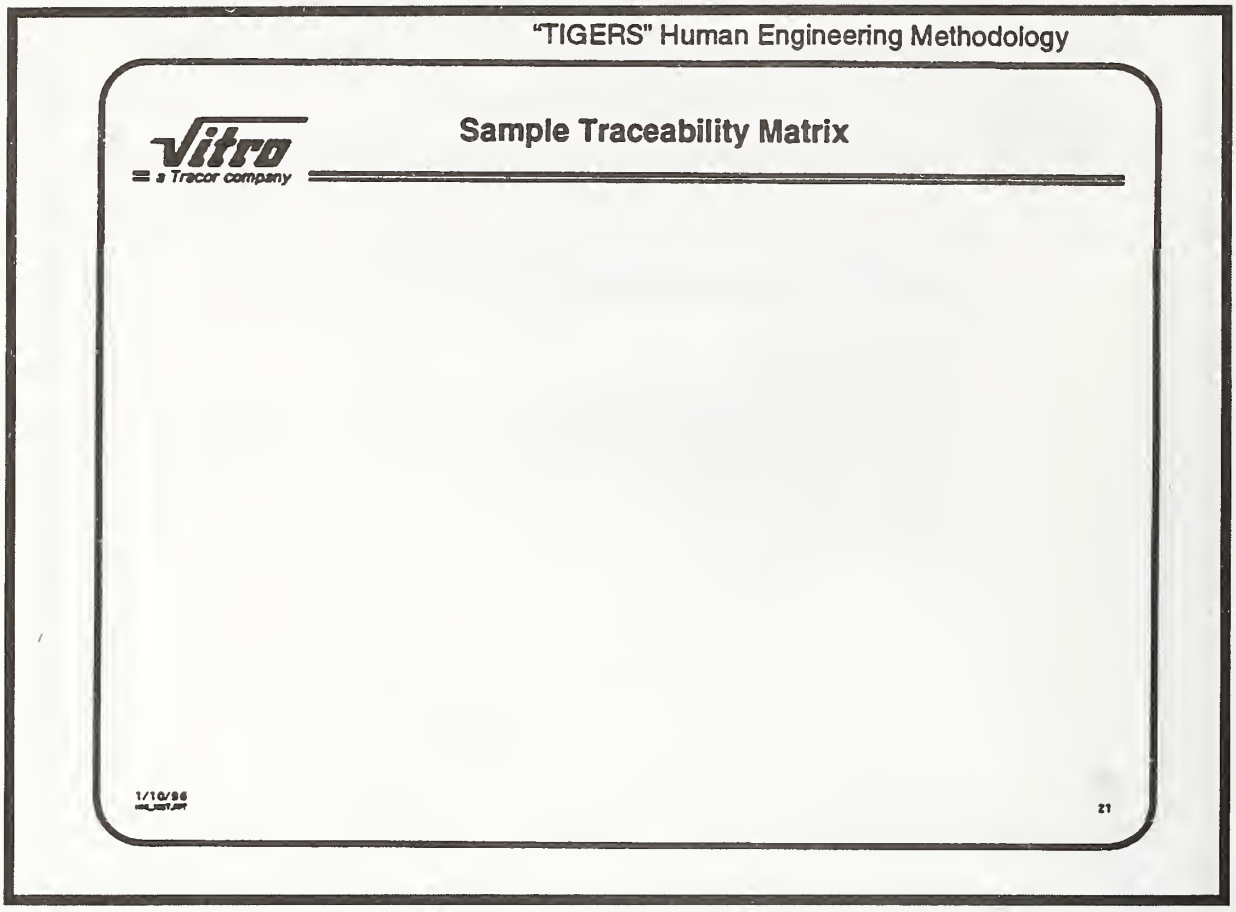


Figure 20

(intentionally blank)

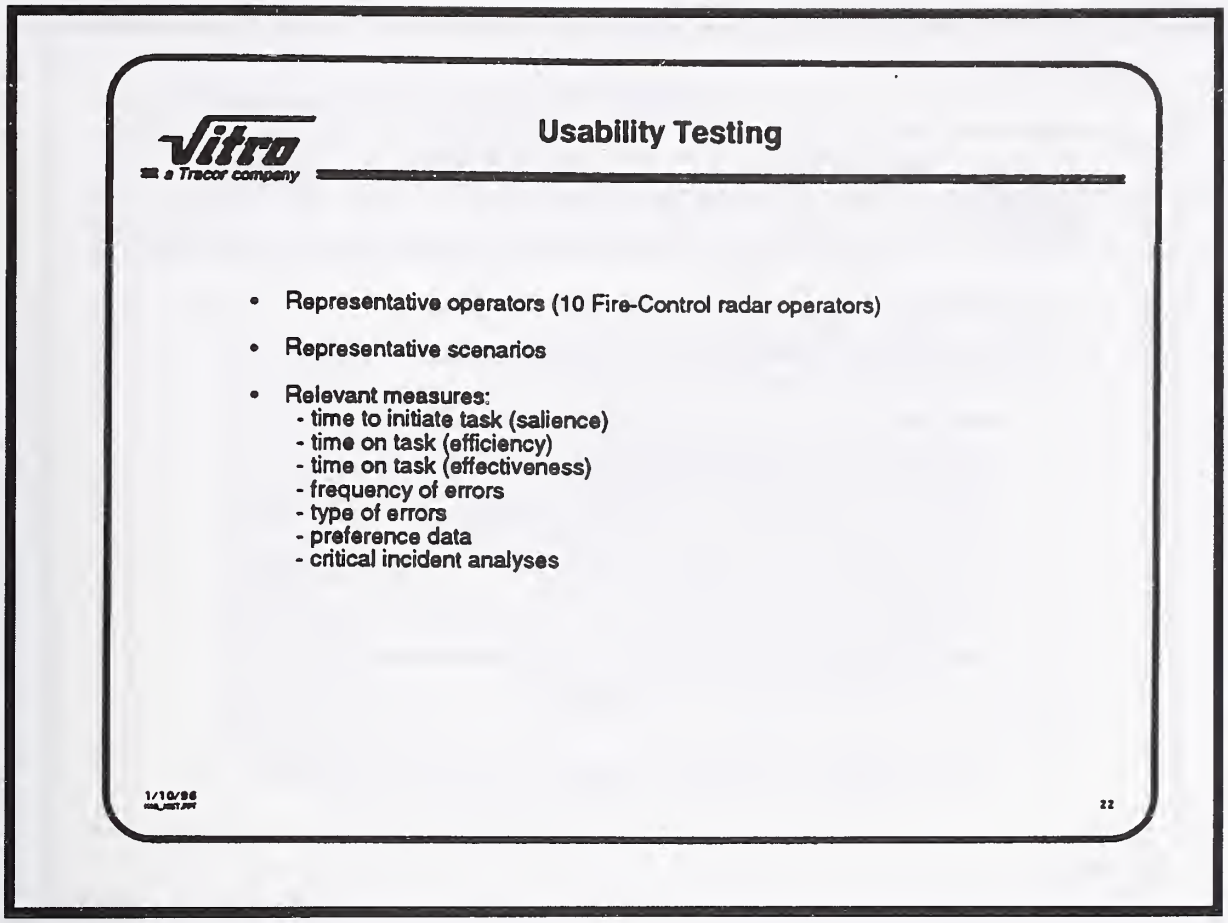

Vitro
a Tracor company
Usability Testing
• Representative operators (10 Fire-Control radar operators)
• Representative scenarios
• Relevant measures:
- time to initiate task (salience)
- time on task (efficiency)
- time on task (effectiveness)
- frequency of errors
- type of errors
- preference data
- critical incident analyses
1/10/96
22

Figure 21

Vitro
a Tracor company

Usability Results

- Operator acceptance of approach validated
- Data required by operators was (mostly) validated, but location of that information was sub-optimal in some cases
- Implementation of certain display elements needs re-engineering: e.g.
  - radar cutouts should be displayed on the map display as a default
  - contrast between land and sea was too great on the map display
  - operators desired to "hook" a target from the engagement schedule
  - operators desired to "hook" a target from any alert pertaining to that target
  - auditory cueing for critical alerts is needed for salience
  - some actions that are required from the operator on the basis of a critical alert should be executable from that alert
  - certain data fields that the operator has to enter in some forms can be auto-filled with default or related data
  - operators desired to have search return data displayed directly on their map display (to correlate with existing tracks)
  - etc.... ~160 distinct design recommendations

Final design specification agreed to by both contractor and Government human engineering activity

1/10/96

23

Figure 22

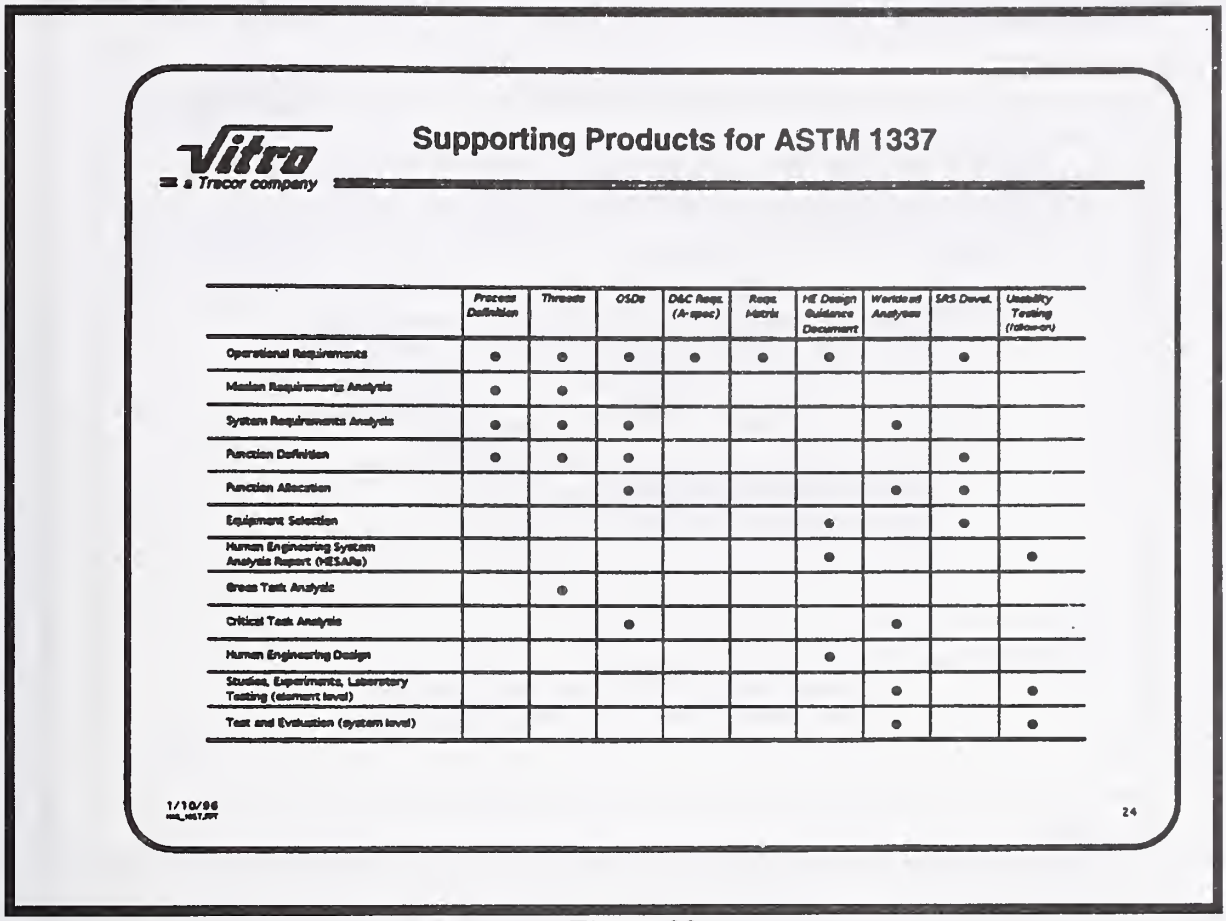

Vitro
a Tracor company
Supporting Products for ASTM 1337
Process Definition
Threats
OSDs
D&C Reqs. (A-spec)
Reqs. Matrix
HE Design Guidance Document
Workload Analyses
SRS Devel.
Usability Testing (follow-on)
Operational Requirements
Mission Requirements Analysis
System Requirements Analysis
Function Definition
Function Allocation
Equipment Selection
Human Engineering System Analysis Report (HESARs)
Gross Task Analysis
Critical Task Analysis
Human Engineering Design
Studies, Experiments, Laboratory Testing (element level)
Test and Evaluation (system level)
1/10/96
24

Figure 23

Vitro
a Tracor company

Application and Usage of Collateral Products

- Requirements traceability matrix provides mapping of system requirements (A-spec) to D&C elements, human engineering principles, display implementation and, ultimately, the SRSs
- Matrix reflects articulation of HW / SW / Human function allocation
- Guidance document provides design parameters for Design Agent
  - Not necessarily design requirements
  - Recommends optimal display principles
  - Provides "compliant" implementation

Primary tools for tracking, auditing and documenting the D&C design process

1/10/96

25

Figure 24

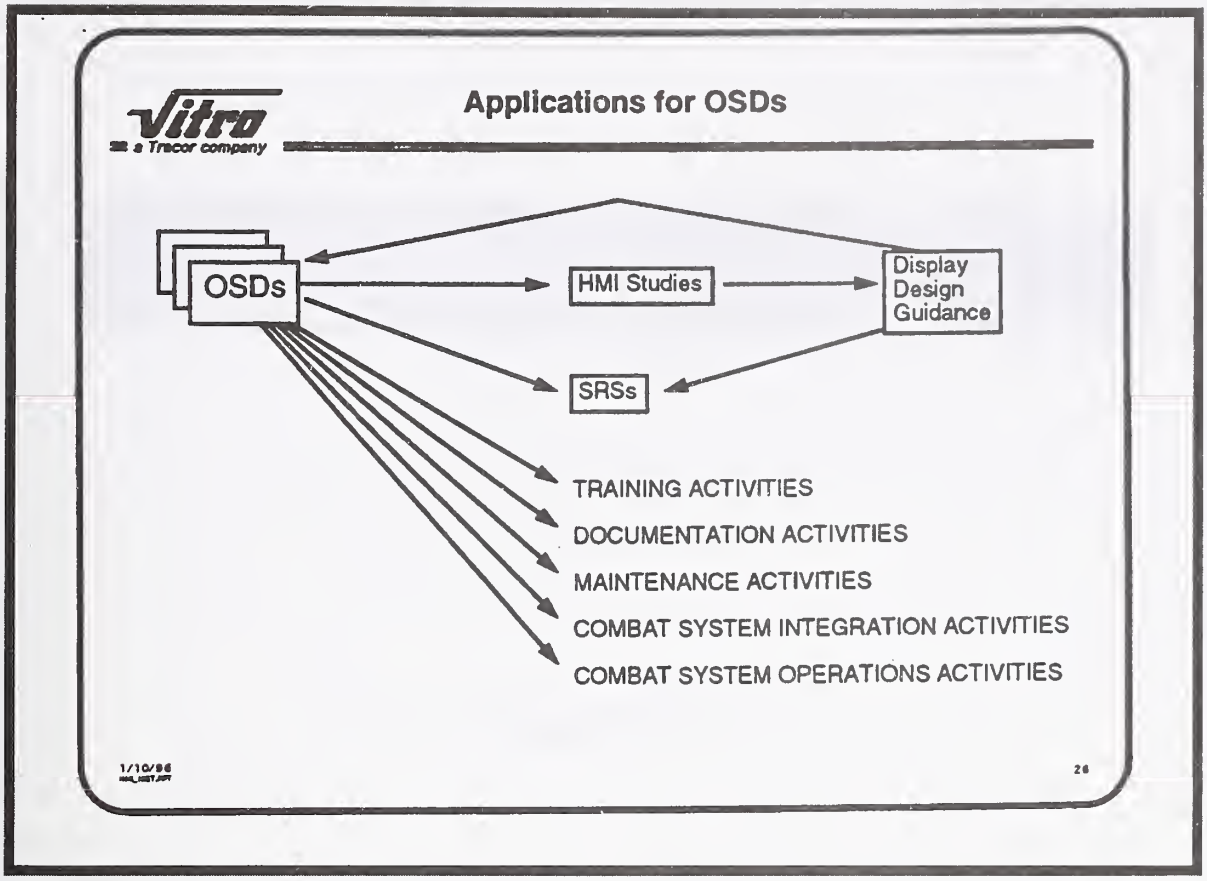

Vitro
a Trecor company
Applications for OSDs
OSDs
HMI Studies
Display
Design
Guidance
SRSs
TRAINING ACTIVITIES
DOCUMENTATION ACTIVITIES
MAINTENANCE ACTIVITIES
COMBAT SYSTEM INTEGRATION ACTIVITIES
COMBAT SYSTEM OPERATIONS ACTIVITIES
1/10/96
26

Figure 25

Usability Engineering in Europe

Nigel Bevan

NPL Usability Services
National Physical Laboratory
Teddington, Middlesex
TW11 0LW, UK

Nigel@hci.npl.co.uk
Tel: +44 181-943 6993
Fax: +44 181-977 7091

Dr. Nigel Bevan, NPL
Usability Engineering in Europe

OK, let me begin by telling you what I'm doing. I'm head of the usability group at the National Physical Laboratory in the UK. It's a pretty similar organization to NIST, except we have been doing work in this field for a long time, around 20 years, so we've built up quite a bit of expertise and in particular we have developed methods for usability specification and measurement. I'm also involved in a number of collaborative European projects. There is funding in Europe to bring together research organisations and industry working on precompetitive collaborative research projects, and it's one of these projects (MUSiC) which funded us to develop usability measurement methods. I'm currently involved in new projects to disseminate usability information in Europe, and I will use these to illustrate some of the European prejudices to usability.

While we are waiting for other people to arrive, why don't you tell me a little bit about what you're doing.

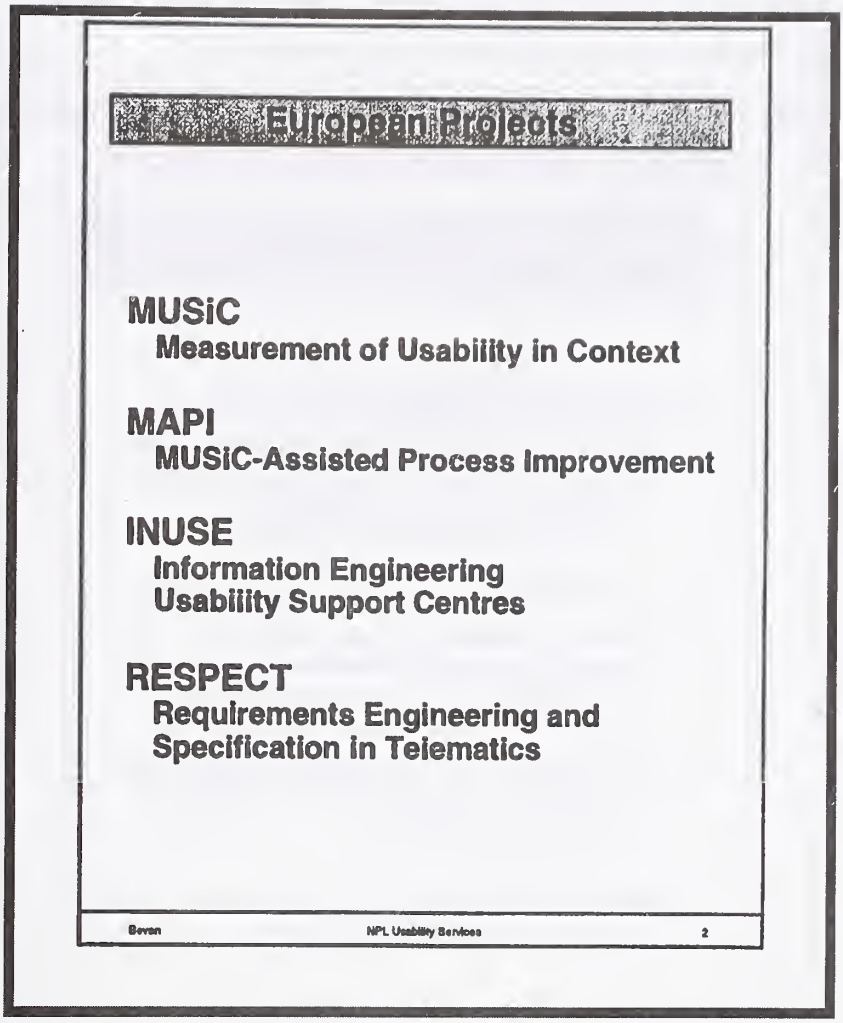


Figure 1

[audience responds]

Bevan: Good. First of all the European legislation which impacts on usability. Some of you may be familiar with this—the European Commission Display Screens Directive. It is implemented as health and safety legislation in European Union countries. This primarily deals with hardware ergonomics: the keyboard has to be adjustable, no reflections on the screen, your chair has to be comfortable and the desk has to be the right height. In addition to these ergonomic issues, it requires that systems are easy to use. I think that in Europe we just have a great respect for standards compared to the US. Organisations respect legislation intended to protect the health and safety of office workers. International standards on ergonomics provide the foundation for these regulations. If an employer complies with ISO 9241 (Ergonomic requirements for office work with visual display terminals) they will also meet the requirements of the legislation.

Being involved in the development of international standards myself, I have got the impression that that the last thing that the American industry wants is international standards. The European

legislation could be said to outlaw the use in offices of command-line operating systems like DOS which do not implement the principles of software ergonomics.

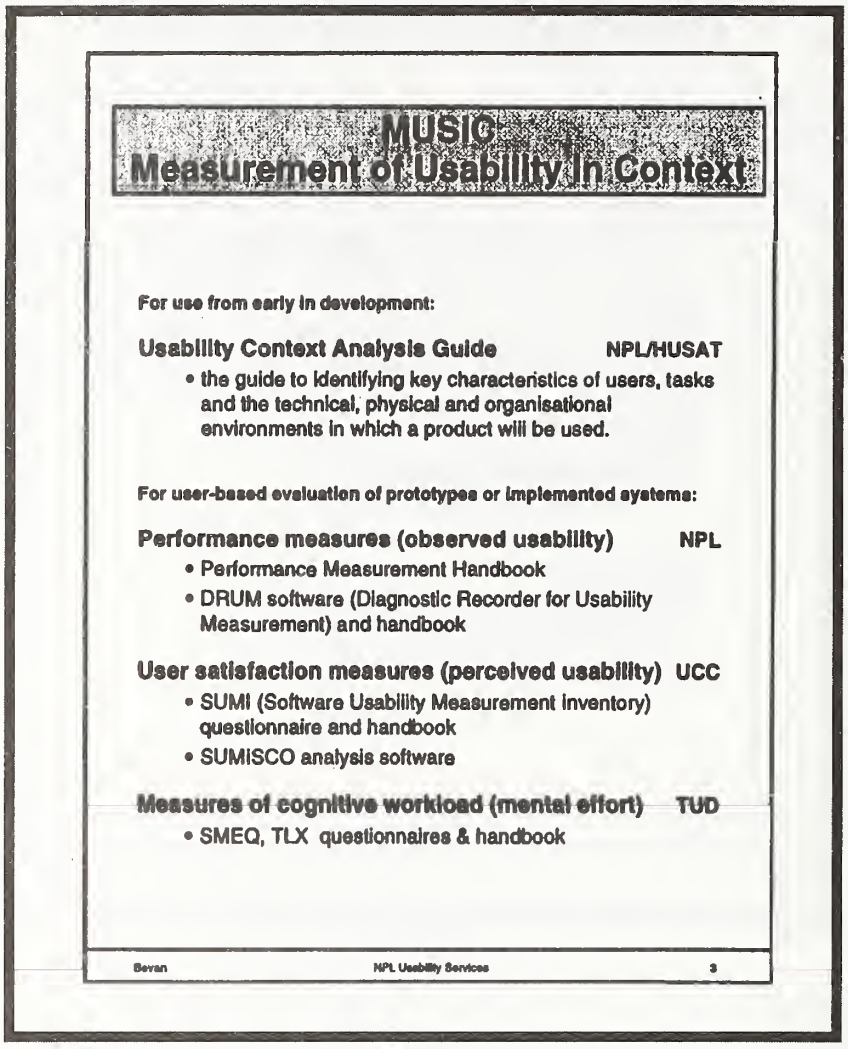


Figure 2

Female: So the practice though, how does that work?

Bevan: It's actually mostly a matter of large organisations doing their best to comply with the legislation. It would only be likely to get to court if the worker's health was impaired. This is quite possible if badly designed software and working practices led to some form of repetitive strain injury. It does indicate a willingness and interest in Europeans trying to legislate for these areas. There's also of course a lot of interest in the use of standards in Europe.

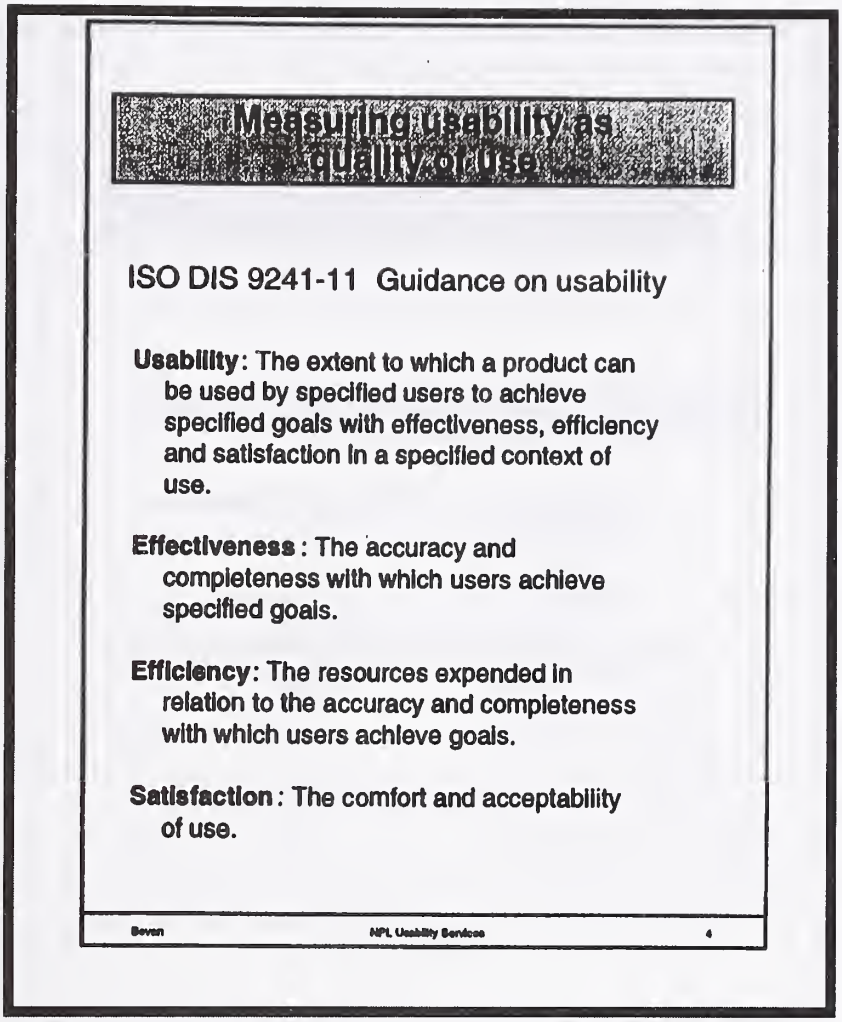


Figure 3

What I'm think I'm going to do is briefly talk about some of the European projects I've been involved in and the influence they have had on the development of standards. The first one was MUSiC (Measurement of Usability in Context). This ran from 1990 to 1993, and helped develop the ISO 9241-11 standard for usability which I edit. It defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." This is a very broad approach to usability. MUSiC developed tools for measuring effectiveness, efficiency and satisfaction, and even more importantly developed Usability Context analysis for identifying and specifying the context of use of a product - the details of the users, tasks and environments.

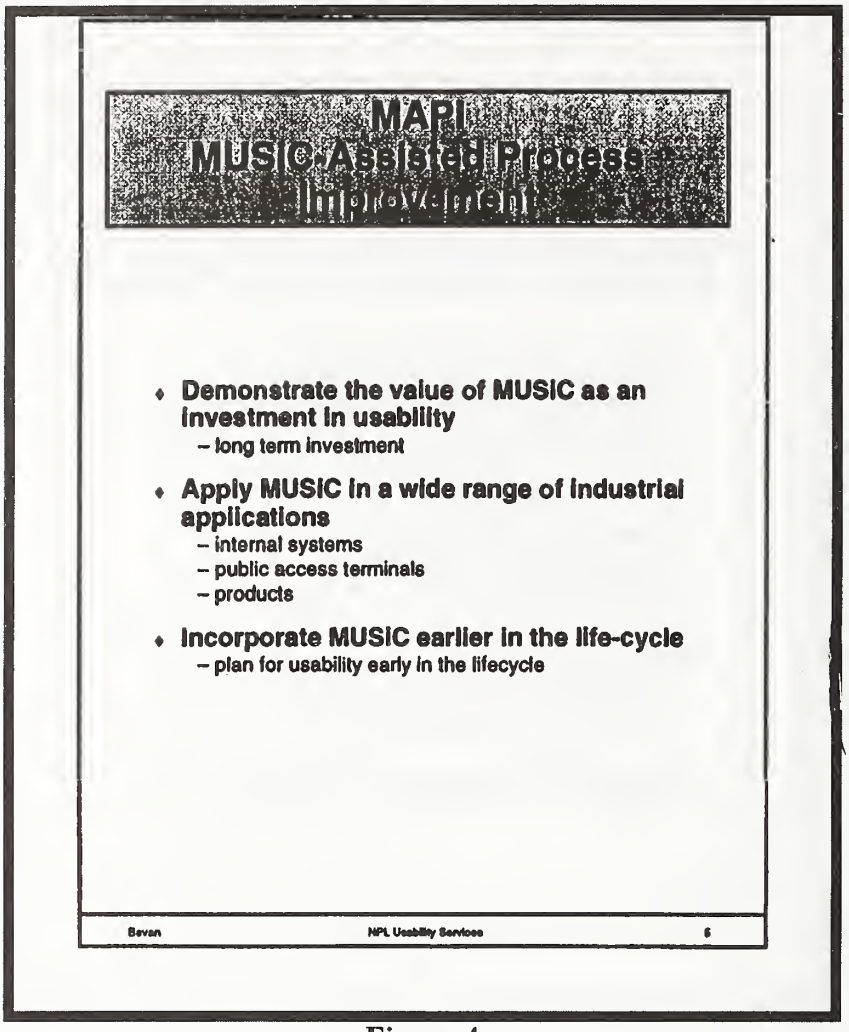


Figure 4

This put Europe in the lead for usability measurement, and this is a topic which is the US has not yet caught up with.

I'm also responsible for two projects, INUSE and RESPECT which have set up a network of European Usability Support centres to transfer information about usability to European industry. I'm also very interested in bringing together usability and quality in those projects. For those of you not familiar with the ISO 9126 standard for software quality characteristics. This is used both here and abroad to provide a structure for defining and measuring software quality. It defines usability as the properties of a product which make it easy to use. This contrasts with Usability which is seen as a quality objective in ISO 9241-11. In this broad view usability means that people get the job done effectively, they get things done right and they get things done efficiently and they are satisfied - they enjoy doing it. Effectiveness and efficiency are very close to productivity as a business objective.

So there are actually two complementary approaches to usability: usability as a user interface issue is only one contribution to usability as effectiveness, efficiency and satisfaction. To achieve that requires that not only is the product easy to use, but also has to have the right

functionality, be efficient (has an acceptable response time) and is reliable. Thus its actually a high level quality objective. So what I have done in the software quality standards ISO 14598-1 (Software product evaluation) and ISO 9126-1 is define quality in use as the combined effect of all the quality properties when a product in use. This is actually the same thing as usability in ISO 9241-11, but its more appropriate to call it quality in use in a software engineering context.

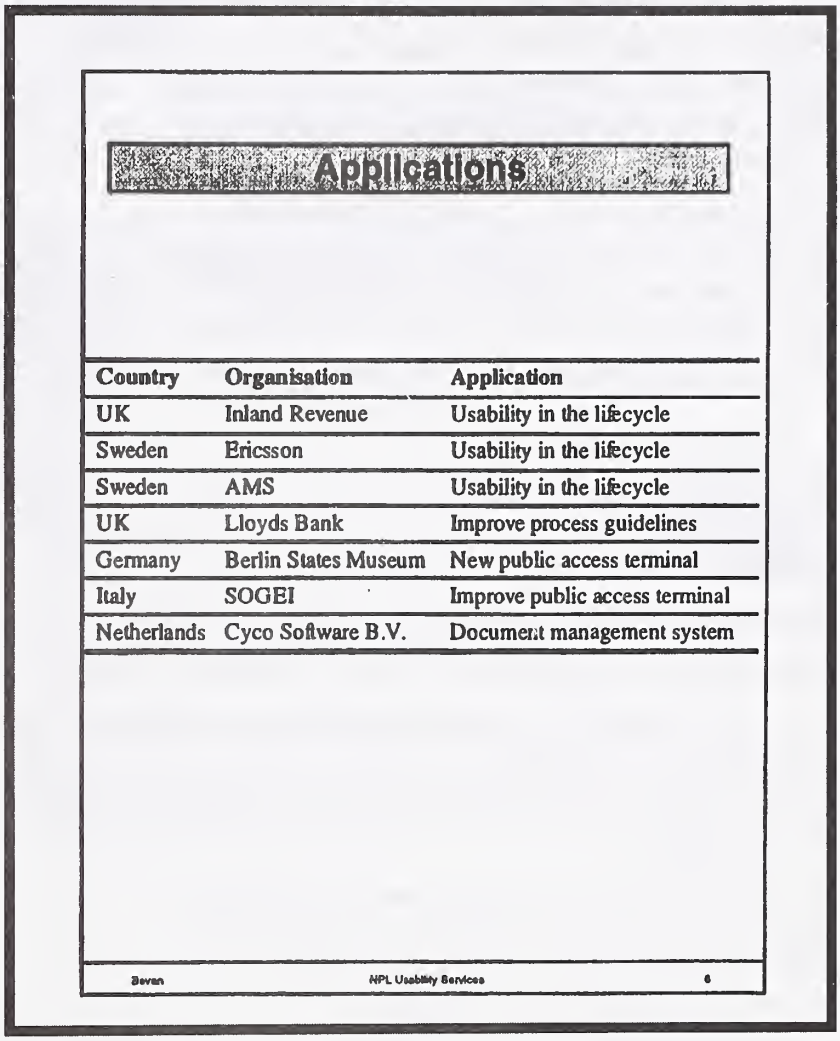


Figure 5

Female: How much of the interest in this in Europe comes out of the general more labor orientation?

Bevan: Scandinavia puts a great emphasis on participation of workers in the design of new systems. So they go further in seeing satisfaction with the use of a system as an essential prerequisite, while in most of Europe satisfaction would be more important as a potential indicator of a problem which might lead to lower productivity. Overall ISO 9241-11 is an economic view of usability which is very consistent with business objectives. If you go to the South of Europe you lose sight of usability altogether. It's a mute topic in the South, though we have got some small usability consultancies from the South of Europe involved in our INUSE

project. In general I think the U.S. has been a bit ahead of Europe, except in the area of quantification and proceduralisation of usability, which fits in with the more formal approaches in Europe.

It seems that there is more emphasis on user centered design in North America. Another standard I'm involved with is ISO 13407 (Human-centred design processes for interactive systems) - this relates the concepts of usability and user center design. Usability at one end is about testing products once they have been built, whereas user center design is about building in usability throughout the design process. Its also a matter of maturity. Organisations start with usability, and as they acknowledge the problems, they realise they have to build in user-centred techniques earlier in design.

An area which is important in Europe is an emphasis on quality, particular probably in the UK, where ISO 9000 registration is very important. I think it's difficult to do business in the UK without ISO 9000 registration. One thing we contributed in the MUSiC project is how to integrate usability with a quality system. MUSiC allows usability to be specified and measured. It can then be a quality objective in the ISO 9000 quality process, and this is mentioned in ISO 9241-11.

What I will say finally and this is that as far as I'm concerned usability in the product is synonymous with a very high quality objective - that the product works in the real world with real user needs. If you can actually quantify these in an objective fashion then you can introduce them to a quality system and they can be part of the software engineering process.

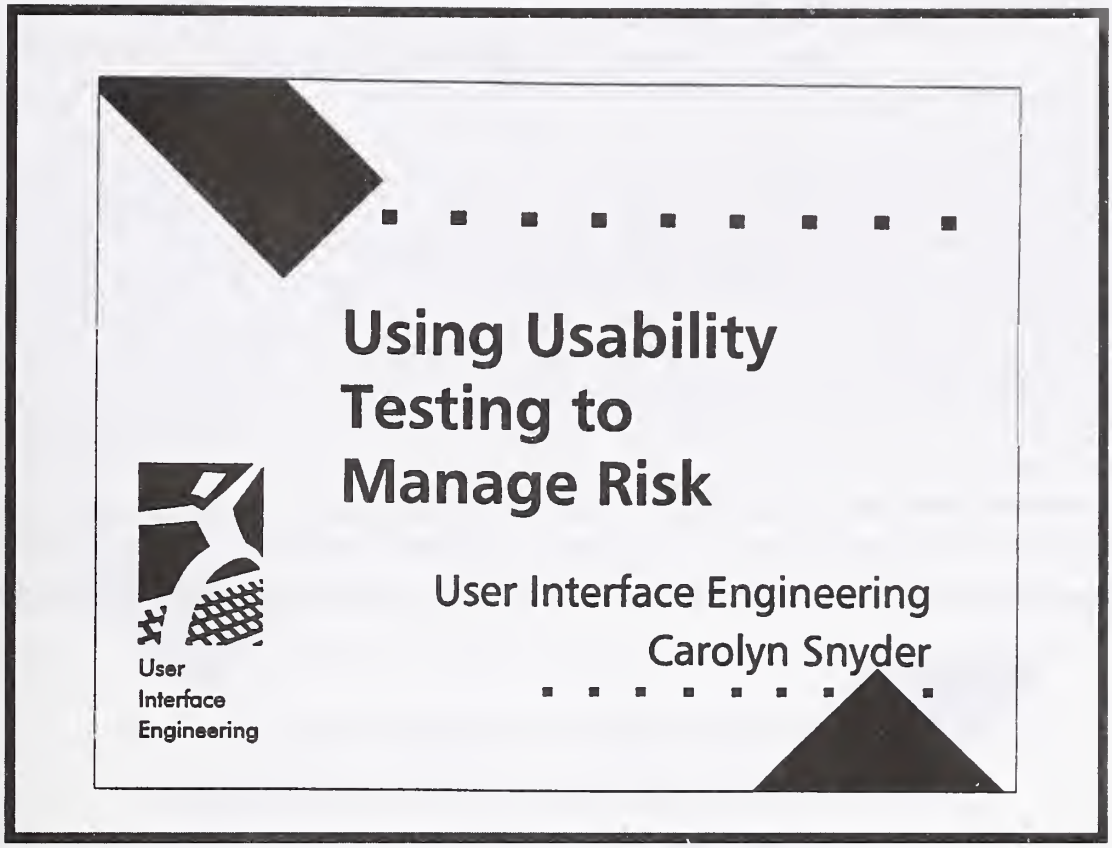
Male: Using that word quality I found software engineers they don't use it in that same sense. When they talk about software quality they talk about a different kind of approach to quality.

Bevan: Absolutely right, which is why quality in use has been defined as a higher level objective in the software quality standards. It means as soon as you buy quality you also buy usability. It seems to me to be a very effective argument for making usability as we understand it the prime objective of design!

Female: Did you say there was a draft of that user center design?

Bevan: That's right the draft of ISO 13407 is being copied.

Downey: It will be on the table. Thank you Nigel.


Using Usability Testing to Manage Risk
User Interface Engineering
Carolyn Snyder
User Interface Engineering

Ms. Carolyn Snyder, User Interface Engineering
Using Usability Testing to Manage Risk

User Interface Engineering is a consulting firm based in the Boston area. We do a lot of work with development teams that want to build more useable products. A good deal of our work is in industry, either on commercial products or internal productivity tools. We also do some work with government agencies and educational institutions. The techniques presented here have been developed and refined with a lot of different types of products, everything from touch screen kiosks in a shopping mall to spread sheets to molecular modeling. We really have had a chance to get a good cross section of development. We also do training and a little bit of independent research in our spare time on topics of UI design that we find to be of interest such as Wizards. This presentation refers not just to commercial software but also to internal productivity tools, so whenever you see the word "product," feel free to substitute "application" or "interface" as

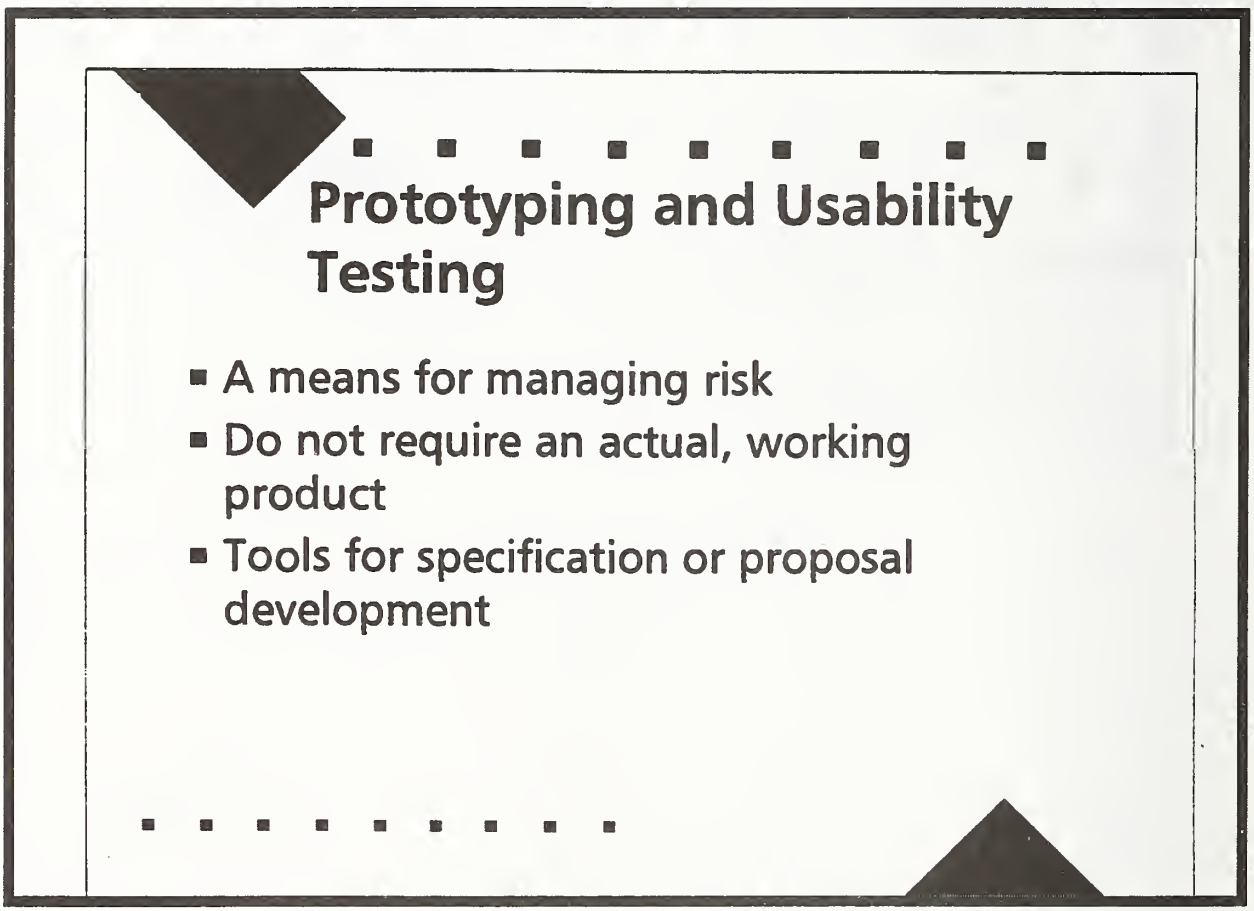


Figure 1

appropriate. Earlier presentations discussed some of the reasons for doing usability testing so this presentation is not going to focus on the why but instead on the how for prototyping and

usability testing. It is a means for managing risk and I think some of the earlier presentations discussed some of the risks in development.

You know a big obvious risk: are we developing the right product? We have worked with design teams that did testing and found out they were developing the completely wrong product. One of my favorite commercials of all time was the Bud commercial where the cave man says, bring me a light and this guy goes scurrying off fur clad savage and brings back fire and says here it is, here's your light, and he says, "That's very nice, but I wanted a Bud Light," and that kind of thing happens all the time in product development. With judicious use of prototyping and usability testing (Figure 1) you can discover: are you building the right thing? Are users going to be able to use it? Can you deploy this without incurring higher training costs or support costs? etc. This discussion is about about how you can address these questions without having to build a product first.

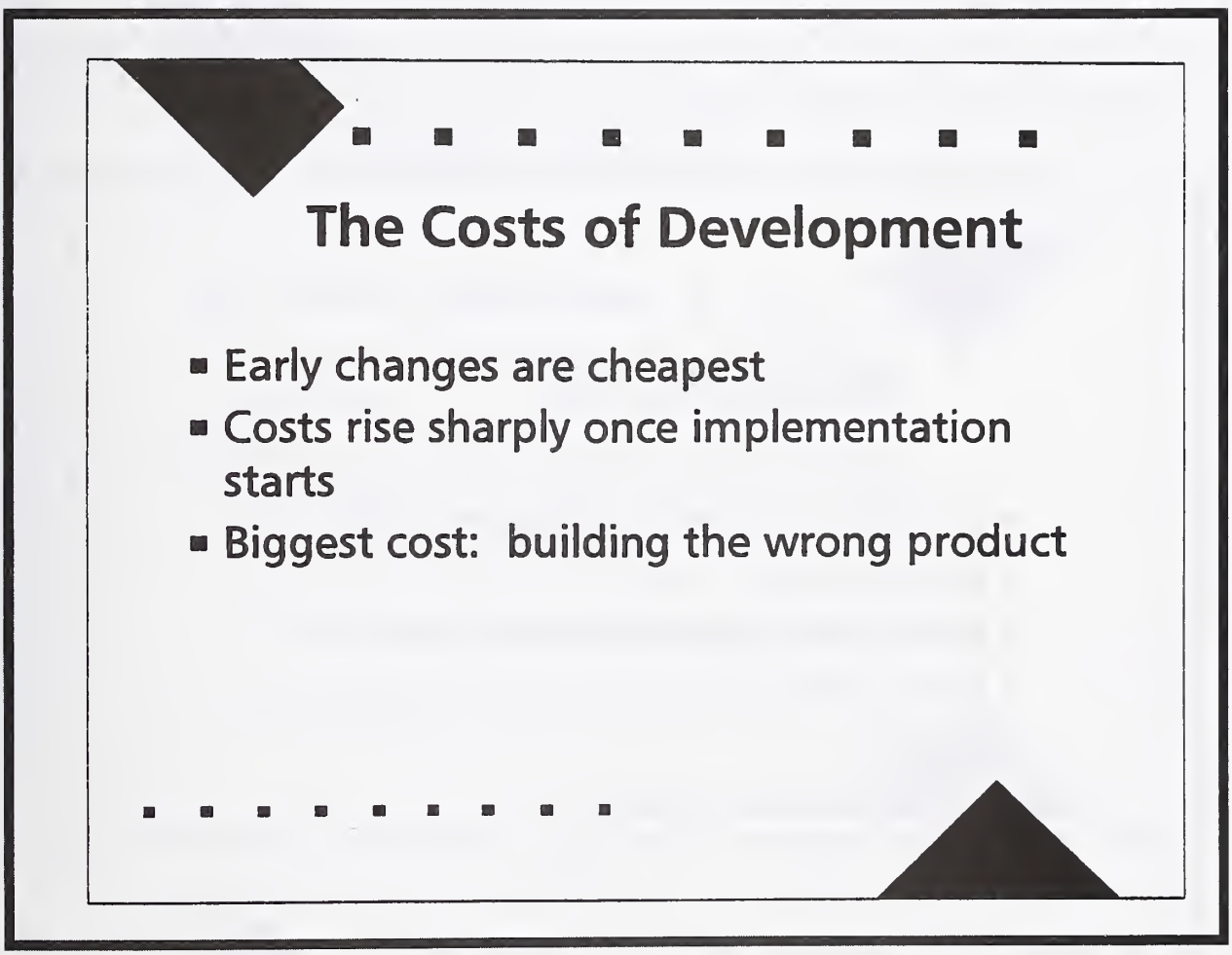


Figure 2

Here is an illustration about development costs (shows slide with increasing slope); our universal trend chart which we use for everything. This is the percentage of number of Republican Presidential candidates. Here's the average annual snowfall in Washington, DC.

Here's development costs over time. Think of this graph as representing a product life cycle where at the left the product is a gleam in someone's eye and on the right is the end of a product life cycle, which is the point where the last user died. Sooner or later users die do die, and then you really get to start over again, we hope not because of anything your product does. The questions is, "At what point does this curve really start to shoot upward?" The horizontal axis is time, the vertical axis is cost associated with making a change to a product.

At the first customer shift or first release of a product the cost is escalating. (Figure 2) Even farther back, where do the costs really start to escalate? Even before integration testing? At requirements stage costs are already escalating.

In all those areas, there are costs associated with making changes, but it's when implementation starts that those costs really start to go up. Once you've written code or started documentation of something, that code or that documentation takes on a life of its own and it becomes much harder to change things, once you've committed actual development resources.

So, what you want is a technique that will allow you to explore changes to the design back earlier in the specification stage before changes affect a lot of work that has already been done. The strategy is to do this as an iterative process (Figure 3) in the area of greatest risk.

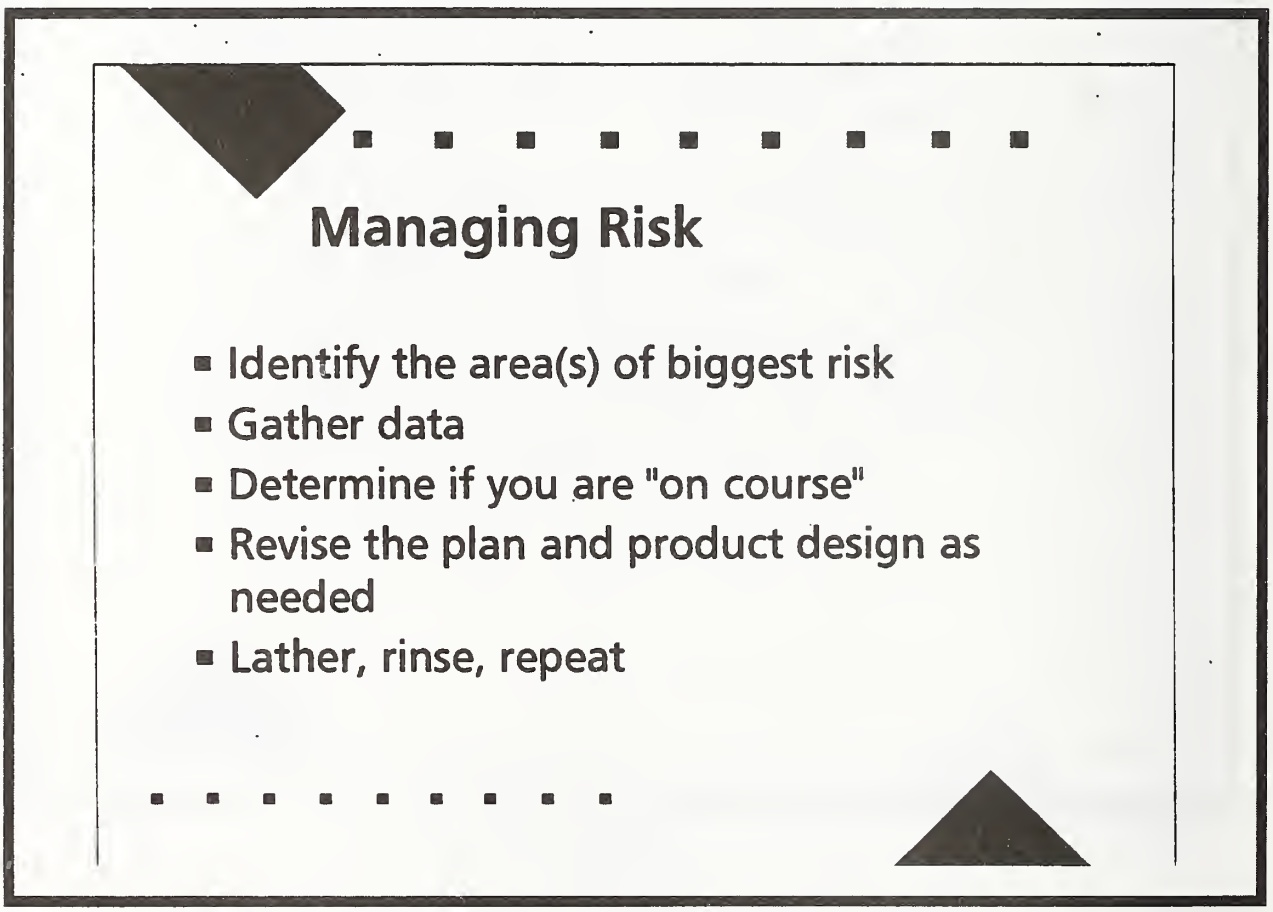


Figure 3

You know for example for a legacy system, where you're taking maybe a character-based system that was developed in the sixties with cobol programmers that have been maintaining it for 30 years, and you are moving to a graphical user interface such as Windows. You have people that have been using the old system for a long time so one of the big risks is that users are going to be able to have the same level of productivity or greater without increasing training costs. As consultants, when we work with teams, we help them figure out what the areas of risk are. Then we can go out and test it, build something, get some data, get some measurements, and based on what is learned, decide what changes are needed. Then, revise the plan and continue that process until time runs out. The more iterations you can go through the more of the risk of a product can be managed.

