

HUMAN COMPUTER INTERACTION

(CS408)

TABLE OF CONTENTS:

LECTURE 1.	8
INTRODUCTION TO HUMAN COMPUTER INTERACTION – PART I	8
LEARNING GOALS.....	8
1.1 RIDDLES FOR THE INFORMATION AGE	10
1.2 ROLE OF HCI.....	14
LECTURE 2.	15
INTRODUCTION TO HUMAN-COMPUTER INTERACTION – PART II	15
LEARNING GOALS.....	15
2.1 DEFINITION OF HCI.....	15
2.2 REASONS OF NON-BRIGHT ASPECTS	15
2.3 HUMAN VERSES COMPUTER.....	18
2.4 SOFTWARE APARTHEID.....	19
LECTURE 3.	22
INTRODUCTION TO HUMAN-COMPUTER INTERACTION – PART III	22
LEARNING GOALS.....	22
3.1 AN INDUSTRY IN DENIAL	23
3.2 TECHNO-RAGE	24
3.3 SUCCESS CRITERIA IN THE NEW ECONOMY.....	26
3.4 COMPUTER + INFORMATION	28
REFERENCES	29
LECTURE 4.	30
GOALS & EVOLUTION OF HUMAN COMPUTER INTERACTION	30
LEARNING GOALS.....	30
4.1 GOALS OF HCI	30
4.2 EVOLUTION OF HCI	34
LECTURE 5.	39
DISCIPLINE OF HUMAN COMPUTER INTERACTION	39
LEARNING GOALS.....	39
5.1 QUALITY	39
5.2 INTERDISCIPLINARY NATURE OF HCI	41
REFERENCE:.....	45
LECTURE 6.	46
COGNITIVE FRAMEWORKS	46
LEARNING GOALS	46
6.1 INTRODUCTION.....	46
6.2 MODES OF COGNITION	48
6.3 HUMAN PROCESSOR MODEL	49
6.4 GOMS.....	50
6.5 RECENT DEVELOPMENT IN COGNITIVE PSYCHOLOGY	51
6.6 EXTERNAL COGNITION	52
6.7 DISTRIBUTED COGNITION.....	53
LECTURE 7.	54
HUMAN INPUT-OUTPUT CHANNELS – PART I	54
LEARNING GOALS.....	54
7.1 INPUT OUTPUT CHANNELS	54

7.2	VISION.....	55
7.3	VISUAL PERCEPTION	57
LECTURE 8.		62
HUMAN INPUT-OUTPUT CHANNELS PART II		62
	LEARNING GOALS.....	62
8.1	COLOR THEORY	62
8.2	STEREOPSIS	66
8.3	READING	69
8.4	HEARING	69
8.5	TOUCH.....	71
8.6	MOVEMENT	72
LECTURE 9.		76
COGNITIVE PROCESS - PART I		76
	LEARNING GOALS.....	76
9.1	ATTENTION	76
9.2	MEMORY	79
9.3	REVISED MEMORY MODEL	84
LECTURE 10.....		85
COGNITIVE PROCESSES - PART II.....		85
	LEARNING GOALS.....	85
10.1	LEARNING	85
10.2	READING, SPEAKING AND LISTENING.....	87
10.3	PROBLEM SOLVING, PLANNING, REASONING AND DECISION-MAKING	88
LECTURE 11.....		93
THE PSYCHOLOGY OF ACTIONS		93
	LEARNING GOALS.....	93
11.1	MENTAL MODEL.....	93
11.2	ERRORS	100
LECTURE 12.....		101
DESIGN PRINCIPLES.....		101
	LEARNING GOALS.....	101
12.1	DESIGN PRINCIPLES	103
LECTURE 13.....		109
THE COMPUTER.....		109
	LEARNING GOALS.....	109
13.1	INPUT DEVICES	109
13.2	TEXT ENTRY DEVICES	110
13.3	POSITIONING, POINTING AND DRAWING.....	115
13.4	DISPLAY DEVICES.....	118
13.5	TOUCH, FEEL AND SMELL	120
13.6	PHYSICAL CONTROLS	120
13.7	ENVIRONMENT AND BIO SENSING	120
LECTURE 14.....		121
INTERACTION.....		121
	LEARNING GOALS.....	121
14.1	THE TERMS OF INTERACTION	121
14.2	DONALD NORMAN’S MODEL	121
14.3	THE INTERACTION FRAMEWORK	122
14.4	FRAMEWORKS AND HCI	123