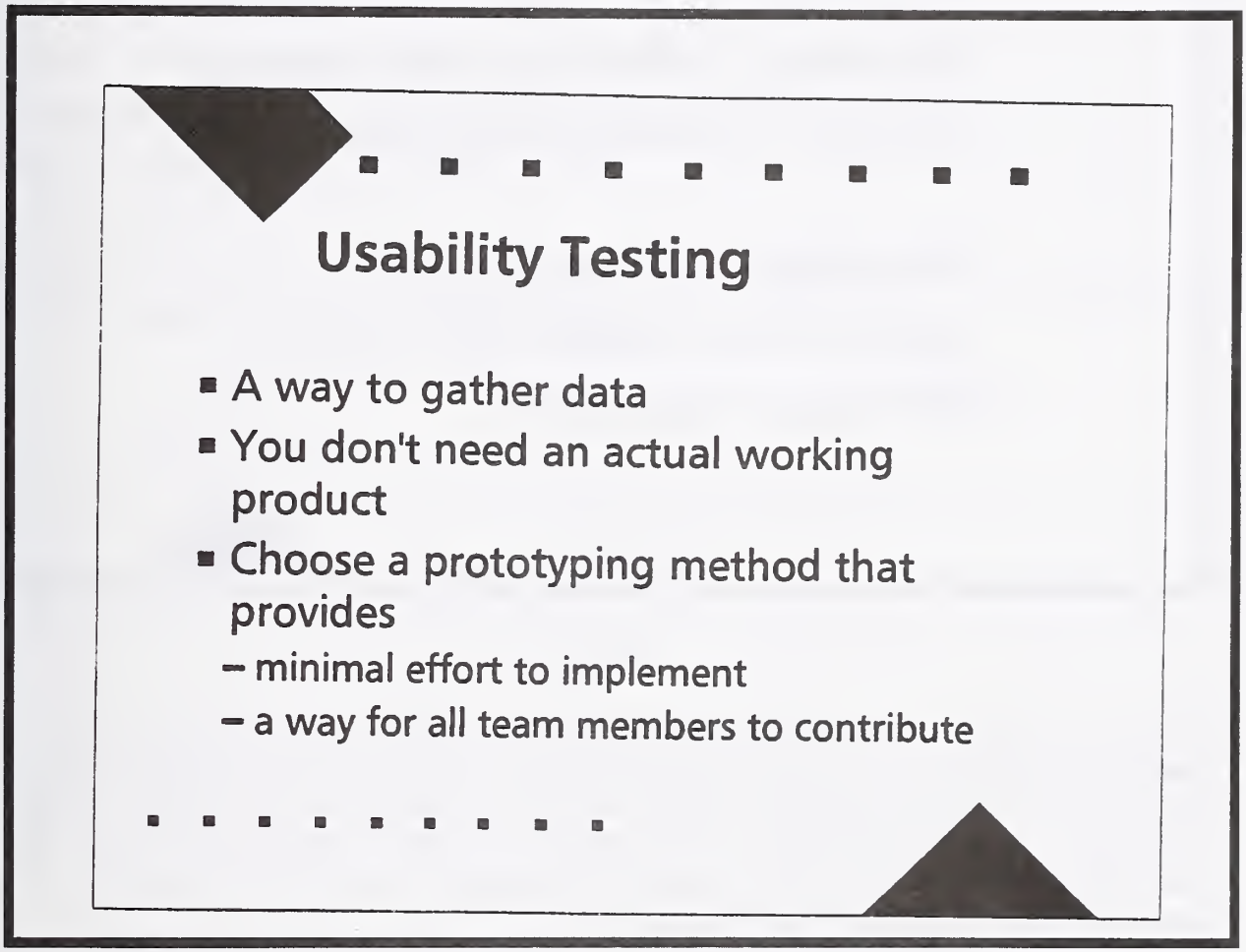


Figure 4

The purpose of prototyping is to get the maximum feedback with the minimum amount of effort so whatever the kind of prototyping that is chosen is looked at accordingly. Our finances suggest that you actually do not need to develop something that runs in order to get a lot of good data about the usability of the end system. (Figure 4) So, you need to look at the amount of effort to implement a prototype. A lot of prototyping methods that are software–based, require a programmer or some programming skills to use the tools. That can become a roadblock to ideas

making it into the product. A way for people to contribute that takes a minimum amount of effort is often a paper prototype.

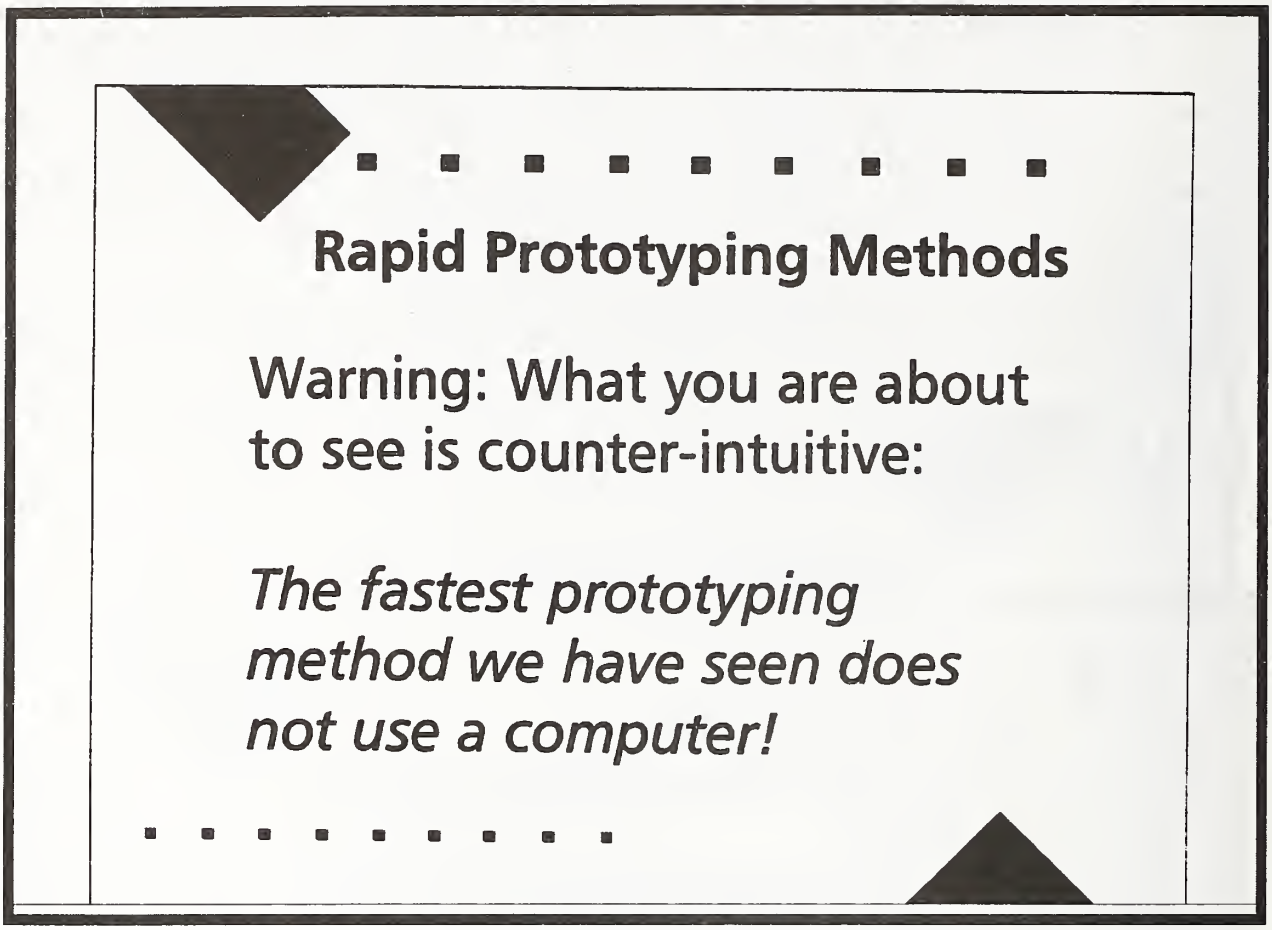


Figure 5

Several earlier presentations referred to paper prototyping. (Figure 5) It is a method that is gaining more and more acceptance among practitioners. Instead of making a running version of a product is you literally put all of the screens on paper. For usability tests of a paper prototype, we usually test with the two users at a time, you get twice as many data points for tests that way. Users work on the tasks that we have assigned them using the paper prototype. A typical dialogue from a test session (short video shown):

Male: This is so leading edge that we're going to be working apparently with a system that we haven't had any time to write any code for yet, so the entire prototype is going to be in with a paper cut type. Trent's been through this once and he survived. The guy to my left here is going to play the computer and he's going to make TSO seem fast.

Male: We're running windows, here we got something like a 186. We will step through it. What we're looking for here is we're trying to figure out given these ideas we came up with a rough interpretation of how we get the functionality. Instead of investing in a lot of code, shifting and then finding out, no one can figure it out, we're trying to find that out early, figure

out why people can't find out so what we're looking for are places where the product seems to work and places where it doesn't seem to work and we clicked on associated life cycle.

Female: Could I ask a question?

Male: What would you do next given this screen?

Male 1: Selected stage ----.

Male: Selected stage ----.

Male 1: I would click on entry first, I would highlight entry.

Male: That was an important message by the way. It was like OK. It was stored on a disk far away. (People talking, not audible)

Male 1: I would try to take V.

Male: (Inaudible)

Male 1: And then this stage.

Male: Right.

Male 1: OK, then I would come down here and I would select test and I would say new system. So here we go. (Inaudible) —— learning 186.

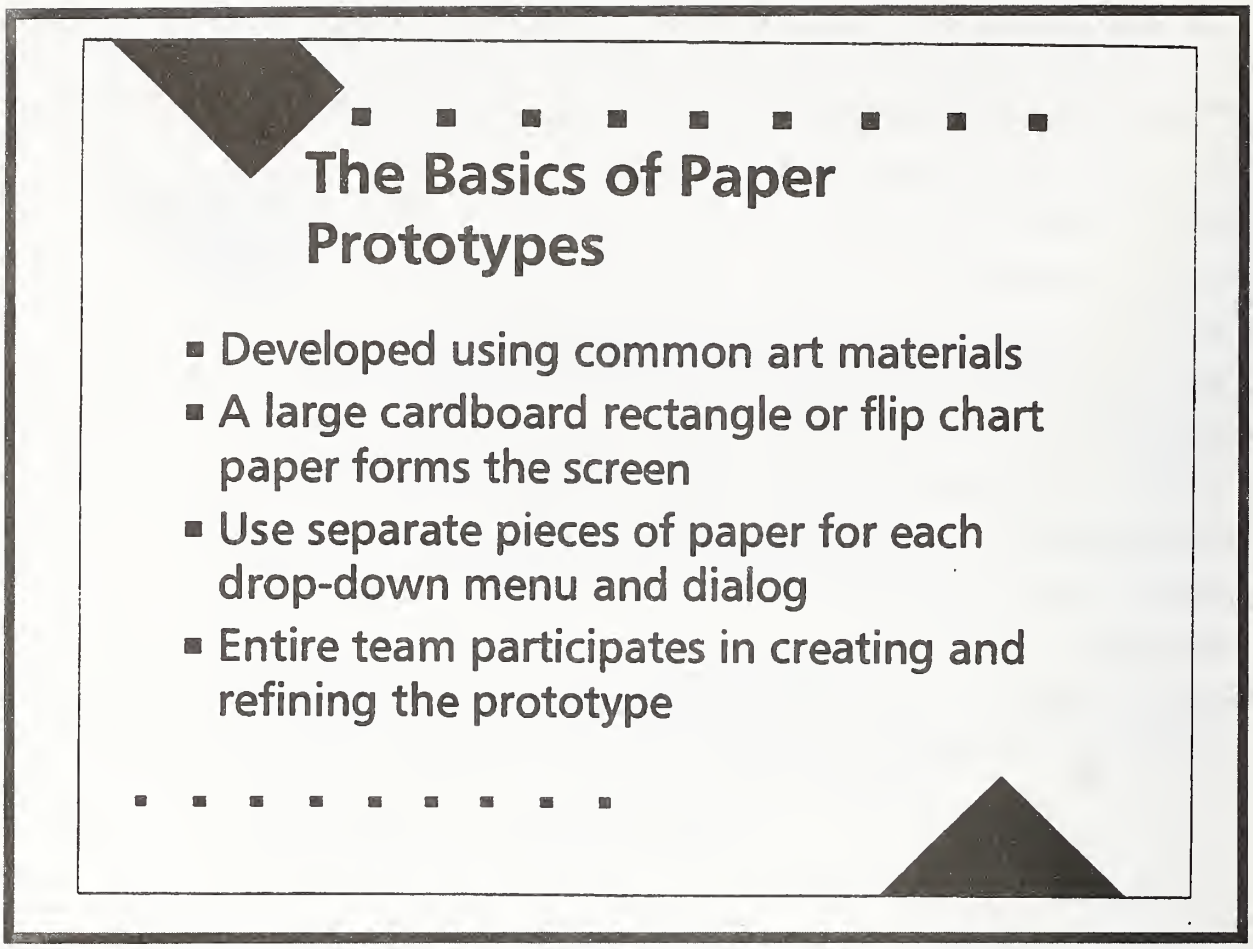


Figure 6

Male: So you would create a system mainframe.

Male 1: Right, so I would say what's this project that you're working on.

Male: TCS how's that? (Inaudible)

Male 1: We have a better TCS system, the customer says that.

Male: How's that?

Male 1: We've got a system in here. At this point I would hit help and see what he's talking about, who needs — — is going to be able to — — at the system level.

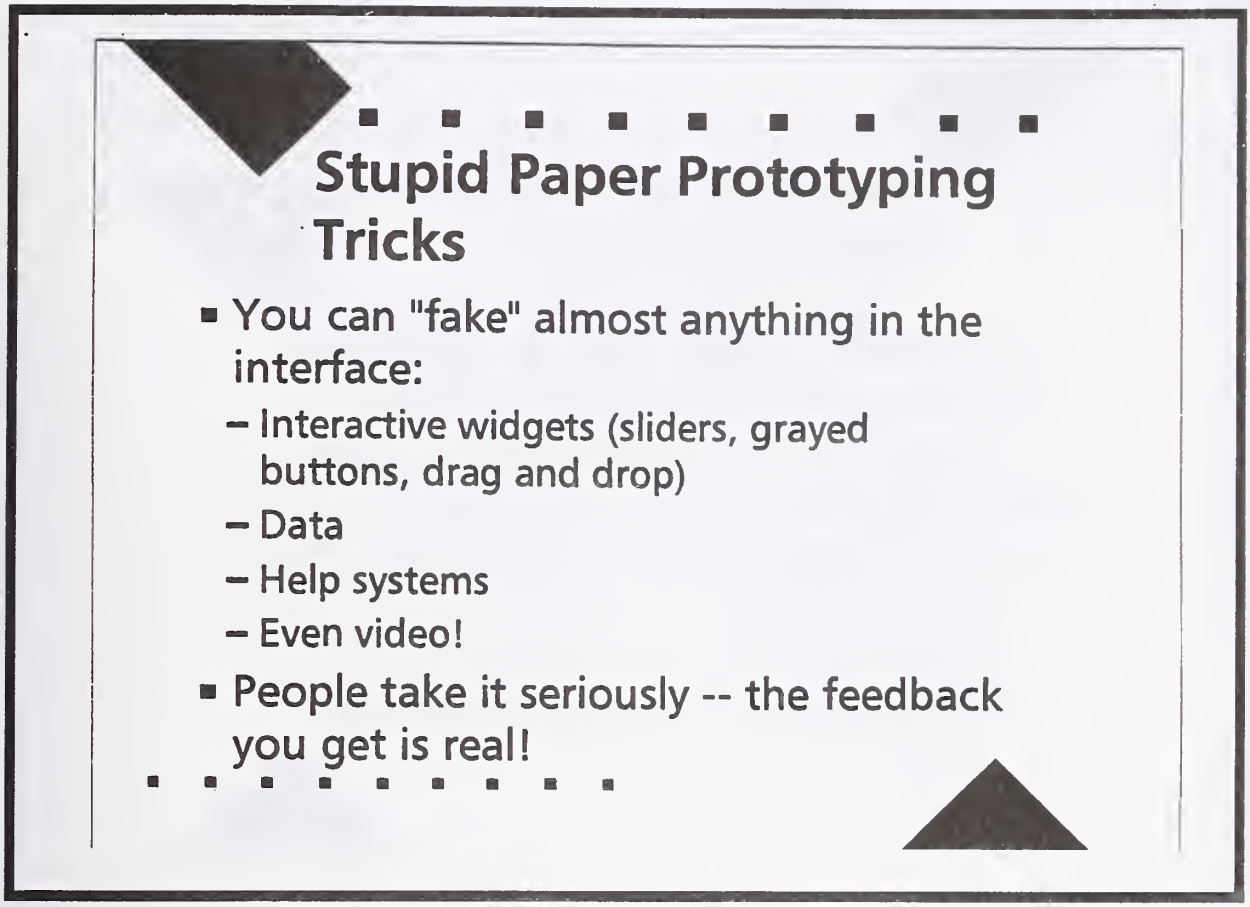


Figure 7

The product that was being developed, this tape was made at about 3:30 on a Wednesday afternoon and the product did not exist Monday morning. We got the team together, came up with a preliminary design and were testing it and getting user feedback in just a couple of days. Another thing to note. They were laughing about, "oh, it's a learning 186; it's adaptive technology." That came up because one of the usability issues that they found during the test, the developer who was right there in the room grabbed a pen and literally changed that particular screen and problem solved right then and there.

As you can see, we make paper prototypes (Figure 6) out of art materials—for example, shirt cardboards taped together. A paper prototype for a spreadsheet that we've used for some development has input on separate pieces of paper, in this case this is the data contents of the spread sheet, so that once the user opens the spreadsheet you would just set that data down and a person plays the computer not explaining the interface, but behaving only as the machine would. The entire team can participate in creating an refining a design, not just programmers, but a technical writer, marketing, management, users themselves can all contribute to the design.

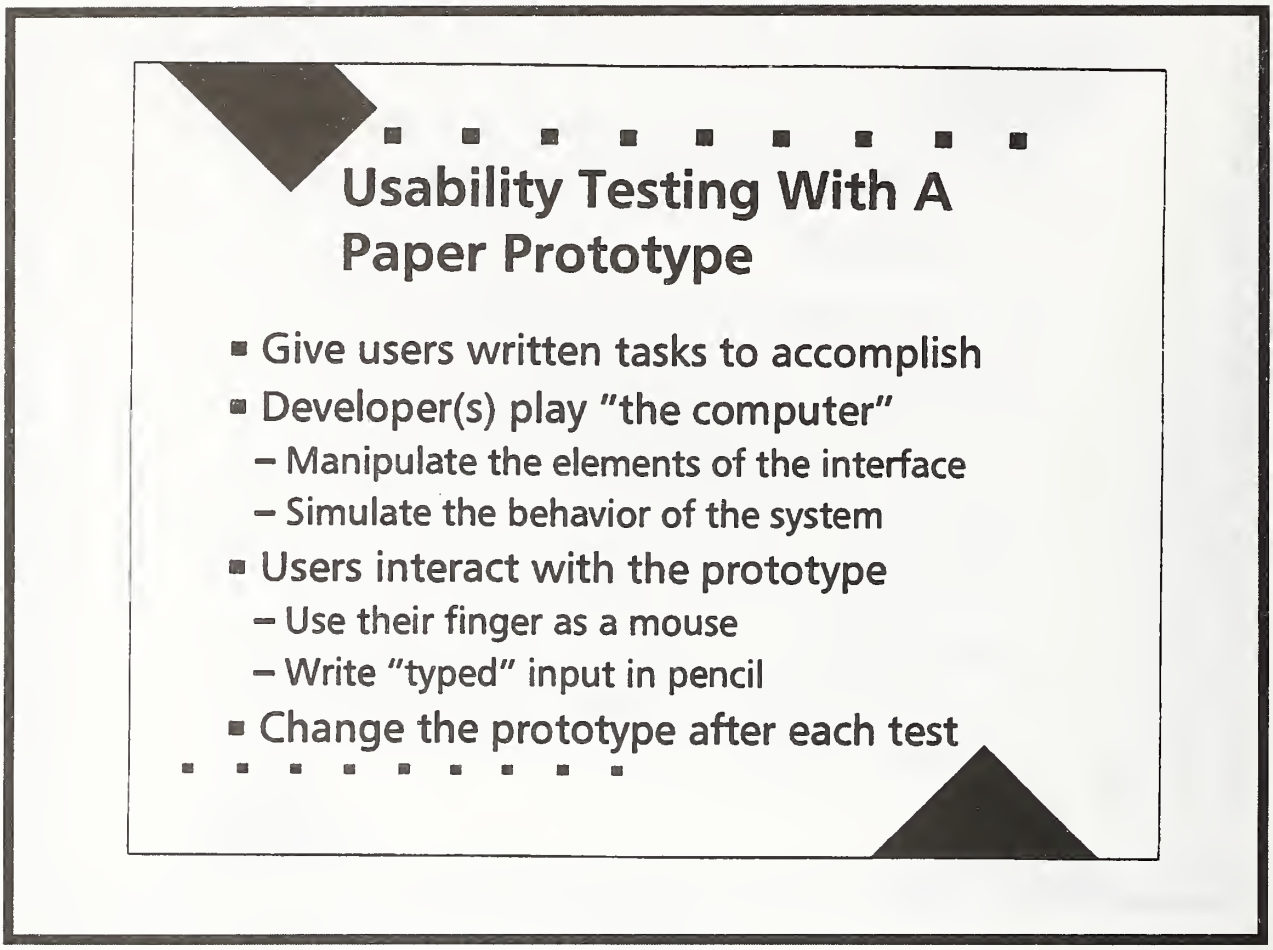


Figure 8

With paper prototyping and a little imagination you can simulate anything (Figure 7); we have done video, with Polaroid photographs, we have prototyped animated tutorials, with props, physical pieces of hardware that we've incorporated into the prototype. Here are some samples of the different ways we simulate some graphical user interface widgets. This is for a system for printed circuit board test equipment. The user is clicking on a button. A rectangle is on a piece of transparency. We use that for highlighting. The dialogue that comes up and the user clicks on something, it becomes highlighted. The buttons were gray until the user clicked on something and now he's putting down dark versions of the buttons. Double clicks, the changing captions say "running," because he's just launched a process. A dialogue box that has a radial button and the dark circle there is on a piece of removal tape so the users can click on a different option. The user clicks the OK button. A dialogue that has a scrolling list of options in it. The user can pick one of them. In this case, the two lines in that dialogue box were file names that the users themselves had entered in earlier so here they are seeing their own data actually reflected in the interface. Even high level managers will take this seriously; they are drawn in and engaged and do provide realistic feedback. In that first example, a user had a question about whether version control was supported and this was something that the team wasn't even attempting to prototype at this point and what they found out from that test was that there was something that needed to

be in the requirements that was important to users. Again, this is a way to manage risk, to find problems in the early stages.

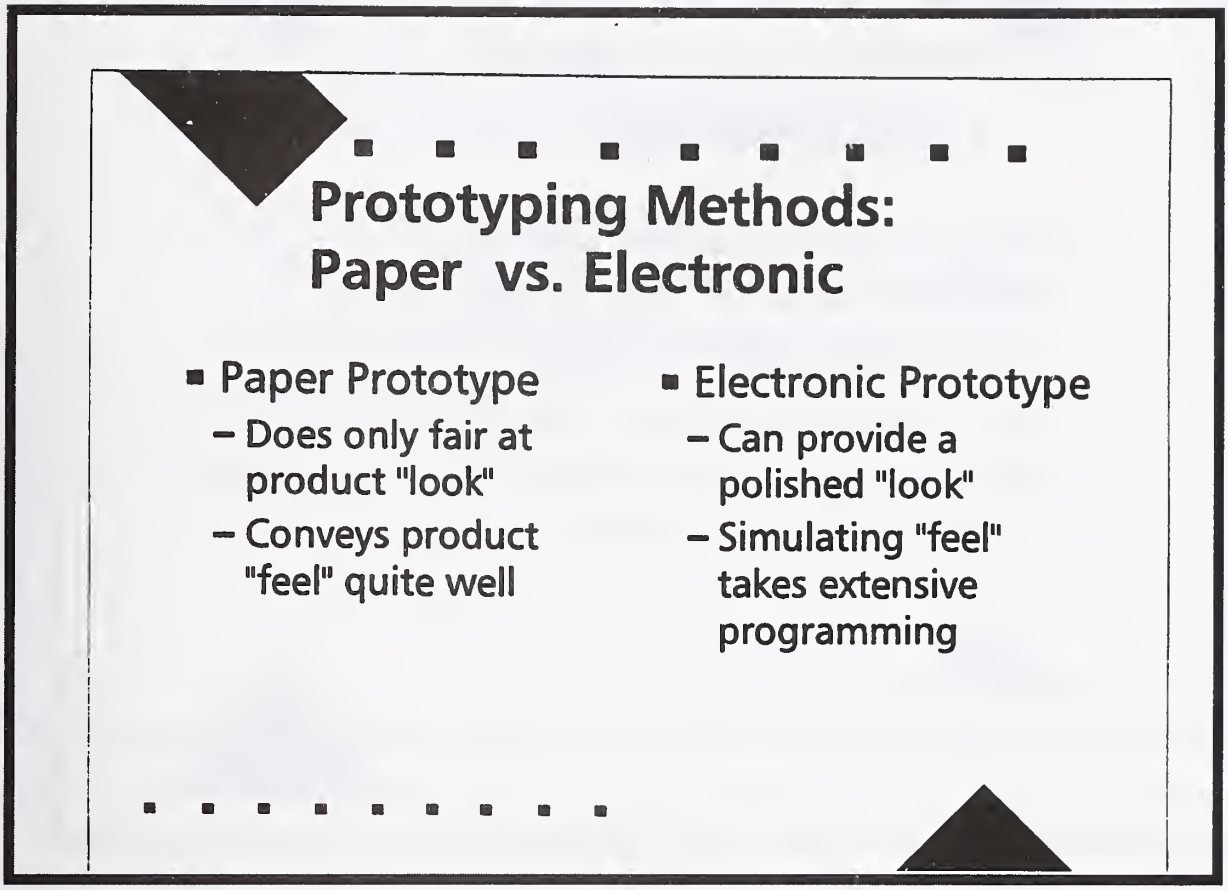


Figure 9

The prototype changes after every test. (Figure 8) Sometimes even during a test in response to what you learned and teams have been able to make substantial changes in just a few iterations with a paper prototype. We've had cases where we come in, do the first usability test and find out that the system is a complete failure. Users cannot use it or hate it or they cannot get their work done and the team will be able to make some changes from one test to the next that make the difference between failure and success. We have seen teams completely redesign their product literally from 4 o'clock in the afternoon until 10 o'clock the next morning. They've made enough changes that they are able to fix a lot of the problems. A lot of times they are small things. Changing a word on a button can make a big difference or just realizing these two screens always come up one right after the other. Why don't we put them together. You can make those sort of changes very quickly in a paper prototype.

The common question is, "Aren't there already prototyping tools on the market that let you do this kind of thing, for example Visual Basic?" Someone that knows how to program might say I can do this same sort of thing in Visual Basic, what's the difference?

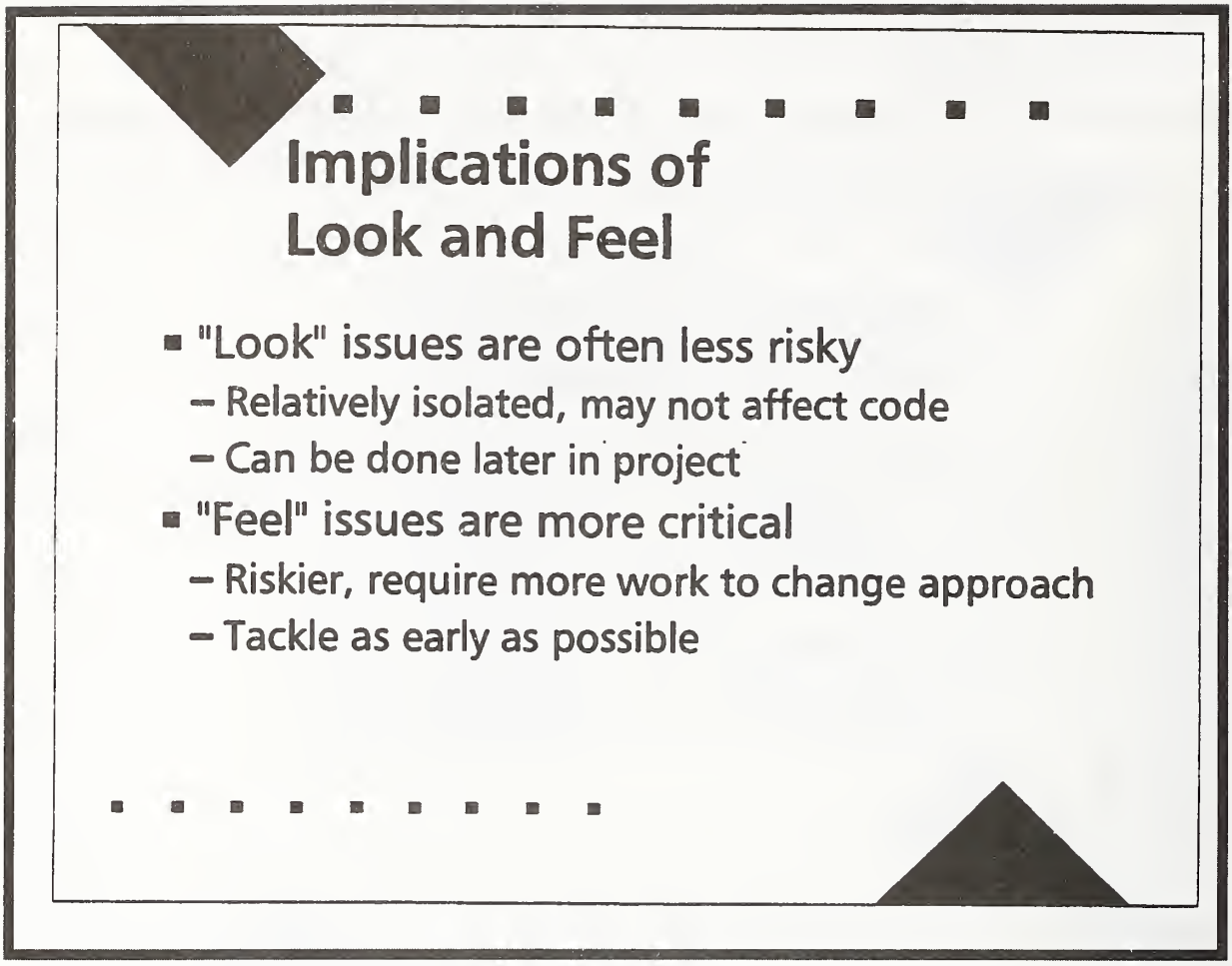


Figure 10

If you think of an interface in terms of look and feel, the visuals of a product, for example, do all the buttons line up, the colors that you're using, that sort of thing, and the feel matching the behavior, does it match the workflow, is it responding appropriately to errors, or even better is it preventing errors? With a paper prototype, the look is crude, hand drawn. We don't use nice straight lines and all that sort of thing and you don't need a lot of artistic ability. Paper prototyping conveys the product feel and the behavior very effectively so users can see what actually going to happen. In electronic prototypes you can have something that looks very nice. You can have 3-D buttons, etc. but in order to simulate the feel can require extensive programming. An earlier presentation listed test-related development costs, close to $35,000. With a paper prototype you avoid a lot of that cost, you don't have the programming that's necessary to really make one of those electronic prototypes do the right thing.

In terms of look and feel (Figure 10), the feel aspect of a product is typically where the risk resides. We have yet to see a product that failed because it had bad icons, for example, where the bitmaps were off. A bigger issue is whether you need toolbar icons at all and what sort of functions should they include. And the look related issues are easier to change a bit later in the project life cycle without incurring lots of rework or development costs. Therefore, you want to tackle the feel issues up front and the paper prototyping lets you do that very effectively.

Figure 11

Paper prototypes are not just for usability testing. (Figure 11) Once you've had a chance to develop a prototype and refine it with some tests, and we typically do this with about four tests and we get users for each test so there are eight data points, we have found the biggest problems, and we also have the paper prototype and design. The prototype itself can be a very effective communication tool for other people working on a project or as input for a specification.

You don't necessarily want to put this in a RFP, but in a requirements spec or an interface spec. A paper prototype gives the designer a chance to experiment and define the key elements. The paper prototype can also be used to show how the system is supposed to work.

It is an effective way to communicate how a design is supposed to look. For example, we have videotaped walkthroughs of prototypes and sent them to development teams that were in remote locations. Sometimes you have one group that's doing the specifying here and somebody else on the other coast who's actually doing the development and this was one medium for communicating, especially people who could not attend the test.

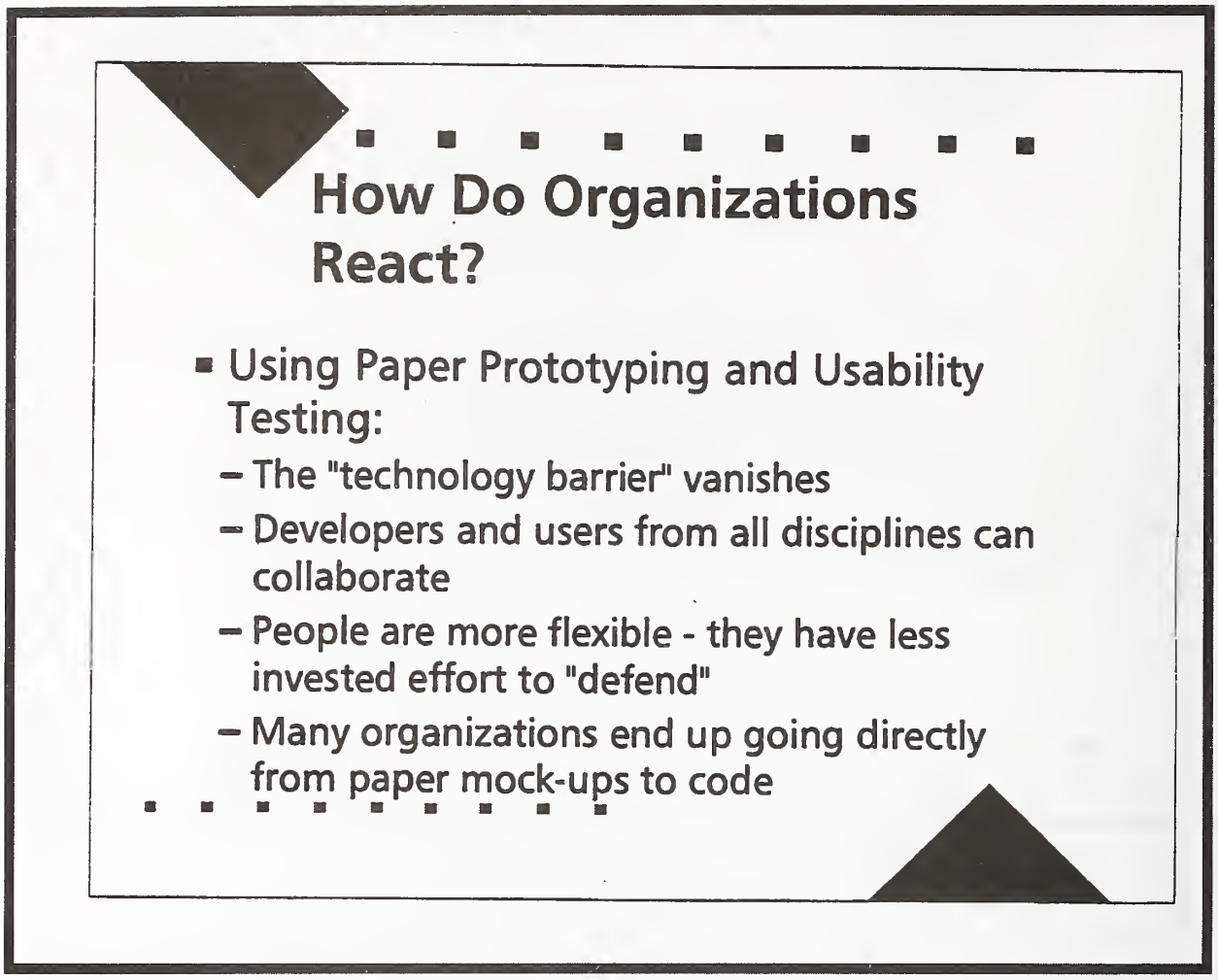


Figure 12

A common question is what happens when organizations adopt paper prototyping. (Figure 12) Surprisingly, the technology barrier that existed previously between the software engineers and other members of the team vanishes. You have a multi-disciplinary team that can now effectively collaborate on the design of an interface. (Figure 13) Another benefit is the flexibility to change something. It is much easier for people to just say, "OK, we'll change after 10 minutes of work," as opposed where they have spent weeks or months working on a prototype.

Discussion: When can you start using paper prototyping?

Paper prototyping is effective often when you're at the state where you're trying to develop a spec or if you have an existing spec and you want a quick early mock-up. Once you have an idea of some requirements is a good time to start doing paper. We expected when we first started using this is that people would do some iterations on paper, get the big bugs out, then go to an electronic prototype and get some more of the bugs out and then finally start coding. What we found was you can find enough of a high risk things. So, a lot of people go directly from a paper prototype.

How often do you use paper prototypes?

This is the bread and butter of our consulting work. We do a lot of usability testing and probably half of our projects involve paper. I have been doing this for about three years and my colleague Jared Spool has been doing this sort of thing for many years. We do this a lot in a variety of different disciplines. It is hard to gather statistics, because there is essentially no control. We're not comparing with and without paper. With paper, we are continuously making changes but because we have a way to measure, we have a way to tell if we're improving a design.

What does the development team do before starting the paper prototyping?

The development team defines usability requirements such as what the users must be able to accomplish these types of tasks in what amount of time and then we measure against those with each design.

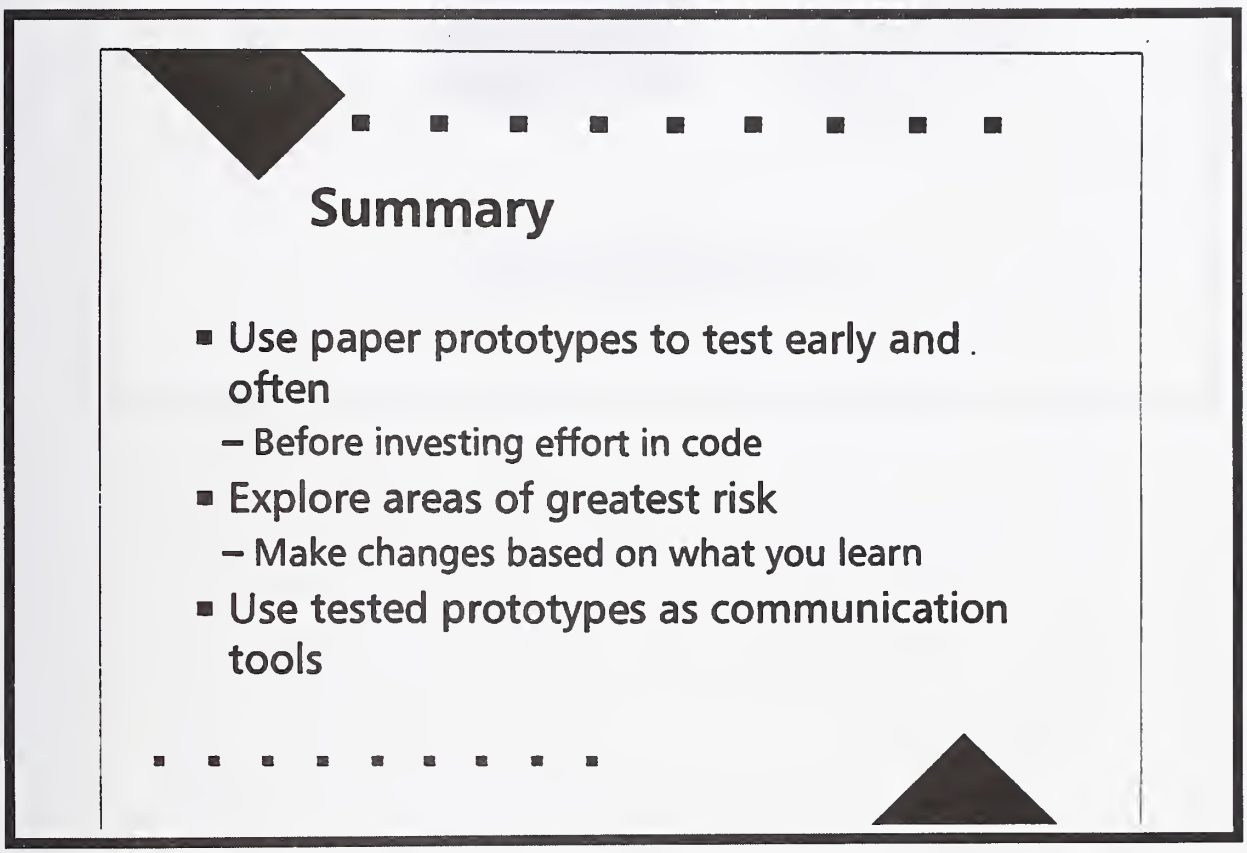
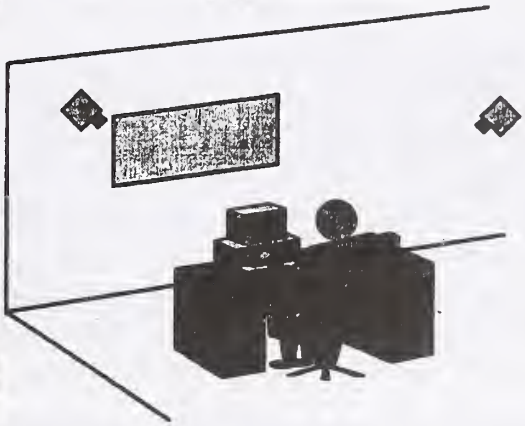


Figure 13

Conduct Usability Testing!



A Major Finding from a Study of Commercial Best Practices

Dr. Pascal J. Gambardella
Computer Sciences Corporation
3170 Fairview Park Drive
Falls Church, Virginia 22042
703-641-2248
pgambard@csc.com

SYMPOSIUM
Usability Engineering: Industry-Government
Collaboration for System Effectiveness and Efficiency
National Institute of Standards and Technology, Gaithersburg, Maryland
January 10, 1996

Dr. Pascal J. Gambardella, Computer Sciences Corporation
Conduct Usability Testing! A Major Finding from a Study of Commercial Best Practices

We have a customer who has been developing a system for six years. It's not quite out yet, it should be out soon. It would have been out sooner except when the user saw it they decided it was not good enough, did not satisfy their needs and was not as good as the existing system which it's going to replace completely. And about six years ago for about a year, they did talk to users. Within this system, I worked as a developer, systems engineer, performance engineer, transitioner of the data and one of my last roles was systems integrator on the new system trying to relate the new system to the old system. In all that time as an engineer/programmer, I never really talked to users. Only when I got to the point of being systems integrator did I actually talk to representatives of users, not quite the user yet but the representatives.

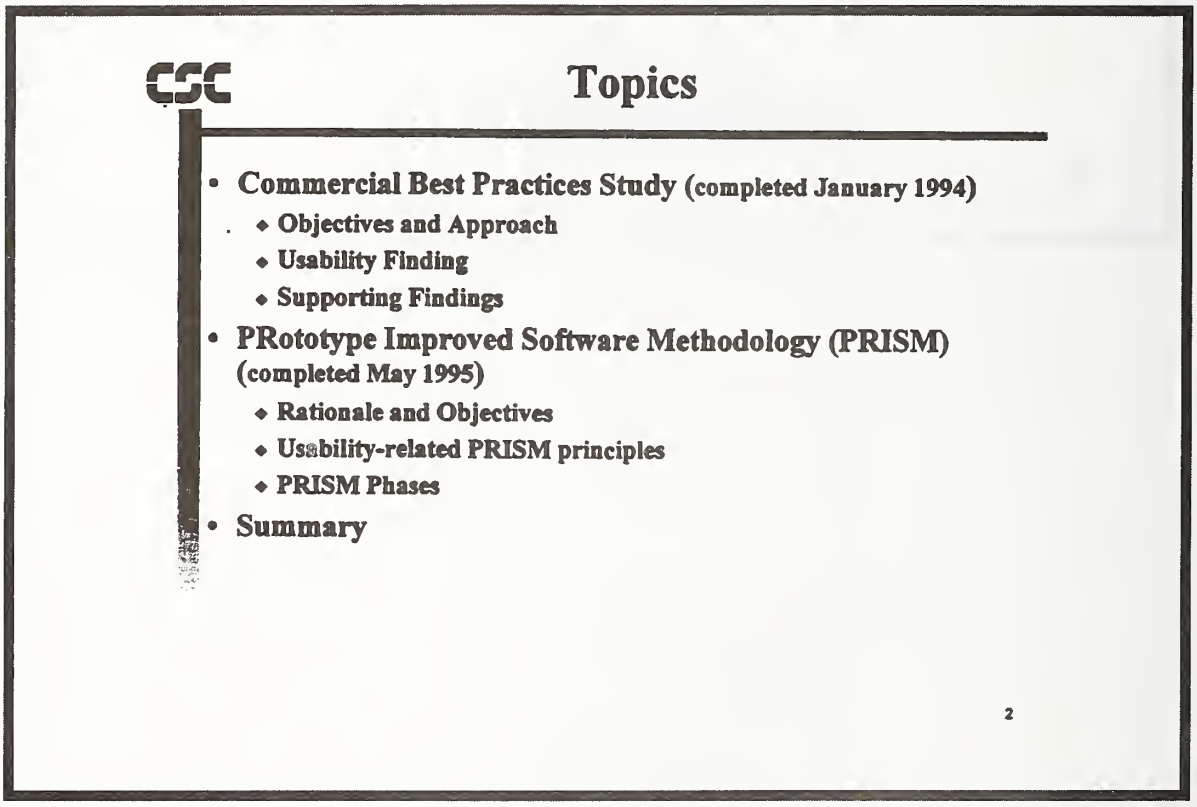


Figure 1

Now in this project, they realized they had a problem. They decided that when the system finally went on-line they wanted a different way of doing business. They wanted a different

methodology. They sent us to find out how the commercial world develops software. On a four month tour of the country which led us to California twice, to Texas, to Boston, we did just that.

The first part of this presentation is a discussion of the commercial best practice study we did and its findings. Our intention wasn't to find out about usability. But that's what we discovered after the process. I will discuss the findings we had about usability and also about supporting findings that are important in context. This was in the fall of 1993 and we finished our study in January, got it unclassified by April and distributed. After that, the government spent time with the TIGER team. We then put together a development methodology based on these best practices, delivered in May 1995. I will point out where usability engineering comes in.

The government wanted us to discover how these commercial companies determined customer needs, how they are capturing document requirements, how they validate requirements, how they develop software. They were concerned about upgrades and how to select among the various different methodologies to develop software. Other issues are how to tailor methodologies to different companies' needs, situations and cultures.

These figures (Figures 2 and 3) describe the objectives of our commercial best practices study. We talked to eight companies, we did interviews, we taped the interviews where we could, and we taped after the interviews. We conducted research before the interviews about each company to understand its culture. We then produced a study and summarized our findings. After our interviews, we sent the companies back a 12 page report on what we thought they said. All but one company gave us feedback, and we went through several iterations to ensure a good understanding of what happened.

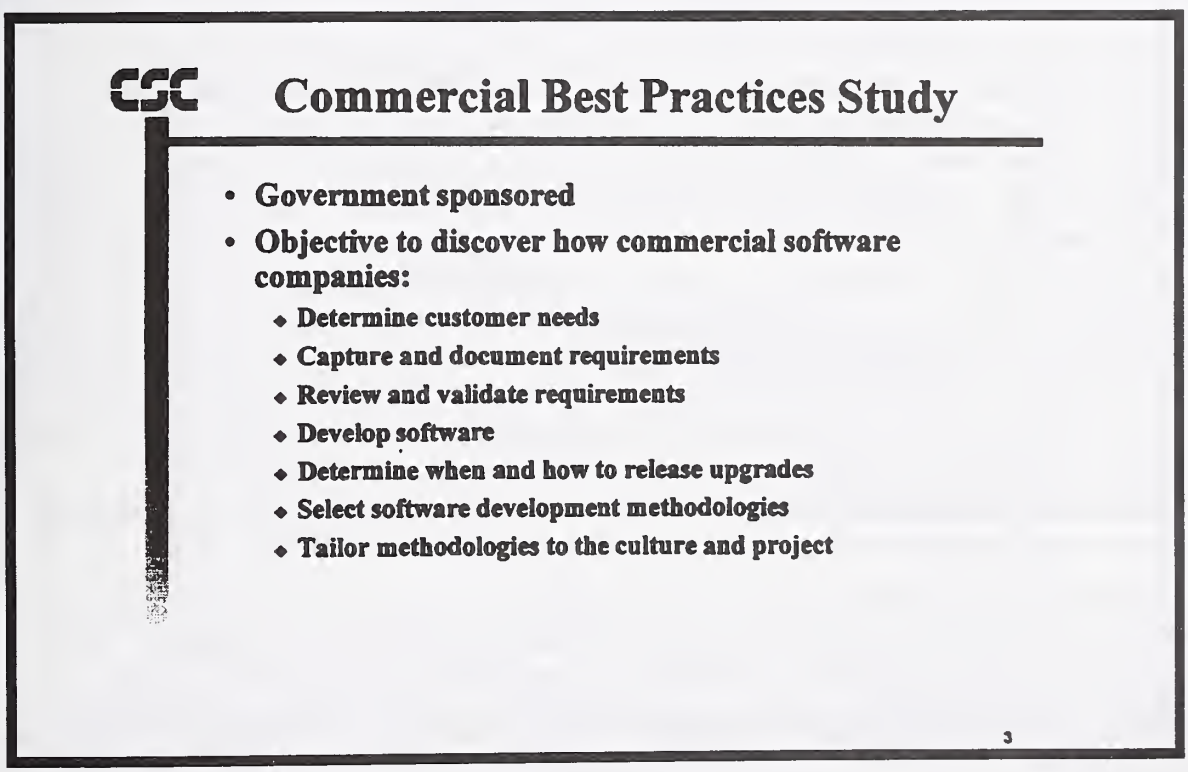


Figure 2

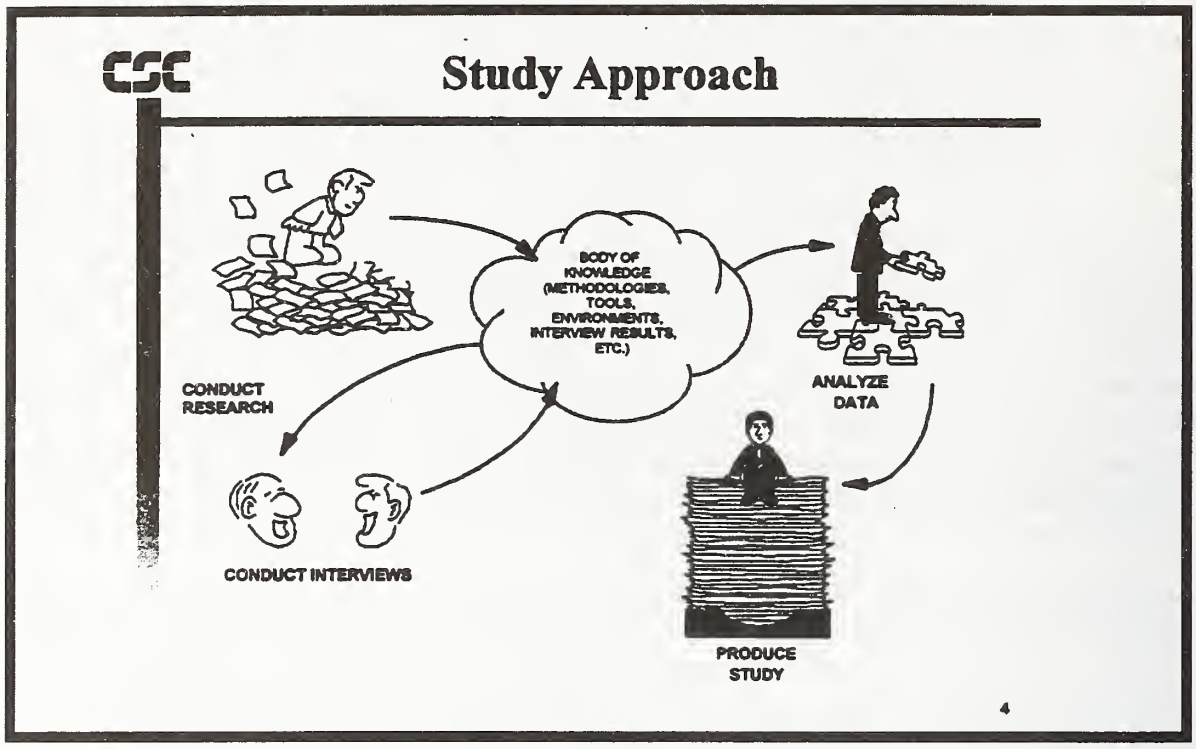


Figure 3

For example, (Figure 4) we went to Borland where we talked to the team that did Quattro Pro. We visited Sun Microsystems. Most of what we learned about them is proprietary. I work for Computer Science Corporation, but I work for the government part, and we had a commercial arm that does commercial work, and we visited that and neither of us knew much about it, developing software for the commercial world so it was as if we were going to a different company. We also went to Fidelity Investments, that part of Fidelity Investments that develops in house software. We went to TI where they develop in-house software. We also went to Lotus, Microsoft, and Oracle. The companies were selected on several criteria: who would talk to us in four months; the type of work they do; whether they were developing software internally or commercially; and what types of applications they deal with. We wanted to get a wide variety of applications. Note that this was in the fall of 1993, and many of these companies are parts of other companies so part of what we learned then is probably no longer completely true today. At that time, companies like Oracle had not really worked much with usability, but they were trying to establish a usability lab. We also visited labs at Lotus, Fidelity, Maryland National Information Technology Center. At UP Module Technology, we talked about user partnering modules, a methodology for putting information on a computer to test user reactions.

We had several results presented in this figure (Figure 6) and, after we looked at the data, we looked at the results in terms of the government's categories, gathering and validating requirements, and doing upgrades. We found 10 related findings which seemed to occur frequently. One of the findings that we thought significant is that it is important to conduct usability testing not just at the beginning, but throughout the entire cycle. (Figure 7) Later in

Interviews with Software Development Companies

	OFFICE AUTOMATION	MULTI-PLATFORM	INTERNAL SOFTWARE DEVELOPMENT	OPERATING SYSTEM	CONSULTING
BORLAND	✓	✓			
SUN MICRO	✓	✓		✓	
CSC CONS.					✓
FIDELITY			✓		
TI			✓		
LOTUS	✓	✓			
MICROSOFT	✓	✓		✓	✓
ORACLE		✓			✓

5

Figure 4

Lab Visits and Other Interviews

- Visited usability labs at:
  - Lotus Development Co.
  - Fidelity Investments, Inc.
  - Maryland National Information Technology Center
    - — Usability resources now managed by Man-Made Systems Corporation
- UP Module Technology, Inc.
  - Modules gauge users' reactions to software while they are using it
  - The developer embeds dialog boxes, queries, and other means of gathering user input

6

Figure 5

Summary of Study Findings

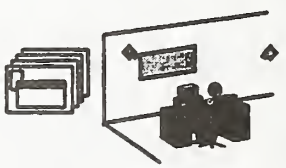
- **Usability Finding**
  - **Conduct Usability Testing Throughout the Life-Cycle**
- ***Supporting Findings***
  - ***User Involvement is Critical***
  - ***Solicit User Input using Multiple Means***
  - ***Use Iterative Development Methods***
  - ***Employ Small, Heterogeneous Development Teams***
- Other Findings
  - Choose a Flexible Methodology
  - Provide a Supportive Management Environment
  - Develop a Clear Vision
- **Must be judiciously employed; they do not guarantee success**

7

Figure 6

Usability Findings

Conduct Usability Testing Throughout the Life-Cycle



- **Providing early access to users helps ensure software better meets the users' needs**
  - **Users input can influence interface and functionality**
  - **Users are best reference point for efficient software operations**
- **Catching errors early reduces the cost to fix the errors**

8

Figure 7

this presentation, we show you how we put that into a methodology for the government. Some of the findings that were expected but some that the government was surprised were based on user involvement being critical to the effort. Here "government" refers to our particular client. Soliciting user input, we discovered, was what the commercial world did.

Another finding which also supports usability is to develop software iteratively, employing heterogeneous teams, composed of not just those developers, but testers and specification writers. Borland had a technical writer in at the very beginning when they were defining requirements so that the wording of the requirement would be clear. If the requirement was too long or unclear, they had to rewrite it before they even started doing development. A flexible methodology was also recognized as important, e.g., one for costs, or one for very rapid application development or one for Oracle development. These findings we subjected to the government were not.

We briefed this study to a lot of different government agencies, especially in the intelligence community and DOD 607. The major finding we had was conduct usability testing throughout the cycle. I'm not going to say much of this, it has been discussed today but we felt and we noticed that it was important that you have early access to users.

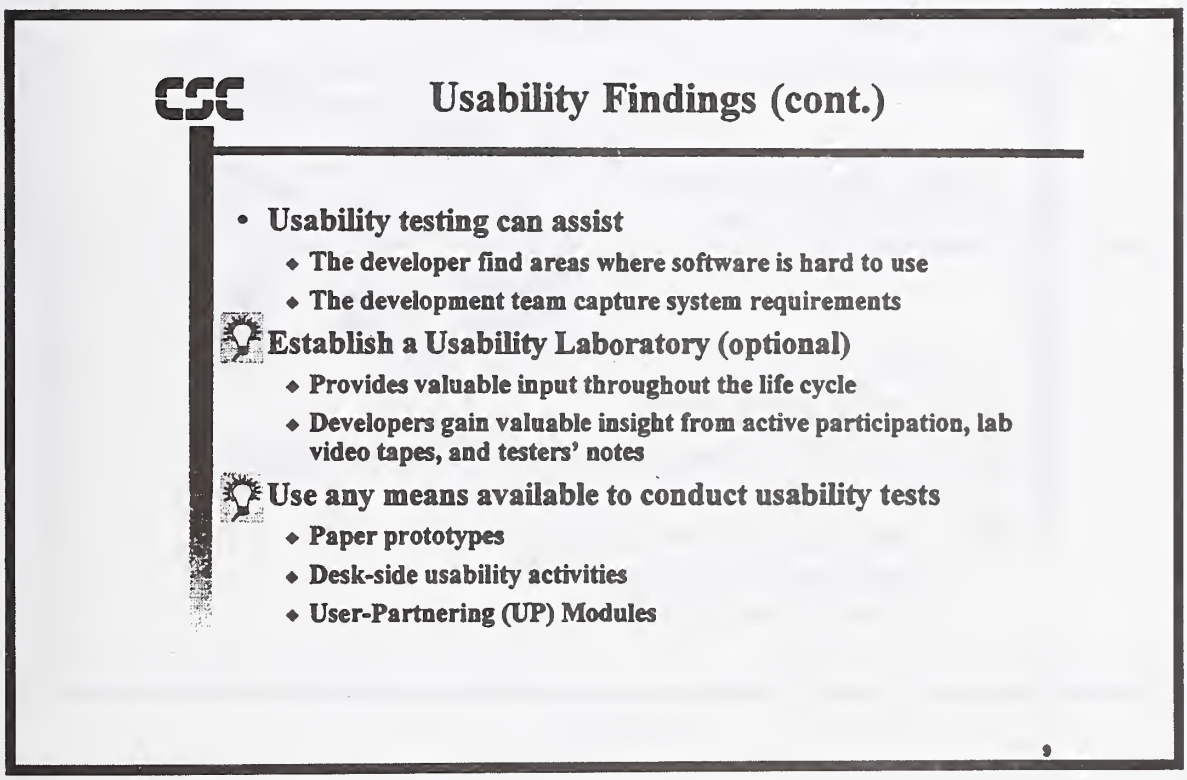


Figure 8

We gave reasons why it was important, such as helping find areas where software is hard to use. (Figure 8) We also recommended establishing a usability lab. Of course, that's not the only way to do usability testing or usability engineering and we discussed alternatives. Then we recommended using paper prototypes and other means of usability testing. User partnering

modules and Borland did not do usability testing with labs or contractors. They just did their work with in-house users, and observed users at desk-side. It wasn't a formal process at that time.

Earlier we heard about critical incidents happening during usability testing. Here is an example of something that really affected us and our customer. We went to Lotus Development Corporation and we interviewed people about their software development environment. Then we got a tour of the usability lab. Mary Beth Butler gives the tour and the user for the usability test didn't show up, so colleague of mine said he would do it, i.e, perform this test on a new version of Lotus. So he went into a room with a one way mirror. Mary Beth, the developer and I are in another room, and he has a microphone, and we are looking at him on two different screens. He narrates describing exactly what he is doing and he could not finish the process. It turned out that an icon should have changed when it went from one part of the window to another and it did not. Because it did not change, he was confused about what to do next. This was a critical incident for the developer.

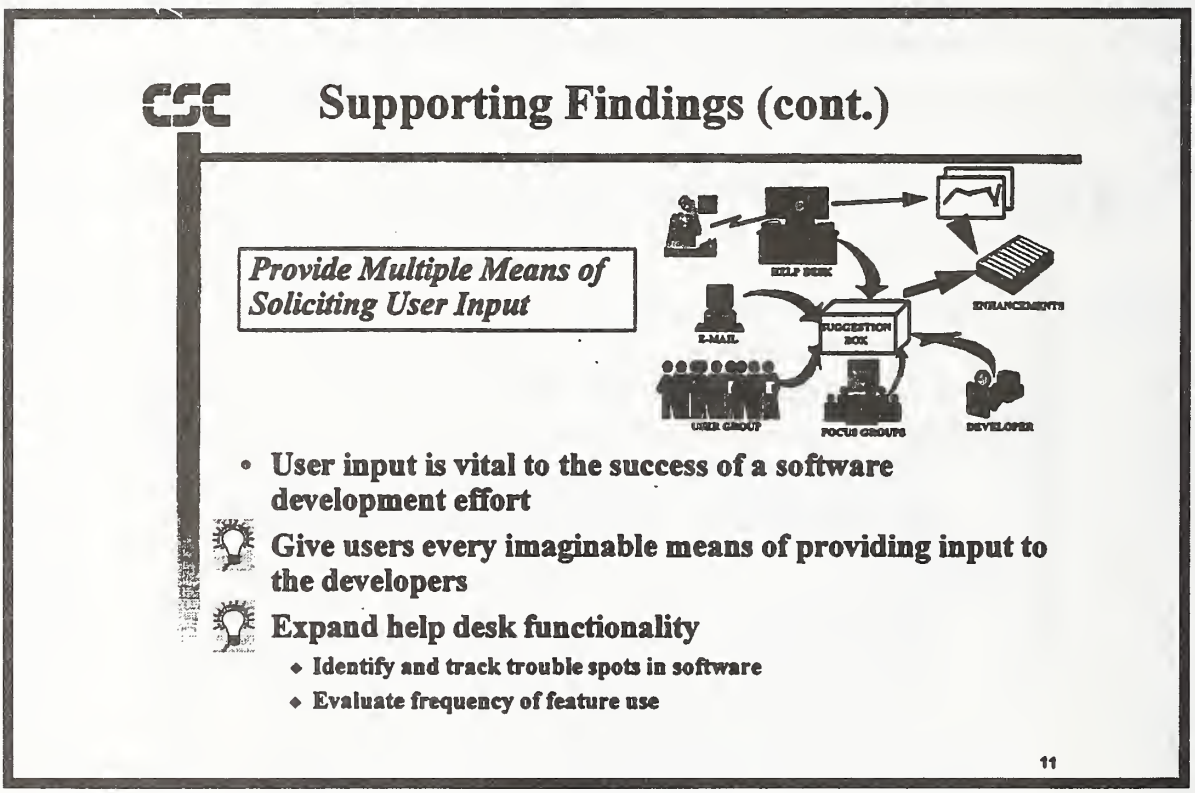


Figure 9

One of the important points that we mentioned earlier today is that user involvement is critical for testing. (Figure 9) A lot of companies put users on the development team, including CSC. It was also important to get as many different means of user input as possible, help desks, focus groups, user groups, E-Mail, as many as possible. (Figure 10) When we first presented this slide (Figure 11) to our company before we presented it to the government, it took about 2 hours to discuss the importance of developing software iteratively. How do you do that in an

environment which, sometimes, is traditional, like Oracle development? But, it was critical and it was a way of minimizing risk and we thought it was important to capture the requirements throughout the life

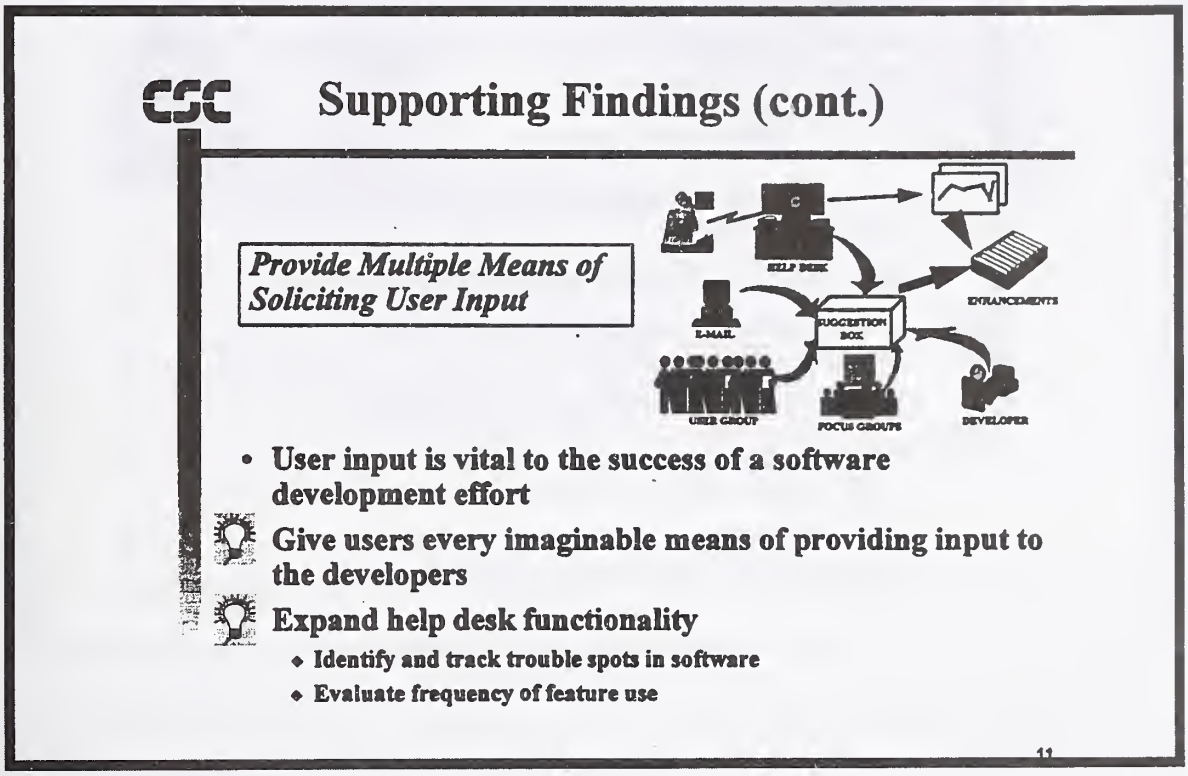


Figure 10

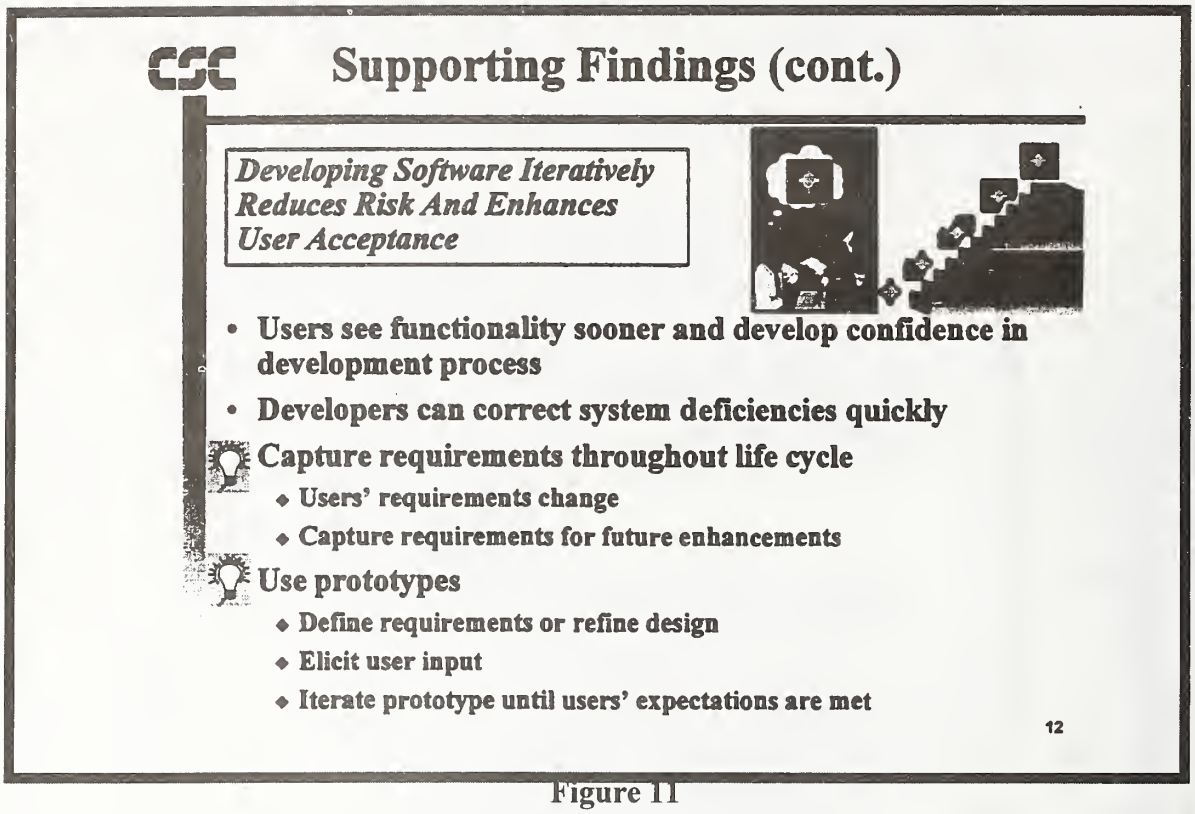


Figure 11

cycle. You want to develop software using the Oracle method, it is valid in certain cases and it is a lot of context, but with the Oracle method requirements you stop requirement development and at a certain point you develop code. In the real world this does not happen. Requirements development does not really stop; you just add more and more ECP's and enhancement until you eventually you're further along but you haven't got a product out yet. So we should say develop iteratively, but give the user something of value in a fixed amount of time.

This figure (Figure 12) describes the importance of employing a small heterogeneous team.

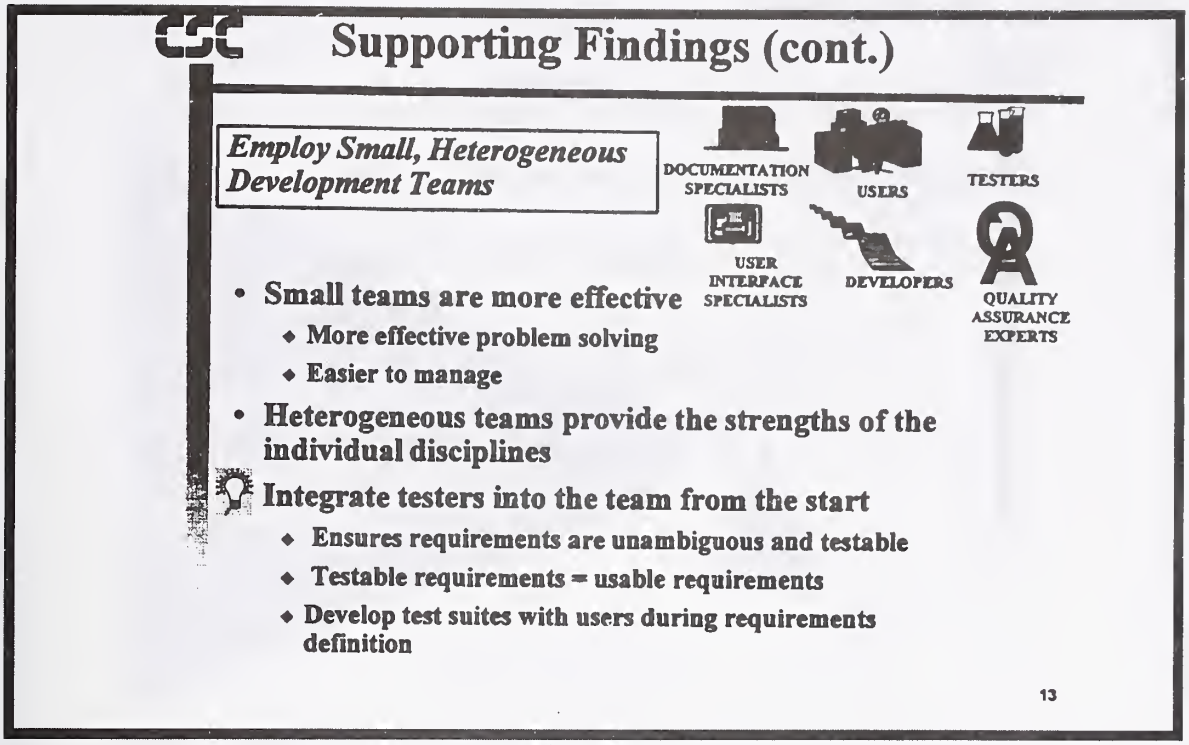


Figure 12

The government wanted a way of doing business, so we gave them a conceptual and logical methodology. Now this methodology (Figure 13) is based on CSC's catalyst methodology. Abut the same but it is to a certain level and the government is implementing this methodology in a new facility they are building. We called it PRISM, Prototype Improved Software Methodology. (Figure 15)

And the rationale (Figure 14) for this: we had some statistics about the current way of doing business that you've been doing is it really working, and it was clear to them why it wasn't working, but we also had some research reasons and the environment was changing. Users are different. Many users are more computer literate. As they discovered, people cannot wait six to seven years for a new capability or new system.

We presented this methodology based on our best practices, and we wanted to give them a methodology that was team based with users no just participating at the end but from the very beginning throughout the entire process. And the team is not just the users, it is the government

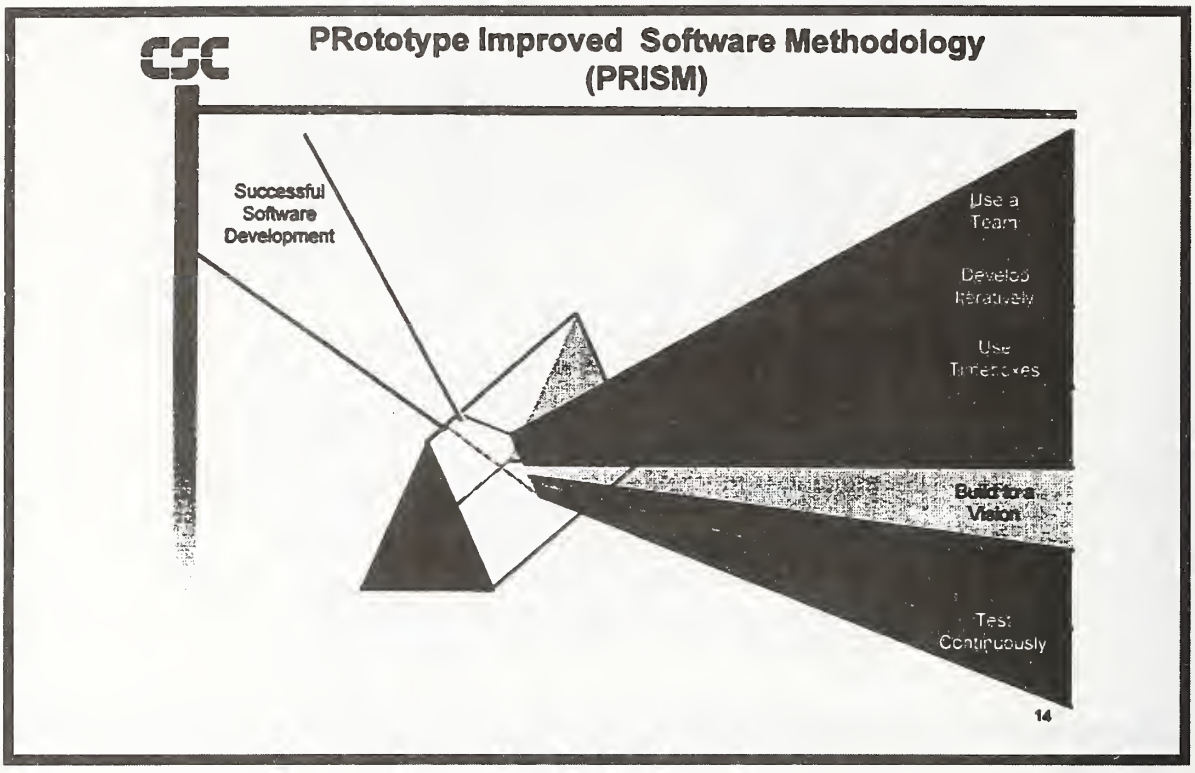


Figure 13

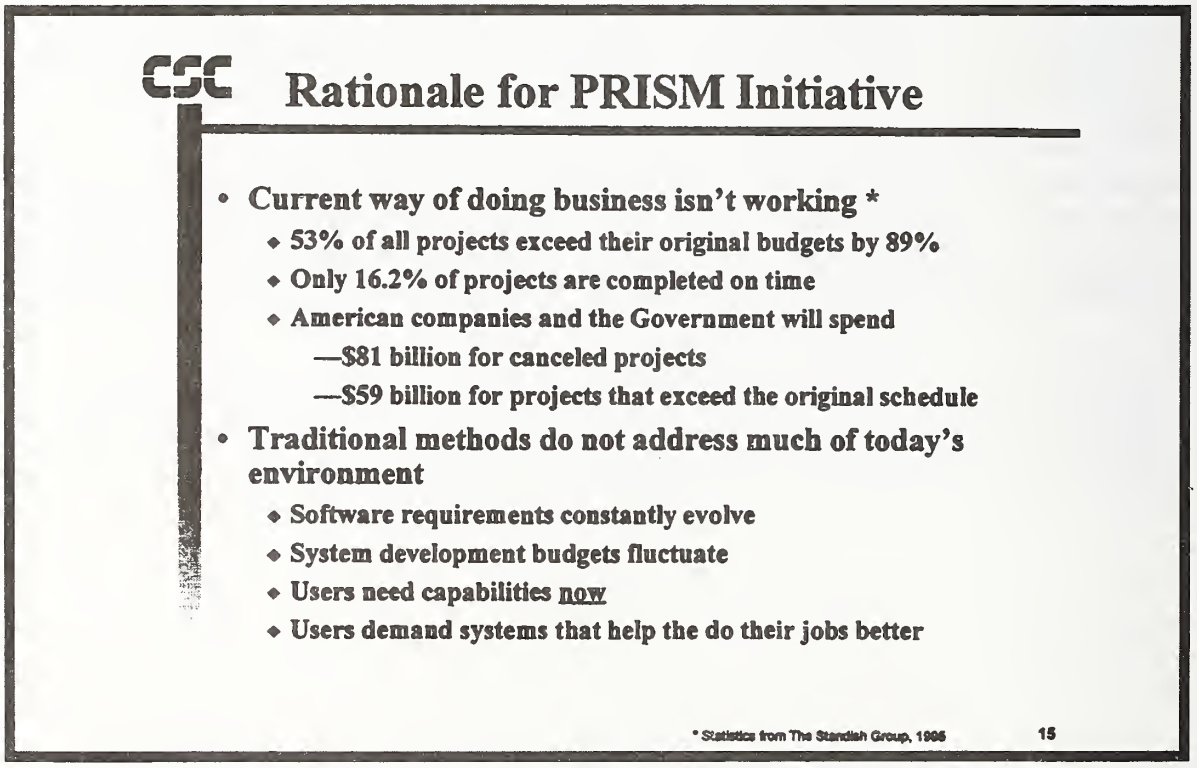


Figure 14

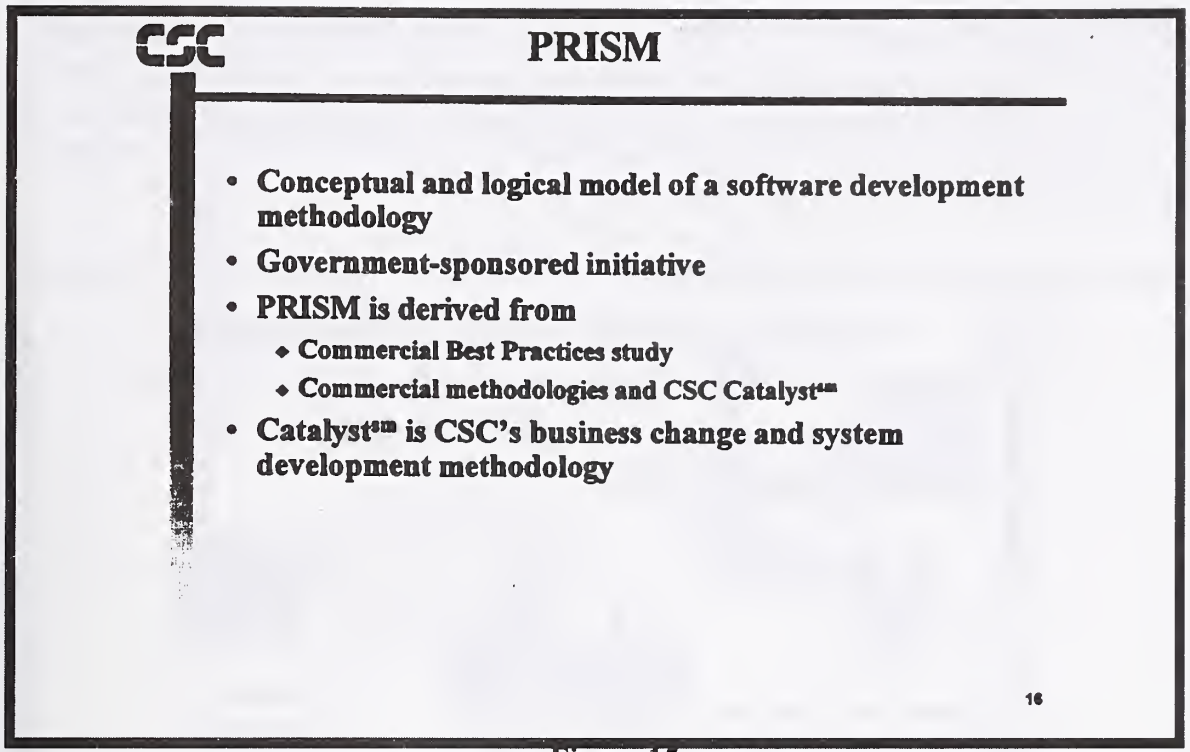


Figure 15

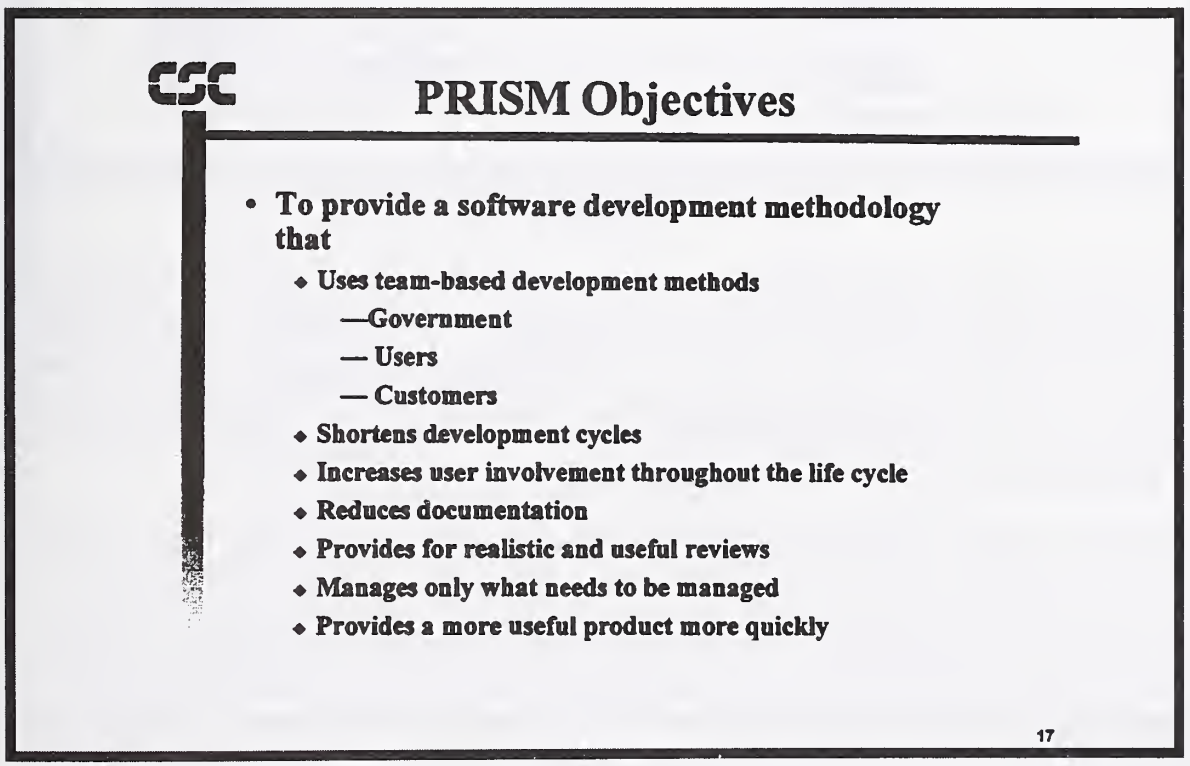


Figure 16

plus the customers. (Figure 16) The particular customer we had developed systems for their organization but part of their organization comes to them to develop a system. We wanted to give them something that had a shorter life cycles and has user involvement throughout the cycle and reduced documentation. We thought it was important to not have so much documentation only the documentation that was needed.

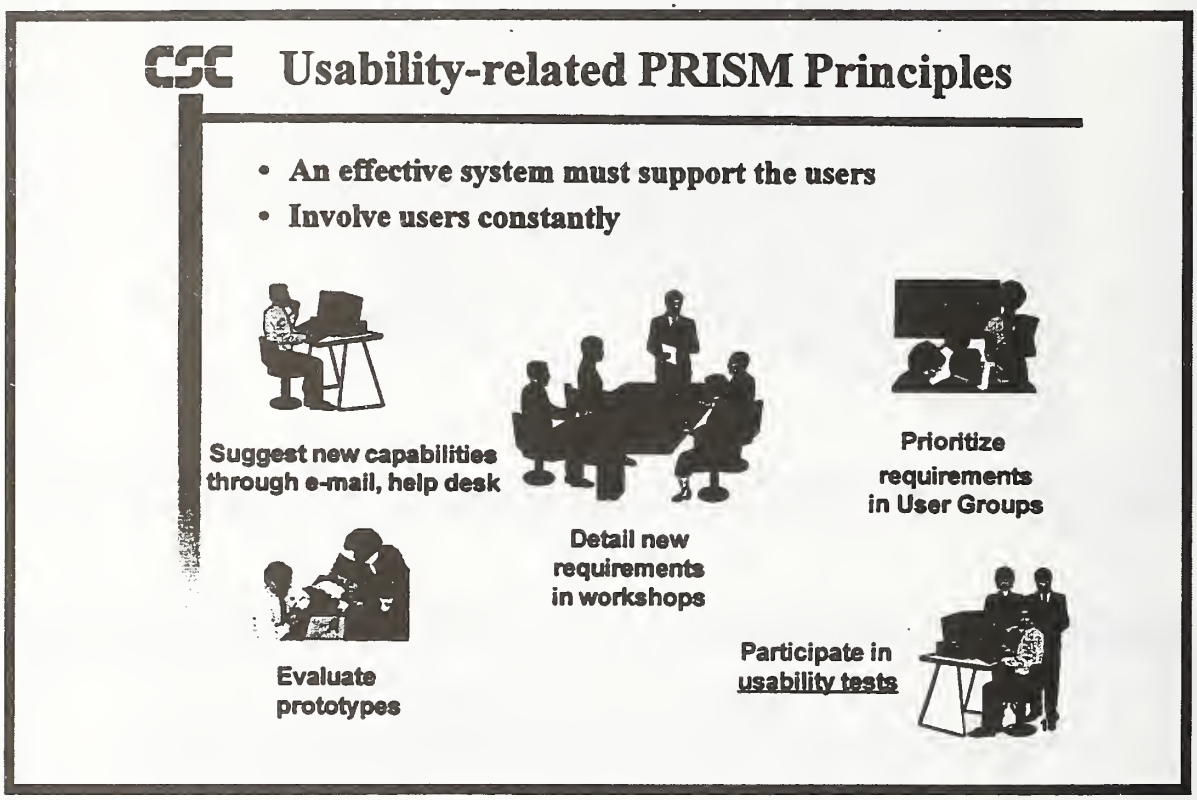


Figure 17

Now I'll just go through and highlight how we thought it was important and how usability came to be involved. For example we said an effective system should support users and it should support them constantly and one way to do that is for them to participate in usability testing. (Figure 17) Test early, don't wait until the end. (Figure 18) Test as you go along, have testers on the team as you go along. Have testers help write the requirements or be there to show that the requirements are testable. Have usability tests.

Many times in the government environment you'll come in and they'll already have a vision or they have an idea of what they wanted and they'll come in with just a spec. This is what we want you to do in RFP. We thought it was important (Figure 19) to get the developers, the programmers in early and help them with their vision about what they really want. Not only develop the team, but build to a vision, use iterate development, develop useful documents. Use a methodology, a single methodology that could work both with development and maintenance. So we gave the

Usability-related PRISM Principles(Cont.)

- Test the software as it is developed
  - Develop a little, code a little, test a little
  - Catch errors early to reduce cost and impact
  - Developers test their code
    - —Unit tests
    - —Integration tests
  - Testers test the code
    - —Integration tests
    - —Regression tests
    - —Stress/Volume tests
  - Users test the applications
    - —Usability tests
    - —Acceptance tests

19

Figure 18

Other PRISM Principles

- Approach software development as a team
- Build to a vision
- Use iterative development
- Develop useful documents
- Conduct useful reviews
- Use a single methodology for development and maintenance
- Evaluate all perspectives of change

20

Figure 19

methodology that had a phase in which they decided what the vision and strategy was, (Figure 20) had an architecture phase, sort of like a systems engineering phase, a development phase and a deployment phase. For example, let me point out where usability fits in. In the vision and strategy phase sometimes they might want to create a prototype, a concept prototype of exactly what they want, what kind of system they want. It could be a paper prototype. This paper prototype could also be a concept of operations. People in government they talk about concept of operations, how we're going to operate, what the procedures are.

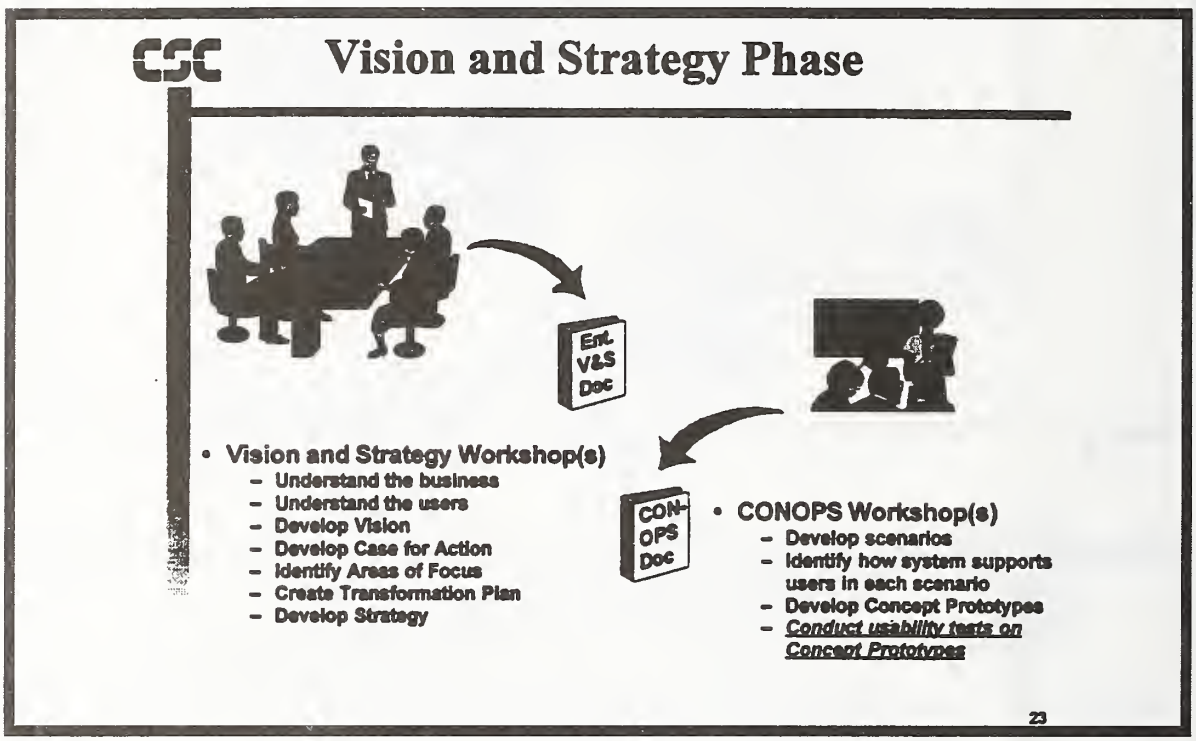


Figure 20

Now going down from the vision and strategy to the architecture phase, (Figure 21) it had two phases. One was developing the architecture but one was gathering the needs and suggestions of the users, what exactly do they need, what are the different ways of doing that and in the architecture development part we'd prescribe a testing strategy document which included usability testing and here would include looking at the usability requirements, what would they be?

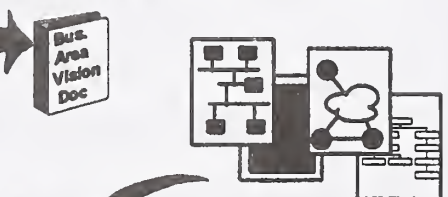
Here's a picture of that slide. (Figure 22) And it is important in the government, not as much in the commercial world, to have documents that are permanent documents instead of temporary for the particular project and how does what you're doing fit into those documents and we did that. Now in the architecture phase we had another component called needs analysis. How do you get new requirements. How do you go about evaluating them, testing them and so we had a process where for example the help desk might receive requirements with suggestions from users and developers. A systems integrator, which is a role for the government, would take these requests from the users and submit it to a formal board which could decide to accept or reject it. A paper

Architecture Phase

- Provides foundation for all development efforts
- Consists of two activities
  - Architecture Development
    - —Defines visions for each Business Area
    - —Develops Architecture Document
    - —*Prescribes a Testing Strategy Document*
  - Needs Analysis
    - —Gathers suggestions and potential requirements
    - —Develops Case for Action for each
    - —Evaluates each against Vision and Strategy and CONOPS
    - —Evaluates each for technical feasibility
    - —Produces a set of prioritized, costed requirements
    - —Assigns requirements to releases and development paths

Figure 21

Architecture Development Activity



Bus. Area Vision Doc

Arch. Doc

Test Strat. Doc

- Business Area Vision Workshop(s)
  - Understand the business area (BA)
  - Understand users' perspective of BA
  - Identify principles, constraints, assumptions
  - Develop Business Area vision

- Architecture Document
  - Describes roles/responsibilities
  - Identifies technical infrastructure
  - Specifies system and development standards

- Testing Strategy Document
  - Broadly describes testing approach
  - Provides guidelines on user and customer involvement in testing
  - Tailored for each capability

Figure 22

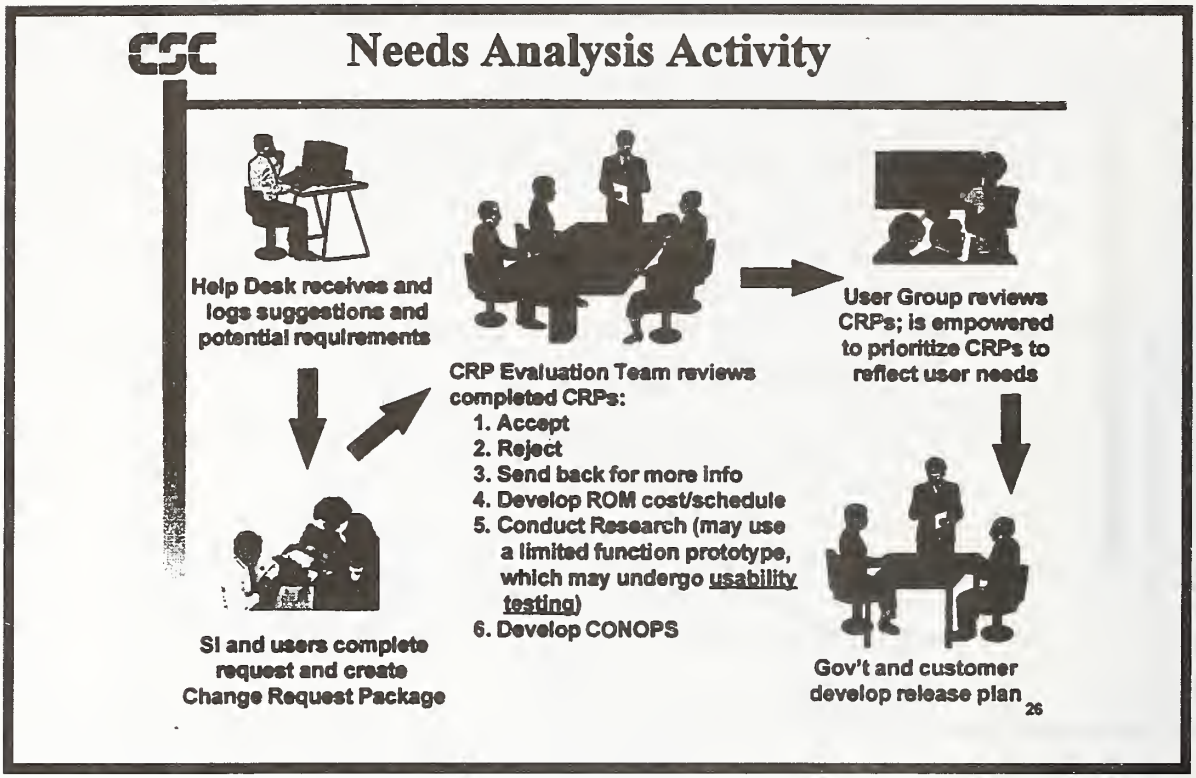


Figure 23

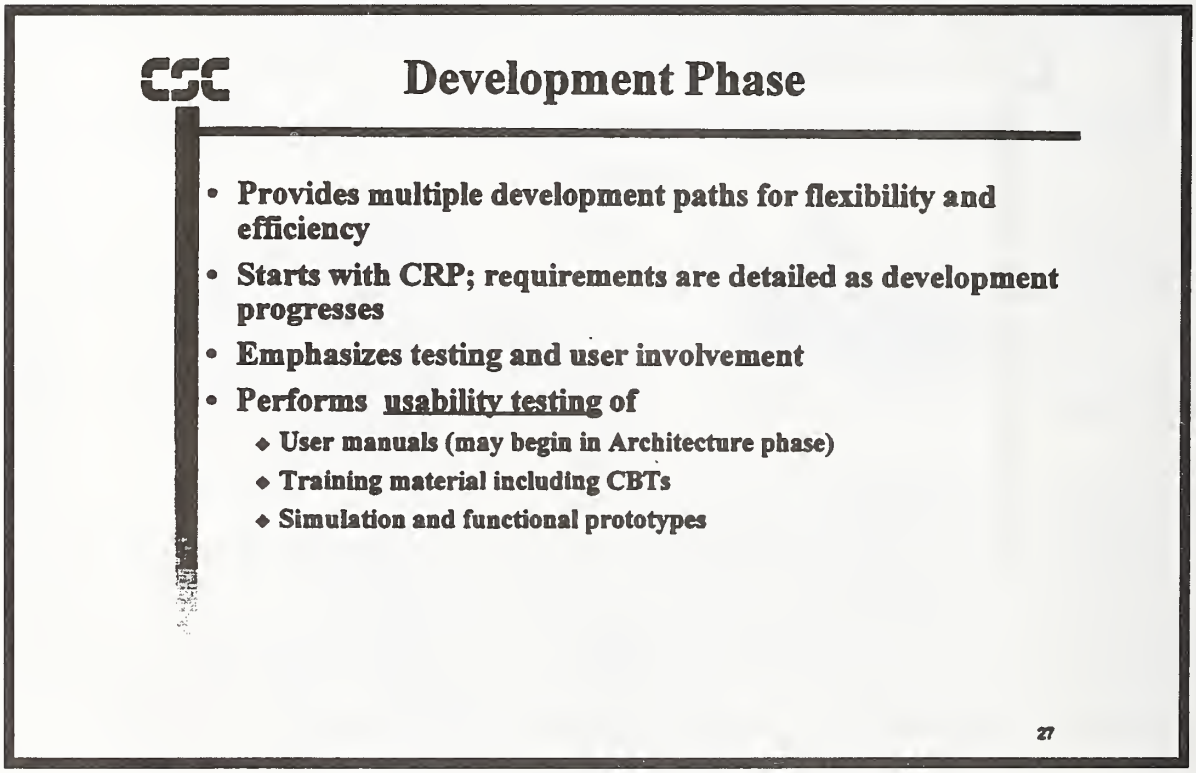


Figure 24

prototype or a more substantial prototype could then be built. Then, the user group would evaluate that and eventually it would be put into a bin where it would get prioritized later on for research depending on budget.

These are our findings. The phases help to place them in an environment where they would be useful and how would you use them if you did? How would you use these environments and these concepts?

We, as a company, are not the development contact for this project, so we can't follow through with this as I would like to. Again in the development phase once you've done that then you would test the manuals in fact you might even test it earlier than the development phase. Training materials including computer based trainings and any prototypes.

Again you'd be performing usability tests as you were developing them. Finally, when you deployed to the field you might also want to test frequently by using a pilot site first and then to the field.

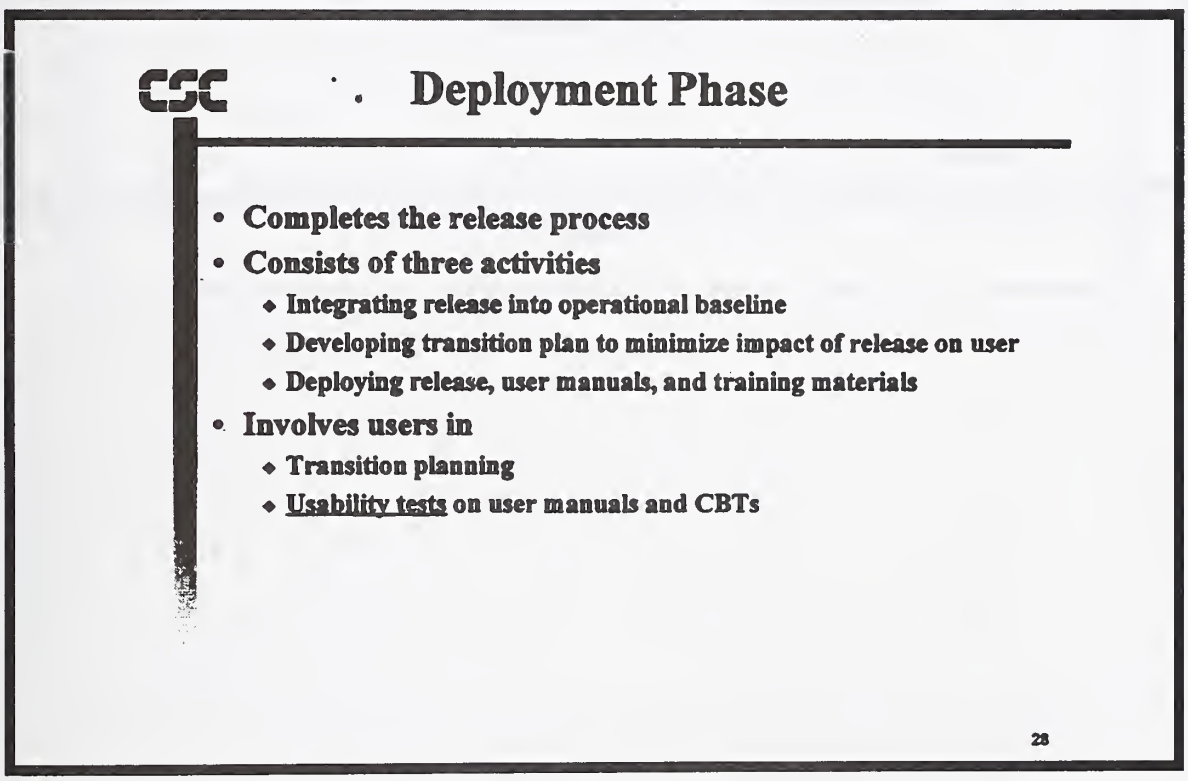


Figure 25

Our major finding was conduct usability testing which really means to consider usability engineering as a significant component of your development processor. Use iterative development, having user access and involvement and soliciting that input through multiple means. I described how a year later we gave the government a way of doing this, a methodology or framework for accomplishing this. The methodology was validated by an additional government consultant.