14.5	INTERACTION STYLES.....	127
LECTURE 15.....		130
INTERACTION PARADIGMS.....		130
	LEARNING GOALS.....	130
15.1	THE WIMP INTERFACES	130
15.2	INTERACTION PARADIGMS	134
LECTURE 16.....		141
HCI PROCESS AND MODELS.....		141
	LEARNING GOALS.....	141
LECTURE 17.....		148
HCI PROCESS AND METHODOLOGIES		148
	LEARNING GOALS.....	148
17.1	LIFECYCLE MODELS	149
17.2	LIFECYCLE MODELS IN HCI	152
LECTURE 18.....		157
GOAL-DIRECTED DESIGN METHODOLOGIES		157
	LEARNING GOALS.....	157
18.1	GOAL-DIRECTED DESIGN MODEL	157
18.2	A PROCESS OVERVIEW	158
18.3	TYPES OF USERS	162
LECTURE 19.....		166
USER RESEARCH PART-I		166
	LEARNING GOALS.....	166
19.1	TYPES OF QUALITATIVE RESEARCH.....	167
LECTURE 20.....		172
USER RESEARCH PART-II.....		172
	LEARNING GOALS.....	172
20.1	USER-CENTERED APPROACH	172
20.2	ETHNOGRAPHY FRAMEWORK.....	174
20.3	PREPARING FOR ETHNOGRAPHIC INTERVIEWS	178
20.4	PUTTING A PLAN TOGETHER.....	180
LECTURE 21.....		181
USER RESEARCH PART-III		181
LECTURE 22.....		185
USER MODELING		185
	LEARNING GOALS.....	185
22.1	WHY MODEL?	185
22.2	PERSONAS	186
22.3	GOALS	191
22.4	TYPES OF GOALS	191
22.5	CONSTRUCTING PERSONAS	194
LECTURE 23.....		198
REQUIREMENTS.....		198
	LEARNING GOALS.....	198
23.1	NARRATIVE AS A DESIGN TOOL.....	198
23.2	ENVISIONING SOLUTIONS WITH PERSONA-BASED DESIGN	200

LECTURE 24.....	205
FRAMEWORK AND REFINEMENTS.....	205
LEARNING GOALS.....	205
24.1 DEFINING THE INTERACTION FRAMEWORK.....	205
24.2 PROTOTYPING	209
LECTURE 25.....	212
DESIGN SYNTHESIS.....	212
LEARNING GOALS.....	212
25.1 INTERACTION DESIGN PRINCIPLES	212
25.2 INTERACTION DESIGN PATTERNS	216
25.3 INTERACTION DESIGN IMPERATIVES	218
LECTURE 26.....	220
BEHAVIOR & FORM PART I	220
LEARNING GOALS.....	220
26.1 SOFTWARE POSTURE.....	220
26.2 POSTURES FOR THE DESKTOP	220
LECTURE 27.....	229
BEHAVIOR & FORM PART II.....	229
LEARNING GOALS.....	229
27.1 POSTURES FOR THE WEB.....	229
27.2 WEB PORTALS	230
27.3 POSTURES FOR OTHER PLATFORMS	231
27.4 FLOW AND TRANSPARENCY	233
27.5 ORCHESTRATION.....	236
LECTURE 28.....	245
BEHAVIOR & FORM PART III	245
LEARNING GOALS.....	245
28.1 ELIMINATING EXCISE.....	245
28.2 NAVIGATION AND INFLECTION	248
LECTURE 29.....	256
EVALUATION – PART I.....	256
LEARNING GOALS.....	256
29.1 EVALUATION PARADIGMS AND TECHNIQUES	258
LECTURE 30.....	264
EVALUATION – PART II	264
LEARNING GOALS.....	264
30.1 DECIDE: A FRAMEWORK TO GUIDE EVALUATION.....	264
LECTURE 31.....	270
EVALUATION – PART VII	270
LEARNING GOALS.....	270
LECTURE 32.....	279
EVALUATION IV	279
LEARNING GOALS.....	279
32.1 SCENE FROM A MALL	279
32.2 WEB NAVIGATION	281