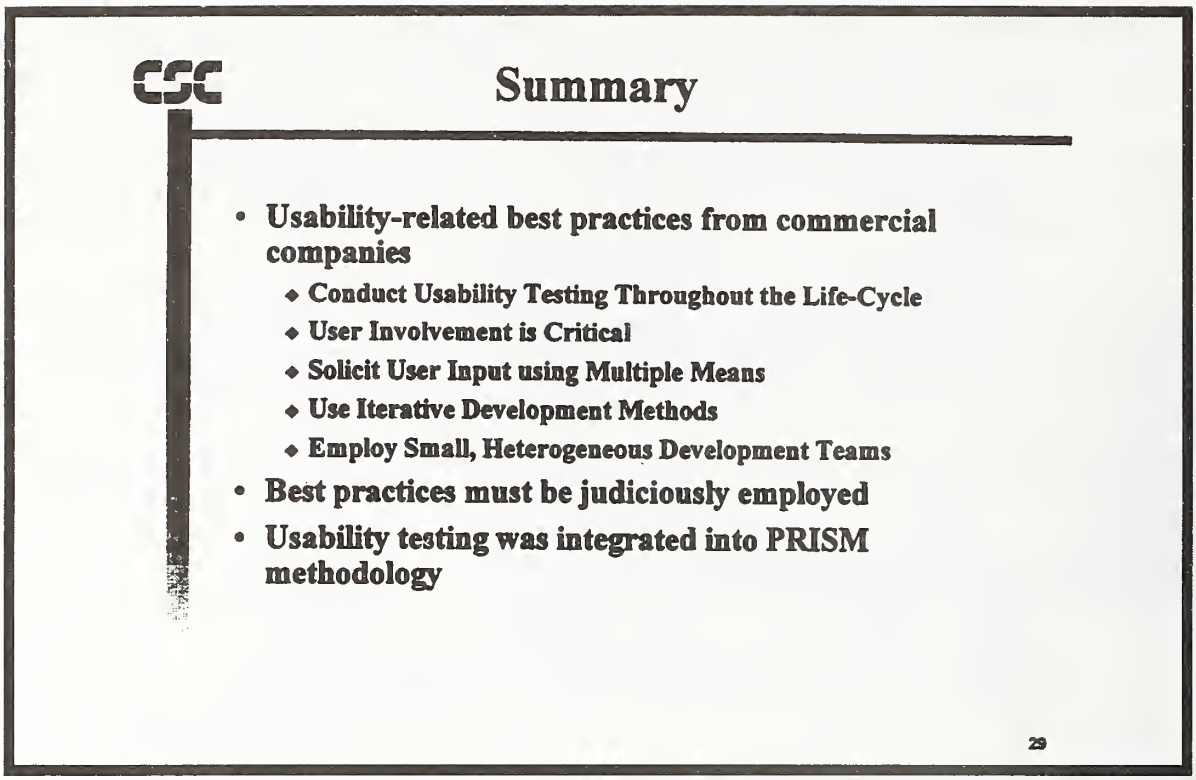

Summary
• Usability-related best practices from commercial companies
◆ Conduct Usability Testing Throughout the Life-Cycle
◆ User Involvement is Critical
◆ Solicit User Input using Multiple Means
◆ Use Iterative Development Methods
◆ Employ Small, Heterogeneous Development Teams
• Best practices must be judiciously employed
• Usability testing was integrated into PRISM methodology
29

Figure 26

Usability Issues in Complex Government Systems

Dr. Donna L. Cuomo
Jill Lynn Drury

January 10, 1996

MITRE

Dr. Donna Cuomo/Ms. Jill Drury, MITRE
Usability Issues in Complex Government Systems

I'd like to start with a question. First of all, how many people had to fly in here from various parts by show of hands? OK, those people who had to fly in, how many people thought about what their air traffic controller was seeing on their display as they were landing? Chances are they had to pick out your aircraft from hundreds of other aircraft. Maybe just dozens, but still quite a few. They might have had to call up auxiliary information on the speed and heading of your aircraft and look at the other aircrafts around you and maybe call up the flight plan. I bring this up, because it is an example of a real time complex system procured by the government. We all have a stake in seeing that this system and others like it are procured with user interfaces that

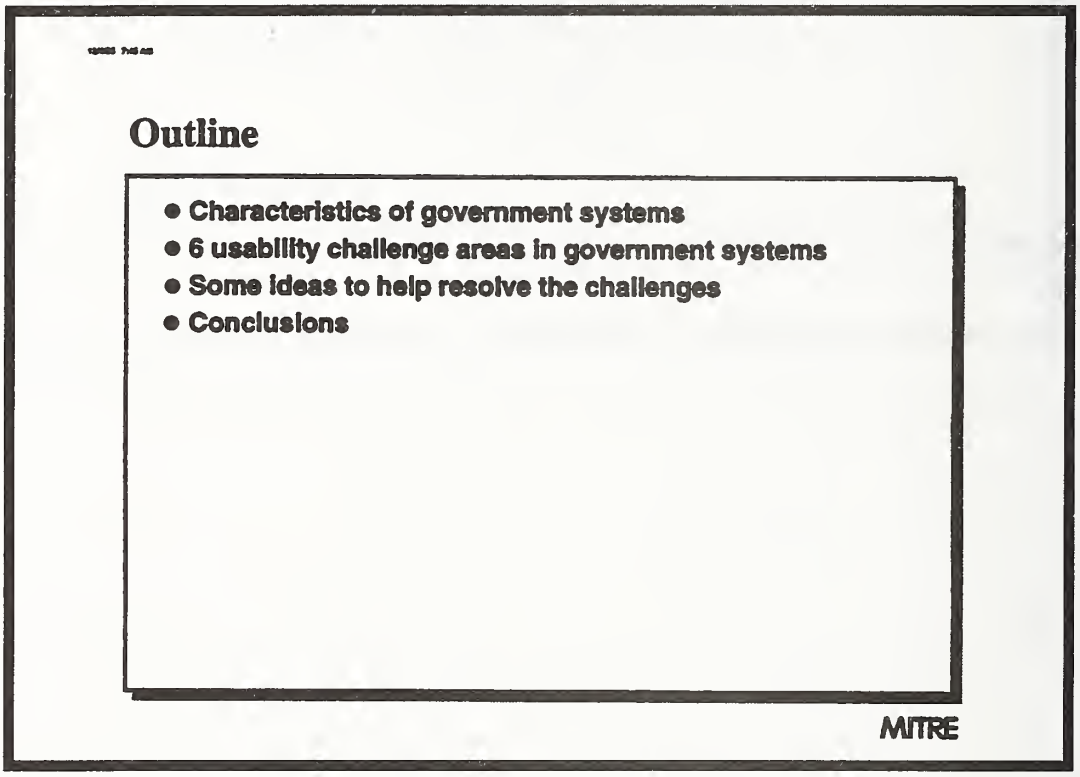


Figure 1

help instead of hinder the users. So our purpose today is to maybe raise some awareness of some of these issues relating to complex government systems and to spark some discussions that may benefit the community. I will have succeeded today if you take away from this talk a feeling that there really are some challenges out there in this area, but there is something we can do about it. I would like to start by generalizing some characteristics of complex government systems that contribute to their design complexity and then I'm going to talk about six areas in which there are usability challenges and, finally, talk about some ideas that might help resolve those challenges.

Aircraft control systems are systems that have a very low tolerance for errors. (Figure 1) They operate in real time. They may also have broad scope with hundreds of displays or hundreds of

business rules, what they call Standard Operational Procedures in the military. This also contributes to design complexity. (Figure 2)

They might also support obscure tasks done only by a few people who are very highly trained.

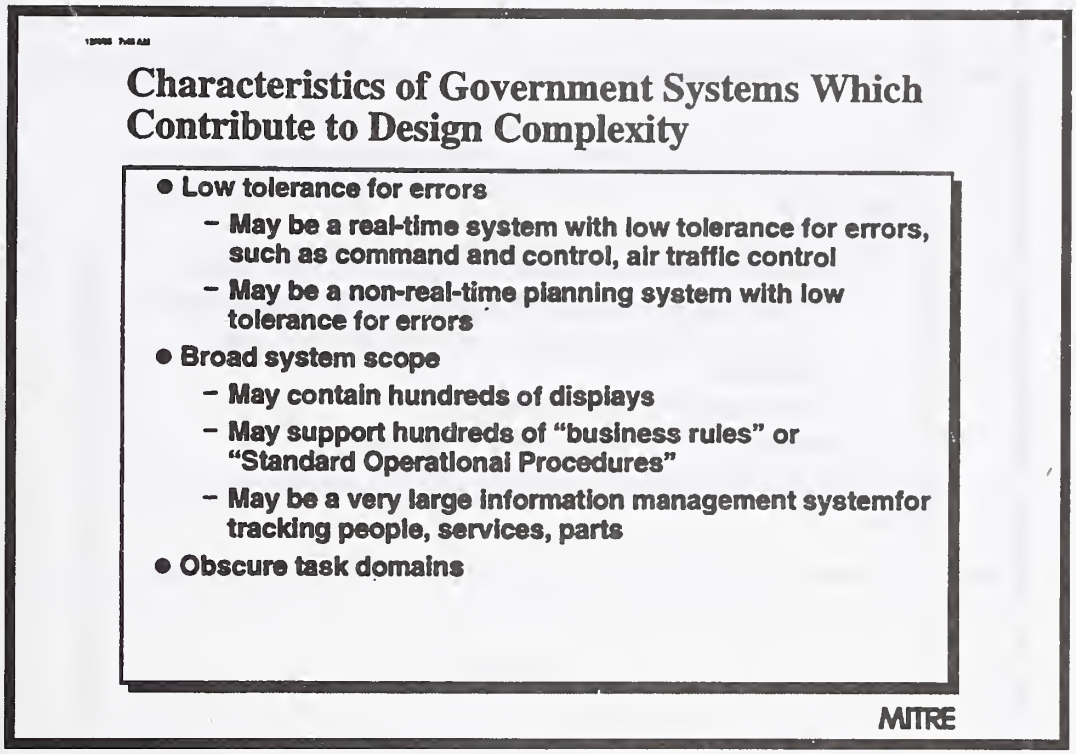


Figure 2

The tasks may also be done by a vast number of people. For example, think of all the aircraft maintainers that there must be in the Air Force or that may have to support people who are in more than one service using the same product. There could be a lot of variability in the work environment. The same product may have to be used in an aircraft, on the flight line or it may be in a ship as well. All of these things have to be acquired using the government acquisition process. I'm actually rather surprised that no one has really brought this up today as an issue because I know that we spent a couple of CHI (computer human interaction) workshops, in 1994 and 1995 talking about the difficulties that government systems have in getting good user interfaces, because of the government-mandated acquisition process. This process makes it difficult to pull in techniques already in use in the commercial world. The ideas from the previous presentation that have been floating around the commercial world for a while, but they are just now taking hold in the government arena. In this presentation we are concentrating on the design complexity issue, but if you want to see a quick summary of some of the problems involved in government acquisition in terms of their possible effects on user interfaces a good reference is the CHI Bulletins of October 1994 and 1995.

Figure 3 lists six major challenges related to just the fact of government acquisitions impacting potentially your user interface development. They arise from the process that you might use when trying to acquire a usable interface. You see some of these same sorts of ideas in the Star life cycle that Rex Hartson talked about in an earlier presentation example.

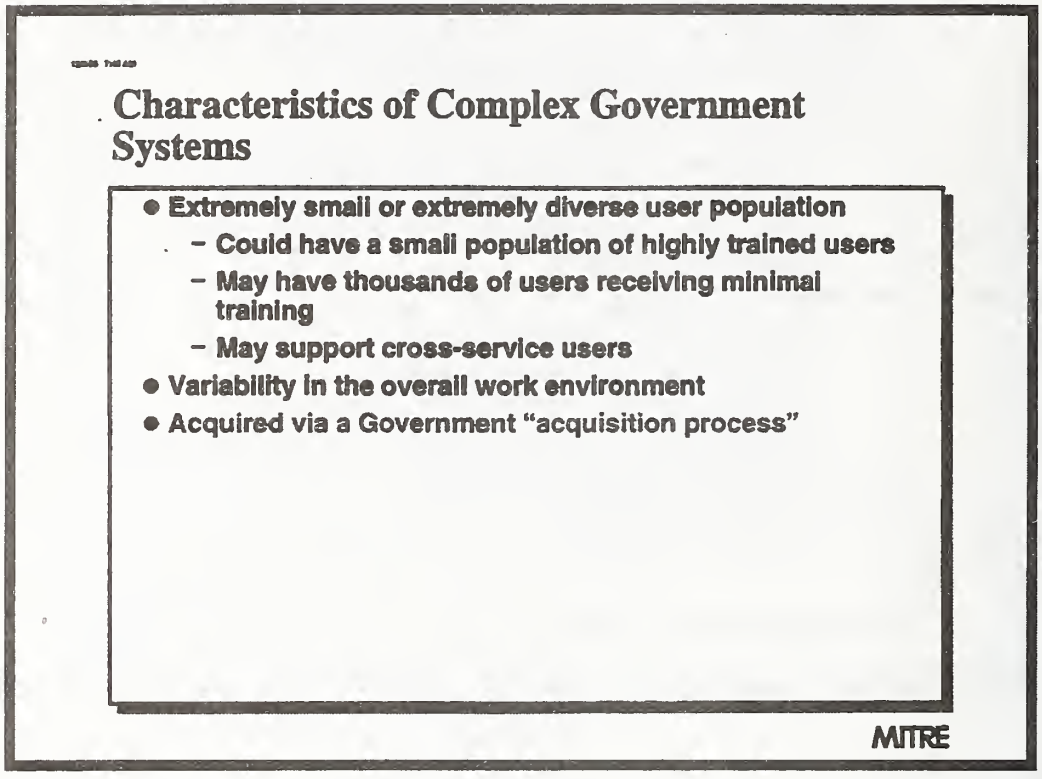


Figure 3

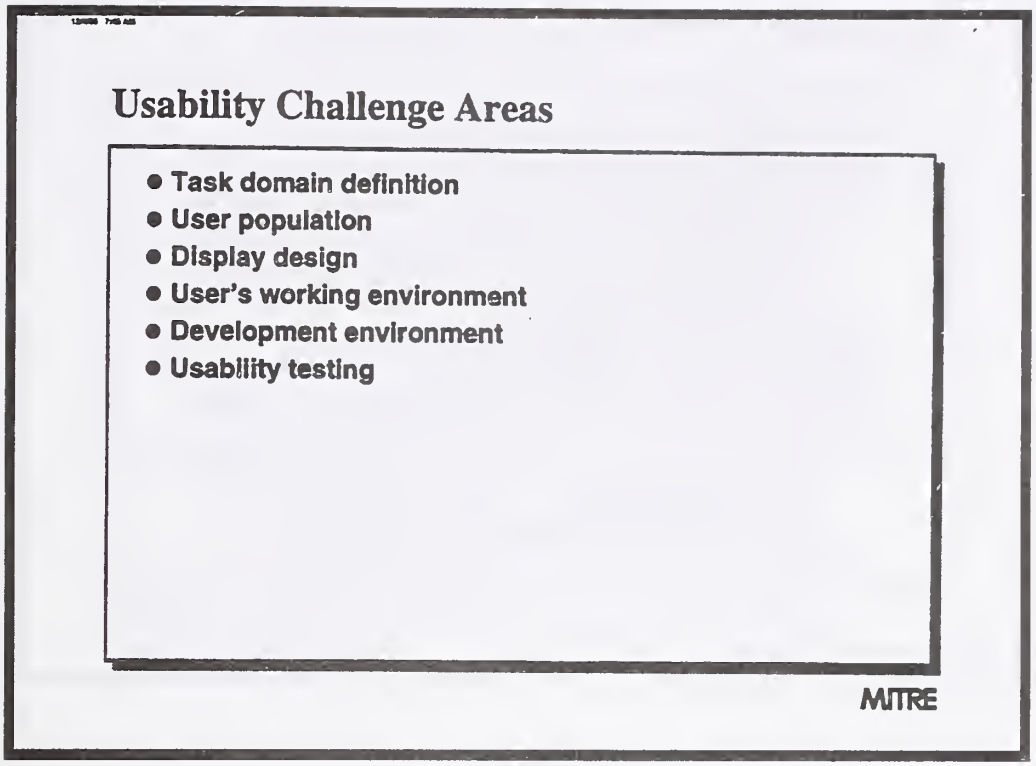


Figure 4

So the first challenge area has to do with getting task domain knowledge. (Figure 4) It is universally accepted by human computer interaction people that you need to have a good feeling for the tasks that the users have to do before starting to design or evaluate the design of a user interface. In some government arenas, it is very difficult to get that knowledge. For example, I once was fortunate to spend time in a windowless bunker blasted out of the side of the mountain in Northern Norway watching users, but I don't get that kind of access very often. Your average HCI specialist is very familiar with office automation products, because we all use some kind of word processor, maybe a spreadsheet, etc. They typically do not know so much about air traffic control systems, air defense systems, or ballistic missile defense. The situation is even worse when these tasks are completely new. No one has really done things in quite this way before or if they are classified where it's difficult to gain access. Also when you do get to talk with users they often talk about what they need in terms of the way they do things now and they may not have a visionary view of what is possible because what they are doing now is very constrained by 60's or 70's technology in some cases.

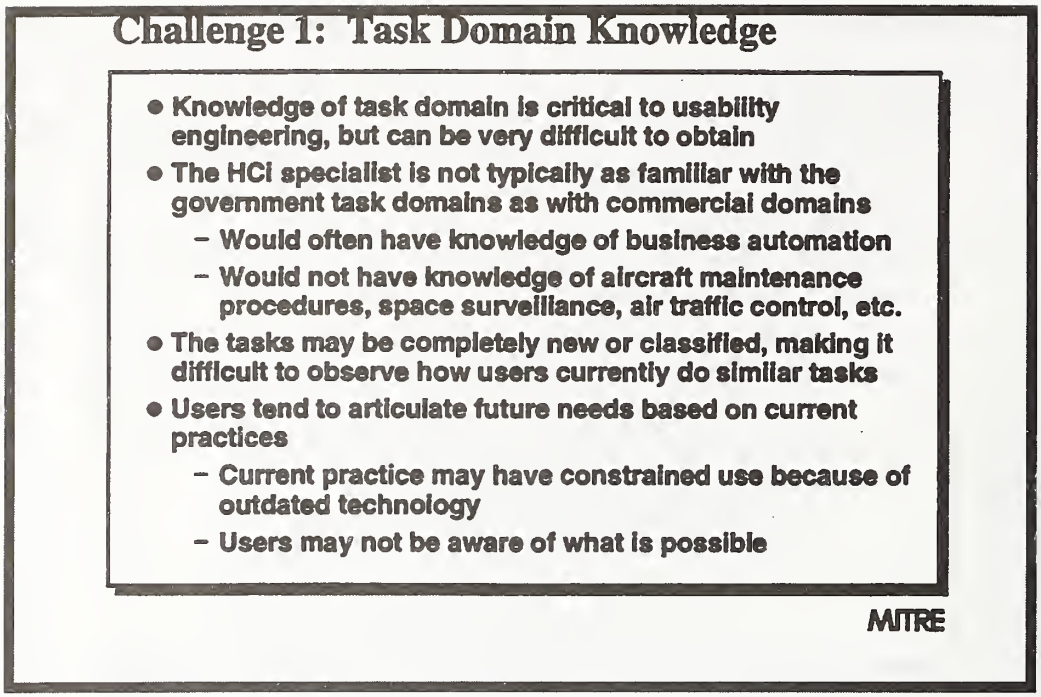


Figure 5

It is important to get HCI specialists together in with the requirements definitions people and with the contractors who have some good domain knowledge. (Figure 5) If possible, it is also very good to visit users in their work space. If this is not possible, we suggest that you try to get access to simulations or training systems or handbooks of the users. If the tasks are completely new, we suggest spending extra time with those people who are defining the requirements because chances are they had to go to bat to get that system approved to begin with so they are a good source of knowledge. If possible, have them develop detailed operational scenarios. We found that to be very helpful. Of course, the best thing is to have a user represented at work full time with the HCI team.

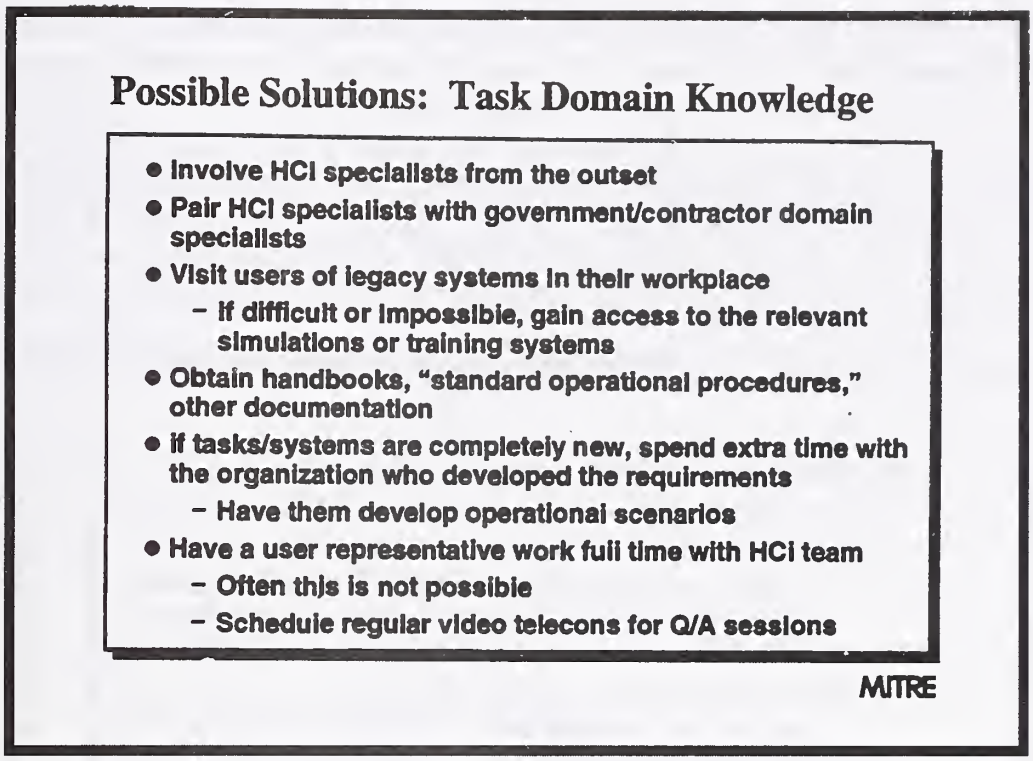


Figure 6

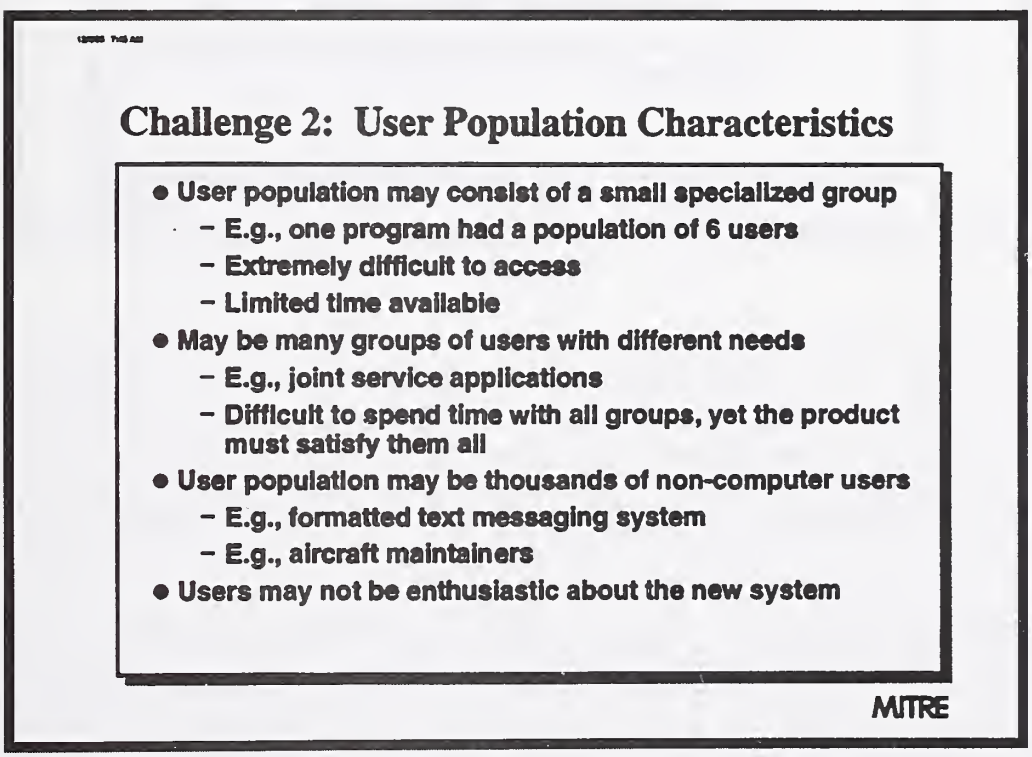


Figure 7

The second challenge (Figure 6) is very much related to the first one because as you examine the task domain you are also looking closely at the users who are doing those tasks. These users may be very difficult to access if there's only a small group of them or they may be a large group of users with many sub-groups having different needs such as somebody from the Army and Air Force and Navy all trying to get onto the same program. Your population may also be thousands of people who don't normally work with computers. In addition, it really doesn't help that a new system or an upgrade to an existing system is mandated from above, because the user may not be really enthusiastic about helping you define something for this new system because they are

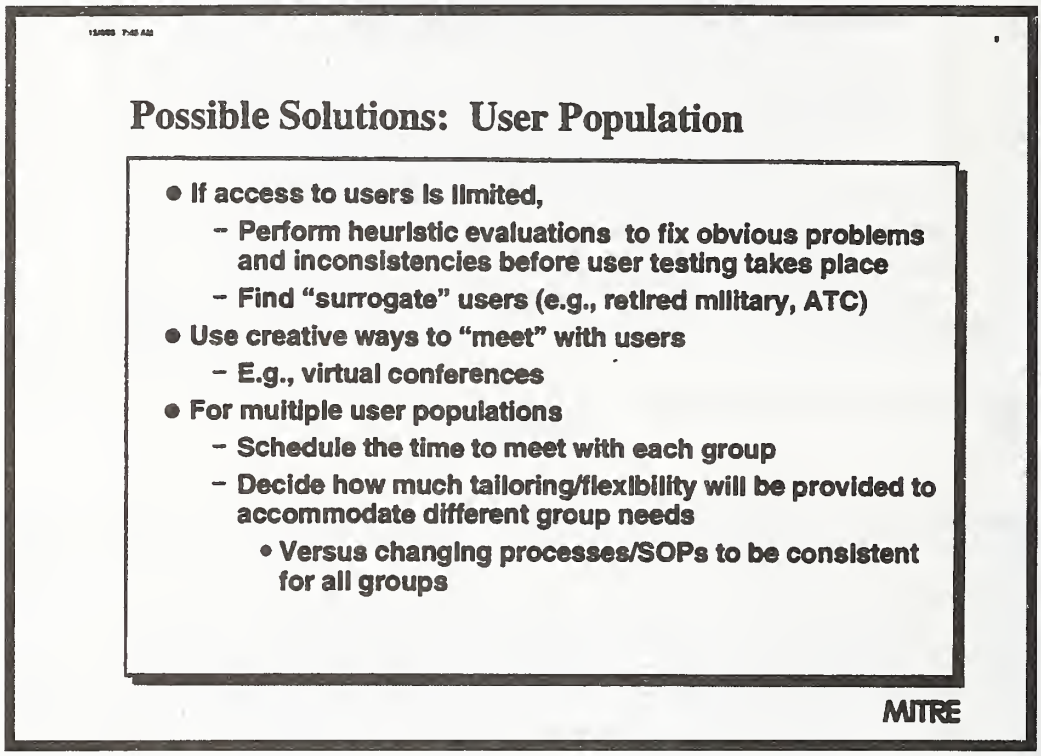


Figure 8

doing just fine as is.. What are some possible solutions? (Figure 7) We talked about some of them just a couple charts ago when we said that you could try to gain access to their simulations or their training materials if user access is scarce. I noticed earlier somebody was talking about heuristic evaluations. For those of you who weren't familiar with this term before, a heuristic is basically a rule of thumb. To do a heuristic evaluation an HCI specialist just sits down and goes through each screen one by one, it could even be on paper, and looks to see whether these rules of thumb are being violated. For example, are there clearly marked exits? Is the interface internally consistent? This minimizes user involvement, if you don't have much of their time. It could save them for more domain dependent problems or for more subtle problems. We also try to get creative in ways to meet with users. For the situation where you have multiple user populations there is something to discuss very early on and that is whether or not you're going to tailor your user interface for the different groups so you have actually different versions or whether you're going to try to get the groups to come together and agree on some kind of common standard operational procedures so that you can design knowing that everyone will

pretty much be doing business the same way. You just can't assume that people are going to be doing business the same way. There are different types groups and it makes a big difference in the way that you do business based upon which option you pursue.

So the third challenge (Figure 8) is the heart of this presentation, and there are several aspects to the challenge of display design. Complex displays may include map-based system with overlays showing thousands of pieces of radar data, tracks, for example, showing storms as they move across. You might have a list of many, many items of data or tables of data. Imagine all the flight plans for the Washington area for a twenty-four hour period. You might also have to have all the documentation for a system on-line so you can imagine how many pages that would be for a bigger defense system. Another challenge has to do with real time systems and how they would need to capture the essence of what's happening in real times. Picture an operator who has to see whether or not an aircraft is following it's flight plan. You need a graphical representation of that aircraft. You also need another representation of the true path that the aircraft should be flying on according to its flight plan, and you might also need a little box that shows where the aircraft is expected to be in relationship to its flight plan that might be moving along.

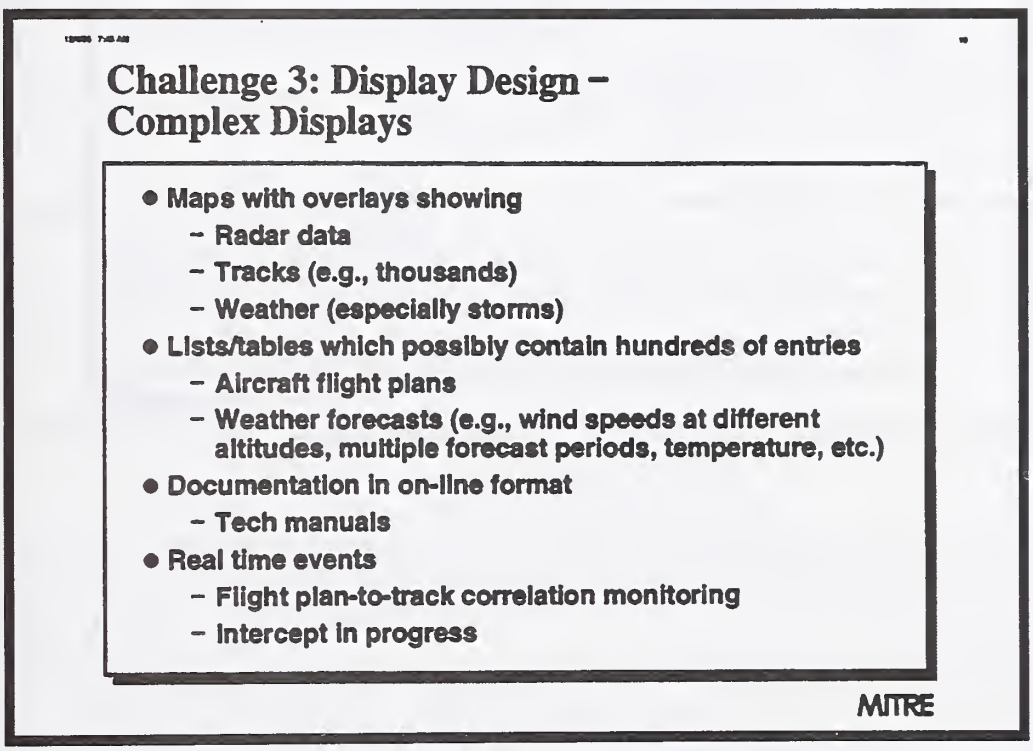


Figure 9

Here's another example of a complex display (Figure 9) where you see, in the middle, an intercept in progress. You see this is where a intercepted aircraft is expected to be flying and this where you might expect to intercept an aircraft that's moving along in time. You'll see here that the operators for our air defense systems often ask for just a very simple map outline. This is intended to be a coastline because they have to look at so many other pieces of data they don't want a whole lot of clutter introduced by the map itself. In this particular example, there are seventeen tracks, seven radars, three other ground based installations, and three orbit points and

the intercept. Just imagine if you had 50 or 100 tracks that are moving in here which would not be unusual for a system like this. These tracks are also pretty well-behaved right now, because you don't see them flying on top of one another at different altitudes. They can also be over flying one of the radars, for example, and you would see all your symbology right on top of one another. You'll have operators playing around with different magnifications to try to see what's going on.

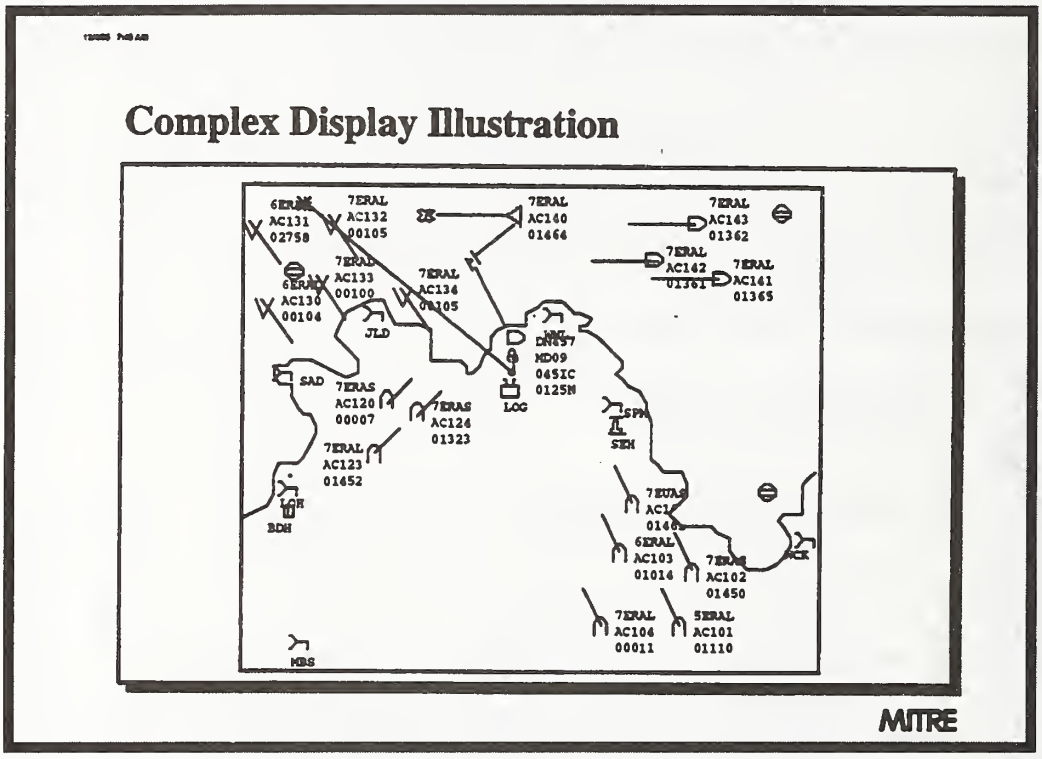


Figure 10

Another aspect to this challenge has to do with just the sheer number of displays and the large amount of functionality that many of these systems have to encompass. (Figure 10) Just the sheer scope of it almost always means that it's not just one person who is designing the interface. It's not just one person who is developing it. So once you've broken that up into different people or maybe even among different organizations, maintaining consistency is a problem. It's hard for one person to conceptualize and design even if you have one person who is kind of designated as the overall person who tries to look at everything. It's hard to keep that all in your head, especially when the requirements can be changing and fuzzy. This kind of system is all the more difficult and time consuming to document and to prototype, even, unfortunately in a paper prototype just because there's so much of it. Still, you need to do these activities.

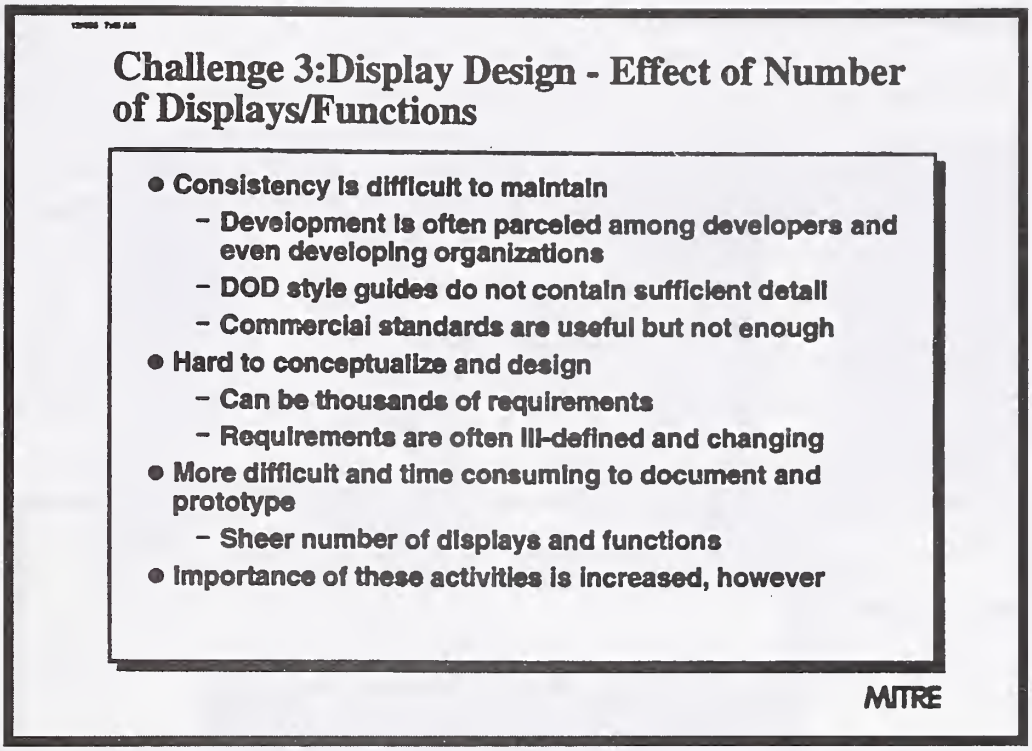


Figure 11

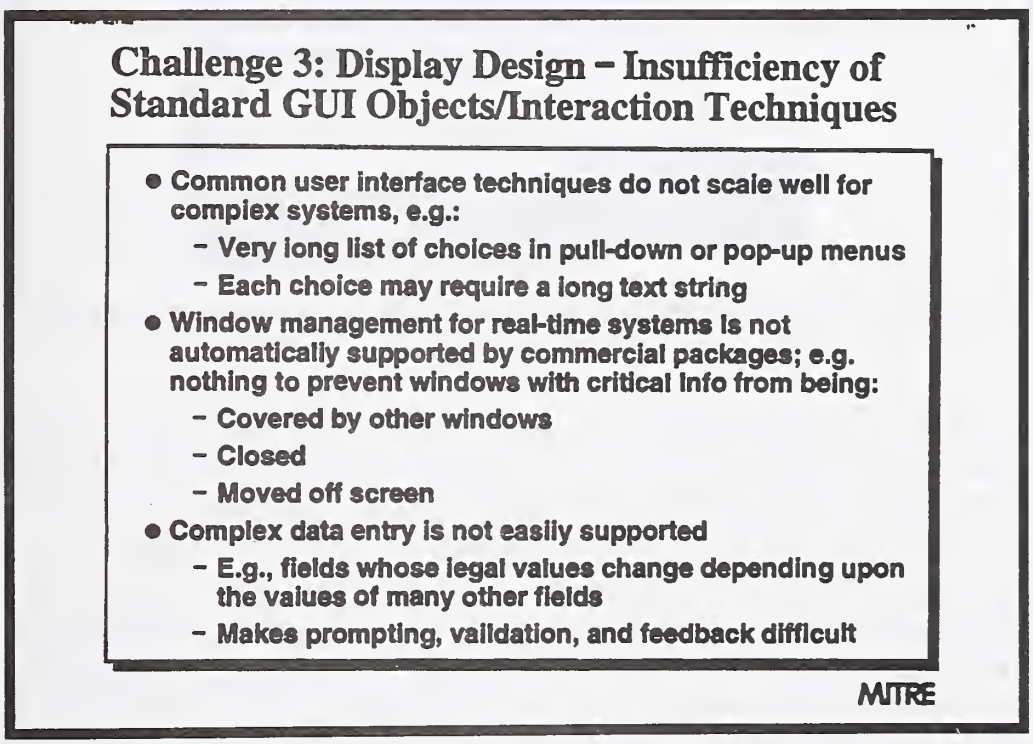


Figure 12

The third aspect of this challenge (Figure 3) has to do with the fact that many standard graphical user interface techniques and interaction techniques do not work well in this context. Earlier someone mentioned the fact that when you have some real time systems you may not have time to pull down a menu. If you have hundreds of menu items there may be just too many to fit on a pull down or a pop up menu. We have also seen names of individual menu items that were just too long to put in a standard kind of menu. Window management for real time systems is not often supported well. For example, if you've got the intercept in progress that I showed you on the other chart and an information window popped up automatically on top of it, by the time the operator moved it off or closed it the intercept could already have occurred unseen. When a user is trying to enter data, that's sometimes a problem because the legal values for one field may differ depending upon what is filled in for another field. So it makes prompting, validation and feedback difficult.

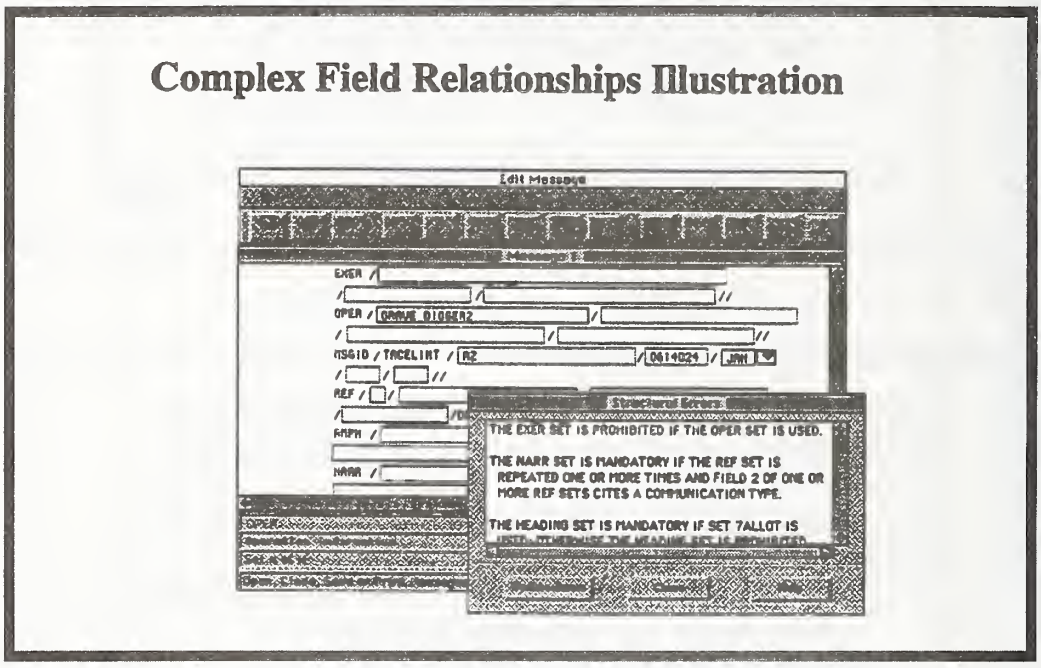


Figure 13

I've got an illustration of that in Figure 12. You see here that the active field is one of the operation information set fields and that if this is filled in here then this field on the top of the display next to EXCR, the exercise set, cannot be filled in. The rules get even more arcane if you bother to read through some of those on that illustration, but I'm going to move on.

Here is an illustration (Figure 13) of some window management going on. The window on the bottom called Engagement Summary is sitting a little bit on top of a 3-D globe window that is showing a map–based display. That is allowed according to this system, but in the corner under "alert"—that window cannot be closed, cannot be moved, and cannot be covered. One of the things that's showing is the number of detected nuclear destinations. The convention used here in fact is that the light blue windows cannot be covered up, moved or closed where other ones with the gray at the top have more freedom on what they can do. The point of this is that this

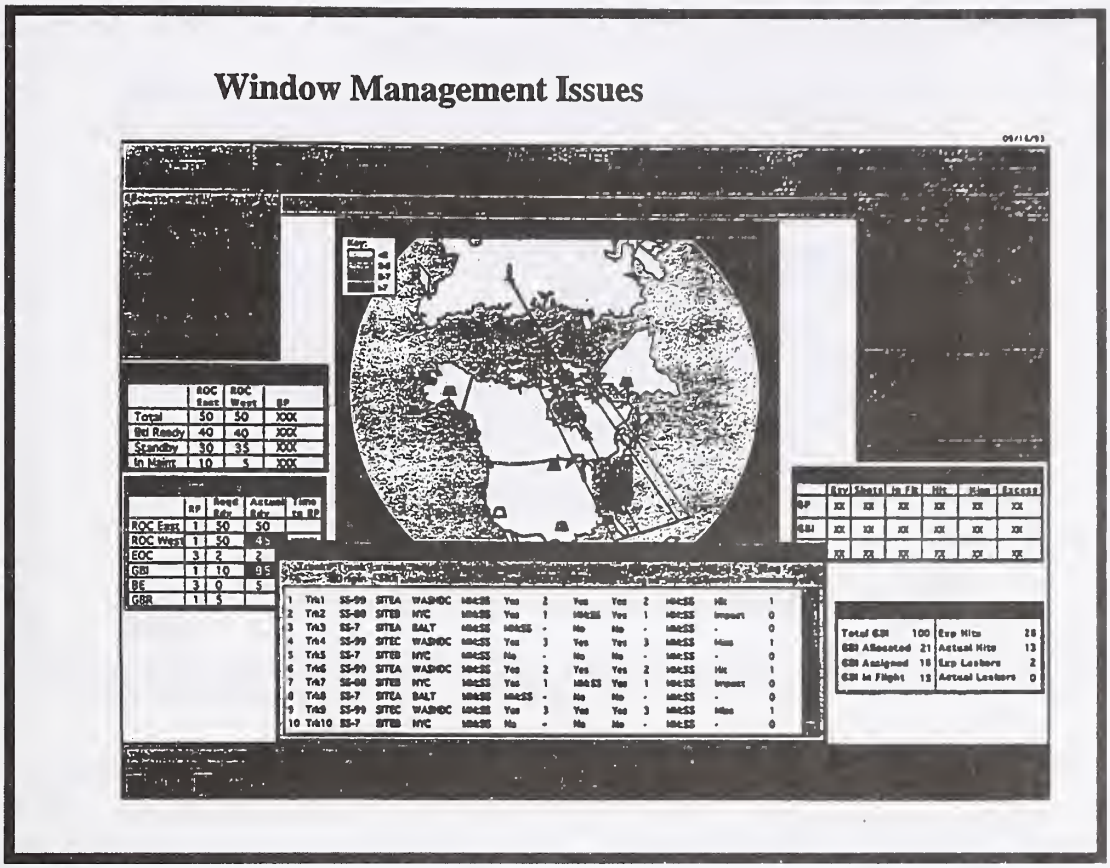


Figure 14

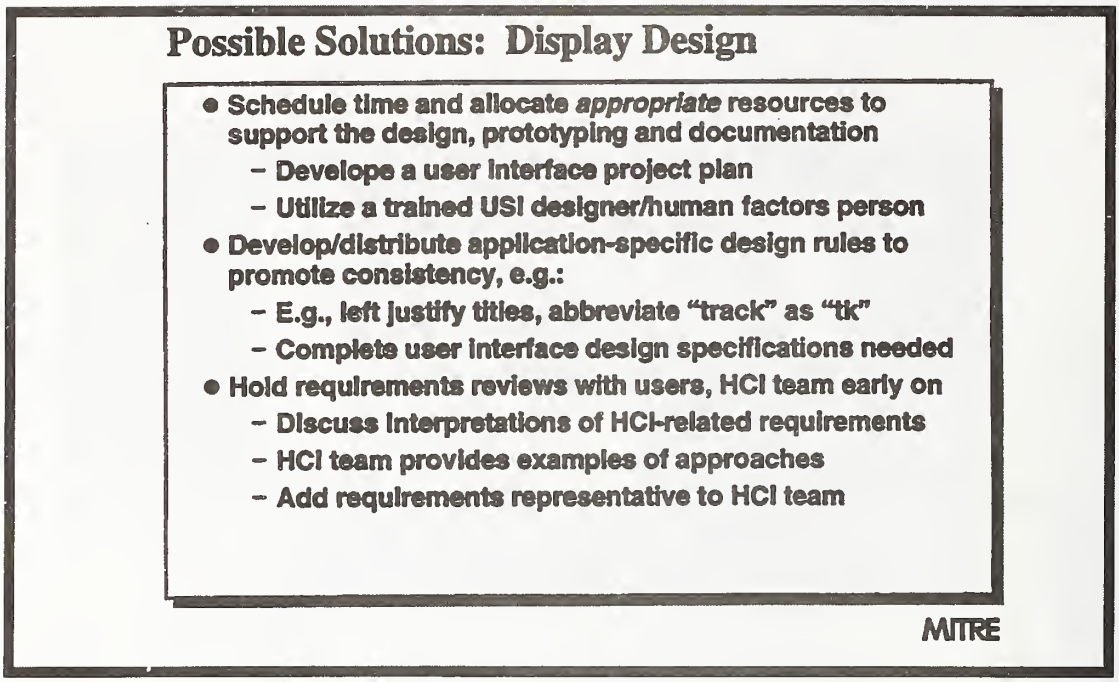


Figure 15

kind of set up didn't just come right out of the box. You couldn't just open up your standard Window Builder and get this kind of support in what you need to do. The point is you have to think about this for these kinds of systems.

So moving on to some potential solutions. (Figure 14) It becomes very important in these kinds of systems to schedule the time and resources that support the design and prototyping that you need and to have people who are trained in this area work in this area on the project. One thing that helps is the developers or designers coming up with specific design rules to promote consistency when multiple people work together. These little rules can seem really picky. For example, always abbreviate a certain word a certain way or always left justify titles. But, you need this level of detail. It is important also to hold requirements reviews with users and very early in the process to discuss interpretations of requirements that effect human computer interaction. To show examples of approaches to users so that users can express their preferences in terms of the full range of what is possible. The human computer interaction team should include a requirements representative.

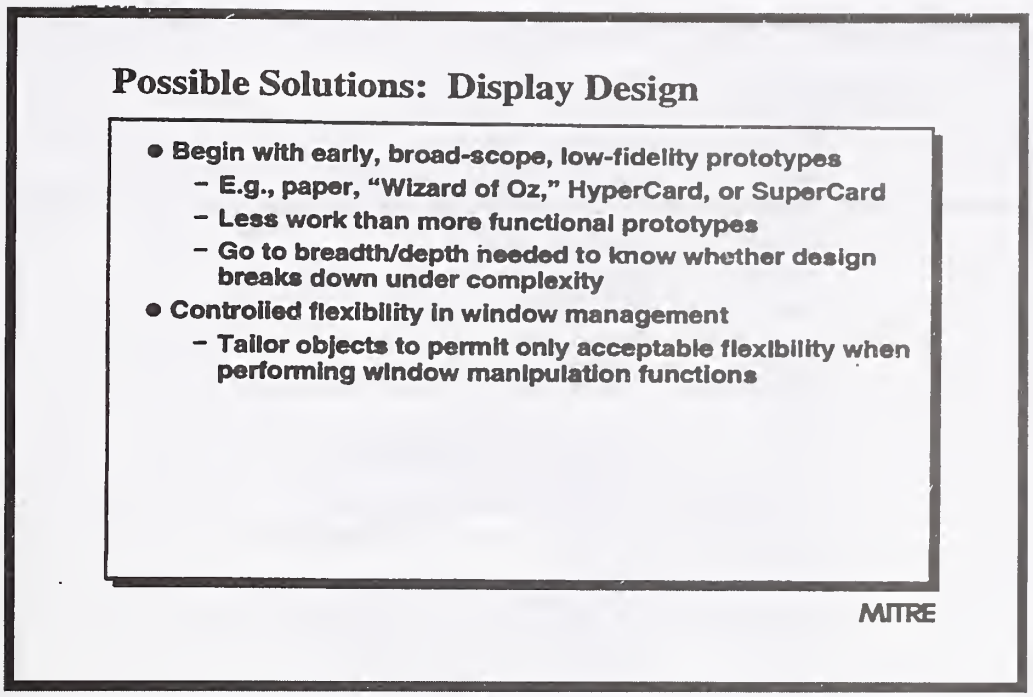


Figure 16

We do advocate starting with early prototypes and so called low fidelity prototypes that are great to start with as we talked about earlier. (Figure 15) Just going to the breadth and depth needed to be able to tell where the design breaks down due to complexity. We also advocate something that's called controlled flexibility in window management and that is getting back to the point of looking at your windows carefully and only allowing the user to do that which is reasonable under the circumstances. In other words, keep some windows from being closed or moved according to the kind of job those windows are doing and what the users are doing with those windows.

Window Properties Definitions

Table Field	Description
Resizable	Whether or not the window can be resized (Determines the presence or absence of resize handles in the window border).
Iconizable	Whether or not the window can be iconized (Determines the presence or absence of the iconify/maximize buttons in the window border)
Default size	The default size of the window.
Minimum size	If a window is resizable, the smallest size that a window can be resized.
Maximum size	If a window is resizable, the largest size that a window can be resized.
Maintain Aspect Ratio	Whether or not the window must maintain the original height to width ratio when resized.
Scale Contents of window when resized smaller than default.	When resized, a window can crop the contents, resize the contents of the window, or compress the contents of the window closer together.
Scrollable when resized	When a window is cropped scrollbars may need to appear. If so, then the window must be built in a Scrolled Window.
Modal	A window can be modal or non-modal.
Close Mechanisms	The method(s) that are used to close the window.
Contents of window contextual menu	The menu options that are provided in the contextual window menu.
Primary Background Color	This is the color of the window background and its children unless otherwise specified.
Window controls	Window controls may be menu items from a menubar or push buttons.
Default Button	This is the default button (if any) on the window.
Default Cursor Location	This is the location that the cursor initially appears when the window is opened.

Figure 17

Window Element Definitions

Window Element Characteristic	Description
Label	The text that appears on the screen to identify an window object.
Description	A brief description of a window object.
Type	Text field, Label, Option menu, Check box, Radio button, Push button, List, Text field with drop down menu, etc.
Format/ # of characters	If the window object is a text field, whether or not the text in the text field is alpha, numeric, or alphanumeric and whether the text in the text field is a fixed or variable length and the number of characters in the text field.
Static/Variable	The window object can either change (variable) or it does not change (static).
Editable/Non-editable; Selectable/Non-selectable	The window object can either be editable or noneditable. The window object can either be selected or it cannot be selected. It can be selected with a mouse, a keyboard, or both mouse and keyboard.
Mandatory/Optional	The window object must be addressed (mandatory) by the user or it can be optionally addressed by the user.
Default	The default value for the window object.
Color	The background and foreground color of the window object.

Figure 18

Properties of windows need to be defined very early on and everyone needs to be aware of everyone on the development team and the design team needs to be aware of some of these decisions that are made about windows, for example, whether they should be iconifiable, resizable, etc. (Figures 16, 17 and 18 show some definitions of properties, elements and control.)

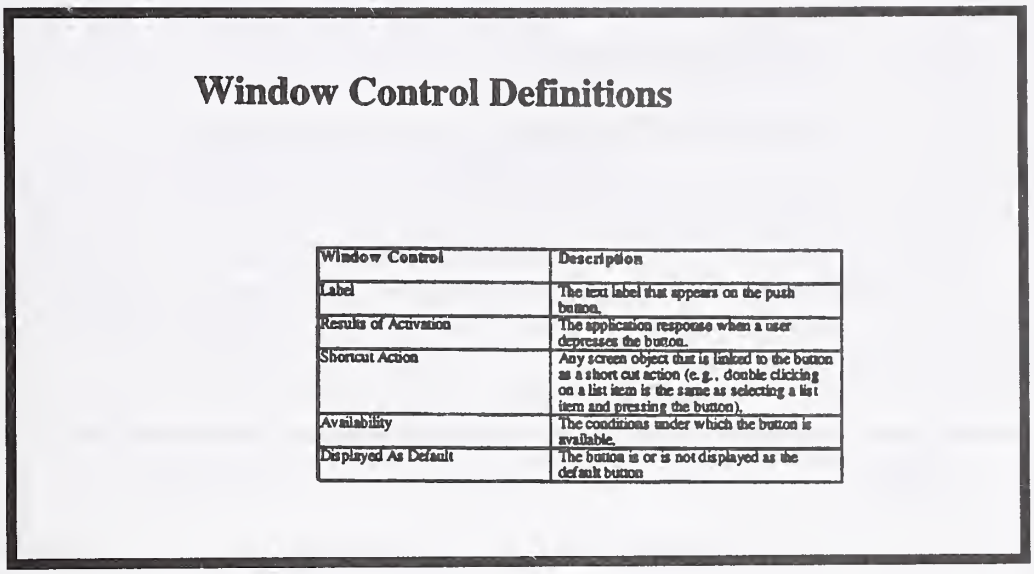


Figure 19

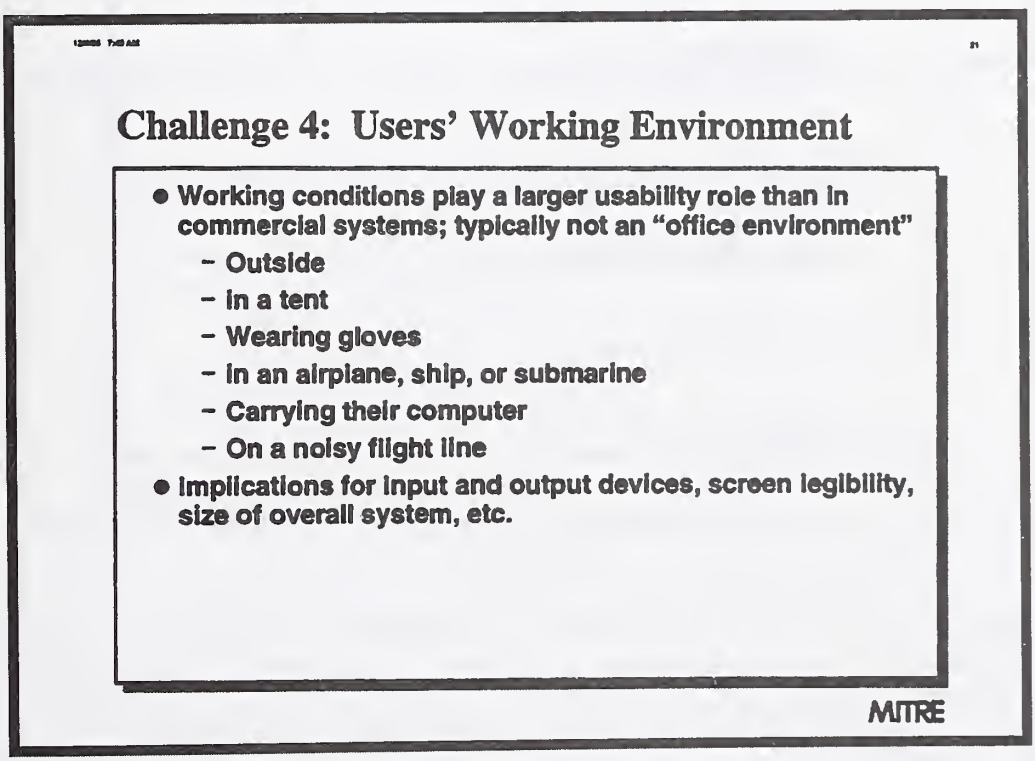


Figure 20

Challenge four has to do with the user's working environment. (Figure 19) I think you can appreciate that many government, especially military, people would be working in the non-office environment, such as outside in a tent or flight line. These decisions should be made knowingly and this information should be dispersed to the design team and the development team.

So these conditions just have to be taken into account, when the designing process occurs. We advocate attempting to quantify your environment conditions (Figure 20) and just look at the suitability of your various input or output devices in light of those conditions.

Challenge 5 (Figure 21) has to do with the development environment and we've heard a little bit about this so far in terms of people having to use COTS in their development. The project may also have to run under UNIX and Windows or maybe on workstations and personal digital assistants. It might also have to use certain builder tools and unfortunately the look and feel of the finished user interface may often be influenced by the builder tools. The style guides are out there, but they are just really not detailed enough to provide all the guidance someone needs when coming with the user interface.

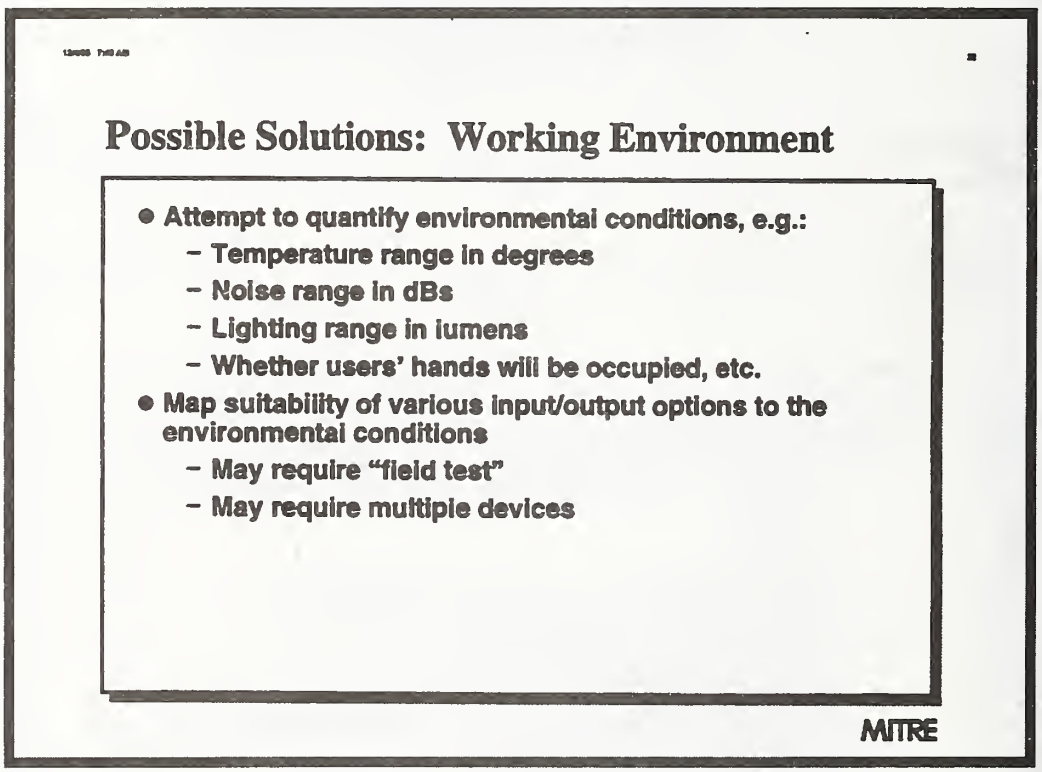


Figure 21

To mitigate the challenges of the development environment look at minimum configurations such as designing for a twelve inch display, if that is the smallest display. Look at all the platforms that you've got to work with. When using COTS applications, we advocate trying to find those that have interfaces that are customizable. When using an interface builder's tool, the look and feel should be dictated by the user's requirements and not the built-in capabilities of that tool. The most important point on this (Figure 22) is to maintain the user interface

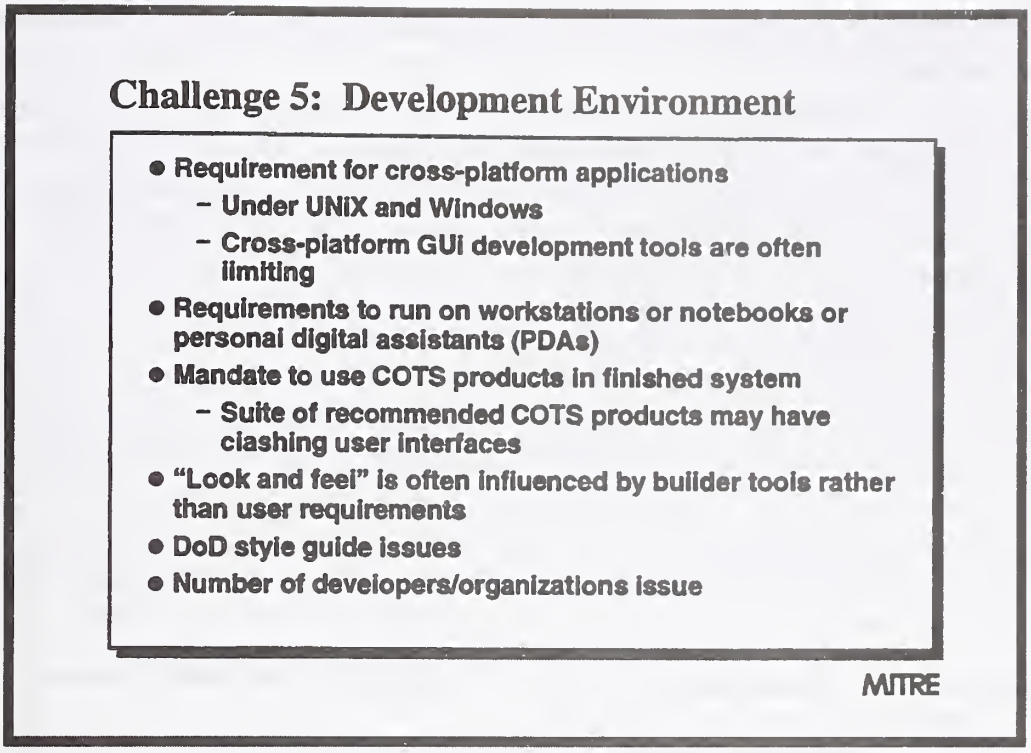


Figure 22

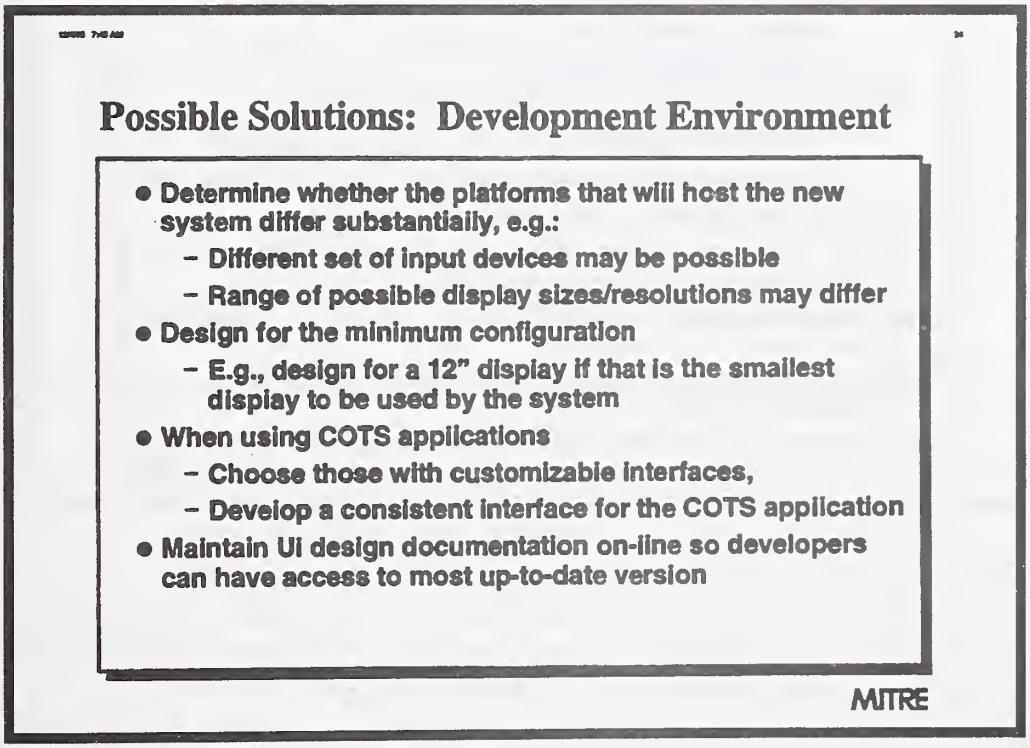


Figure 23

documentation, if possible on-line, so that everyone has access to a common, up-to-date version of the user interface documentation.

The last challenge, has to do with usability testing. (Figure 23) We found that project managers can find usability tests suspect if they didn't basically replicate key parts of the user's environment. For example, it is difficult to show how a user might use their interface under the stress of missile launch. Flight line environment might also be difficult to replicate. Also, I found that managers of projects can sometimes be leery of usability testing, because they are afraid that users are going to just keep asking for one more little change after another. They have seen it happen on a few programs to date and want to avoid driving up the cost. People have also seen high fidelity prototypes that are very detailed that are very realistic take a lot of time to develop and cost a lot. It is also difficult to decide some usability tests to show what expert system use may be like when these complex systems normally would take quite a while to learn under the best of circumstances. With a brief test you may be showing more about how quickly the user can be trained rather than how they will be working with the system every day. Likability is when you throw up a couple of charts of screen shots and everyone agrees they are pretty. It turns out that the colors would give them a blinding headache in two hours.

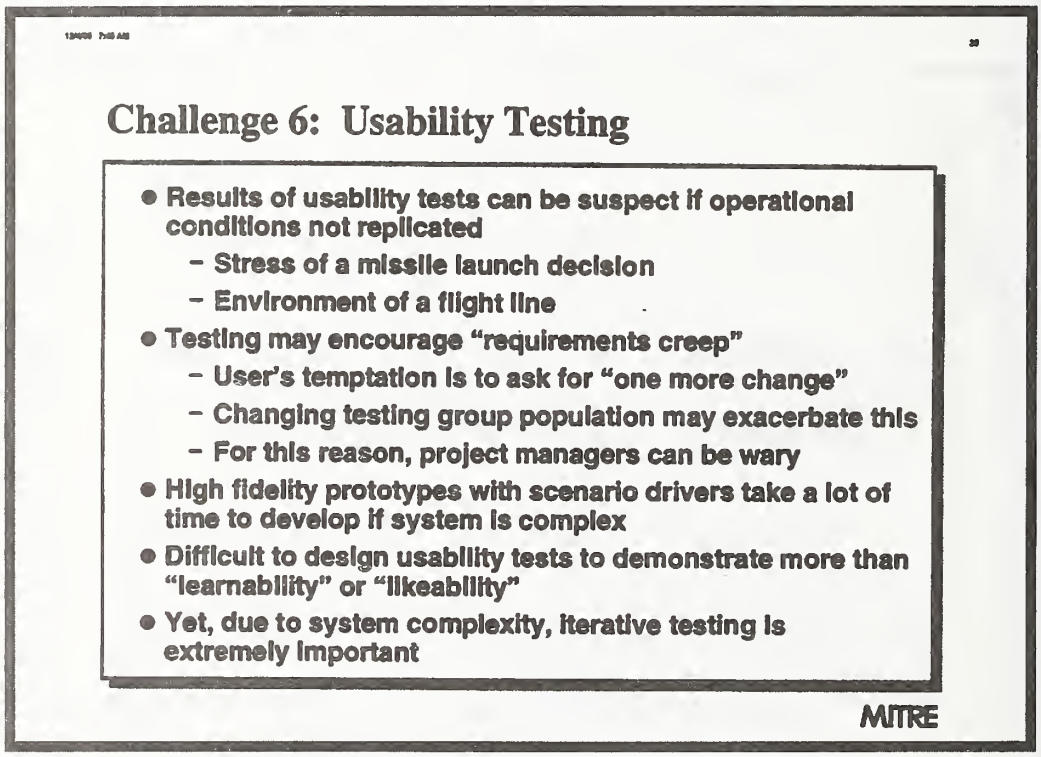


Figure 24

Despite the fact that there are difficulties with coming up with good usability tests for complex systems we really do feel it's important. We really do feel it is important and worthwhile. Leave time in the schedule for testing with users, incorporating their feedback. (Figure 24) Of course, this is best done when the program is just getting off the ground so you can have some input into the schedule. We found that even just quick feedback sessions with users can really help and that

you don't have to wait for the book or the report to come out six months later, especially if the developers are included in on the feedback session. They can really be thunderstruck at how users have had trouble with their creation. Another way to get around the possibility of the requirements—creating problem is to plan a fixed number of tests, make the users aware that these times are when their input is needed and where they can really make an impact. We suggest that you use high fidelity prototypes just for those critical functions. It is very expensive to use high fidelity prototypes, in general, but it is worthwhile taking some time to look at those functions that are really going to be critical and to maybe spend some more time making a prototype more realistic for those functions. This might involve replicating key parts of the environment. For example, we know at AT&T lab they actually have reproduced a city street including strategically placed speakers so that it sounds like a bus is going by 10 feet from you. It was that lab that they used to try to figure out whether or not people would be able to use their telephones.

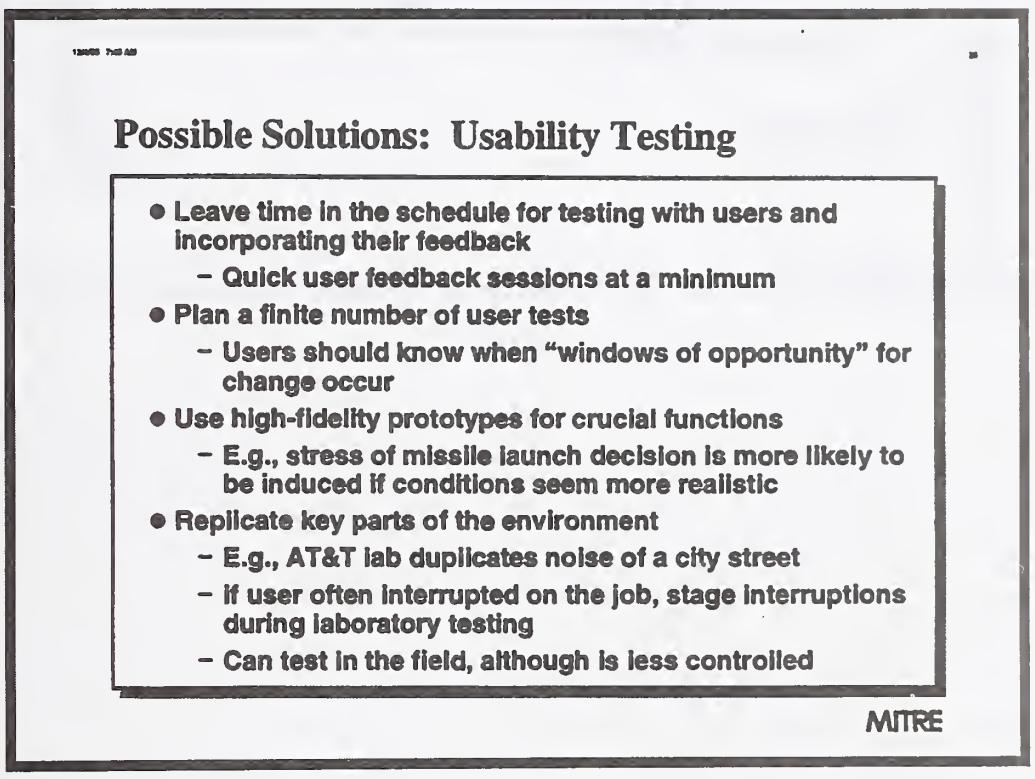


Figure 25

In conclusion, we think there are some real challenges (Figure 25) when the underlying functionality of the system is complex, that special efforts are needed to make the users a part of the system, helping to develop and evaluate it. It is definitely worthwhile to take some extra time to figure what that is in the beginning. The challenge is to figure out what would best benefit the usability of the system as a whole.

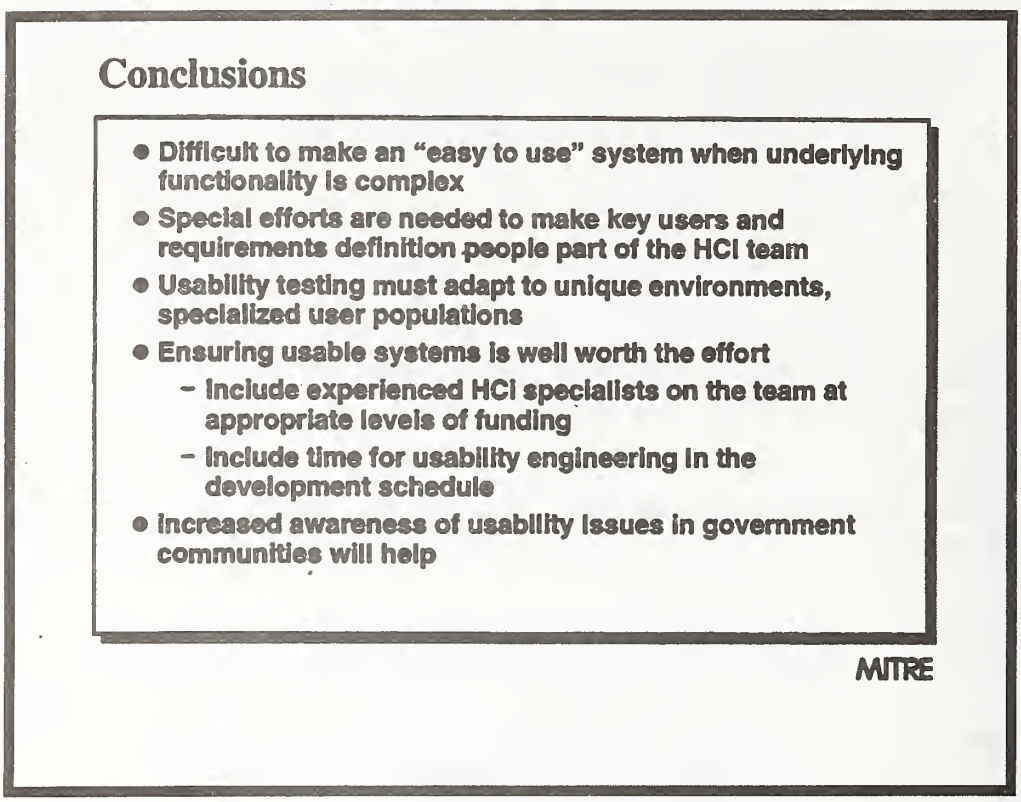
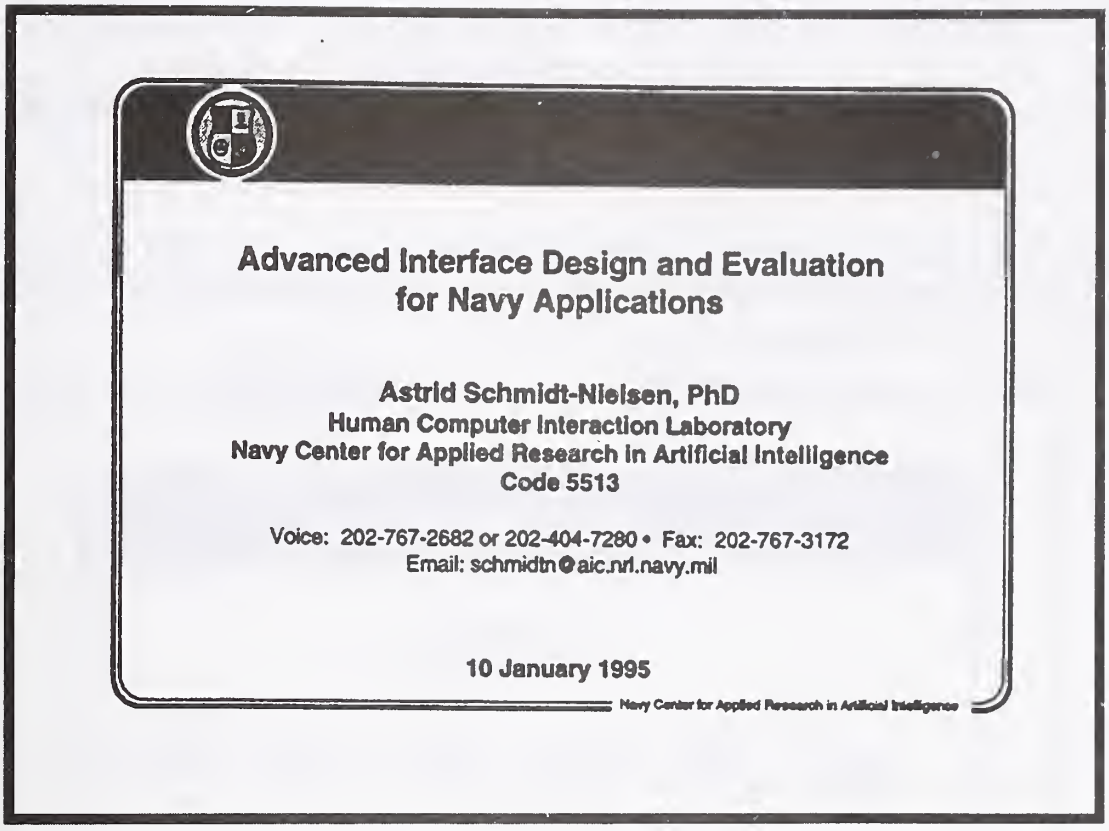

Conclusions
Difficult to make an "easy to use" system when underlying functionality is complex
Special efforts are needed to make key users and requirements definition people part of the HCI team
Usability testing must adapt to unique environments, specialized user populations
Ensuring usable systems is well worth the effort
Include experienced HCI specialists on the team at appropriate levels of funding
Include time for usability engineering in the development schedule
Increased awareness of usability issues in government communities will help
MITRE

Figure 26


Advanced Interface Design and Evaluation
for Navy Applications
Astrid Schmidt-Nielsen, PhD
Human Computer Interaction Laboratory
Navy Center for Applied Research in Artificial Intelligence
Code 5513
Voice: 202-767-2682 or 202-404-7280 • Fax: 202-767-3172
Email: schmidtn@aic.nrl.navy.mil
10 January 1995
Navy Center for Applied Research in Artificial Intelligence

Dr. Astrid Schmidt-Nielsen, HCI Laboratory, Navy Center for Applied Research in Artificial Intelligence
Advanced Interface Design and Evaluation of Navy Applications

We are designing new interaction techniques and new interaction methods with the cognitive, perceptual, and motor abilities of the user in mind. Our testing occurs before the new techniques appear in operational systems.

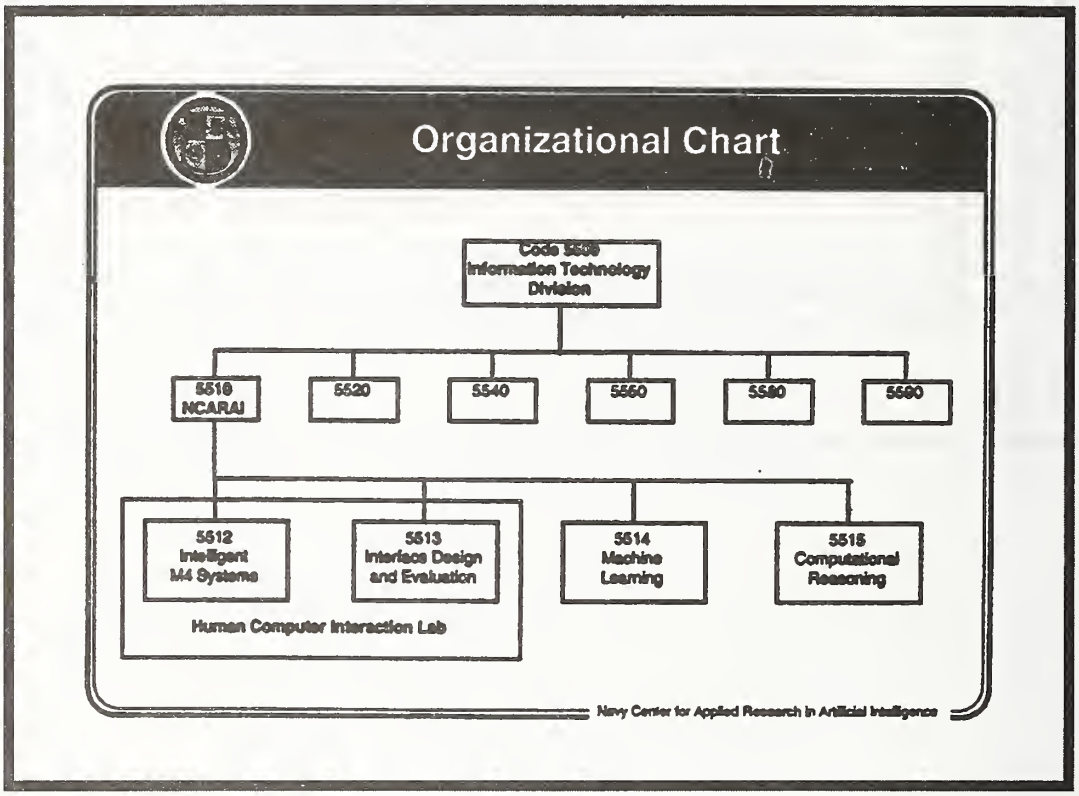


Figure 1

This is our division's organizational chart. (Figure 1) One of the reasons for showing this chart is that when I came to NRL, I was the only psychologist there. There were psychologists right after World War II, but the human factors group was disbanded in the 60's. Under the Navy Center for Applied Research and Artificial Intelligence we have two sections: the Natural Language group with multi-media interests and the interface design and evaluation section which has a mix of psychologists, computer scientists, and linguists. It is an excellent environment for doing interdisciplinary work of the kind that we're interested in. This figure (Figure 2) lists our staff members.

I will concentrate on describing the first three topics from this list of six. (Figure 3) Our philosophy of Multimodel Interaction (Figure 4) is to use people's sensory-motor and perceptual channels in the best possible way to give natural feeling interactions. So we don't want to overload one channel. We want to give people a chance to use some rather unusual channels.

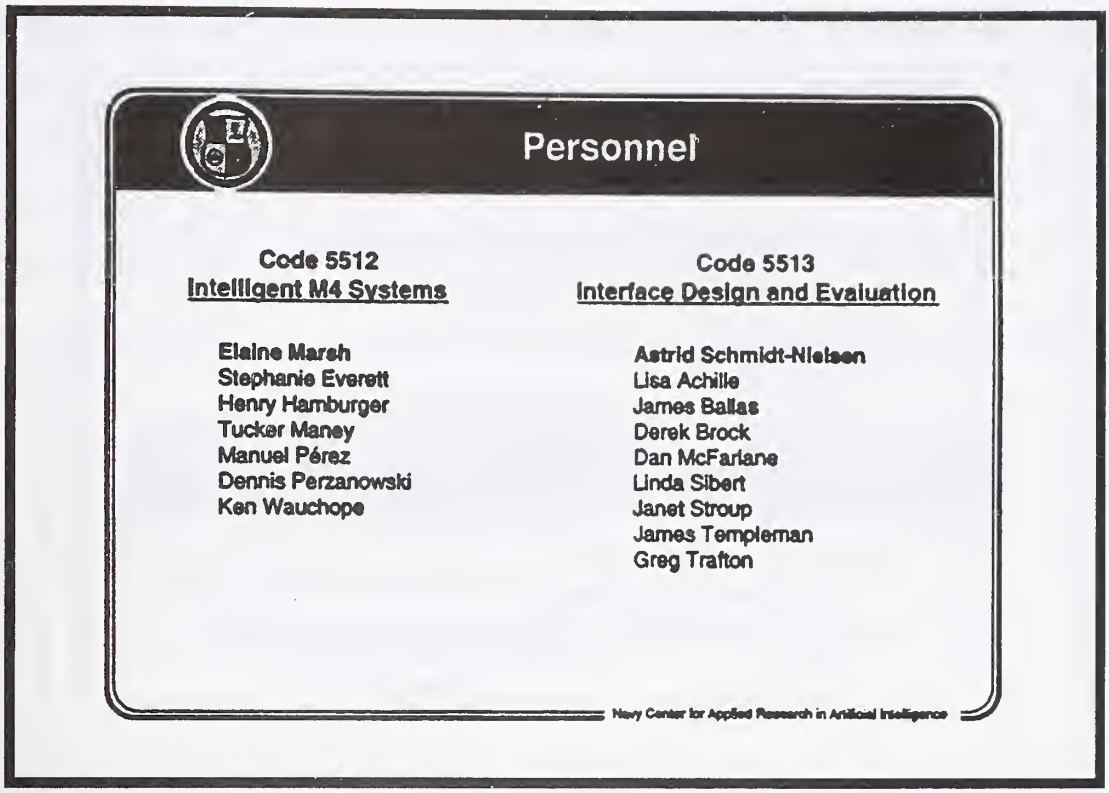


Figure 2

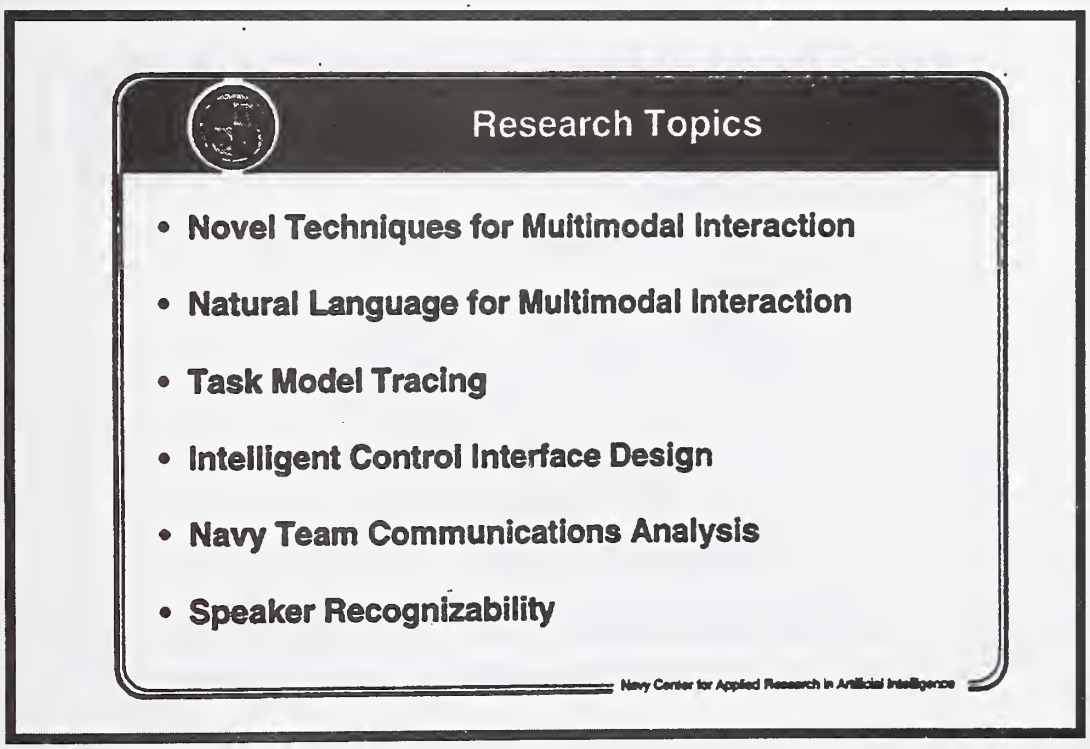


Figure 3

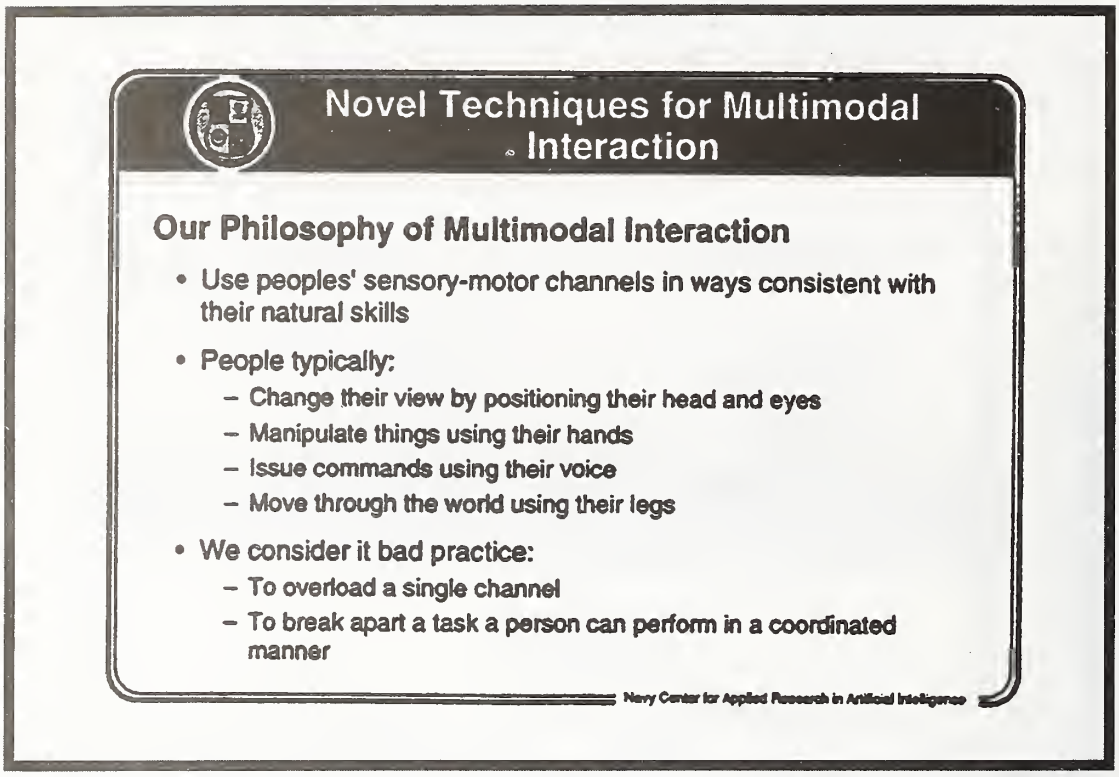


Figure 4

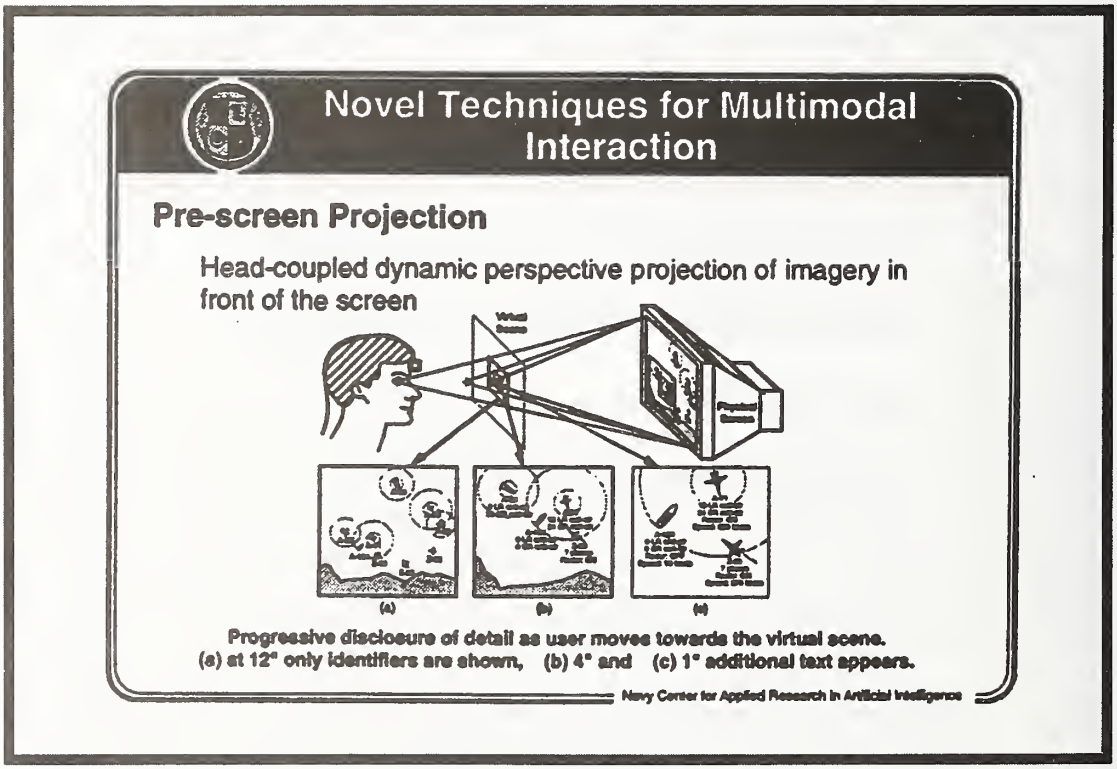


Figure 5

In this particular case (Figure 5),we are interested in how people navigate and maneuver through virtual environments. We have developed and are currently testing an interaction technique called Pre-screen Projection. A person wears a tracker on their head and as they move their head back and forth, the screen pans and zooms in a natural way allowing the person to move their head to change view. It's also somewhat faster, so if you move your head in not only do you zoom in, but you get greater detail. You might get information that is not on a map: you get information about the ships and planes on the map. This is currently under testing.

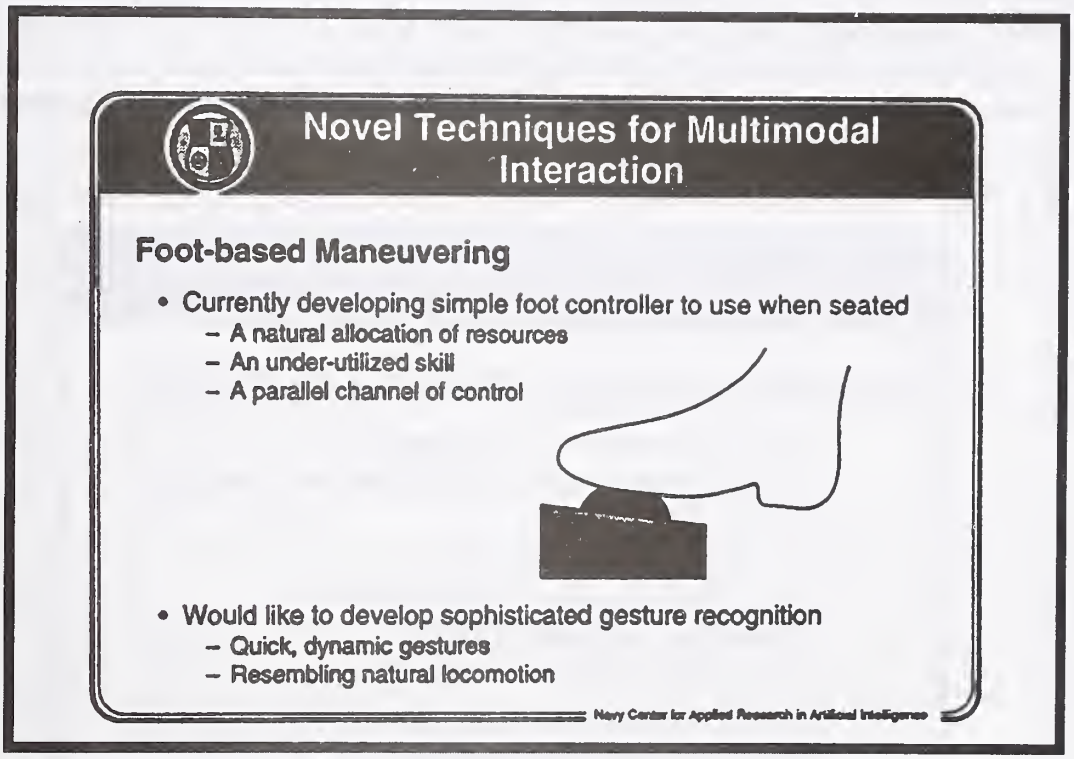


Figure 6

We are looking at some new techniques for maneuvering. (Figures 6 and 7) We're going to be looking at how people can maneuver and control their environment in 3-D and in virtual environments in 2-D. A lot of the devices that are being used now like the mouse and the track ball are really devices that you would like to use to control other things besides moving through data. So if you try to maneuver naturally through an environment you first move forward then you find the object then you stop and use the mouse to grab it or pick it up or whatever.

What we'd like to do is develop some other control methods (Figure 8) like the Mongol warrior who is on horse back and sighting the enemy and pulling his weapon and aiming it all at the same time rather than moving forward and then doing this. Now you have to grasp the bow and arrow with the mouse. So we're trying to develop more natural methods of maneuvering through these environments.

Figure 6 embodies a lot about iterative design. We are doing is developing a taxonomy of ways in which people can maneuver and input devices and different ways they can handle them as they relate to the user's capabilities. One novel technique that we're going to be developing is foot-

based maneuvering and Jim Templeman has been working on this for some time now. He actually has begun to develop a prototype foot-based device. This is a very natural device, because people use their feet all the time. We're looking at ways in which people can control things in concert. Feet are a natural thing to use in concert. You use them to drive your car. You use your feet to walk. We don't have to simulate walking, people are very good at learning controls like skiing or skateboarding or many other ways of controlling things with their feet. What we really want to do is to start to taxonomize the tasks, find out how people can use their feet naturally to classify the kinds of motions that are natural and what they are likely to mean to the person. For example, you can find out whether pressing forward on something might mean move forward or that sliding a slider might be more comfortable.

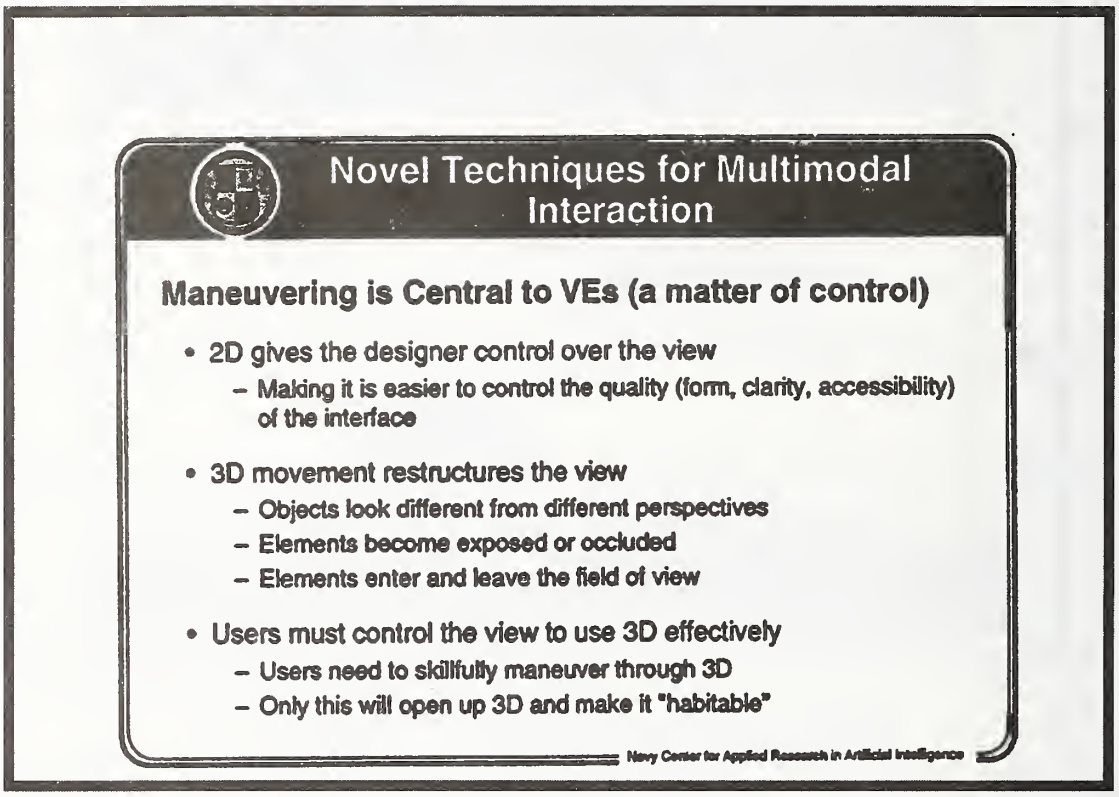


Figure 7

At this point we already have a prototype foot based control. We're looking at different mappings. This particular one has five degrees of freedom. We can look at which mappings are best for which kinds of control and how to apply two feet in a coordinated manner. It might be that moving the feet apart would mean one thing and moving them together would mean another or moving them sideways. So we're looking at which kinds of actions and motions and controls are going to be most natural for controlling particular motions through environment. Another issue that Jim Templeman has identified is that if you can have opponent sensors what happens if you're pressing forward and backward at the same time. Does this hold the image? Does it slow you down? Does it speed you up? What kinds of things do you get when you get coordinated actions?

Our next one is some of the sensor technologies that are out there. The magnetic trackers that are used the pressure sensors a lot of these are currently used more for finding out how athletes move and that kind of thing. They certainly exist and a lot of the earlier foot controls like treadmills and so on still require the hands to steer them. A lot of them have just been a peddle like essentially a foot based mouse which is not what we're interested in.

We are linking the auditory and visual feedback. (Figure 9) There are other people on the project who are interested in things like gesture recognition. We're also looking at designating sets of objects and what kinds of things are best for designating and manipulating sets of objects as opposed to just a single object. At each step we're going to do the kind of empirical iterative evaluation that everybody has been talking about, but we're at a much more basic level of what we're developing than the kinds of systems that I think have been talked about.

Now I'd like to discuss Greg Trafton's work on task model tracing. (Figure 18) Task model tracing is used to facilitate user interactions by context sensitive recognition of repetitive actions and then offering to help—useful for intelligent tutoring system. In general, I do not like adaptive interfaces. You don't want the system changing with you, but this is more flexible than just developing a macro to do a repetitive task. It's a system that learns with the user.

So what it does is trace what the user is doing in a context sensitive way. It follows the user in order to be able to anticipate future actions. It can also incorporate a model of the task, but it doesn't tell the user what to do about the task.

What Greg is using to do this is ACT-R which is a computational model of how people solve problems to follow the user's actions and as the user is going determine what kinds of things that the user is trying to do. ACT-R has a rational analysis system and it allows you to select among alternatives with a satisfying rule. That means you don't have to have the best possible solution; you have one that satisfies what's going on. As a cognitive model, this one was selected rather than a GOMS model or some other because it has the right granularity for the kinds of tasks that we would like to apply it to.

We plan on demonstrating it using a synthetic stripe planning task (Figure 19) where we model the task and then trace the user actions. Now ACT-R has been used in tutoring systems where the tutor knows for example what the solutions to the problem is, what the right answer to the task is and it guides the student and if the student goes off course it offers the student advice. This is very different. The model does not offer you advice it learns as the user does repetitive tasks and offers to complete tasks if it sees that it can do it faster than you can. It doesn't guide you. It does what you would have done and you can choose to use it or not to use it. So it develops anticipations, but they are context sensitive which is the advantage of using a cognitive model like ACT-R.

Models can be used to anticipate user's actions and then facilitate the actions where the model can do it faster. While it can trace certain sequences, it is best for highly repetitive tasks. I mean if you're trying to use it as an architect to do house designs there's too much creative input, but if you were using it to program your VCR it could begin to trace things where if you did the same sequence every night it could anticipate and do that sequence for you.