LECTURE 33.....	294
EVALUATION V	294
LEARNING GOALS.....	294
33.1 TRY THE TRUNK TEST.....	296
LECTURE 34.....	300
EVALUATION – PART VI.....	300
LEARNING GOALS.....	300
LECTURE 35.....	304
EVALUATION – PART VII.....	304
LEARNING GOALS.....	304
35.1 THE RELATIONSHIP BETWEEN EVALUATION AND USABILITY?.....	304
LECTURE 36.....	310
BEHAVIOR & FORM – PART IV	310
LEARNING GOALS.....	310
36.1 UNDERSTANDING UNDO.....	310
36.2 TYPES AND VARIANTS OF	312
36.3 INCREMENTAL AND PROCEDURAL ACTIONS.....	312
36.4 SINGLE AND MULTIPLE UNDO	312
36.5 OTHER MODELS FOR UNDO-LIKE BEHAVIOR.....	316
36.6 RETHINKING FILES AND SAVE	319
36.7 ARCHIVING.....	321
36.8 IMPLEMENTATION MODEL VERSUS MENTAL MODEL	322
36.9 DISPENSING WITH THE IMPLEMENTATION MODEL OF THE FILE SYSTEM.....	323
36.10 DESIGNING A UNIFIED FILE PRESENTATION MODEL	323
LECTURE 37.....	325
BEHAVIOR & FORM - PART V.....	325
LEARNING GOALS.....	325
37.1 UNIFIED DOCUMENT MANAGEMENT	325
37.2 CREATING A MILESTONE COPY OF THE DOCUMENT	328
37.3 ARE DISKS AND FILES SYSTEMS A FEATURE?	329
37.4 TIME FOR CHANGE.....	330
37.5 MAKING SOFTWARE CONSIDERATE.....	330
37.6 CONSIDERATE SOFTWARE IS POSSIBLE.....	336
37.7 MAKING SOFTWARE SMARTS:.....	336
37.8 PUTTING THE IDLE CYCLES TO WORK	336
37.9 GIVING SOFTWARE A MEMORY	337
37.10 TASK COHERENCE.....	338
37.11 ACTIONS TO REMEMBER	339
37.12 APPLYING MEMORY TO YOUR APPLICATIONS	340
37.13 MEMORY MAKES A DIFFERENCE	342
LECTURE 38.....	343
BEHAVIOR & FORM – PART VI	343
LEARNING GOALS.....	343
38.1 DESIGNING LOOK AND FEEL	343
38.2 PRINCIPLES OF VISUAL INTERFACE DESIGN	345
LECTURE 39.....	348
BEHAVIOR & FORM – PART VII.....	348
LEARNING GOALS.....	348
39.1 PROVIDE VISUAL STRUCTURE AND FLOW AT EACH LEVEL OF ORGANIZATION.....	348

39.2	PRINCIPLES OF VISUAL INFORMATION DESIGN	354
39.3	USE OF TEXT AND COLOR IN VISUAL INTERFACES.....	356
39.4	CONSISTENCY AND STANDARDS	358
LECTURE 40.....		361
OBSERVING USER.....		361
	LEARNING GOALS.....	361
40.1	WHAT AND WHEN TO OBSERVE.....	361
40.2	HOW TO OBSERVE.....	361
40.3	DATA COLLECTION	365
40.4	INDIRECT OBSERVATION: TRACKING USERS' ACTIVITIES	366
40.5	ANALYZING, INTERPRETING, AND PRESENTING THE DATA	367
LECTURE 41.....		370
ASKING USERS.....		370
	LEARNING GOALS.....	370
41.1	INTRODUCTION.....	370
41.2	ASKING USERS: INTERVIEWS.....	370
41.3	ASKING USERS: QUESTIONNAIRES	374
41.4	ASKING EXPERTS: WALKTHROUGHS	379
LECTURE 42.....		381
COMMUNICATING USERS.....		381
	LEARNING GOALS.....	381
42.1	ELIMINATING ERRORS	381
42.2	POSITIVE FEEDBACK.....	385
42.3	NOTIFYING AND CONFIRMING	387
42.4	ALERTS AND CONFIRMATIONS.....	387
42.5	ELIMINATING CONFIRMATIONS	390
42.6	REPLACING DIALOGS: RICH MODELESS FEEDBACK.....	391
42.7	RICH VISUAL MODELESS FEEDBACK	391
LECTURE 43.....		393
INFORMATION RETRIEVAL		393
	LEARNING GOALS.....	393
43.1	AUDIBLE FEEDBACK.....	393
43.2	OTHER COMMUNICATION WITH USERS.....	395
43.3	IMPROVING DATA RETRIEVAL	401
LECTURE 44.....		406
EMERGING PARADIGMS.....		406
	LEARNING GOALS.....	406
44.1	ACCESSIBILITY	408
LECTURE 45.....		413
CONCLUSION		413
	LEARNING GOALS.....	413
45.1	WEARABLE COMPUTING	414
45.2	TANGIBLE BITS	416
45.3	ATTENTIVE ENVIRONMENTS.....	418

Lecture 1.

Introduction to Human Computer Interaction – Part I

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

Answer what is the significance of Human Computer Interaction (HCI)

Discuss and argue about why Human computer Interaction (HCI) is important with reference to the way in which technology has developed during past forty years

Describe a formal definition of HCI.

At the end of this lecture you will be told about the course contents. This will be a brief overview of the topics that we will discuss in this course and the structure of the course.

Run for your lives---invasion has begun---the computers are invading.

Now it is twenty first century and during the past thirty years technology has advanced to such an extent that almost everyone come in contact with computers in one way or another. Look around yourself how many things are there which have some kind of computer embedded in them? Think about a minute about what you use in a typical day; ATM, cell phone, VCR, remote control, ticketing machine, digital personal organizers, calculator, watch, photocopier, toaster, bank, air conditioner, broadcasting, satellite, microwave, medical equipment, factories, companies....the list is endless. Computers are everywhere. We are surrounded by computers. Now they are part of our everyday life.

Traditional notion of computers is no more. Unlike in the early days of computing, when only highly skilled technical people used computers, nowadays the range of knowledge and experience of different users is very broad. Computers are no more just on your table. Now computer has become a tool of everyday use. They are everywhere, at everyplace and in everything. They are penetrating in every aspect of our life. They are taking our lives.

When computers first appeared on the commercial scene in the 1950s, they were extremely difficult to use, cumbersome and at times unpredictable. There were a number of reasons for this;

They were very large and expensive machines, so that by comparison human labor (that is, 'people time') was an inexpensive resource.

They were used only by technical specialists – scientists and engineers – who were familiar with the intricacies of off-line programming using punch cards.

Little was known about how to make them easier to use.

None of these conditions holds today: computers have become much less expensive, users come from every walk of life, and we understand a great deal more about how to fit the machines to people's needs and their work.

Dramatic decrease in the cost of computing resources have resulted from new technological advances, the most significant being the development of the silicon chip. The ability not only to miniaturize circuits but also to pack large number of them on to tiny, individual chips paved the way for his development of powerful computers with large storage capacity. In less than thirty years computers changed from being huge machines housed in large, air-conditioned rooms to much smaller machines, including some that can easily be carried around by children. Computers have also become more reliable and today's machines do not suffer from overheating like their ancestors. Computing has entered a new era and is becoming ubiquitous.

The development of the first personal computers in the 1970s was a major landmark because these machines provided interactive computing power for individual users at low cost. Consequently, instead of just a handful of highly experienced programmers being the only users, people from all walks of life – commerce, farming, education, retailing, defense, manufacturing and entertainment – began using computer systems. Computers are becoming increasingly POWERFUL.