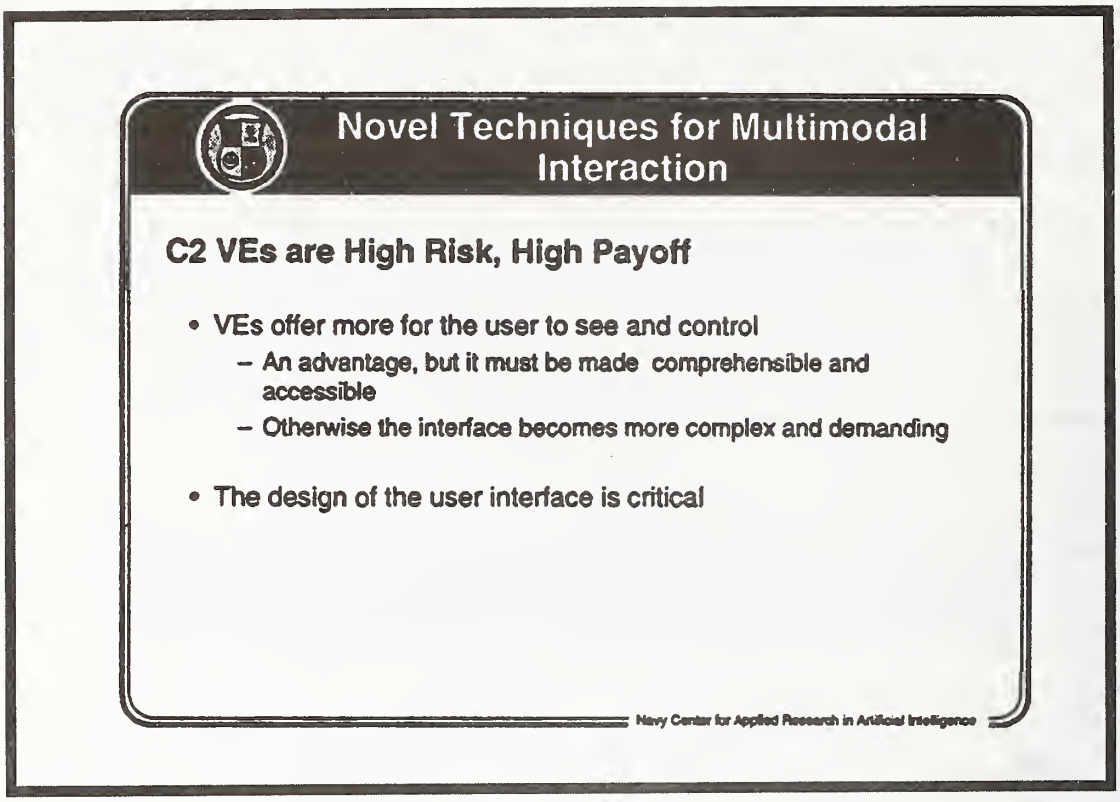


Figure 8

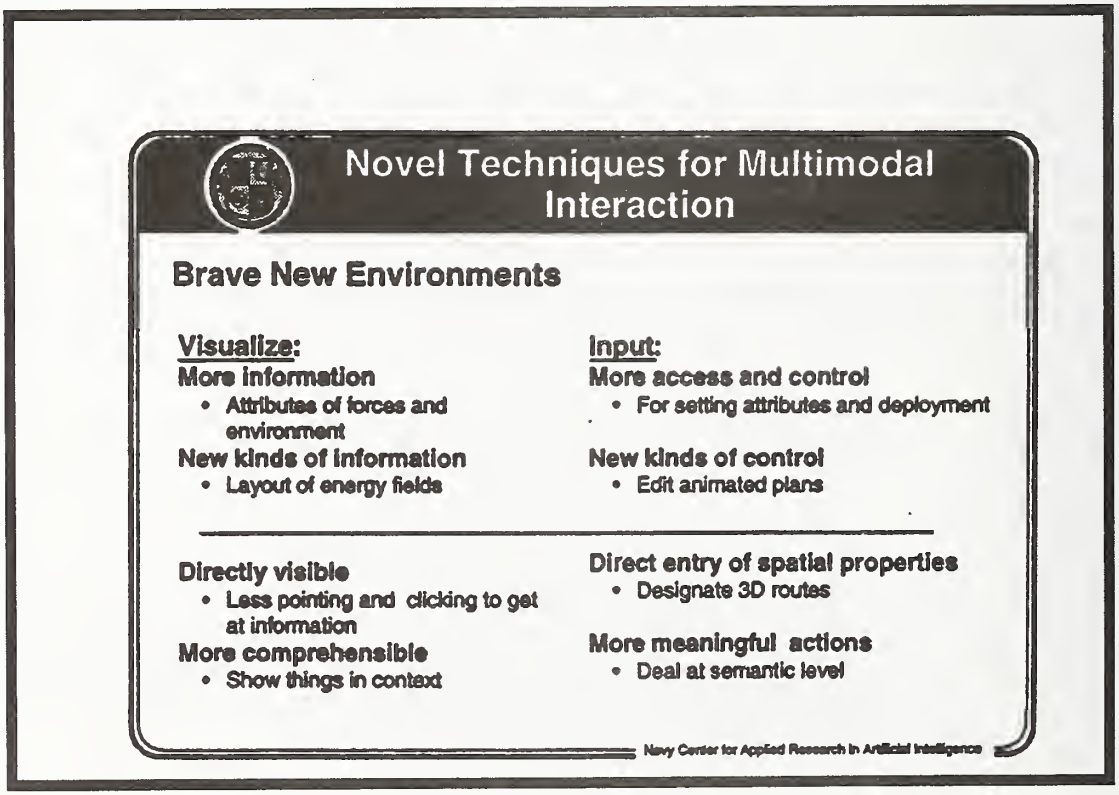


Figure 9

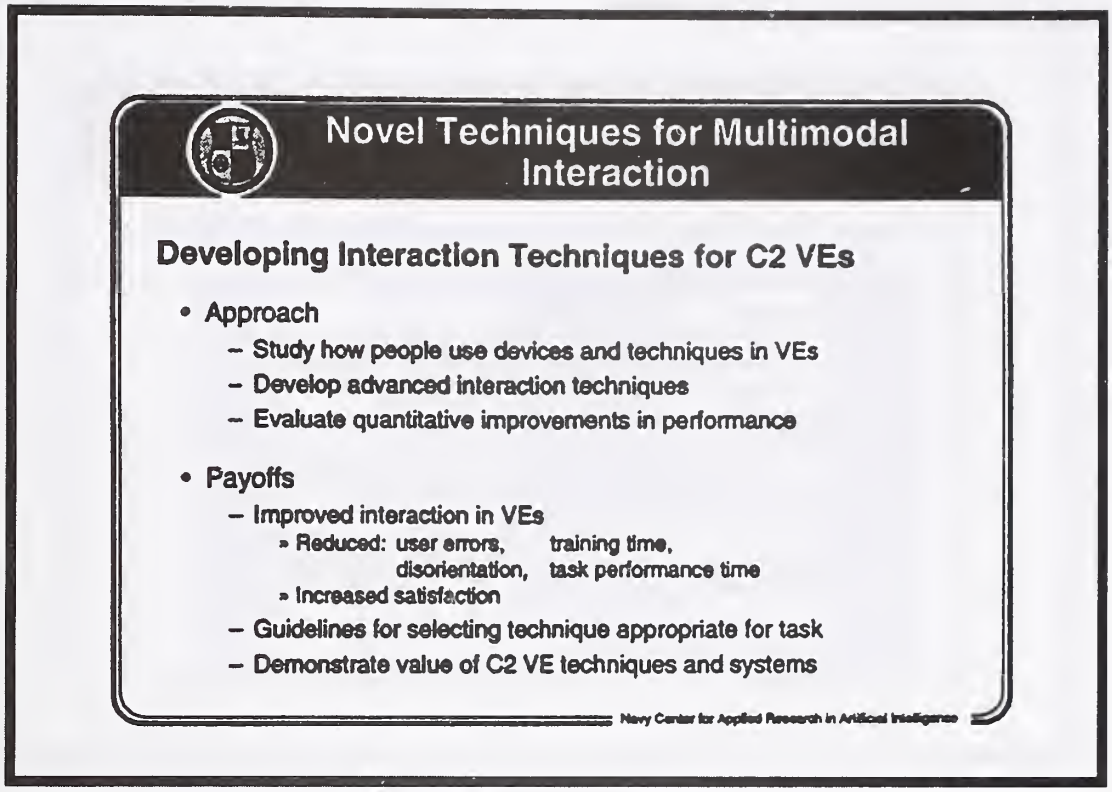


Figure 10

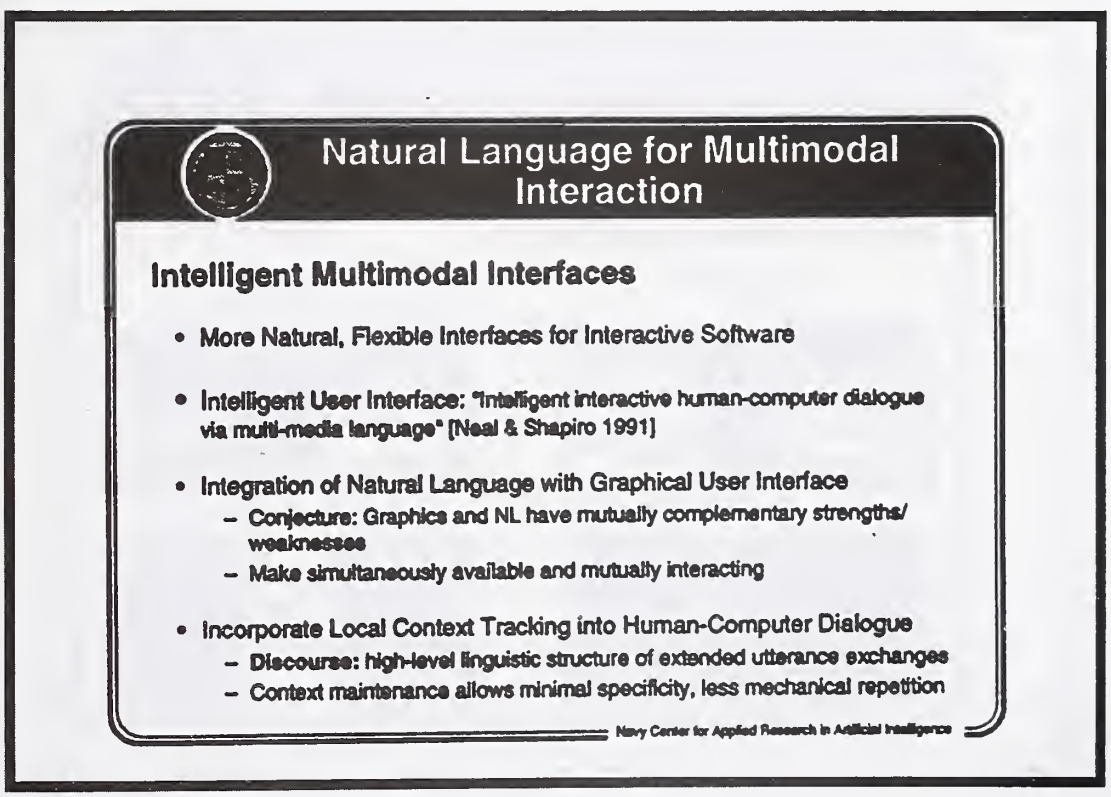


Figure 11

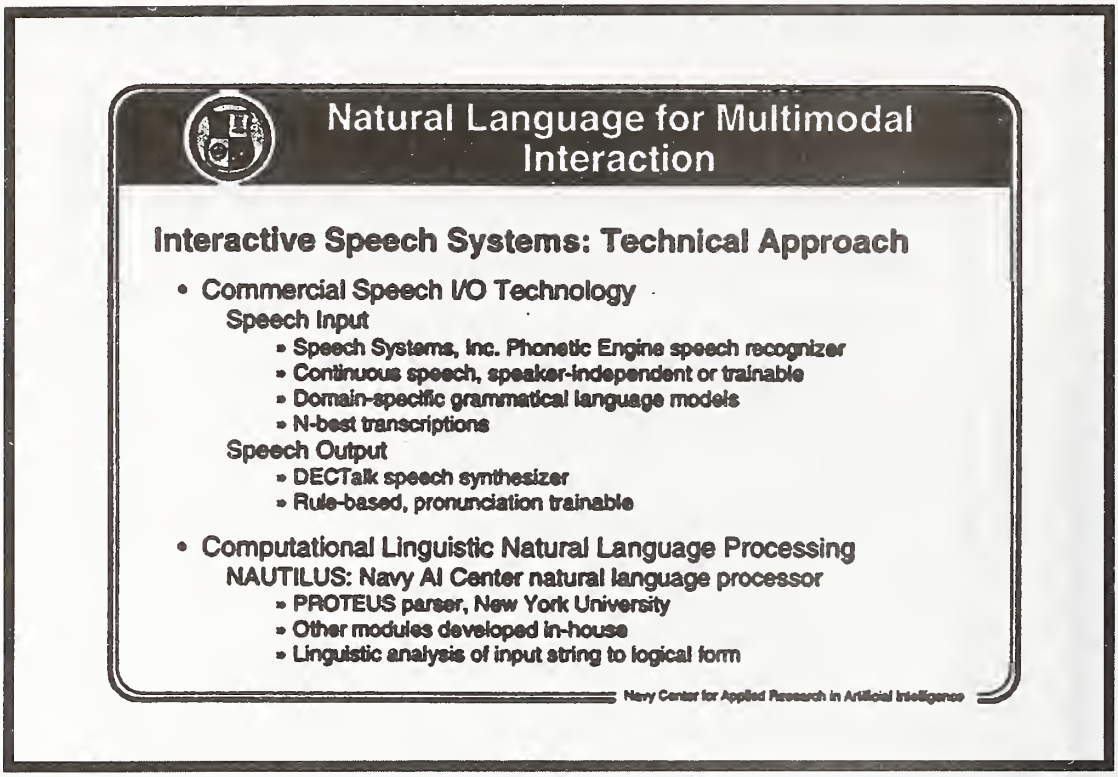


Figure 12

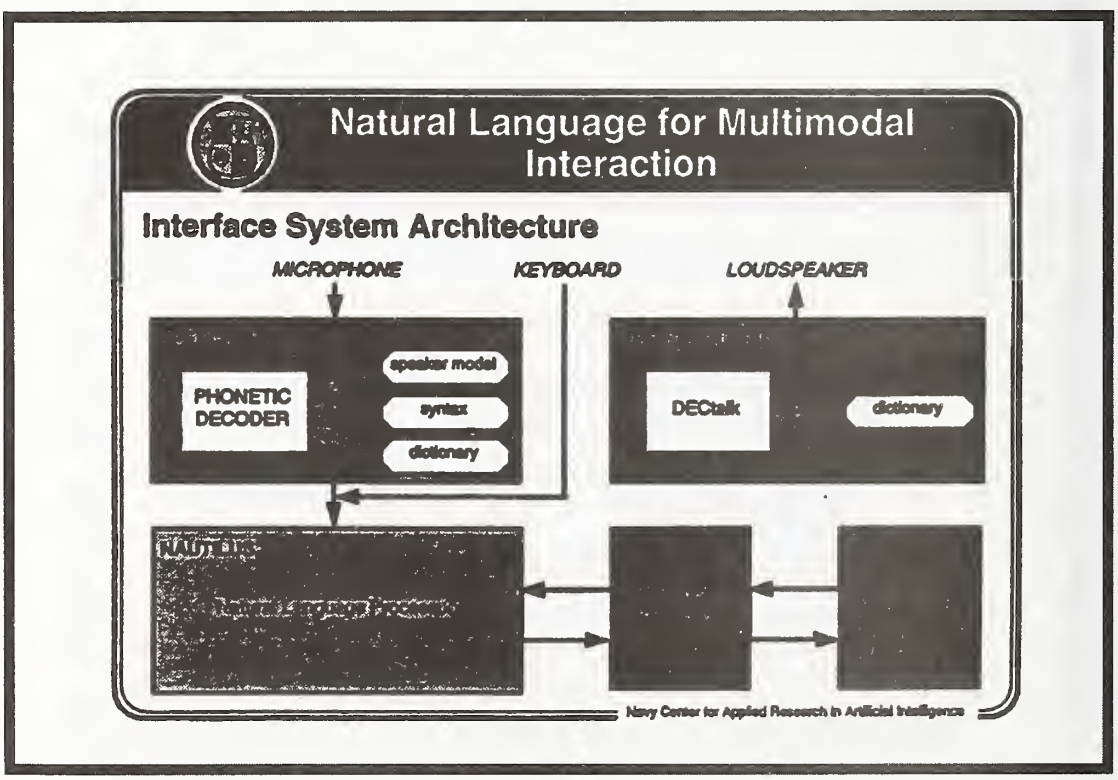


Figure 13

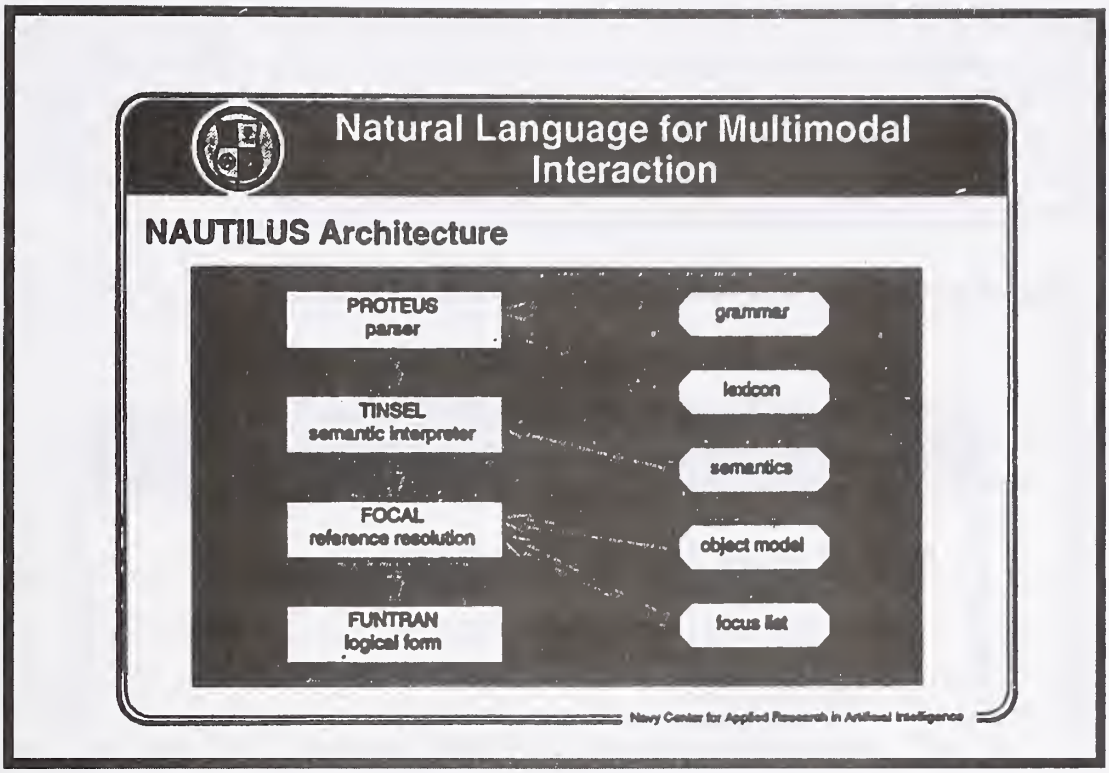


Figure 14

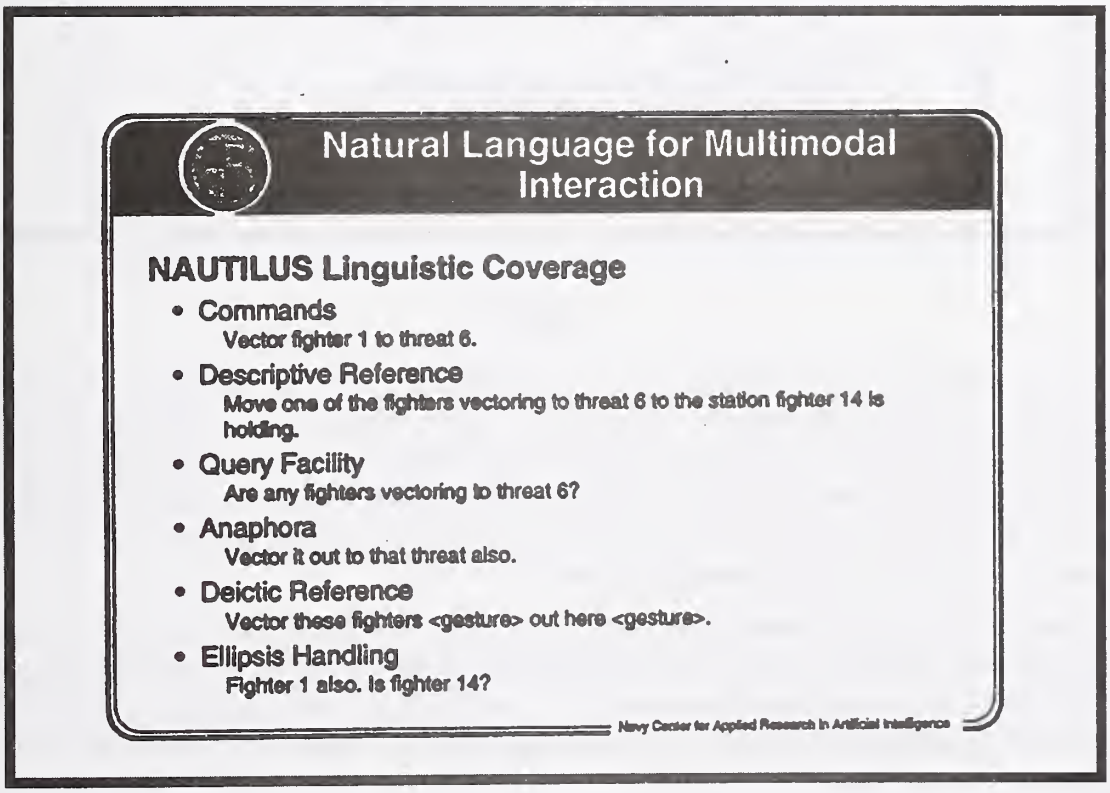


Figure 15

The third area is really a success story, but it is a success story that will allow us to continue developing maneuvering techniques for virtual environments. In this particular case, the virtual reality lab at NRL, which is a different group from ours, had developed a model of part of the U.S.S. Shadwell which is a fire training ship that is now stationed in Mobile that is used as a fire training research ship. They have fire fighters come in, and they do contained fires to learn about how fires spread through ships and how best to control fires in ships.

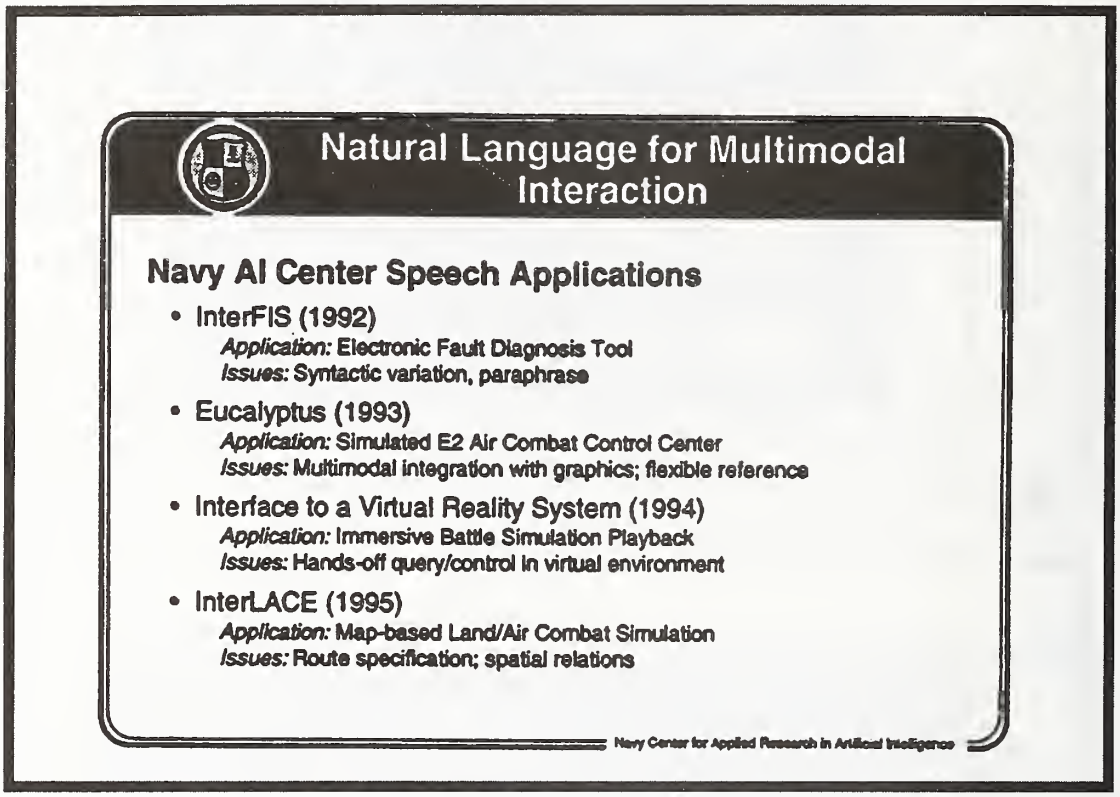


Figure 16

One of the things that we've done is a very quick evaluation. Note that the experiment has some design flaws which always happens when you're doing something in the real world. We used this model of the Shadwell to train fire fighters before they went in. So the two major problems with fire fighting that people have know about are how to behave in low visibility and knowing the ship. Half a group of firefighters trained and prepared as they usually do by looking at diagrams of the space that they were going into.

The other half did a walk through in the virtual environment with simulated smoke and then the two groups went through two tasks. At first they had to navigate to an unknown location with low visibility and in this case there was no fire. The way they simulated the low visibility was that they put a low visibility screen over the oxygen masks so that you basically can only see three feet in front of you.

What they found was that the people who had the virtual environment training got through 30 seconds faster than the other group. So that's a really big improvement in their performance in finding the location. In the second task, the groups were actually going to a simulated fire so

there was real smoke and they were going through the part of the ship. In that one, as you can see the VE trained people made no wrong turns. They felt that they knew where they were going. Their comments afterwards confirmed that they thought they knew the space. The training really made a difference in the two groups. I think the real advantage to having made a real life demonstration is that we can now go on to show that if we can get improved navigation methods and improved maneuvering methods and improved interactions techniques for virtual environments, this will benefit Navy training in noticeable ways.

I would like to acknowledge all of the other people who were involved in the virtual Shadwell experiment besides our own group. (Figure 35)

Here is a list of some of the other areas in which we're working on interaction methods and evaluation for various Navy problems. There are more than that, but if you want to ask about one of the things I haven't talked about and if we have time for questions you can ask about these too. Yes.

Question: How much time occurred between the training and the experiment?

In this case the virtual environment training was right before and the feeling was instead of just seeing the diagram they really knew the space. It's not really a realistic thing because this is an experimental ship. These people volunteered, they came in. Each ship is different, except they don't learn that particular one, but you could assume that a fire fighter on another ship might already know the ship, but if you're just doing the fire fighting training then it certainly speeds up their learning of the things they need to learn to get there quickly.

Ms. Allyn C. Dillman, National Assistant for Training for the Professional Airways System Specialists Union (PASS)
The Role of Users in the Development of Computer Systems for the National Airspace System

Everyone else that has gone before me today has made my job much easier by telling you why you need to include the users in your system design. Now what I'm going to do is introduce you to the user and talk some specifics about the diverse users that are in our particular industry and some of the problems we have encountered in the Federal Aviation Administration with new technology coming down the pike. Not all of my examples will be specific to software, but when it comes to system design a lot of the problems do cross the software and hardware lines. I am with Professional Airways Systems Specialists (PASS).

We are the labor union that represents 10,000 Federal aviation employees. We represent the people in the airway facilities, flight standards and aviation standards organizations.

Our airway facilities systems specialists, electronic technicians and computer operators install, maintain, repair, operate and certify over 31,000 facilities and equipment systems in the national air traffic control system. These systems stretch from Deadhorse, Alaska to San Juan, Puerto Rico, from Bangor, Maine all the way over to Guam. Our flight standards aviation safety inspectors are the ones who grant the pilots that fly in the system their license. They grant the license to the mechanics that maintain the airplanes and they certify that those airplanes are maintained in the safest standards possible. Our aviation standards procedures specialists are the ones who take the FAA airplanes out and flight test the navigation systems to make sure that they work like we believe they are going to. They are the ones who come up with the flight procedures for the pilots to follow. PASS also includes the clerical and administrative support positions for the FAA. Every employee represented by PASS deals with a multitude of software systems every single day that we go to work. I mention the air traffic controllers; they are our brother and sisters in the FAA.

We have a very close working relationship with both NAATC, the National Association of Air Traffic Controllers, and NAATS, the National Association of Air Traffic Specialists. The importance of having a cooperative working relationship out in the field is demonstrated by the fact that with the current FAA reform movement underway these three unions have bonded together in a coalition to help the FAA get it is personnel and procurement act together. Between the three unions we represent 30,000 of the FAA's 48,000 total employees. All of the people in these unions. All the people that work for the FAA are dedicated professionals who want to sustain the safety of the most efficient air traffic system in the world. We move 30,000,000 aircraft per year through that system. Computers are the way we do it. Air to ground communications, radar systems, controller data displays, computerized cockpit systems that the pilots deal with. All these use a multitude of software programs, too many for us to even try to count. Our payroll and personnel systems, our training systems, technical LAN management, our field communications. That all depends on computers and when we get a bad system in the

field that doesn't work and we have to spend extra time and money trying to make it do our job, it makes us less productive and less effective and that wastes you, the taxpayer's money.

People are a basic component of any computer system, just like the hardware and the software and we are the people. The success or failure of any system can depend largely on management and labor being able to interact with that system. There are systems that we can just shut off if they don't work, count the money as gone and leave. There are other systems out there in that air traffic world that we have to figure out how to work around. That makes the controller's job and technician's job a whole lot harder. Private industry has determined that it's not cost effective to leave us out of the design phases of software or hardware systems. They are asking their workers that put quarter panels on cars, how can you do your job better and faster? If we put in a new computer system, how can it help you? We, the ones who are responsible for the public's flying safety, deserve the same consideration.

Let's talk a little bit about loss of productivity. To give you a minor example in the big picture, last year the FAA went to a cc:Mail program for our interagency communications. The person in charge set up the LAN and gave us another specific communications program to dial into that LAN as outside users. I spent four months fighting that. He would not listen to any of the other users or myself that complained it was not working, we can't stay connected for more than 5 or 10 minutes, we need to get another communications program so that we can get our cc:Mail when we're on the road or when we're at home. He didn't want to do it. That was his decision and that was what he wanted. After four months, I was lucky because of my national position in the union I said, I'm not fighting this anymore I've got to have cc:Mail so I can do my job which is to communicate with FAA employees in management on the issues we're working on in a fast and efficient way. I pulled my cc:Mailbox out of the local LAN and had it set up in the regional LAN in Atlanta. Now if I need to distribute a letter to 50 people with the FAA I can do it in one city. I'm not spending three or four hours just trying to get it to go somewhere away from my computer. That also incurred additional costs because now I have to make long distance phone calls to Atlanta to upload and download my cc:Mail instead of just the local call. As I said, that's a small inconvenience in the big picture of the FAA.

Moving up the spectrum of cost and inconvenience is the FAA's CBI program, Computer Based Instruction. Several years ago, the FAA decided the leading edge of technology: Computer Based Instruction. They went out and poured a lot of money into a "playdough" platform. They didn't take any end user input when they designed these courses. They put it out in the field and said here it is. Field managers decided that the additional down time that technicians were spending trying to struggle their way through these courses because they were so user unfriendly. wasn't worth it. They went ahead and sent the users to Oklahoma City for resident training at our FAA academy and just put those training programs on the shelf. Another problem with those training program were that they were very, very boring for our technicians that are used to complex computer systems. Because of that they might get through and check the boxes to finish the test, but that didn't mean they really learned it. Then they had to spend extra time in the subsequent classes catching up. Again, this is going to cost the FAA additional money.

A lot of these playdough courses have been targeted to be redesigned, and right now, the estimated cost is six thousand dollars per class hour to change that technology over again. So if you take a 40 hour CBI course that needs to be reconverted, which is very, very short for our AS

technical training courses, it's going to cost another 240,000 dollars on top of whatever it costs in the first place to convert that course. Right now we have around 75 courses they've targeted. Now the real problem here is Airway Facilities was only given 6 million dollars for their technical training budget for 1996. That was 31% less than our calculated training requirements said we needed. Not counting the cost for the course redesign we need to do. That boils down to the fact that we have 300 technicians we hired last September and October that don't have the money to go to resident training at our academy in Oklahoma City and they don't have the money to take the computer courses or redesign the computer courses that we can't use. So we don't even have the good money to throw after the bad and that goes across all agency lines in the Government today.

Added to the loss of worker productivity and costs of another 240,000 dollars are the ramifications of our particular industry. Only a few industries have the safety considerations that the FAA has. As someone put it earlier today, when our systems don't work people die, airplanes crash and that's what we don't want to see. That's our number one concern as a field employee is to keep that from happening. When a system outage happens, as you've heard about in the papers recently, controllers have back up systems they go to and they go to those back up systems and they work, but the controller can't handle as much traffic. Therefore, airplanes are held on the ground or rerouted until they can go into that controller's airspace. These delays have widespread economic impacts on the airlines and other business. It cost the airlines thousands, almost millions, of dollars to have those airplanes sitting on the ground unable to go and it costs you all, the business travelers, money and time when you can't get to where you need to go. We will put up with those economic impacts, because what we won't let happen is those planes go into the sky before it's safe for them to.

What happens when an FAA system goes down as it did in Chicago Center? Well, not only does the news media show up, but then we have Congressional hearings and then we have IG reports. So it's much better to give us a new system, or a system that works right in the first place than to go back and have someone scrutinize why it didn't work because everybody gets crucified then. Because of the high value that we in the field in the FAA, in general, place on safety, the field employees are taking a proactive stance in getting involved in system design, hardware, software, whatever. We don't want to see the same problems with the new systems that are going to come into the field to replace this aging equipment, and yes, folks, we do have systems out there that have vacuum tubes, because we're way behind on the new technology. Too often in the past with the systems that are already operational we've been handed systems that took us one, two, three or more years to get to work right because they were handed to us and they said here it is, use it, this is what we've got today, we're going to get you another new system in five or six years. Well five or six years never comes.

Let's look at the difference between when we got those old systems and 1996. In 1991 Airways Facilities (AF) had 19,000 systems to maintain out in the field. We had 11,000 technicians. In 1996 we have 6,000 technicians and we have over 31,000 systems. Some of them are more sophisticated and they don't require as much technical attention, but, as I said, we do still have some out there that are the problem children. We do not have the time, the people or the money to mess with systems that come into the field that don't work right. In the FAA, the situation is improving because the employees have taken a proactive stance and because management has

figured out that we need to know exactly what an air traffic controller needs to do his job. We need to know how the air traffic controller and the airway facilities technician interact in the field to get the job done. We've started working with partnership. This means that we now have national representatives on every new system that's coming down the pike. A national labor rep for PAS and a national labor rep for NAPTA to give them that field perspective. Will it work? Can the controller do his job? Can we, AF, keep it going or get it going again if it does stop? This is what they have to know to design the system.

A system that is a good example of how critical our involvement in the initial phases of system design are is the voice switching communications system, also called VSCS. This is an air-to-ground communication system that is being deployed in the twenty air traffic control centers around the country right now. When that contract was first let and that system was designed, no field input was gathered. All the FAA, Air Traffic and AF input came from Washington-based management types. As a result they decided that because of the reliability figures on that system they didn't need to purchase an additional system called the Voice Training and Backup System, VTABS. Note that we have had a national representative working on VSCS for a little over two years. He has done a great job of collecting user input from the focus groups. This is the field employee's way of getting input to management and to the system designers. We have focus groups that work on various subsystems or various subparts of this system, such as training and certification requirements. We have consistent subgroups, consistent reps working with the VSCS system so you don't run into the problem of one group comes in and says, we need this to be done in purple and blue and another group comes in and says we need it green and black. We have consistent representatives that coordinate what really needs to be done.

Back to the VTABS system, as soon as our representative got involved, the first thing he stated was, "Folks, we need to buy this VTABS system; we need this back up system." Upper level management said, "No, the statement of work is already done. The contract is already let; it's going to cost too much to go back and buy this system. We can live without it because of the reliability figures." In the last six to seven months there have been several system outages with the VSCS. Now the FAA is going back and buying VTABS. It is going to cost them more time and money now than it would have if they had done it in the first place. That's why the Federal employees need to be involved in the first part of the system design. One of the ways we do that and what gives us the basis for doing that is the PAS Employee Involvement Program.

We started our Employee Involvement Program, which works with workgroups to bring field suggestions to management, in about 1985. In your handouts, there is a copy of an Executive Order 12871 signed by President Clinton in October 1993. Basically, that Executive Order outlines our EIP program and it says that management and labor are going to work together in a collaborative relationship to help make the government more efficient, more effective and provide better customer service to our customers, the taxpayer and the flying public in the FAA's case. The EI program is not in all agencies. Some agencies haven't caught on to partnership yet, but it is coming as a cultural change that you're going to see more and more of out there. It's a resource that you really, I can't stress this enough, you really need to draw on to find out what the end user needs to do their job. Because we don't want to spend hours and hours of trying to make the software or the hardware do what we want it to do. We want to input the time and attendance data. We want to be more efficient, more effective and provide better customer

service to our customers, the taxpayer and the flying public in the FAA's case. The EI program is not in all agencies. Some agencies have not caught on to partnership yet, but it is coming as a cultural change that you're going to see more and more of out there. It is a resource that you really need to draw on to find out what the end user needs to do their job. Because we don't want to spend hours and hours of trying to make the software or the hardware do what we want it to do. We want to input the time and attendance data. We want do our word processing. We want to track our airplanes through the sky. That's what we want to do and today a lot of people have told you how you can do that by including the user. I'll take any questions.

Male: (Inaudible)

Dillman: One of the issues would be the downsizing. You know we acknowledge that the job of an FAA technician in the year 2000 is not going to be what it was in the year 1960 or 1970 or 1980. That's a fact of life and we're ready to make that change. Where you find people resistant to change it's when it's dumped on them. When I was a secretary in FAA Security, one day UPS came in and they dumped five PC's in our reception area. Our office was now automated. Talk about minimal user training. We didn't even have anybody in the office that knew how to set it up, but all of a sudden within six months we're supposed to be fully automated and running this for our security reports. When you do that on systems with the complexity of the display system replacement which is coming out to replace the host computer, the VSCS system and all these it has a ripple effect. We want to see that new technology come out. We just want to see it come out right and we want to be involved in making it work.

Male: (Inaudible)

Dillman: Our goal is get on the integrated product teams, we're not there yet. We have run into a lot of resistance outside of the airway facilities and air traffic organizations in the FAA as far as letting us sit at the table when the contract is let. Even with VSCS, the AAS, Advanced Automation System is another good example. We were in on the testing of that, but by the time that we were let in to do the testing, you had requirements creep, because things that we needed to do our job in the field. Remember a controller at the center in Boston may not need exactly the same thing on their screen that a controller in Houston center needs. There's a lot of site modifications that have to be done.

Male: (Inaudible)

Dillman: VTABS, that is my best example. If we had been sitting at the table when that backup system was said we don't need that because of the reliability figures our folks would have said, well, OK, if you do not want to do that let us at least figure out how to make the VSCS processor work if we do not have VTABS. Let's see what changes we can do to the controller's PVD upstairs so that we don't have to buy VTABS. They did not do that and now it is too late to do anything with VSCS. We've got it so we've got to go back and get the other system. Does that answer your question?

Male: (Inaudible)

Dillman: We didn't particularly like VTABS. It's just right now VTABS was the only system that existed to back up the VSCS. When VSCS goes down because the back up channels run through the main processor and it takes 20 minutes for a cold start. The controller has no back

up communications while that processor is down. We consider that unacceptable from a user's standpoint. The controller considered it unacceptable and AAF considered it unacceptable.

Male: (Inaudible)

Dillman: We are not just one person sitting there, we're 30,000 people sitting there. When we talk the FAA listens because we're 30,000 of the 48,000. As far as what the new procurement is going to look like we don't know because right now they are sitting over in Washington trying to figure out what they are going to do after April 1st when they have to have a new procurement system. We are involved in helping figure out what they are going to do and we intend to have a very strong voice at that table.

Needs Assessment/Open Forum

Laura Downey
laura.downey@nist.gov
301-975-4659

Sharon Laskowski
sharon.laskowski@nist.gov
301-975-4535

NIST

Laskowski: We are asking you for your opinions. (Figure 1) To get started we put together a few ideas of our own.

Downey: These are some ways NIST could support collaboration between industry and government in usability engineering and in HCI. (Figure 2) One of the ideas we've come up with is the establishment of some type of usability lab at NIST where people could come and bring their systems or prototypes. We could give demonstrations of some our systems and prototypes.

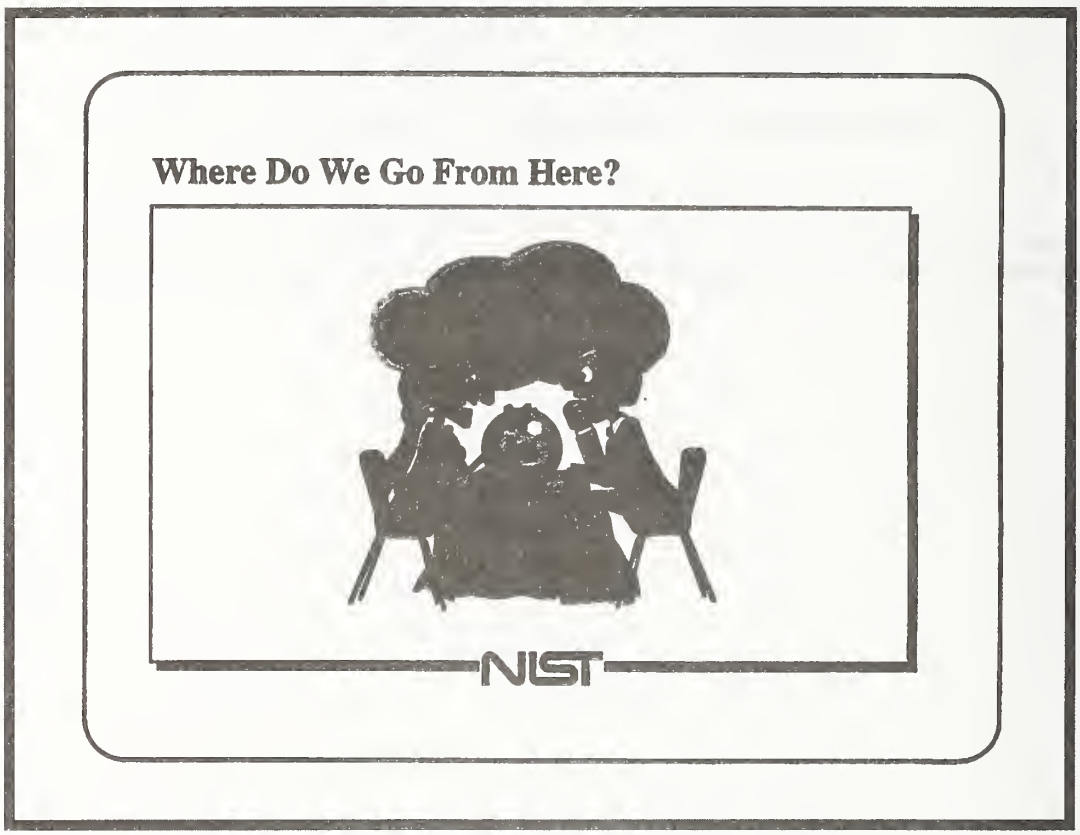


Figure 1

We might be able do some collaborative evaluations to help develop some more measurements in the field. The question is: is there demand for such a lab? Second, we could collect and analyze some government case studies. You've heard a lot of success stories today and maybe we should collect those together and do some analysis. Third, we could serve as an HCI guidelines clearing house. You've heard about a lot them, but we don't have one central place right now to go to and we could serve in that function. Fourth, today is an example of another function we serve, that is conducting a technical exchange and education by facilitating symposiums and workshops.

Laskowski: A good portion of the work we do at NIST has to do with analyzing the latest technology and how we can facilitate its transition into real applications more quickly. For example, the Web usability is quite an issue now. Many people are designing Web pages --many of which are not usable products. NIST could provide a large source of information to the general public and industry and researchers across the country to help.

Downey: Those are a few of our ideas and we're going to open this up to the audience and take suggestions. (Figure 3)

Male: (Inaudible)

Downey: You would like us to form an electronic mailing list of people that are here today or to provide a mailing list that people can join.

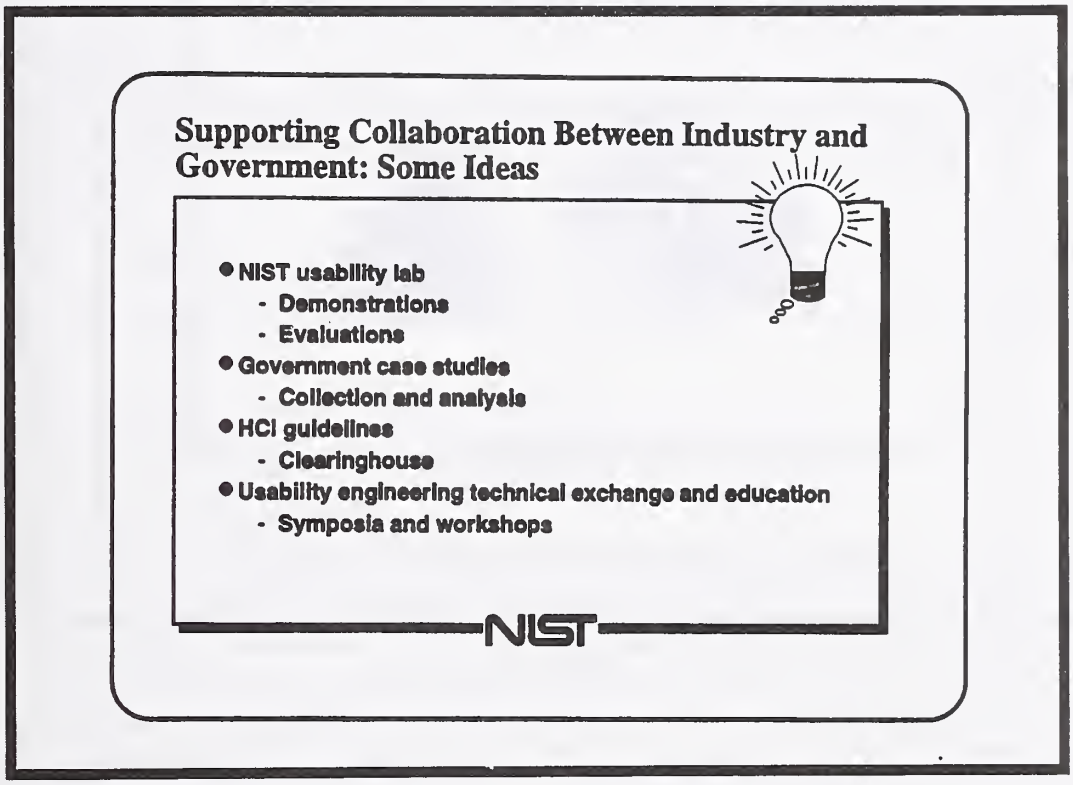


Figure 2

Male: (Inaudible)

Downey: A list server.

Laskowski: There is something that does exist called GOV.HCI and that's people within the Government and within industry that are interested in some of the issues we talked about today, but I think it grew out of some different special interest group workshops.

Female: Where did government HCI originate?

Laskowski: That came out of the CHI 1994 workshops.

How busy is the government's HCI?

Downey: I don't have a good feeling of how much traffic there is on GOV.HCI.

Female: (Inaudible)

Downey: Maybe it's something that should be publicized more.

Female: (Inaudible)

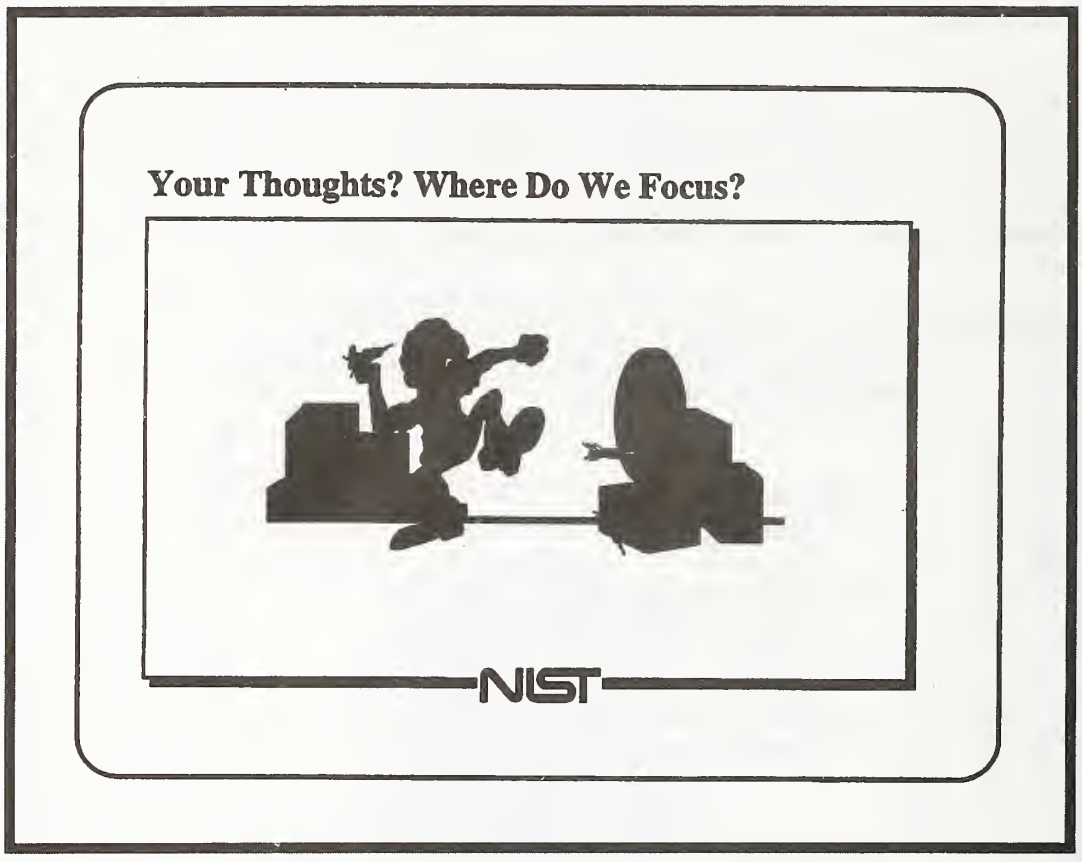


Figure 3

Laskowski: To summarize: we could revive the list. We would have to get in contact with the people that had previously started it up and see if we can collaborate. We are also trying to collect everyone's E-mail that has attended today, which I think is one of the things you can put on your feedback sheets. Please fill out your feedback sheets.

Male: I think it is grand, but I guess I would just like to know when you are going to start. You are positioning NIST in the sense of facilitating, promoting, sponsoring rather than expertise leadership is that correct?

Laskowski: We do have a certain amount of expertise in HCI.

Male: You do? Well, I wasn't sure how it would be reflected.

Laskowski: We have a division called Information Access and User Interfaces. We have four groups that deal in some of the component technologies and the infrastructures to support advanced interfaces. We are doing some work in HCI and usability.

Male: I'm not an expert in this area at all. I don't know the process of going through, what are the barriers, what are the problems, what are the difficulties. So I guess I would say in particular in the Federal agency of introducing usability engineering. When you know what they are and I think there's a lot of experts and I've heard some of the areas referred to, but I don't know if it's all been assembled and you have the top six things that preclude it happening successfully. If those had been addressed and if we know exactly what the problems are and a rendering of the solutions that falls within the scope of the things that you're actively going to pursue. I have no reason to believe we haven't done that I just bring it up as, has it actually been looked at and is that built into the proposed list of things you like to do. Because it strikes me as being right on target.

Downey: I think it ties in somewhat to one of the suggestions we made about gathering the set of Government case studies that were successful versus unsuccessful.

Male: (Inaudible)

Downey: And I think Rex has done some work in that with SSA and some of the other Federal agencies and certainly there are some other people, you've heard the success stories.

Male: (Inaudible)

Laskowski: Are there Federal standards developed in FIPS?

Downey: I understand that FIPS adopts some of the other existing ones out there. So there is that baseline.

Female: What about a working group or consortium?

Downey: That is a good suggestion, that we put together inter-agency groups.

Female: What about including unions?

Downey: Certainly they would need to be represented in that capacity as being end users. I would agree with you that the members of the union are very important. When I worked with Social Security Administration developing a prototype, we did work with several union members, and we had a very successful product and prototype and they were ecstatic to be involved. So I know first hand that it is very important and we should come up with some other ways of doing that. I agree.

Female: There are a number of questions, problems in the government that have not been covered.

Downey: That is one of the things that I guess I didn't focus on enough on the bullet is to continue to run forums like this. What is the audience's opinion on continuing symposia and/or workshops, something like that. Can I have a show of hands? OK, that's pretty positive.

Nigel Bevan: Can we target certain areas like procurements?

Downey: I think that we would probably have that in any symposium or workshop. We could do both. We could specifically target certain things. It's probably important to try to get some procurement officials involved. A lot of people have talked about the specing of usability engineering in the RFP, which is something that we're still hoping to address also.

Female: Can a template for how to include usability be developed?

Downey: I think having a template or a framework is not saying that this is the be all and end all. It's not the magic pill.

Male: But contracting officers are notorious and famous for taking a document and incorporating it by reference, leaving the contractor to figure out how to respond because there's no clue from the rest of the document as to what you're supposed to do.

Downey: I think that's a very good point and obviously one we'd have to address if we had some type of working group that was going to try to even attempt to come up with a template. So that's a definite barrier that we would have to try to rise above.

Laskowski: Anybody else.

Male: First of all, I feel like symposia are like preaching to the choir. I think most people here already know about usability. They think it's a good thing; that is why they are spending their day here. I see it would be useful to have a list server and thing like this generating more effort on more outreach activities. One of these things exist elsewhere, for instance ---- I think monthly -----. I don't know exactly what Friday it's held on, but I believe it's a monthly event.

Downey: Second Friday.

Male: I've been out of town for a while.

Male: (Inaudible)

Male: So there may be some Government issues that I don't know about. (Inaudible) I guess the other thing I would suggest is to schedule more activities like this, get on the schedule of events within the government.

Downey: Right which is kind of what I mean when we were saying go for the software engineering group. If there was a conference to submit something or talk to the people that are organizing it so saying that we wanted to do a presentation on the usability engineering. One of the problems I've seen is how to get the word out to the people that we're really trying to target. Like you said, I knew that we would have some people that were pro-usability already today. We were hoping to reach some people that were interested in learning about usability engineering which is one of our other major focuses today. We tried various Government related publications and so on and so forth so if you had any suggestions on how to reach those other users then we would certainly listen. We've got some arms popping up, go ahead.

Male: (Inaudible)

Downey: I would agree with you to a certain point, but also something that someone brought up in Nigel's European perspective is that it's quality in that realm means something very different than quality in use, which is to use Nigel's term when we're talking about usability engineering, they are talking about all the "ilities," reliability and testability, which is a very different issue. So we'd have to get that buy in that it's the terminology. There's a lot of context and usability in the terminology that means different things to different sets of users. How about the woman here in the back in red?

Male: Where did you announce this symposia? Did you contact CIOs?

Laskowski: Is there an electronic list of CIO's for Government agencies? We've looked for one. Do you know if one exists?

Male: There is one in the Defense Authorization field. (Inaudible)

Laskowski: It was difficult to find a complete mailing list, plus an electronic mailing list, but a lot of that information is not gathered or if it was, we did not have access to it at that time or did not know about it.

Downey: We tried to target various newsgroups, but frankly between just the two of us we did not have time to check everyone's Web page.

Laskowski: We did do some searches on the Web for places to post conferences.

Downey: On your feedback or if you want to just drop a note in our feedback box, I'd be happy to take any possible sources of contacts so we can compile such a list.

Male: Once you have a list of recognized best practices and things that promote, then take it to agencies.

Making Usability Work in the Organization

Symposium on Usability Engineering
National Institute of Standards andTechnology

George Casaday
casaday@usable.enet.dec.com
http://home.aol.com/GeoCasaday
casaday@aol.com

January 10, 1996

1

Mr. George Casaday, DEC
Making Usability Work in the Organization

The topic "Making Usability Work in the Organization" is not something I would have presumed to talk about on my own, but since I got it, it gave me an opportunity to think about things that have really been of concern to me for several years. In about 1989, I stopped being a programmer, stopped being a software developer, and got into this crazy usability and HCI design consulting business. Although I've worked mostly inside my company, just because of the way we're organized, working means selling. It means actually succeeding, that is making usability work in my client organization. So I guess I've seen this working in the organization business from both sides: at times, a user of usability consulting and then as a supplier. So I'm going to give you some personal sorts of views on some ideas. I guess partly because it's late in the day and this is a closing talk I have to be entertaining and I think I also have to be thoughtfully reflective and there's a fine line between that and pontificating. So if I step over, forgive me.

Another sort of thing that struck me about this talk was just where would this kind of a title come from, "Making Usability Work in the Organization." If you think about it, it is kind of a strange title. Rocket scientists don't have talks on making thermo-dynamics work in the organization, if you see what I mean. So in a way to me it's a kind of puzzling title. So I thought about it and

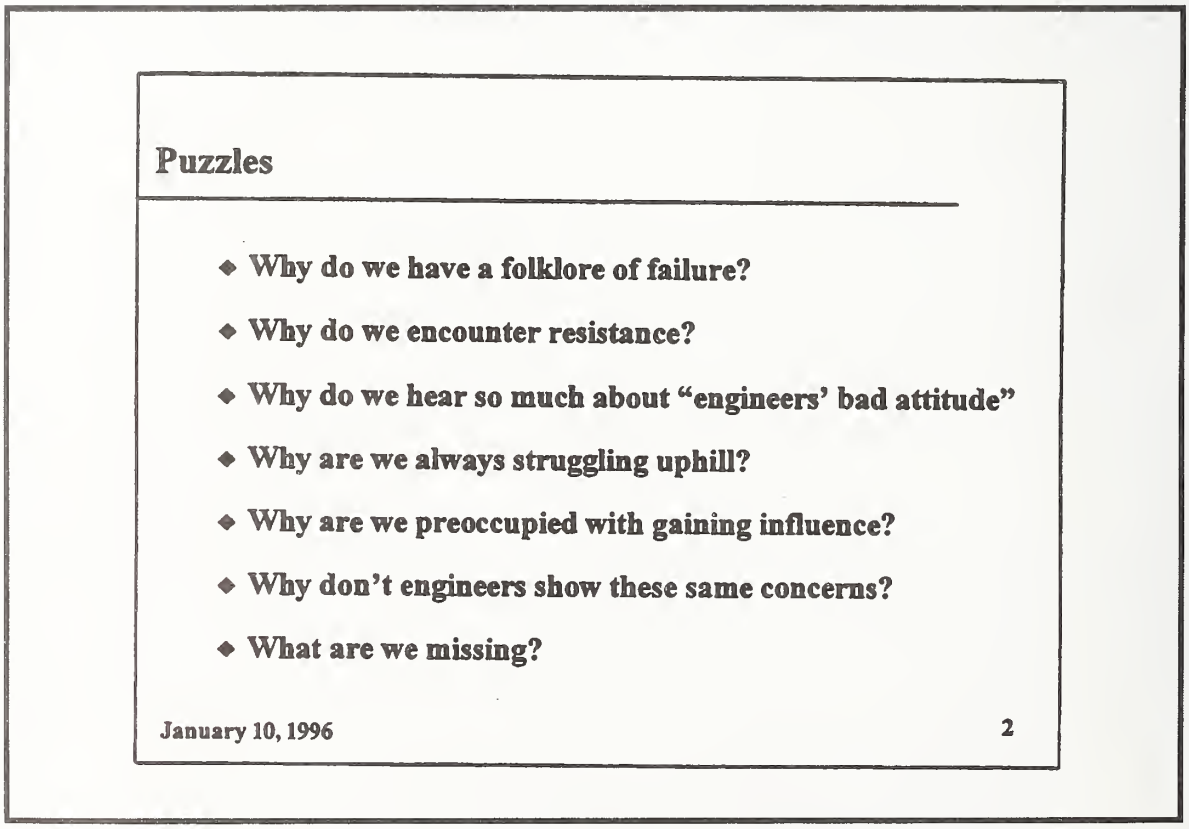


Figure 1

actually made a list of puzzling things and I want to mention those. Why do we have a folklore of failure? By "we," I mean those of us who are actually on the usability consulting side trying to be heard in the organization side. That's who I mean by we, people who would show up at the CHI conference or show up at the UPA conference or things like that. Why do we have success stories? Why isn't success just a part of our standard operating procedure? Why do we have to bring out success stories? Why do we talk about the resistance we encounter so much? I mean, aren't we saving the world after all? Do people want unusable systems? I don't think so. Why do we hear so much about engineer's bad attitudes? I'm an engineer, and honest to God, I am a Boy Scout. We're always struggling uphill coming and going both ways. We're preoccupied with gaining influence. That's very strange. Engineers and developers don't have those same kinds of concerns. They just don't. Most engineers I know are hiding, trying to avoid things, not trying to get more responsibilities. Well, what are we missing? Bounded rationality being what it is, we probably don't know what they all are, but we're probably missing lots of things.

We all have points of view, and I have at least these two points of view of an engineer and a usability specialist. Actually, before that I was a scientist so I'm very schizophrenic. It seems to me that in thinking back on conversations I've had, people I've talked to, my own problems doing this and that, the points of view that we bring to things and that I've brought to things can make a lot of difference. So that's really what I'm going to focus on tonight.

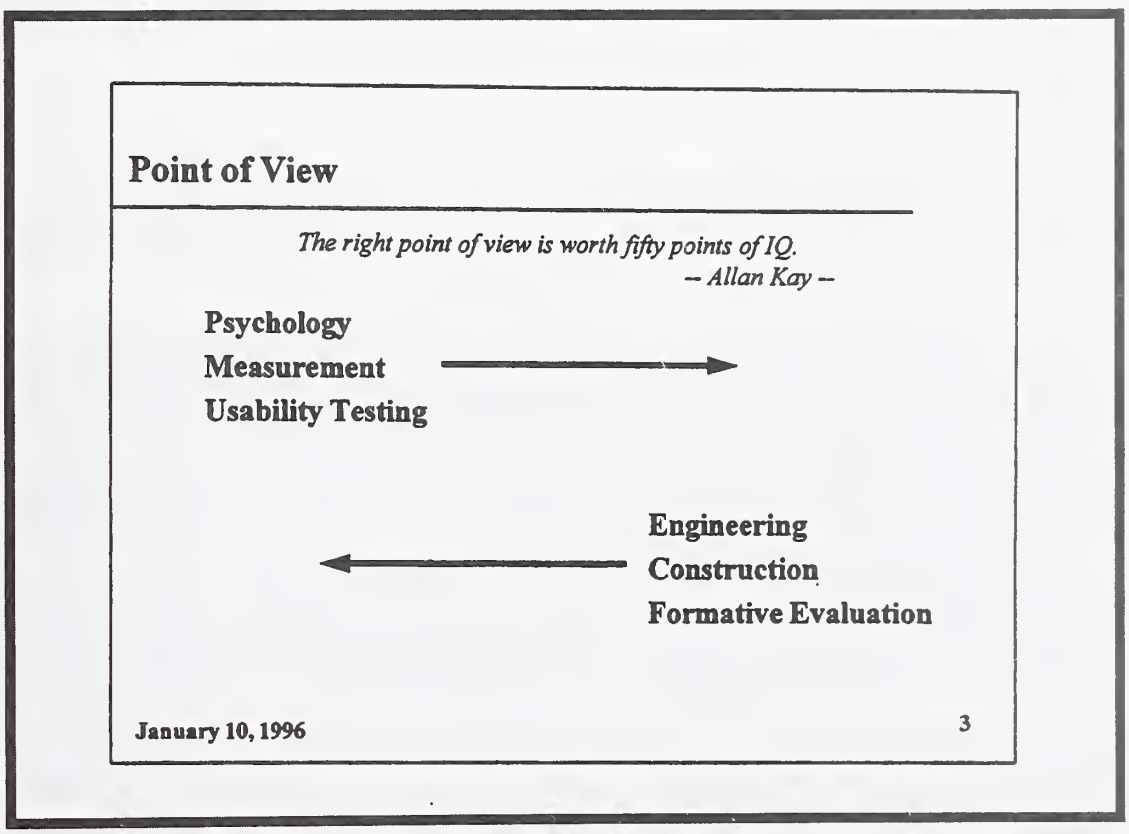


Figure 2

Here's what I mean by point of view (Figure 3). A typical project situation is that you have someone who is a Ph.D. psychologist who spent their life doing the most careful measurements of natural phenomenon and they are doing usability testing and get statistically good results.

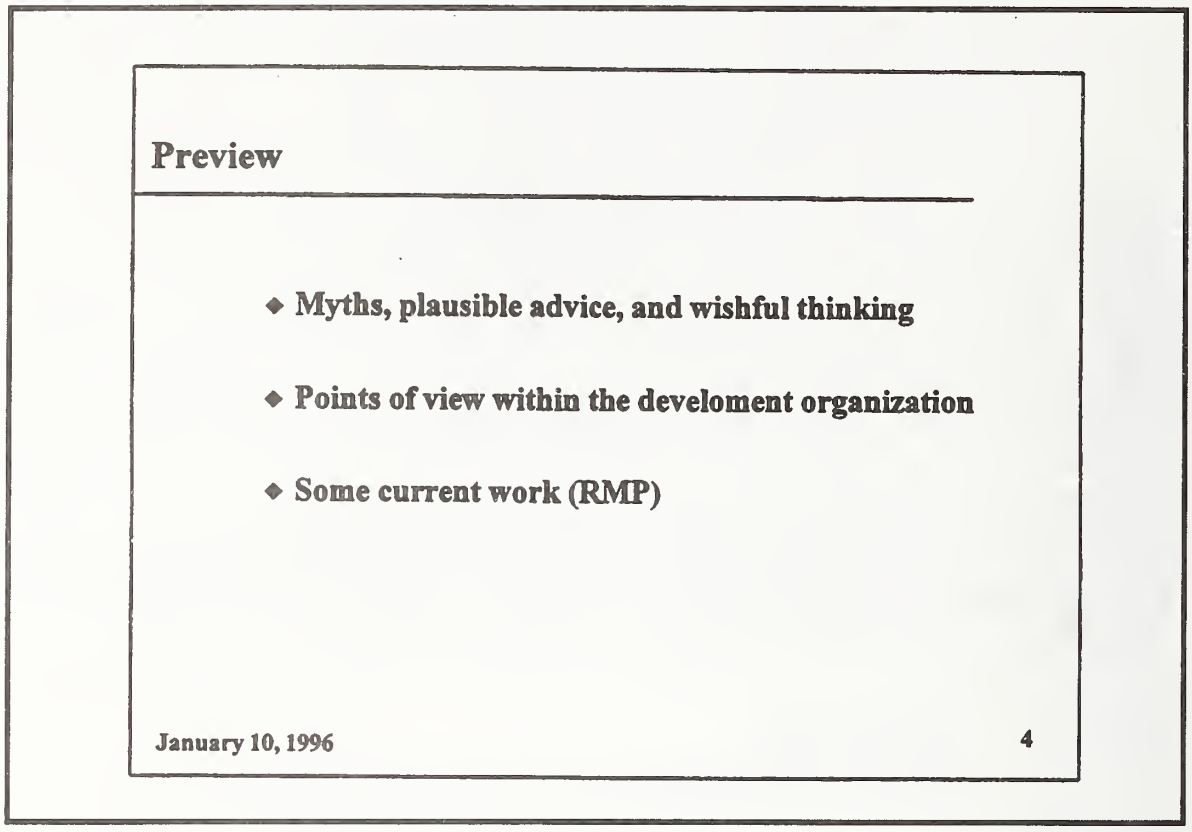


Figure 3

You've got a software engineer who has never done any science in his life, but has been building things since he was eight. He's doing something that we would call maybe formative evaluation, but he just thinks he's evaluating as you go along making sure things work. These people can talk right by each other and I've seen it happen many times in projects. They are just different points of view. The psychologist doesn't think, this is not a knock on psychology, it's just a point of view it's where you're coming from, what you care for and what you take care of. Maybe the job is not done until the evaluation is done very carefully and you know exactly what's good enough or maybe the job is done when the system works at all, these are different points of view. Actually I was really proud of this example because I thought it was a really great one until about two days ago I actually stumbled across the book I had borrowed it from and had scribbled in the margins several years ago. This is a great example you've got to use it sometimes.

First, I want kind of hit three topics: myths, plausible advice and wishful thinking. (Figures 5 and 6) This is a collection of things that are on two slides, just a list. There are things that we, meaning us usability people, believe or haven't questioned for a long time or guesses or things we hope are true, that, in a way, we don't check. We share among ourselves because no one questions them, but they won't necessarily stand up to scrutiny on the outside. In other words your engineers and your developers may not believe you. I believe that those things, even if they

are only just questionable, tend to ghetto-ize us. They separate us from the development organizations that we have to influence. So I'm going to just go through some of those and we'll have a few laughs.

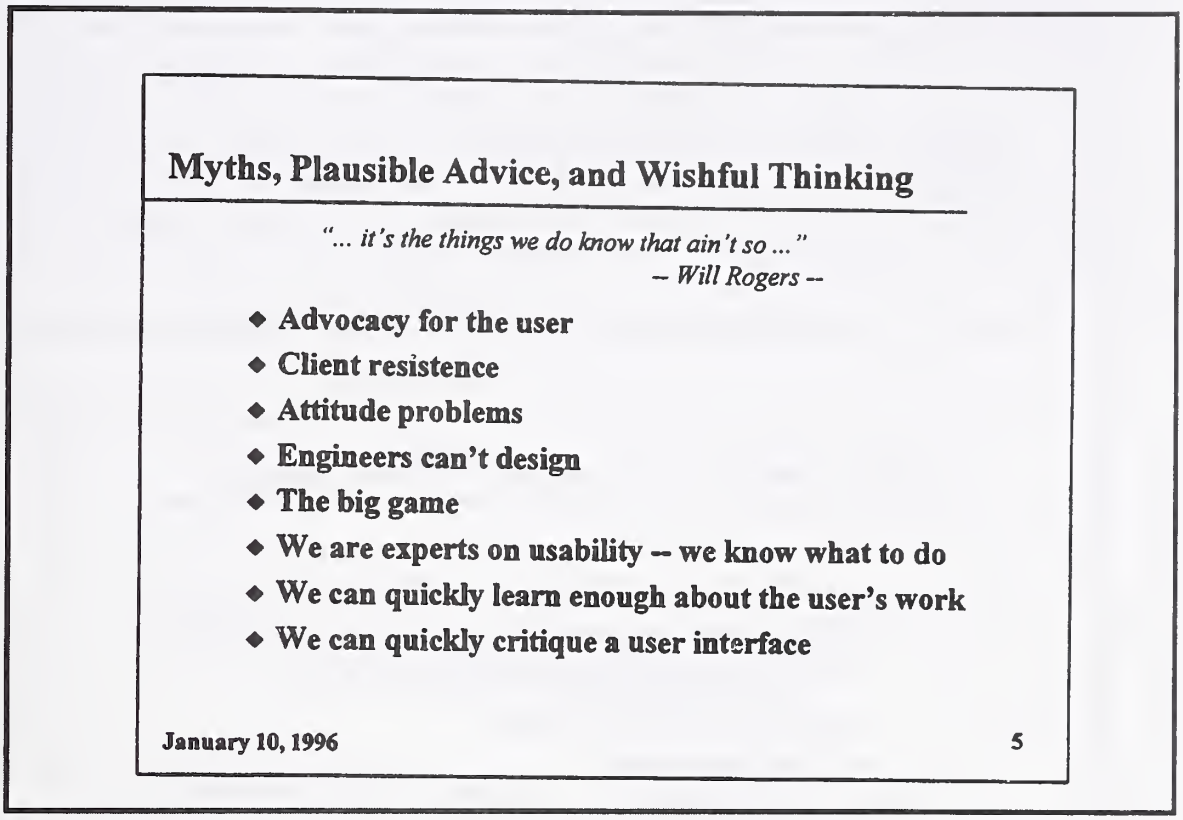


Figure 4

The second bullet (Figure 4), "points of view within the development organization. are legitimate role differentiations within any organization and within development organizations. Those different roles produce different points of view and different concerns. I'm going to talk about some of those at least as they appear to me. Now keep in mind that most of what I know is from watching the inner workings of a large, multi-national corporation that mostly does system level software. So it's a particular point of view. The last thing which I'll talk about is a process that I and a few of my colleagues have developed and used called RMP, which stands for Requirements, Models, Prototypes. It is a process for HCI or user interface design so it doesn't address all of usability, only a small subset of it, but I'll give you a quick overview of that and how it works in with the points of view within an organization.

So let's look at that list. (Figures 5 and 6) Myths, plausible advice and wishful thinking. I don't know which of these are which and some of them may actually not be any of those. Some of them may be for real. Advocacy for the user: should we be advocates for the user? I think in a way that's equivalent to getting a lawyer. If you want to pick a fight, truly you don't want advocacy which turns into an adversarial situation. What you really need is resolution of different points of view and constructive compromises. These actually come in a kind of list because I wrote them down stream of consciousness, sometime in November. It was really

trying to figure out what these notes meant after several months of delay. Advocacy can very easily look like what we call client resistant. That is the developer's bad attitude. In fact, it looks like a bad attitude from both sides. It looks like we're being snobbish and they look like they are resisting. We can conclude that engineers can't design like white men can't jump and women can't preach and things. Then finally that leads to "the big game" which is a phrase that the family psychologist Virginia Santera invented. (I think that's where I got it.) "The big game" is the game about who gets to tell whom what to do. Who gets to give orders and who has to take orders. Actually, we can say we should get to give the orders because we're the experts. That's very shaky ground and what's really needed truly I believe is the team effort where there is mutual respect and where the big game just is not played. Where pulling rank just doesn't happen.

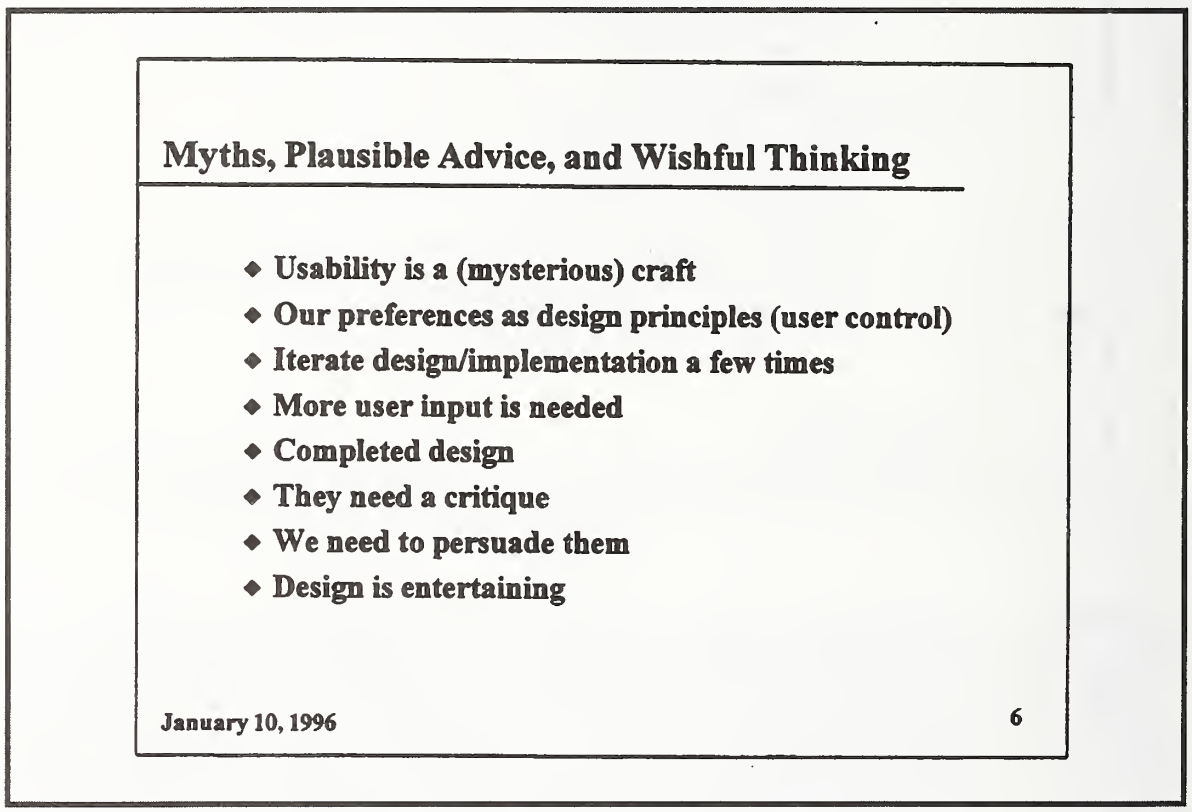


Figure 5

A couple of other things on this list. We can quickly learn enough about the user's work, that is, perhaps as consultants we drop into a project with truly a lot of expertise about human factors, but maybe not much expertise about the task domain. It's very tempting to start talking about what the user might want based on insufficient information. Remember you're probably working with people who have been working in that task domain and have years of experience. We can quickly critique a user interface. Well, we can superficially, but not in any depth, not at the depth where you're looking at the interaction of why things are they way they are. So these are some things that if we're not aware of then they can separate us. They can ghetto-ize us, in fact.

Usability is a mysterious craft. If you sell that to a development manager what he hears or she hears is, "Oh, an uncontrolled outside dependency, please go away." Because managers do not want to be in the hands of prima donnas or people who are mysteriously great. Our preferences as design principles, for example, user control, is another myth. There is something you can say in most usability drinking sessions after a conference like this, we have to always keep the user in control and make the user feel in control. Now that's actually a personal preference, because lots of people who are in this business are very interested in control. See, I'm not so I'm very sensitive to this. I would really be a passenger, if I had a choice, rather than a driver. So, the feeling of control is maybe not the point. Maybe the point for some people is just getting their work done and they don't care who is in control. So maybe user control is a personal preference and not a deep principle of HCI design. Iterative design and implementation a few times is another myth. I have never, ever, been in any project that went back to consider design after any code was written. More user input is needed is probably true, but at some point more user input won't

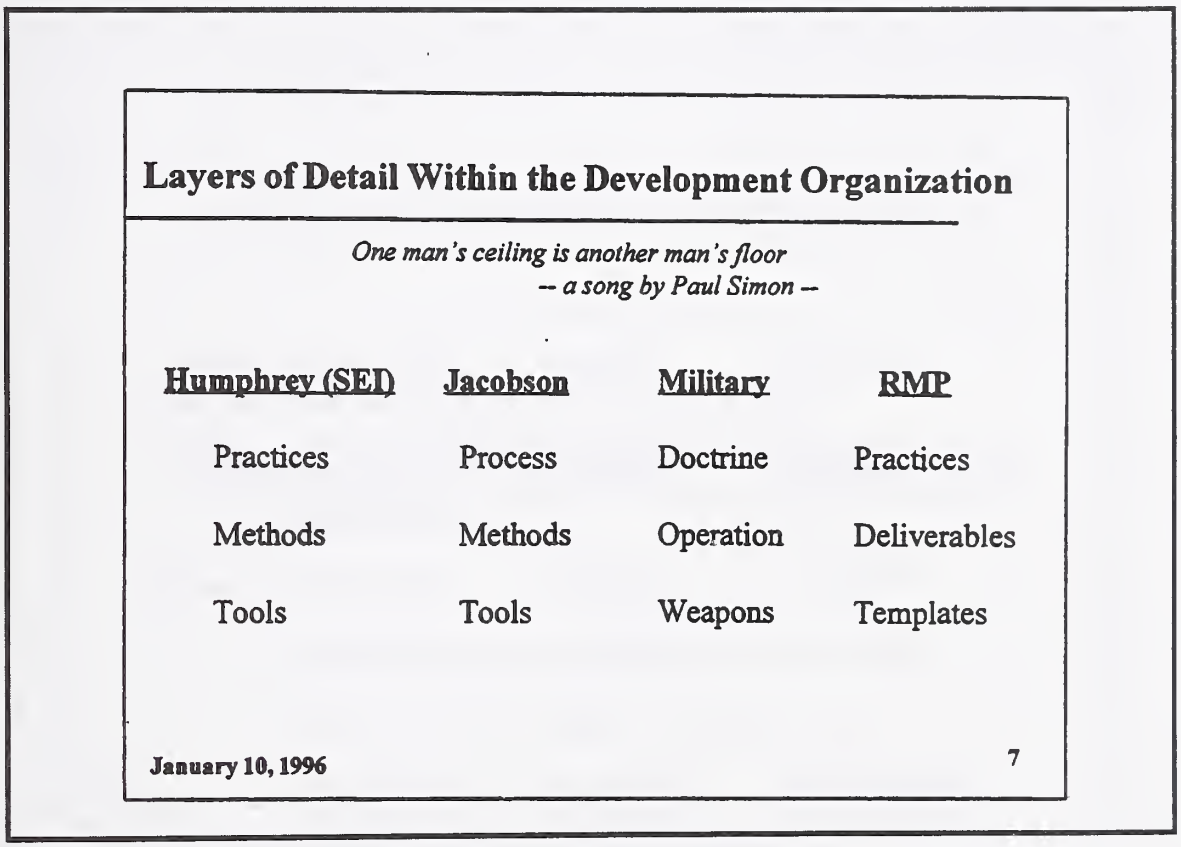


Figure 6

substitute for design and, unfortunately, the notion of design being entertaining is probably not true either. Design is excruciatingly hard work and detailed work. More user input won't solve that problem. The myth or thinking of wishful thinking of the completed design if we could only get it right then we could hand it off to engineering and they could program it, but actually designs are never completed. No one's designs are ever completed. There's always improvisation. So in my opinion a lot of the quality of the hand-off from, say, a designer to

development depends on how aware the developers are of the rationale behind the design so that when they have to improvise, and they will, they do a pretty good job. I've pursued that with developers at almost every level of skill and it works really well. I've just seen programmers do great jobs of improvising.

They need a critique but more. I really find that they need to be informed and they need to participate in developing the design rationale. Programmers need to participate in the design as much as users do. They work better. So these are a collection of things that tend to form our point of view and if we could just see the flip side of them we could see why it is we are not always as successful as we would like to be in dealing with development organizations. So that's the story of myths, plausible advice and wishful thinking, as much as we're going to pursue it now. With any luck this will provoke some afterward things.

I want to move on to the next topic: the legitimate and necessary separation of concerns in a development organization. This is going to be fairly simplistic, but maybe there's something to learn anyway.

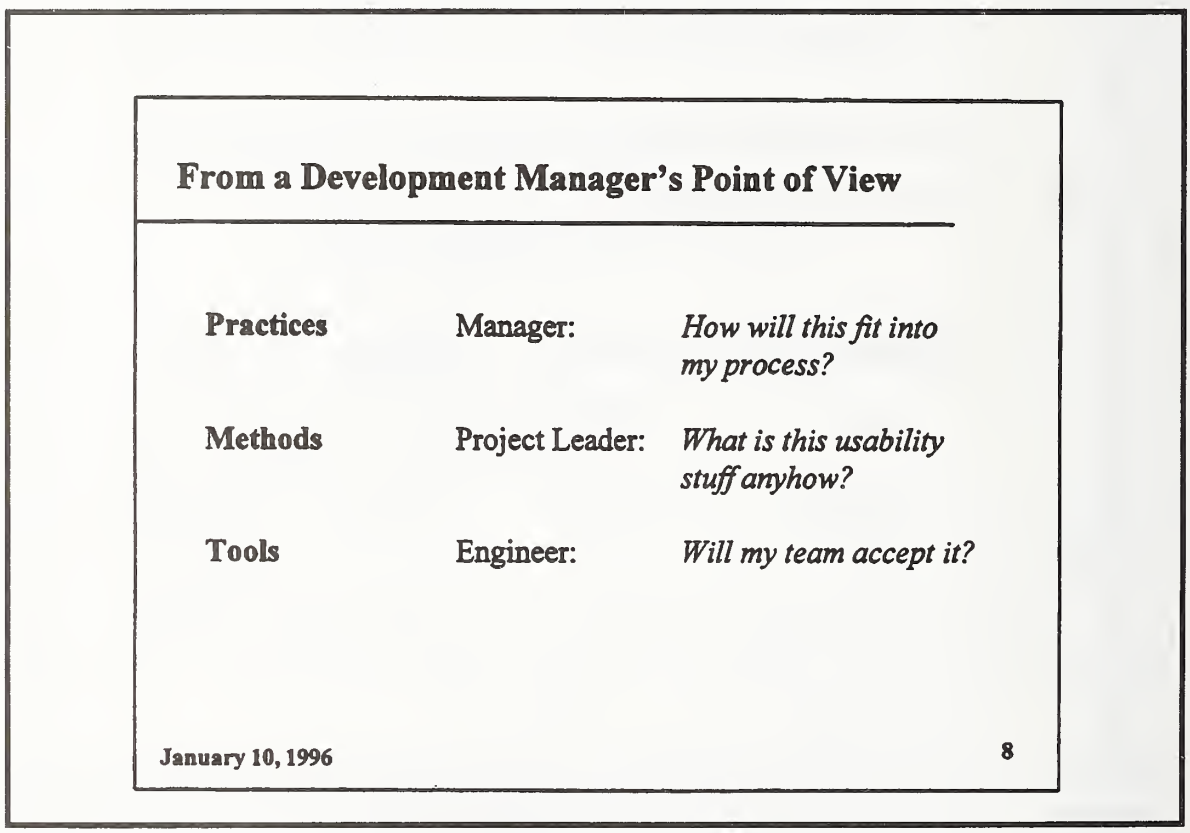


Figure 7

Look around all sorts of places and you can see in management structures a kind of layer of details (Figure 7) that people are concerned with. This is a set of words from Watts Humphrey's book in which he distinguishes methods and then practices about that to explain how to deploy the methods and tools. Jacobson talks about process methods and tools. Military has from the bottom up weapons, operations, doctrine. I just put that in because I'm from Boston and I have

lots of friends who like to wear those hats and do the muskets and march around on Patriot's Day and reenact the Revolutionary War. So the weapon is the smooth musket and the operation is the manual of arms which they practice and practice to get perfect and then the doctrine is something like don't fire until you see the whites of their eyes. There are these layers of concerns. In RMP (requirements, models, prototypes) what we have tried to do is address those levels of concern all together so that we can deal with those levels of concern in a development organization.

Things look very interesting from a development managers point of view. (Figure 8 These are quotes from a development manager, a good development manager, and this one is excellent. She is a manager of a group of something between 30 and 40 developers with a budget of a few million dollars. She reports to a group manager so the group manager said to her, on the previous version of this product we've been getting an awful lot of complaints about how hard it is to use and I want you to make those phone calls stop on the next version. So Ms. R said to me, come on down I want to talk, can you make this stop. So I said, "Sure," "OK, how.?" Remember, she's only got an hour to listen. How are you going to do that and, furthermore, tell me how to tell my manager that this problem is going to stop and tell me how to tell him in about five minutes? So that's what she's interested in. She wants to know if her problem is going to be solved. I didn't have to sell her that she had a problem or wanted a solution, what I had to sell her was that she should listen to me.

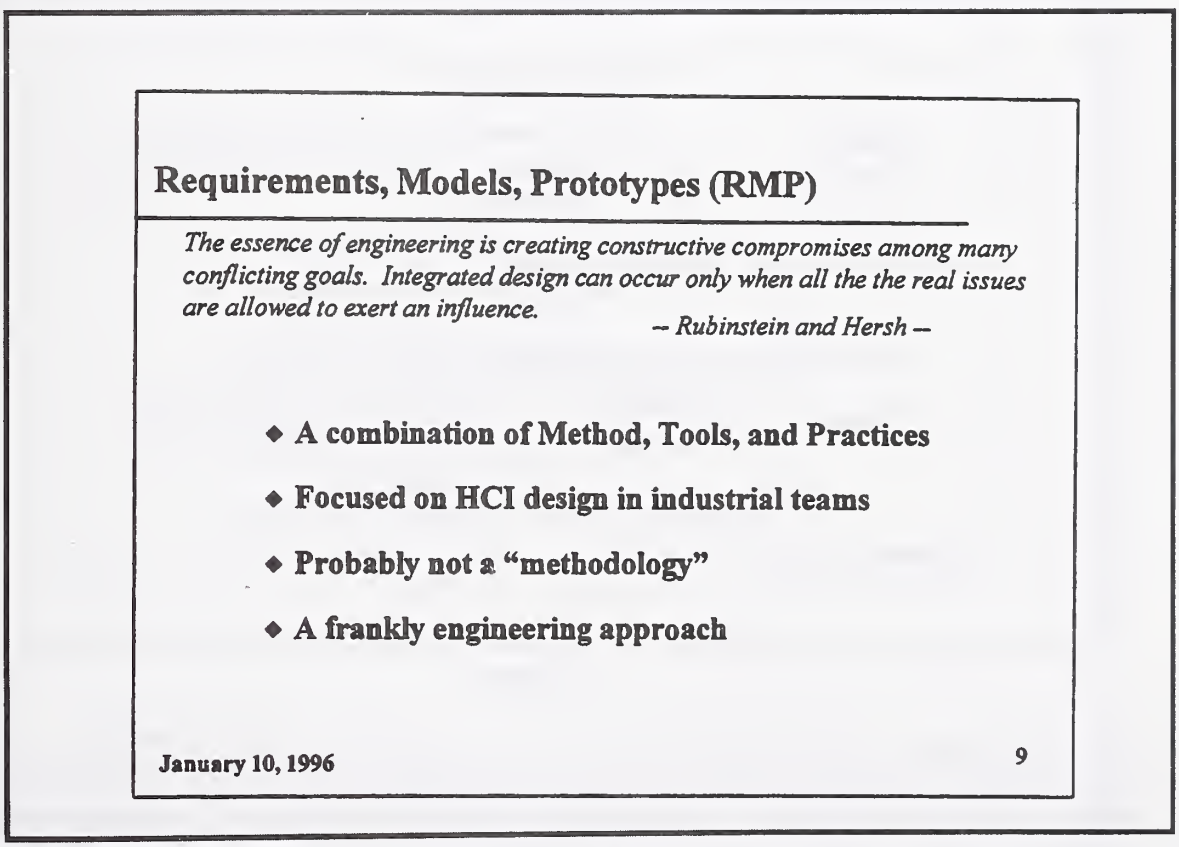


Figure 8

I had a manager once who explained the whole world to me, he said, "George, why should I believe you?" So this manager actually was very savvy and very good, really running this gang of cats really well. So she was able to actually speak for her staff at these various levels and that's what was really cool. She would say, "OK, how is this going to fit into my process? I already have a development process. Whatever you're going to do, how is it going to fit in; how is it going to avoid disrupting what I'm doing?" She says, "I have project leaders, and what they want to know is, 'What is this stuff anyway? What is this usability stuff?' I mean, concretely speaking, how am I going to work this into my development schedule?'" Down at the individual contributor level she said, "Will my engineers accept it?" And it turns out what she meant is what they want to know is how is this going to actually effect what they actually have to do when they come in in the morning. So you've got these level of details. So those are really very different points of view in a development organization and we have had a shot at dealing with them and I'm going to spend the next about 10 minutes telling you what we've got and then that will be it.

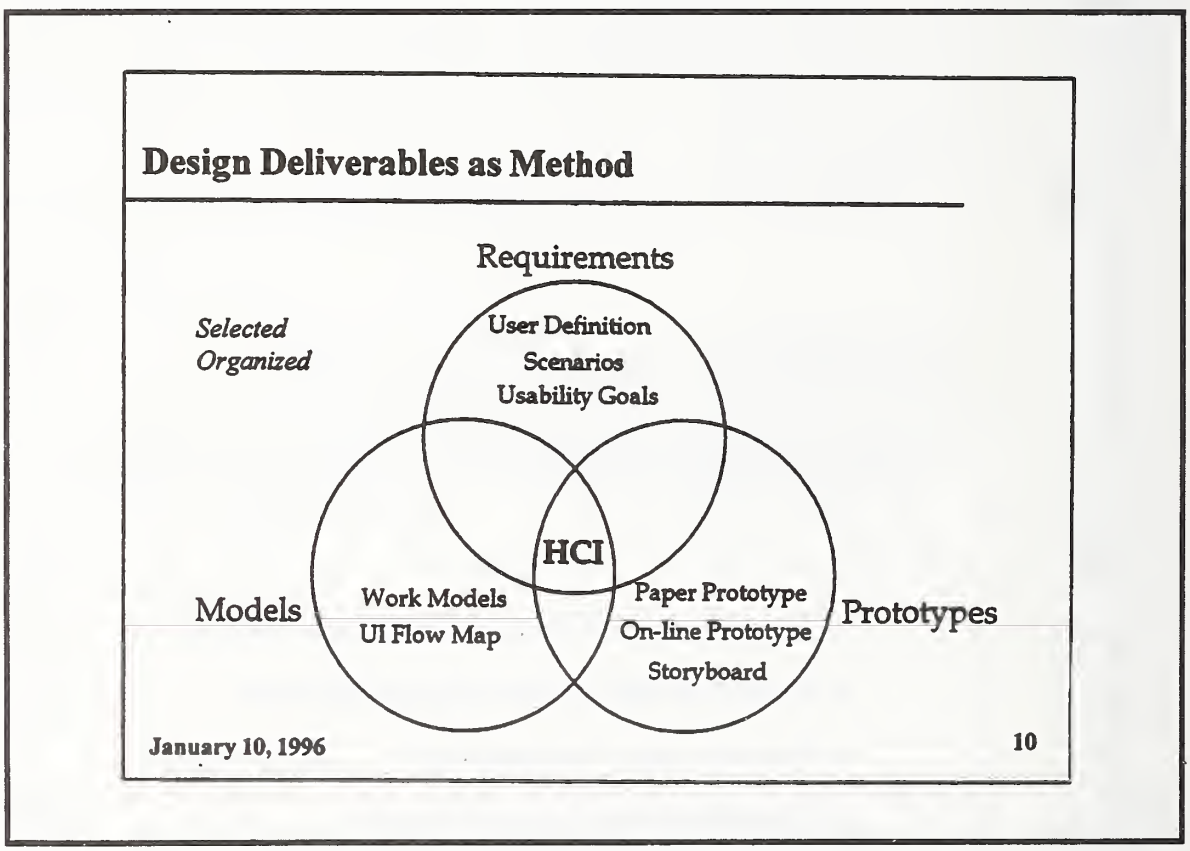


Figure 9

This is the summary slide on requirements, models and prototypes. (Figure 9) It's a combinations of methods, tools and practices. It is focused on HCI design, not usability in the large because frankly that's more than I can bite off. It's probably not a methodology in spite of having an acronym. What I'm going to describe is a framework and a way of thinking about doing usability work in an industrial team, but it's very tolerant of interlopers and snap-ins and changes in variation. So it's extremely flexible, much too much to really be a methodology. It's

frankly an engineering approach. If anyone knows where this came from I'd love to know. I know it's not original with me, but the idea is that there are artists and artists believe in intuition and there are crafts people who believe in, this is when the chips are down, who believe in their skills. There are engineers who, when the chips are down, believe in their methods, and that's what they fall back on. So, that's what I mean by a frankly engineering approach. We're not trying to supply process for people who are rocket scientists. We're trying to supply processes that can be grasped at a elementary level and still add value, something to build on. As Nigel said earlier, maybe it's better to do something generic that you can actually do rather than try to do something extremely specialized that requires a lot of skill that you can't really deploy. So we're after something that's actually deployable in real practice.

We'll start at that middle level of the three which is method. (Figure 10) Rather than specify a collection of things to do, activities, we specify deliverables. There are lots of things you can do, but there are not that many deliverables that you can imagine. There is a need to develop some requirements. There is a need to say what it is that if it happens your system will be usable enough. So we've broken that down into user definitions, "Who is the user?" scenarios; what are the individual stories that have to happen as the system issues and what are the usability goals? Now this is the level that the manager is interested in seeing: there is a user definition scenario and usability goals to go do that, a collection of models.

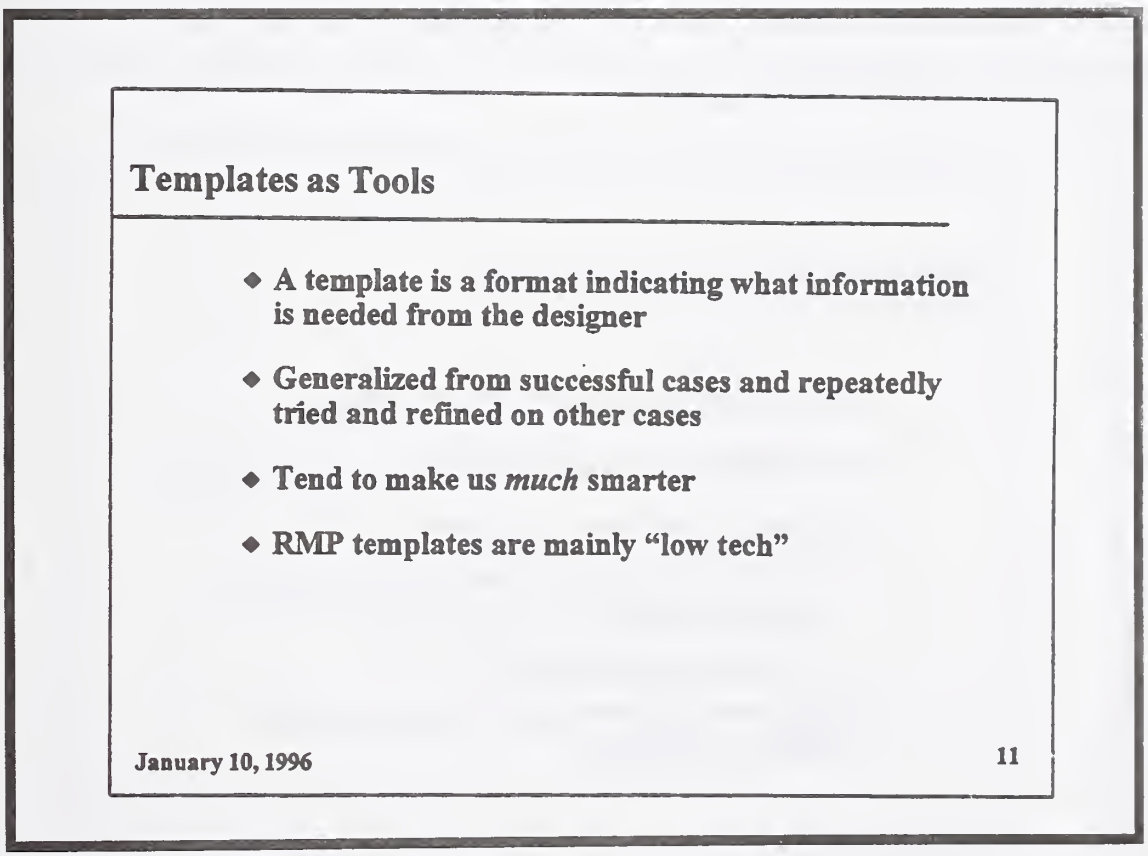


Figure 10

Design, I think, is really a lot about modeling and creating representations of what's going to happen. Models of the work and also models of the user interface that's going to be built, a collection of prototypes. You finally have to find out how stuff is going to work out of practice. We have paper prototypes, on-line prototypes and story boards all for different purposes. There is nothing specified in this about the sequence in which you do things. What I think you really have to do is treat each of these as deliverables and this actually comes to the middle layer where you have a project leader who maybe has a half of dozen people and is responsible for the details of the schedule. This is the user definition. Here is the designated responsible individual, and here is the date for the first draft to be due, the second draft and third draft, until it's done—same thing with work models and prototypes. A specific deliverable, an individual who is responsible for it and the dates for the draft to be due and in practice the drafts of the deliverables can overlap. So that is the sort of stuff that a project manager and a small group leader to know. What individual contributors really need to know is, "What are these deliverables really?"

To deal with that we've created a set of templates, one for each of these work products and I'm going to show you those. I'm going to go into just a little bit more detail on this. A template (Figure 11) is not exactly what you think it is; it is a format that indicates what information is needed from the designer. It doesn't give you the information, but it basically asks the questions that are generalized from the literature from successful experience. They tend to make us much smarter because as a matter of fact it's usually a lot easier to answer a question than to figure out what the right question is to ask. All of the RMP templates are extremely low tech. The highest tech template is a kind of grid that's in visual basic, a lay out grid for screens. Most of them are things like Microsoft Word documents.

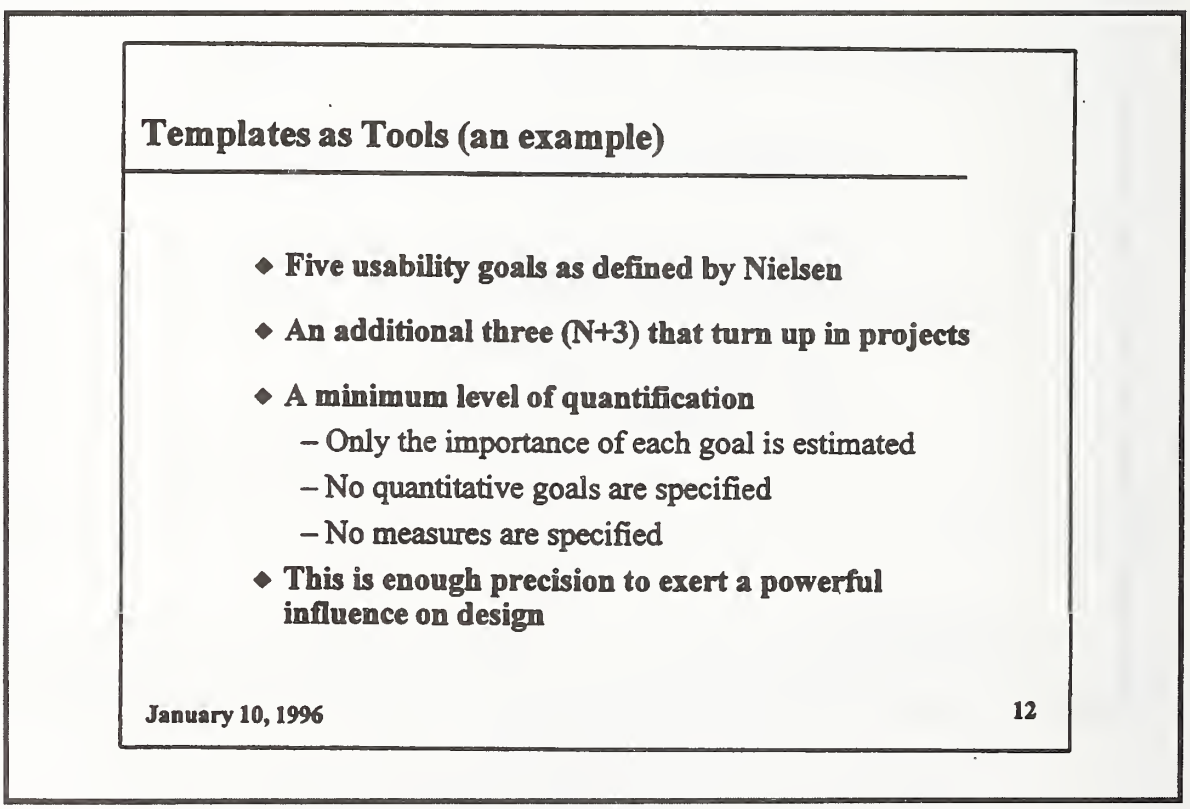


Figure 11

I want to give you one example. This is the example for usability goals. Usability covers a lot of ground and this is a place to start. We started with Jacob Neilsen's five usability goals, and we added three that kept popping up in projects, and we could not add them, and we ended up with those, and we found it very hard to add any more. These are the goals: understandable, learnable, memorable, efficient, reliable, flexible, automated, satisfying. So what we've done in the template is put in the name of each one. We put in a description and an example of each one and we've put in numbers from one to ten so that the developers can estimate the importance of each one.

USABILITY GOALS

We want to build a "usable" product. In fact, usability is not a single quality, but is made up of many goals that can be teased apart and handled somewhat distinctly. It is important to do that so we can focus resources and because some usability goals are actually in conflict and have to be traded off.

We are using eight usability goals that cover most known usability issues. Five of these have been standard in the usability literature and in practice for some time, and three others are new::

1. Understandable
2. Learnable
3. Memorable
4. Efficient (fast)
5. Reliable (user errors)
6. Flexible
7. Automated
8. Satisfying (subjective)

We will start by estimating how important each of these goals is to the product as a whole. When we have rated the importance of each of these goals, we will have a set of usability requirements that are precise enough to guide user interface design and to suggest focused usability testing methods. When we begin to design subcomponents of the user interface, we may find that we need to revise these goals for each of the subcomponents.

Trade-Offs: It is not possible for everything to be very important. First there are limited development resources, and some things will inevitably have to go to the end of the queue. Second, some of the criteria are contradictory; for example a very short leaming curve tends to be incompatible with very great efficiency (speed) in use, and reliability usually conflicts with both flexibility and speed.

Scales: The scales for the usability criteria below run from 1 (no importance) to 10 (essential to product success).

ESTIMATES

1. Understandable: Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that it is important for users to grasp the concepts of the work, system, and user interface. Example: appreciating the distinction between a client and a server. A low rating means that the user can perform in a rote mechanical way. Example: data entry from a paper form.

Comments:

2. Learnable: Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that leaming the system is an important issue. Examples: a requirement for no formal training or a requirement for getting some useful work done in the first day. A low rating means that leaming is not a major issue. Example: Long and rigorous training is assumed in order to get very fast and precise performance, so the user interface does not have to be designed for leaming by exploring.

Comments:

3. Memorable: Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that users will use the system, be away from it for extended periods, then return to use. (Intermittent use) It will be important for them to get back up to speed quickly. Example: a system for filing travel expense reports that may be used only once or twice per quarter. A low rating means that users will use the system more or less daily and so will not forget things once they are learned. Example: e-mail.

Comments:

4. Efficient (Fast): Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that it is important for the user interface to be fast with mouse and keyboard. Example: often used actions in a pull down menu are speeded up through "accelerator" key combinations such as ^X, ^C, ^V for cut copy paste in Windows. A low rating means that speed per se is not important and so can be sacrificed to gain reliability, flexibility, and ease of leaming. Example: confirmation messages after dangerous actions ("Do you really want to start WWIII now?") increase reliability (reduce possibilities for user errors) at the cost of slowing down the dangerous operation.

Comments:

5. Reliable (user errors): Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that there are dangerous errors possible and that users should be protected from them either by preventing them or by making recovery effective. Example: confirmation messages for file deletion. A low rating means that errors can be tolerated in order to gain efficiency or flexibility.

Comments:

6. Flexible: Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that the designers cannot reliably predict the user's work flow, so the system should be designed to let the user improvise. Example: modem GUI's allow great flexibility because they usually let the user execute one of very many actions at any one time and in a sequence determined by the user, not the system. A low rating means that the user does not need to vary the sequence of events or to improvise much. Example: a data entry system may lead a user through a fixed set of questions and data fields in a predetermined order, and this may be the best design for highly predictable systems since it relieves the user of much repetitive decision making.

Comments:

7. Automated: Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that much of the task is so routine that the user should not be bothered with it. Example: rather than having a user specify each name of a series of files to print, a system might automate specifying files by allowing wildcards in the file names of a print command and so replacing typing by an automated process. A low rating means that most operations should be done manually because they cannot be reliably automated or because it is important for the user to

learn and understand what is going on. Example: It is possible to automate $ DELETE and yet it is important not to automate this fully because of the danger of too much damage happening without the user noticing.

Comments:

8. Satisfying Subjectively: Estimate: 1 2 3 4 5 6 7 8 9 1 0

A high rating means that, assuming all the other criteria are acceptable, there is still a need for a pleasant experience for the user. Example: Windows offers a wide range of screen color schemes and amusing screen savers for entertainment value. A low rating means that users need not like the system. Generally this means that as long as the system stays out of the way and is not very noticable it is acceptable. Example: Most of us still use terminals at times. We do not expect pretty graphics on a terminal and consider that aspect of the terminal user interface to be unimportant.

Comments:

I know you can't read this, but you can probably read it right there in your handouts. Basically, all that really needs to be done to fill this in is to say, "OK, let's pick one, Learnable." How important in this product is the learning curve? Is it important for people to get up to speed quickly? Do you not care because you're going to have an infinite training budget? It's really just a matter of making that estimate. Now what we don't say is how you get that information because that's the part that varies. It's very hard to tell someone exactly how or even to give a list of how to make that estimate, but you have to have the estimate. If you can do that for each one of them then you have a set of usability goals that are powerful enough to really guide design and that you can test against. If you're very ambitious you can actually specify quantitative measures to find out how well you're doing on those goals. It is very hard to get people to sign up for a fairly expensive set of quantitative tests until they've seen some groundwork laid. If you say, "OK, learnable is real important; it gives out a rating of nine out of ten." Then you start saying, "Well, if it's that important, shouldn't we invest a little bit in saying exactly what it means to be easy to learn and then shouldn't we test?" So that is how templates work. The idea is to make things almost automatic.

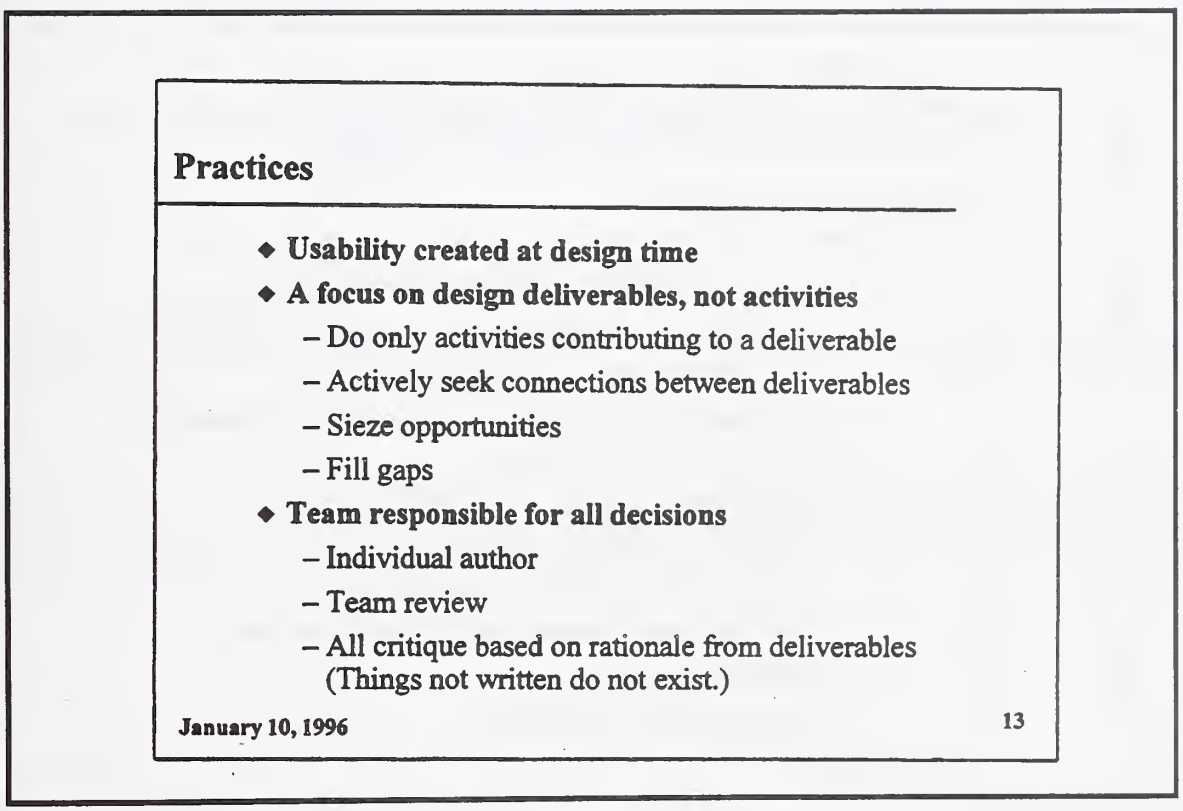


Figure 12

The set of practices for them is small. (Figure 12) Usability created a design time that is not a testing time. A focus on the deliverables, not activities. I think I've said plenty of that. A team that's responsible for all decisions, so shared responsibility, but there's one designated responsible individual. Critiques based on rationale rather than on authority.

Another one is concurrent creation of all the design deliverables. This is not exactly iteration. Iteration implies that there is some sequence of doing things which you do over and over again. We don't actually know of any such sequence. If you think about the Star life cycle that Rex put up this morning, there is an evaluation in the center and lots of things around, but there's no suggested sequence there. There just is not. There doesn't seem to be any sequence that works because there is so much feed forward and feed back of information which I won't elaborate on. The last thing is formative evaluation. We've got a set of ten or so deliverables that we'd like to see produced. They are documents, they can have due dates and they'll go through a series of drafts. So each one of them needs to be evaluated. Many of them cannot be evaluated with users. Most of them cannot be evaluated by testing, but they all can be evaluated somehow. That's the point here. Don't evaluate just the prototypes, evaluate your models, evaluate your goals, and then revise the quality and continue until time is gone. The result of this concurrent approach is that when the clock eventually ticks down to the end and the dinger goes off then you have something. You are not half way through a sequential process. Everything is done and now someone is going to have to improvise. Remember design is never done. So that's what we're advising people to do and in a year or so we'll see how this finally goes.

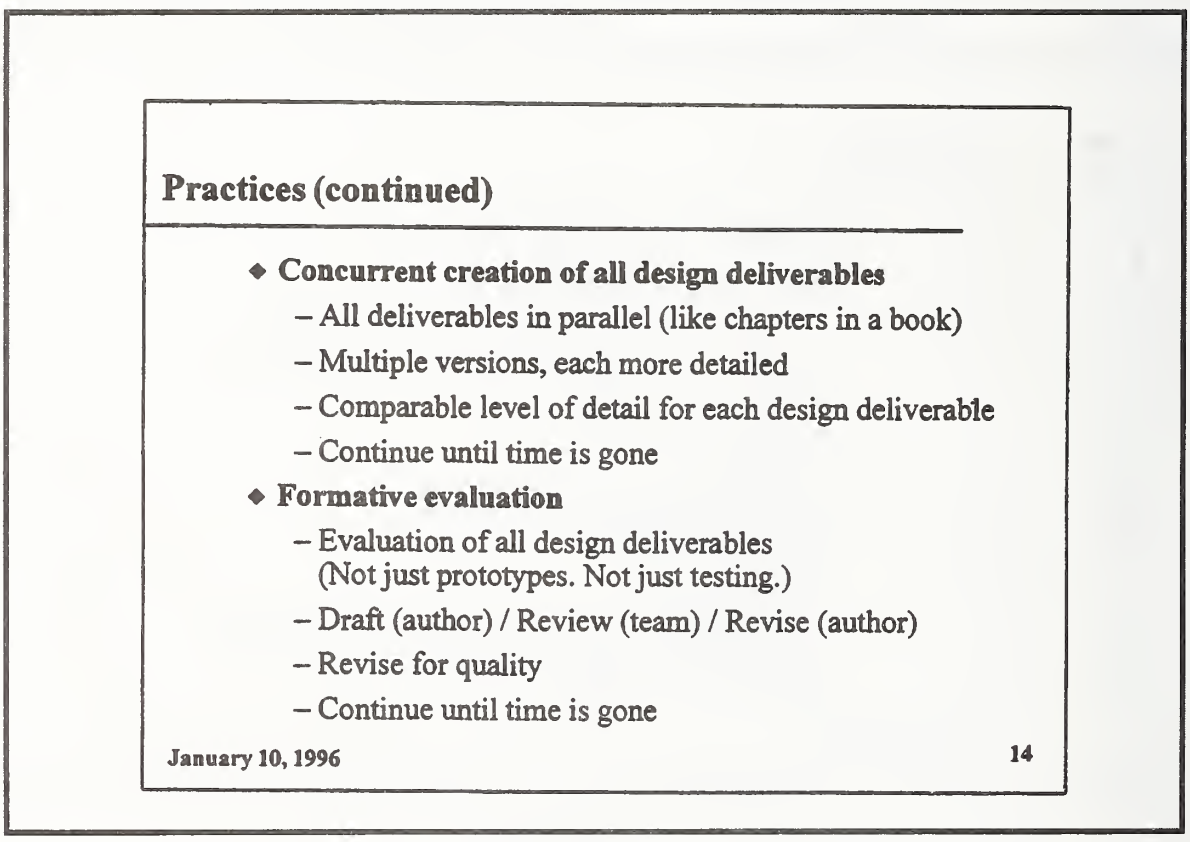


Figure 13

This takes us back to point of view (Figure 15), but we may actually view ourselves as expert navigators. However, others may very well see us as magicians or charlatans, either of those is equally bad. Really development managers and developers don't want magicians coming in and doing something and they certainly don't want charlatans or snake oil sales people. I believe our goal and the way to really make our work effective in an organization is to be convincing,

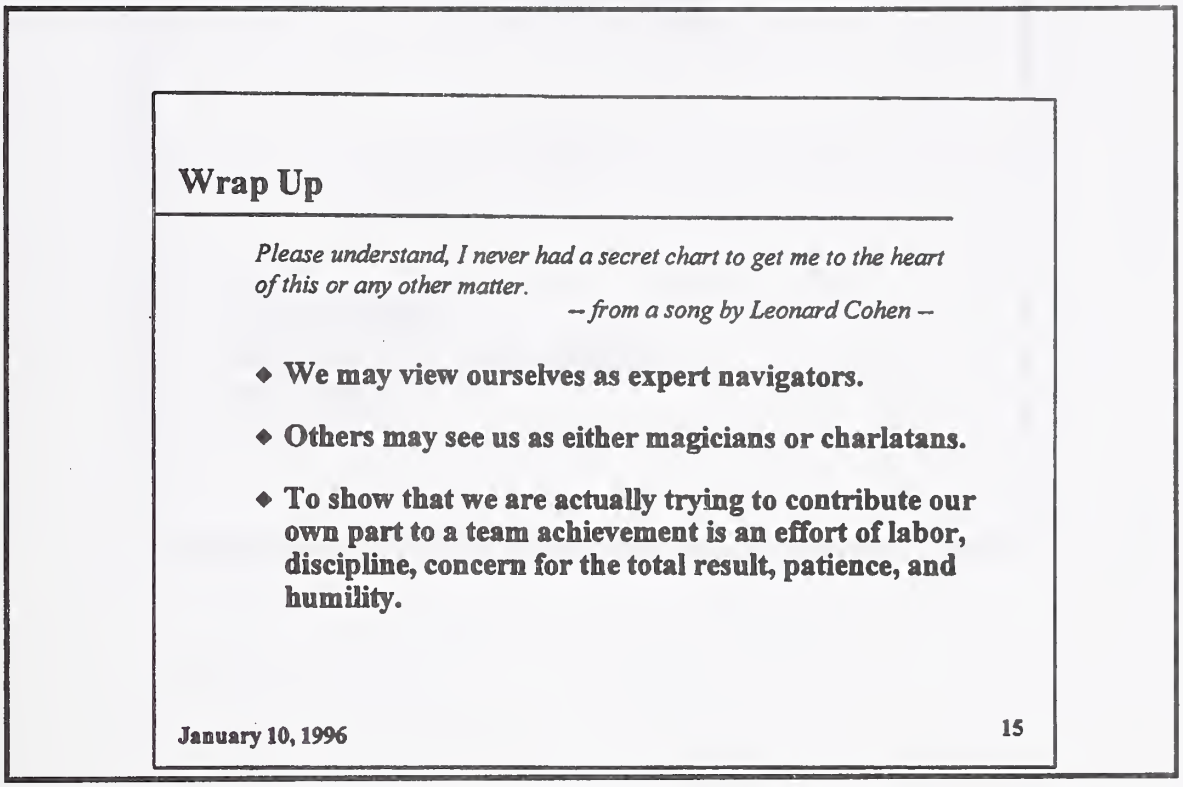


Figure 14

authentic, and truly make a contribution to a team achievement. That's hard and sometimes it involves biting your tongue. It involves patience and it really involves humility too sometimes. I will just stop there. Thank you, very much.

An European Perspective on Engineering Usability: the case of Italian IRS

Anna Giannetti
Software Quality Management
SOGEI Spa

1

(NOTE: This is not a transcription. Ms. Giannetti was unable to attend but submitted this paper for the symposium record.)

"An analysis of the impact of usability requirements in a Large European Government Contract for Public Administration"

by Anna Giannetti

Introduction

This position paper will seek to illustrate the profound modifications going on at the level of European Union for large project in the Public Administration domain and the impact of a Software Quality Management, such as the one fostered by the ISO-9000-1, and the impact they are having on the traditional software development methodology. Issues concerning the role of European Union and European Central Governments as relevant Bodies in fostering human interface and human factor competence and skills will be faced. Furthermore emphasis will be given to the ambivalent role of Government and Public Administration, where a large number of end-users, who should definitely be empowered, are left with the burden of using software systems of a high degree of complexity, often implying a great amount of training, without having the opportunity of communicating their own requirements in a disciplined manner, because of the lack of user-centred software design and development methodologies. Again, the issue of so-called legacy, waterfall software life cycle methodology and support tools, in particular with the amount of investment which they have received during the past few years, is also to be considered to remove economic and technical constraints for enabling the evolution towards new productive models such as Rapid Application Development, Rapid Prototyping or even Participatory Design, at a large industrial scale.

Background

During the past years, the market segment of the Public Administration and Government software providers has undertaken dramatic changes. With the downsizing and rightsizing revolution and the evergrowing market share for the open distributed computing architecture, the large, custom, often centralized systems, were surpassed by a set of integrated solutions, which usually include components off -the- shelf. Competion between custom software and market sofware was also driven by evergrowing user empowerment.

The centralized custom software systems were usually accompanied by massive use of standard business computer language such as COBOL, possible proprietary methodology and support tools which emphasised the "waterfall" approach, and less than ever attention to the ergonomic and human interface aspects as well as user documentation efficacy. The latter point be explained by the fact that, for such big and monolithic systems, a great deal of training was already planned for end-users as they were seen to stay on that particular job for long time, with lowest possible turnover rate, and be justified by the continuity and stability of technology.

Mostly, end-users of such bureaucratic software systems were mainly seen as doing repetitious tasks such as data-entry from bureaucratic forms, and sophisticated exploratory user interfaces were even considered a distracting factor and therefore avoided. Mainframe screens on old VDT were and are still the norm in any large or small Government office.

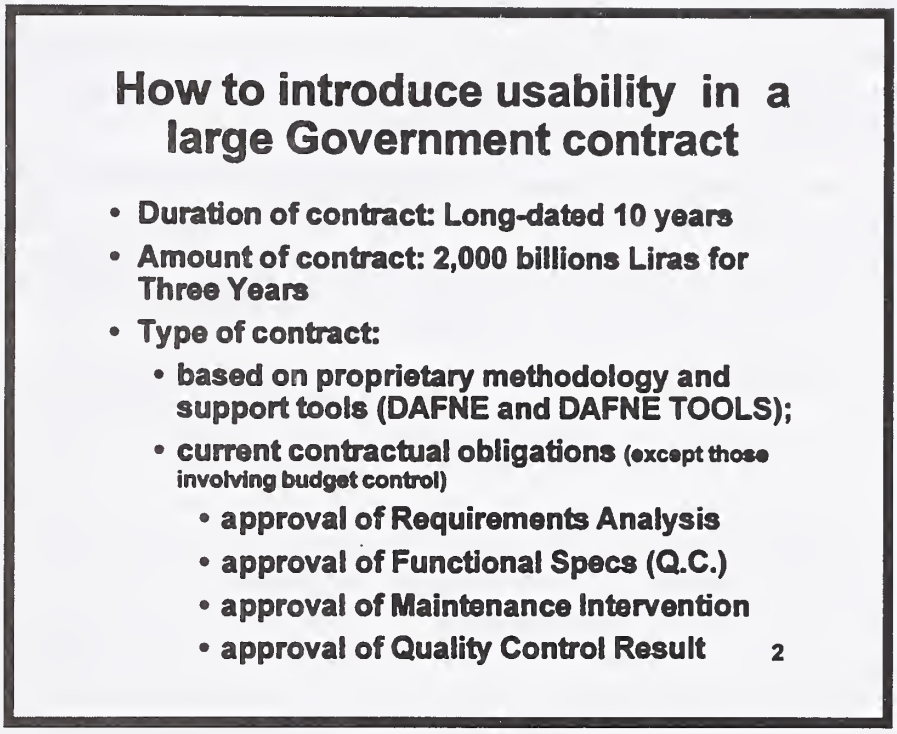


Figure 1

Great investment were made by the largest software providers in tailoring proprietary development methodologies to procurement and contract issues with less emphasis on the actual end-user satisfaction, acceptance and hence welfare. In the later years, with the fastest pace of the technological changes, the international norms on quality processes, software product evaluation and ergonomic of human computer dialogue and above all the sudden impact of Total Quality Management theories and practices, which were based on the principle of "getting it right getting it better", centralized, training-intensive, heavy software systems started being replaced by a set of optimally integrated computer and network solutions with the massive use of personal computing. These solutions were, as much as possible, based on the different user population requirements, and with the consequence of developing software system with the highest degree of operativity, with little or no effort in learning and with readable and really immediately useful user documentation.

Training, as well as user documentation and hence what it is called software support, started being considered as important factors in the overall budget reduction goal and therefore better analysed.

From a management point of view, productivity of end-users also become the central issue and software systems were only one of out several other potential factors for improving productivity.

From a user point of view, acceptance of software system, satisfaction in the interaction, become more than an issue and people started requesting better interaction with such systems, driven also by the alternative market offers.

Again, a better end-user/developer communication become the crux for improving the so-called "time to market," with the offer of new functionalities in an incremental fashion, and reduced development time and early evaluation phases.

Nowadays, the real challenge for the future resides mainly in understanding how to evolve the traditional software life cycle into a usability life cycle within a cost-justifying usability approach. It will include extensive data collection to model and assess current processes and continuous measurements of the benefits of introducing human factor issues in the traditional software design and development.

But since it is quite difficult for the sofware system providers to have enough strenght to change the process and product, the involvement of the client and end-user appear to be mandatory. This is why a Total Quality Management approach is so beneficial: if usability and end-user acceptance is agreed to be a quality factor, amongst others such as realiability or portability, at least initially for the most strategic projects, the client will have a coherent framework in which to calculate and weigh some of the added value (and possible added costs) of having a more usable software system. If we, as industrial researchers, are able to demonstrate that usability is not only a quality factor with its appropriate methods and techniques, but, above all, a cost reduction factor in terms of reduced training cost and reduced operational time with improved end-user performance and productivity needed and are able to quantify it to the client, we will be in the best position to start off the user centred revolution in the Government contracts. Government contractors have got even a better chance: that is to educate the citizens to claim for usable and less prone to error software systems. In fact the citizens are the ultimate users of such systems and they may also push for better and less expensive Governmental systems.

This is a case of introducing usability as a quality factor in the Software Quality system offered by a contractor to a Government Body or Agency. It implies as a first step getting the client to recognize the importance of such quality factor for his own business.

Usability should be recognized in Governmental activities very important since the ultimate users in our cases are not consumers but citizens who pay taxes for having back usable and effective services.

This calls for quality plans which together with the official statement of usability also contains the means to achieve the quality by:

- designing usability into the software system;
- testing the usability into the software system;

- monitoring the usability thoughout the software development;
- assessing usability while the system is in use;
- getting prompts and feedbacks from users;

and doing all this through cost-effective means.

There are at least four important aspects to be considered in order to be successful in this enterprise:

1. the availability of technical norms and regulations which stimulate and support the introduction of usability principles in the processes and products;
2. the availability of updated software development methodologies which embody usability principles;
3. the availability of public procurement rules which enhance the role of the end-users as requirements providers and usability testers.
4. the availability of client and users in more responsible and responsive roles, with the greatest commitment for the global quality of the system.

The norms

International and European technical norms such as EN ISO 9000: quality system, EN ISO 9001/1994: sw design, development, installation and assistance, EN ISO 9003/1994: control and verification , EN ISO 10011/1994: Inspection, EN ISO 9126: Evaluation of sw product quality and EN ISO 9241: Ergonomic Requirements for VDT used in office tasks are all being included in any well managed software process. Relevant legislation at the European Level: such as EEC Council Directive 92/50: "certification of quality for Large Government Contract" and EEC Council Directive 90/27: ergonomy and safety of VDT used for office tasks are increasing awareness in both software providers and users that usability is of paramount importance for improving quality, productivity and time to market. Furthermore, the procedure of certifying software providers on the basis of the quality of their processes, as it is implied by the application of ISO 9000 and related technical norms, is really pushing towards continous process improvements, as it is also testified by the European Initiative called SPICE (Software Process Improvement and Capability Determination), which will need to include human factor issues in software development process. Besides, at the Italian Level, the institution of a Public Authority for Governmental Software Projects has generated great expectations among the users: main objectives are in fact to guarantee a common interface access to public data through the adoption of efficient and interconnected networks. In fact Governmental Information is currently contained in several databases and databanks which are already in most cases interconnected. This information is highly relevant for many sectors of life and economy (ranging from business, professionals, information brokers, publishing company, general public). There is an urgent need in Administrations for a standardization and interchange of data as well as improved access and communication across all levels of governments and transparent communication with the general public or professional associations. A Legislative Decree (DLGS 626/94) on "Ergonomy and Safety for VDT usage" for the

first time in Italy has mentioned the relevance of the the adequacy of the developed software to the actual user task, the ease of learning, the provision of feedbacks to users at any step of the interaction, the speed at which the information should be displayed and presented, etc.., thus paving the way for a progressive introduction of usability principles into the traditional software development process.

The procurement rules

One of the major issues which European Central Governments are facing concerns rules for Public Procurements which are based on a novel relationship between client and provider. To this purpose, EU sponsored initiative such as EUROMETHOD works in the direction of making such relation as much transparent as possible to benefit the open market. In fact, the open market for information systems within the European Union requires a good mutual understanding between costumers and suppliers and Euromethod provides a set of common concepts and requirements for a potential harmonisation of different methodologies, outlook, training and culture to the purpose of imporving the quality and efficiency of the information system development process by promoting flexibility and adaptiveness of such methods. It is very strategic for people involved in such process of public procurement that the development methodologies are adequate to the usability engineering life-cycle. This implies that the conceptual model behind the adoption of EUROMETHOD as a procurement guide should also include the relevant deliverable for User Centred Design, insofar having a beneficial influences on all these European methodologies both Governmental and proprietary, such as SSADM, DAFNE, MEIN, SDM, V-MODEL, MERISE. These methodologies will be then forced to revise their own methodological products so as to include usability specifications to guarantee adequate coverage of user requirements according to the rules and practices of usability. Within the European market, the role of European Government as regulatory body for the Information System Procurement and Development Management is crucial. Another important European Body which may be very influencial on such stable and well founded methodologies is the European Software Institute, a major industrial initiative, supported by the European Unione and founded by fourtheen leading European companies, which, through liasions with the Software Engineering Institute, may also act as promoter of human factors in software development and use as much as SEI is actively involved in such matters.

Development methodologies and contractual documents

The type of specifications provided by the Government are usually very generic and do not mention the usability of the software system. Historically the development of custom software systems was depending from high level specifications, provided by high level senior personnel with little or no attention for actual end-users. The traditional process was then carried out in a bureaucratic "top down fashion", adopting and respecting phases and activities of a methodology and with the help of development tools and quality control procedures. This top down approach was also favoured by a lack of awareness and maybe culture by the side of the client and users.

Major emphasis was given to the bureaucratic aspects of the coding process and of the quality control, mainly enforcing the coverage of requirements, usually without evaluating and tracing user requirements in the actual context of use.

Since quality "at low cost" calls for an early detection of anomalies through early evaluation and test cycles, and quality control procedures become not only a contractual obligation but a crucial step towards the verification and acceptance of the whole software system, more and more attention is being paid to end-user requirements gathering and engineering.

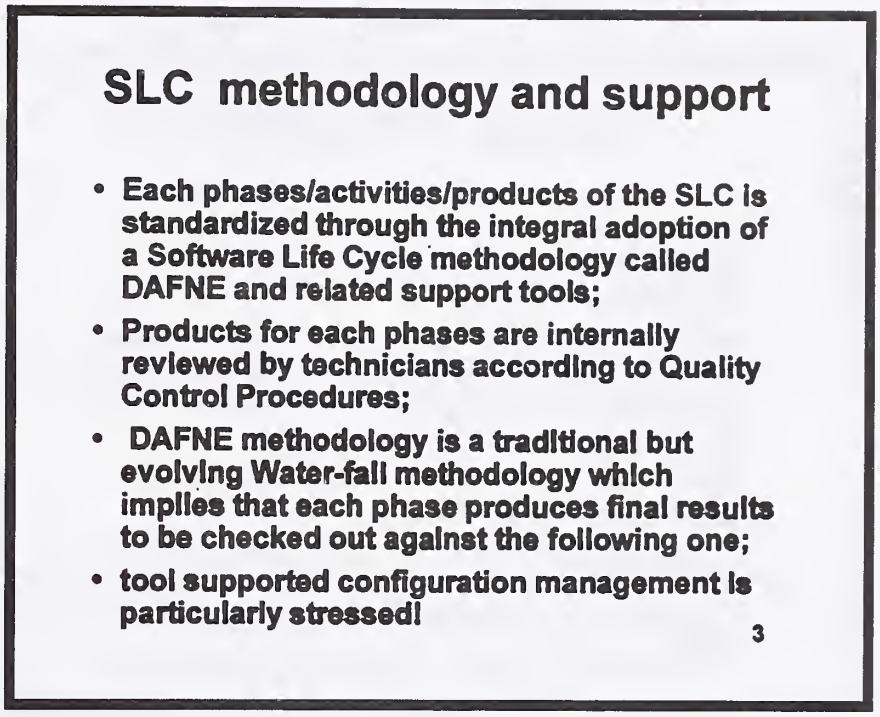


Figure 2

Another influencial factor which is contributing to put Client and Provider relation in the new perspective of usability, is the application of the method or practice of Function Point which is going to be widely adopted as dimensional metric for application development. At the level of Function Point, the user is also involved to analyse the impact and/or weight of a certain interaction modality choice onto the cost of the specific function point (eg. efficiency of end-user interaction), and this issue will definitely contribute to the development of an awareness of the issues related to user interface. During software development there are phases/activities when a specific product, which may be considered as an important milestone for the overall software project, is issued. The redefinition and renegotiation of such documents are seen as a first step towards the acceptance of a usability life cycle. A related issue to be considered is how to track requirements from the specification to the coding and test phases so as to guarantee that usability considerations, hopefully coming directly from the end-users, are not lost during the progressive formalization. In the case which has been presented here, each phase/activity/product of the internal software life cycle is standardized through the

integral adoption of a proprietary methodology and related support tools widely used in Governmental Software projects in Italy. Furthermore, products for each phases are internally reviewed by developers according to Quality Control Procedures. These Quality control procedures were not intended at the very beginning to involve end users and were mainly carried out for contractual purposes. The DAFNE methodology [copyright FINSIEL SpA] is a traditional but ever evolving water-fall methodology which implies that each phase produces final results to be checked out against the following phase.

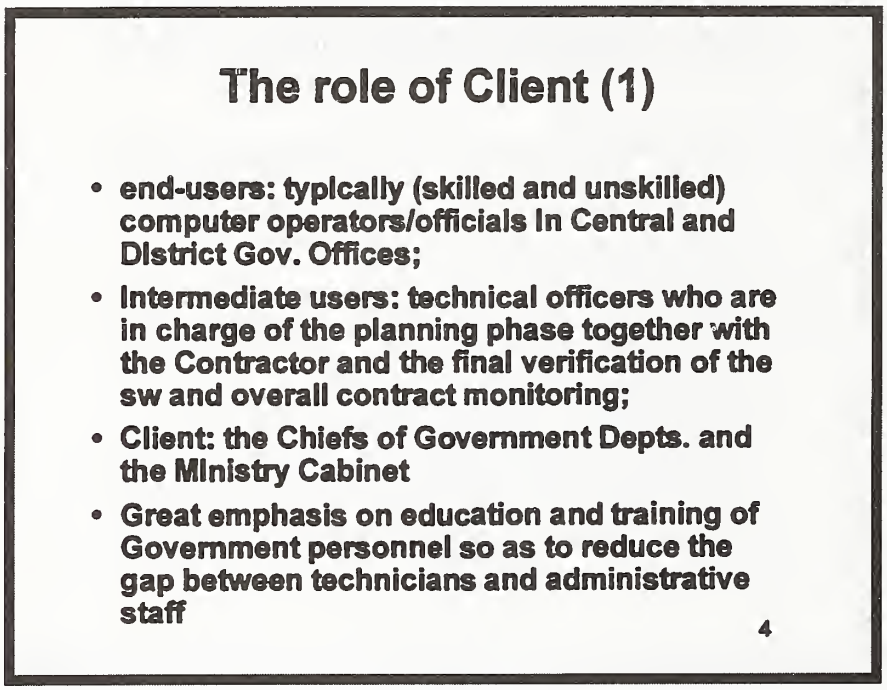


Figure 3

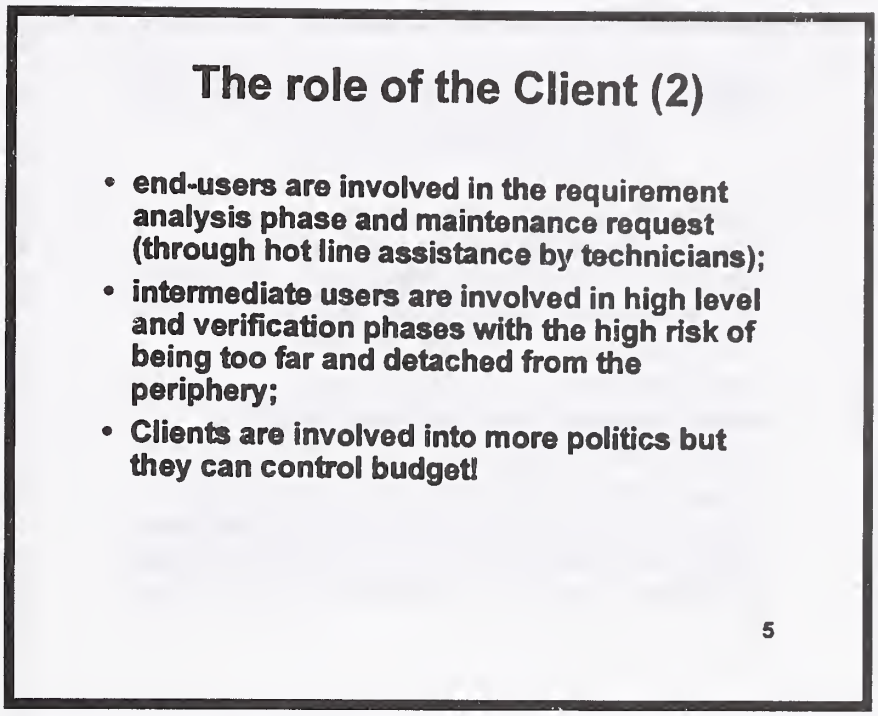


Figure 4

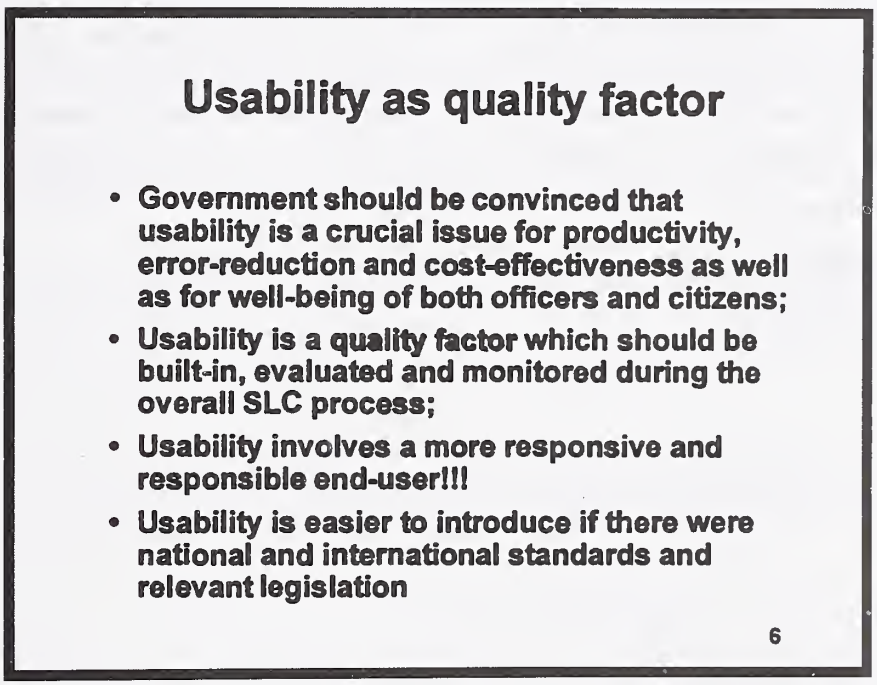


Figure 5

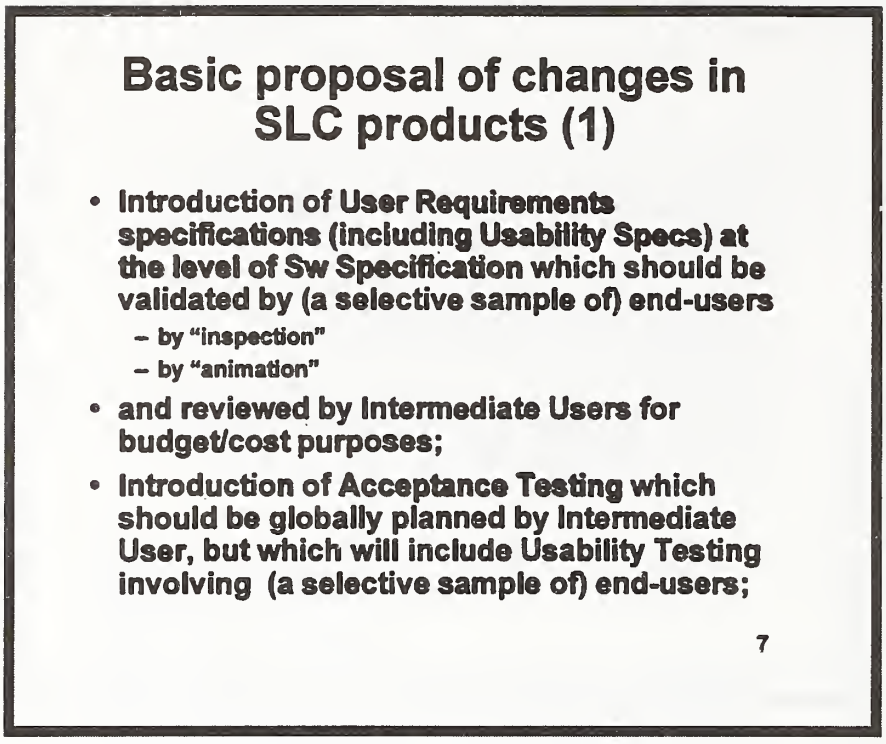


Figure 6

Indeed, with the Total Quality Management issue, these Quality Control procedures begin to be an essential part of the contract thus involving directly the client/user. The first objective of a plan for introduction of usability as quality factor in the software production process is the redefinition of two basic documents in the software life cycle:

- User requirement definition (as evolution of the so-called non-functional requirements);
- Usability Specs (as part of the Quality Specs for the products)

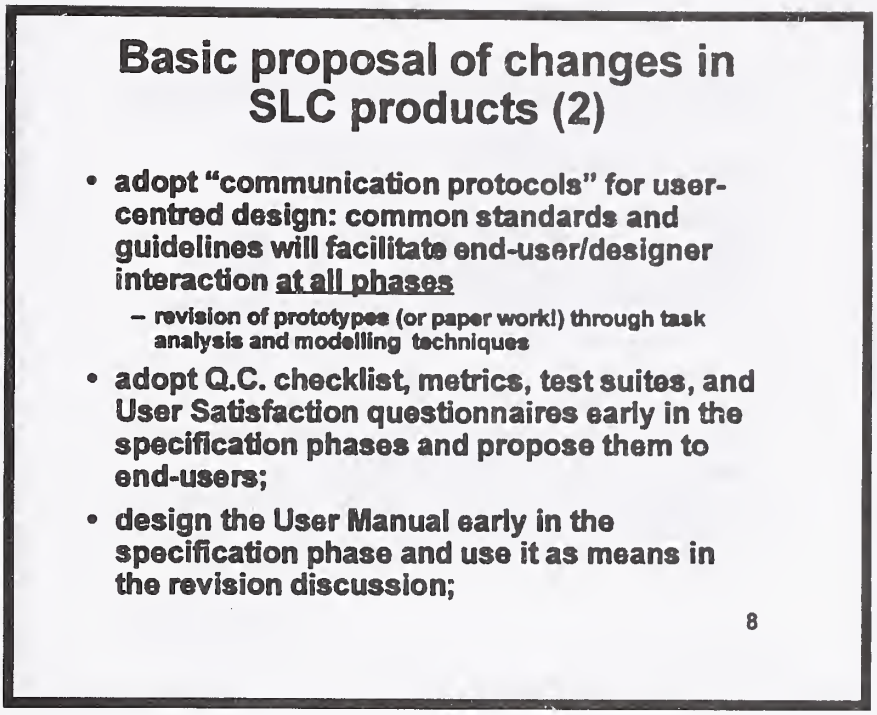


Figure 7

The former document contains the description of the context of use, type and frequency of tasks, type of users, impact on job organisation, etc.. and it will possibly contain formal notation for task description. The latter document contains, for a quality factor such as usability, a set of methods and techniques for testing, evaluating, assessing and thus guaranteeing the standard level of usability for that particular project. The validation or quality control of such documents involves the participation of (a selective sample of) end-users and reviewed by high-level for budget/cost purposes;

In the testing phase, the latter document will be accessed and usability tests will be constructed and documented in an internal report which will then be integrated into the Verification Report. Before the verification phase, all quality characteristics which are considered to be relevant for the objective and the measures are verified and then registered in the official verification report.

Acceptance Testing is globally planned by representative of the Client prior to the actual verification stage and in case it includes Usability Testing, it will be carried out with the involvment of (a selective sample of) end-users.

Role of ergonomists

- Need for International Curriculum:
- "Towards a European Registration Model for Ergonomists", Leiden University, 1992
  - Education, Training, Experience:
    - » Human Characteristics, Work Analysis and Measurement, People and Technology;
  - Registration Bodies (Register Committees, ergonomics societies, Eur.Erg. Title)
- "New Directions in HCI Education, Research and Practice, NSF/ARPA,1994"
  - Nature of Human-Computer Interaction, Use and Context of Computers, etc..
  - other Course Syllabus at several US Universities

9

Figure 8

Standards and legislation: the key for usability!

- At ISO Level:
  - EN ISO 9000: quality system
  - EN ISO 9001/1994: sw design, development, installation and assistance
  - EN ISO 9003/1994: control and verification
  - EN ISO 10011/1994: Inspection
  - ISO 9126: Evaluation of sw product quality
  - EN ISO 9241(1,2,3) : Ergonomic Requirements for VDT terminal used in office tasks
- At European Level:
  - EEC Council Directive 92/50: certification of quality for Large Government Contract
  - EEC Council Directive 90/27 0: ergonomy and safety of VDT used for office tasks

10

Figure 9

Standards and legislation: the key for usability! (2)

- EPHOS (Handbook v.1 1994):
  - European Procurement Handbook for Open Systems
  - guidance for Public Procurers within Member States when procuring information Technology and Telecommunications
  - Relevant EPHOS Module: Virtual Terminal v. 1.0 (1994): Forms profile, Paged Profile, Telnet Profile
- EUROMETHOD (Phase 3 1994):
  - CEE Recommended Methodological Framework (including DAFNE) for project planning, procurement and management, technical documentation for public procurement- key product is requirements specs + interface specs!
- ISO/CEN SC7:
  - Information System Engineering Concepts and Model

11

Figure 10

Standards and legislation: the key for usability! (3)

- ESI
  - European Software Institute (Our major holding is full member!)- Process Assessment and Improvement including Human Factor Issues-Learning Organisation Project
- At the Italian Level
  - Institution of the Authority for Public Administration;
  - Contractual Monitoring;
  - Legislative Decree: DLGS 626/94 on Ergonomy and Safety for VDT usage for Office Tasks:
    - » sw should be adequate to the task
    - » sw should be easy to learn or adaptable to the user
    - » sw system should always provide feedbacks
    - » sw systems should present/handle info at human speed

12

Figure 11

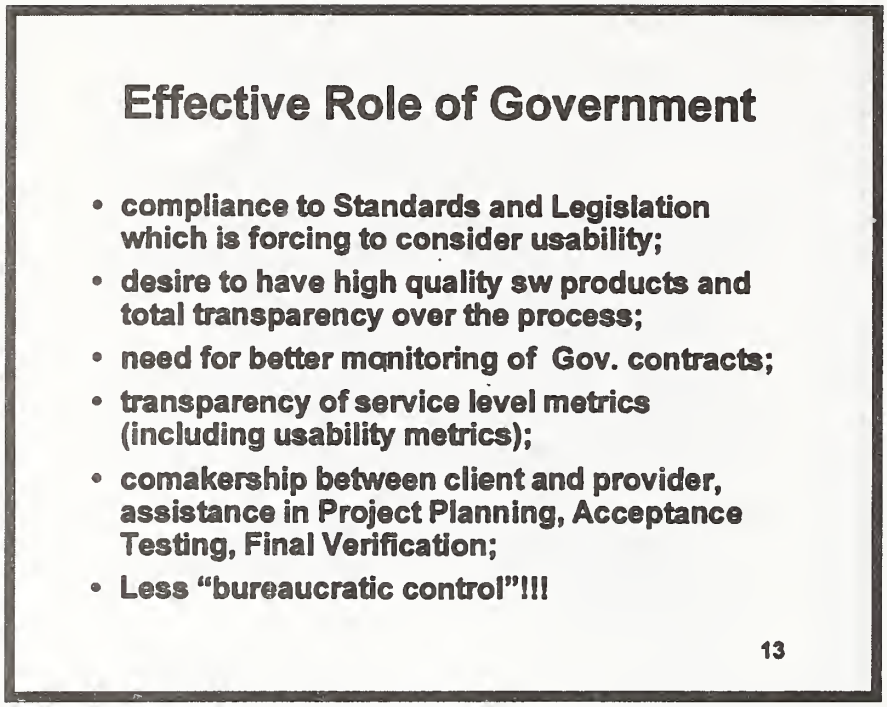


Figure 12

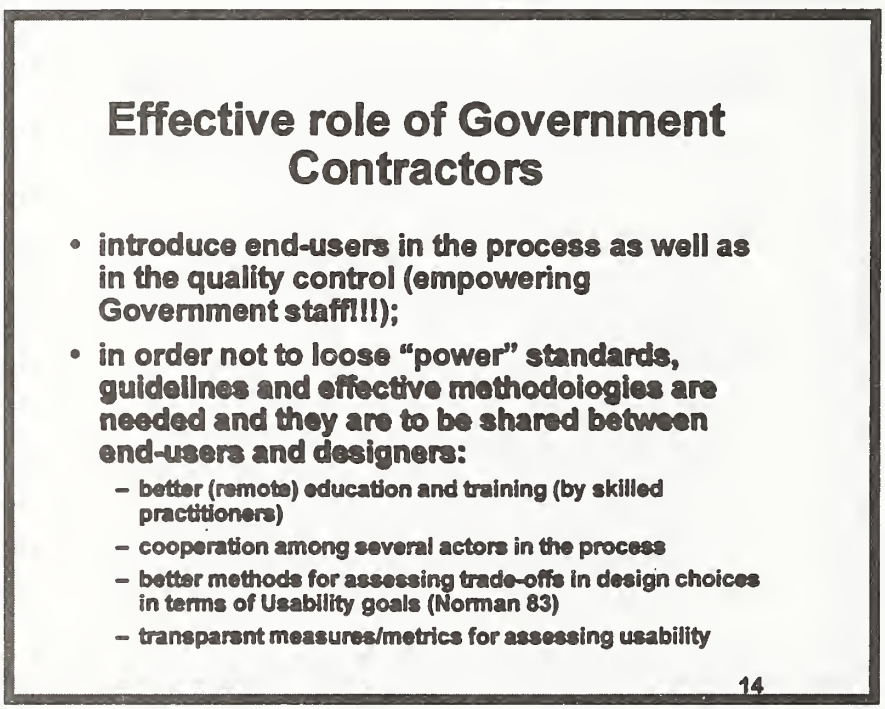


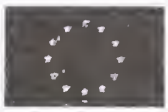
Figure 13

The role of European Union

Progetto MAPI (Music Assisted Process Improvement):

Budget complessivo = 1MECU;

Budget SOGEI: 150,000 ECU



Consortium:

Lloyds Bank-UK (Coordinator)

SOGEI -I (full Partner)

AMS -S (Associate)

Ericsson-S (Associate)

Inland Revenue Service-UK (Associate)

15

Figure 14

The role of European Union

Consortium (cont'):

CYCO-NL (Associate)

Berlin State Museum-G (Associate)

Brameur-UK (Subcontractor)

National Physics Lab-UK (Subcontractor)

Human Factor Research Group -IR (Subcontractor)

WITLab-NL (Subcontractor)

NOMOS-S (Subcontractor)

DATA Management (Subcontractor)

16

Figure 15

The MUSiC method

Progetto ESPRIT 5429 (1990-1993)
Name: Measuring USability In Context
Partners:

– BRAMEUR Ltd	UK
– Data Management S.p.A.	Italia
– SEMA Group sae	Spagna
– National Physical Laboratory	UK
– HUSAT Research Institute	UK
– University College Cork	Irlanda
– Technische Universiteit Delft	Olanda
– WWU University Munster	Germania

17

Figure 16

MUSIC tools

- Usability Context Analyser
- Performance Measurements
  - » (DRUM Video Analysis Supp. tool)
- Subjective User Attitude
  - » SUMI Software Usability Measurement Inventory (Learnability, Efficiency, Affect, Control and Helfulness)- SUMISCORE
  - » Cognitive Workload
  - » Task Load Index
  - » Subjective Mental Effort Questionnaire (SMEQ)

18

Figure 17

New Government Contract proposal (under approval)

- User Requirement Document and Context of Use Definition according to ISO/CD 9241_11
- Usability specs: target value and metrics;
- Definition of Quality Profile for Software projects which may include Usability as one of the quality factors;
- Prototype stage with evaluation sessions:
  - for better elicitation of user requirements;
  - for better understanding of non-functional requirements;
- Better traceability of Usability Requirements throughout the Software Life Cycle;

19

Figure 18

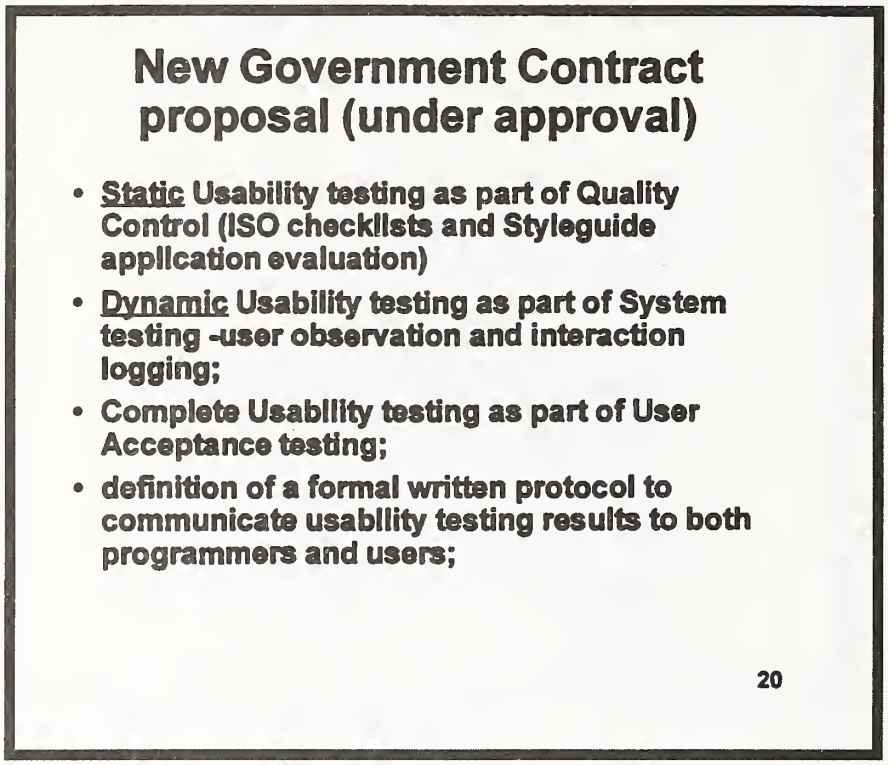


Figure 19

Crucial importance is ascribed to the adoption of so-called "communication protocols" for user-centred design, as much as common standards and guidelines enormously facilitate end-user/designer interaction at all phases of:

- revision of prototypes (or paper work!) through task analysis and modelling techniques
- discussion of usability aspects early in at the specification phase;
- Q.C. checklist, metrics, test suites, and User Satisfaction questionnaires early in the specification phases;
- design the User Manual early in the specification phase and use it as means in the revision discussion;

The user typology

Furthermore, the role of the client should be better specified since, in bureaucratic organisation, people responsible for the budget are usaully centralised and do not usually care too much about the actual end-user. Therefore there is a strong need of empowering end-users (especially at the local and pheripheral level) and motivate them to learn about methods and techniques for requirement gathering, usability testing and global product revision. The major issue is that of disciplining the relationship with the user so as to get a cost-effective involvement. Starting from Central Office which will deal with the

phases of tender and procurement of the system, to the Technical Information Centers which deal with contract monitoring, to the actual end-user of the Central and District Offices which will provide requirements and observations. There is a deep training issue here:

end-users: typically (skilled and unskilled) computer operators/officials in Central and District Gov. Offices; Intermediate users: technical officers who are in charge of the planning phase together with the Contractor and the final verification of the sw and overall contract monitoring; Client: the Chiefs of Government Depts. and the Ministry Cabinet.

Great emphasis on education and training of Government personnel so as to reduce the gap between technicians and administrative staff. End-users are also usually involved in the requirement analysis phase and maintenance request while intermediate users are involved in high level and verification phases with the high risk of being too far and detached from the periphery.

Conclusions

Government should be convinced that usability is a crucial issue for productivity, error-reduction and cost-effectiveness as well as for well-being of both officers and citizens. Usability is a quality factor which should be built-in, evaluated and monitored during the overall SLC process and involves a more responsive and responsible end-user as well as a renegotiation of contractual product and a discipline for user involvement.

Currently Government Agencies need also to be compliant to European Standards and Legislation which is forcing to consider usability in the process and in the product and since they have already invested in a methodological framework for software system development, this investment (both in terms of skills and knowledge and economic resources) should be preserved but at the same time these development methods/techniques should be updated in terms of documents, formal notations, etc... Moreover and because of reduction of the allocated budget, there is a growing need for better monitoring Governmental contracts, both from an economical and from an user acceptance point of view. This also implies that services offered should continously be monitored in a transparent way, including usability metrics both for process and products. Total Quality Management theories and practices imply a "comakership" between client and provider, which includes assistance in Project Planning, Acceptance Testing, Final Verification with less purely "bureaucratic control" and more "involvement and commitment in the overall process". Government Contractors should introduce end-users in the development process as well as in the quality control and, in order not to loose business revenues, standards, guidelines and effective methodologies are needed to be shared as common grounds between end-users and designers implying better (remote) education and training (by skilled practitioners), cooperation among several actors in the process, better methods for assessing trade-offs in design choices in terms, and transparent measures/metrics for assessing usability.

* U.S. GOVERNMENT PRINTING OFFICE: 1997-417-392/72673

ANNOUNCEMENT OF NEW PUBLICATIONS ON INFORMATION TECHNOLOGY

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500–.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published bimonthly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program in support of the efforts of private-sector standardizing organizations.

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and nongovernment). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Institute of Standards
and Technology
Gaithersburg, MD 20899–0001