Computers are performing more and more TASKS. These changes in technology have opened up a wide range of new possibilities for the way in which computers can be used. The sheer costliness and time required to run programs on the early machines dictated the kinds of commercial application in which computers could be used. Business such as banking and accounting, with large-scale record keeping activities, were the first to take up computing technology. Companies that were involved in activities with 'fast' cycles, such as transaction processing for airlines and retailing, could not make use of these machines. They were not sufficiently fast or responsive, but this is not a problem with modern computers.

Computers have also found a place in many private homes. In fact, such has been their pervasiveness that now just about everyone, young or old, able or disabled, skilled or unskilled, is using or is directly affected by computers in one way or another. Machines are leaking into every aspect of lives. So now the concept of life, without computer is same as concept of life without electricity, and obviously it is hard to live without light as well as with computer!

Run for your lives---invasion has begun---the computers are invading.

As computers are penetrating in our daily life, it has some results. The bright side of this invasion is:

Computers are enabling new discoveries
Leading to efficiencies
Making our life easy and convenient

On the not so bright side the result is:

Computers are annoying us
 They are infuriating us
 They even kill a few of us.

In turn, we will be tempted to kill our computers, but we won't dare because we are already utterly, irreversibly dependent on these hopeful monsters that make modern life possible. So we will have to think about them. We will have to think how we can make them better. We need to fundamentally rethink how human and machines interact. And rethink the relationship in deep and novel ways, for the fault for our burgeoning problems lies not with our machines, but with us.

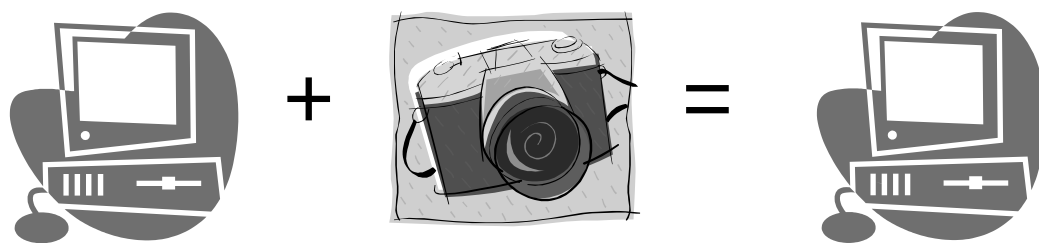
1.1 Riddles for the Information Age

What do you get when you cross a computer with an Airplane?



In December 1995, American Airlines Flight 965 departed from Miami on a regularly scheduled trip to Cali, Columbia. On the landing approach, the pilot of the 757 needed to select the next radio navigation fix, named "ROZO". He entered an "R" into his navigation computer. The computer returned a list of nearby navigation fixes starting with "R" and the pilot selected the first of these, whose latitude and longitude appeared to be correct. Unfortunately, instead of "ROZO", the pilot selected "ROMEO", 132 miles to the northeast. The jet was southbound descending into a valley that runs north-south, and any lateral deviation was dangerous. Following indications on the flight computer, the pilots began an easterly turn and slammed into a granite peak at 10,000 feet. One hundred and fifty two passengers and all eight crewmembers aboard perished. Four passengers survived with serious injuries.

What do you get when you cross a computer with a Camera?



Here is a riddle for the information age: what do you get when you cross a computer with a camera? Answer: A computer! Thirty years ago, a 35mm Pentax Model H, had a small

battery in it that powered the light meter. Like a wristwatch battery, I merely swapped in a new one every couple of years. Fifteen years ago, an electronic camera, a 35mm Canon T70, used two AA batteries to power its rather simple exposure computer and its automatic film drive. It had a simple On/Off switch, so that the batteries wouldn't wear down needlessly.

Five years ago, a first-generation digital camera, had a similar On/Off switch, but this time it had the smarts of a rudimentary computer inside it. So if I forgot to turn it off, it automatically shut down after one minute of inactivity.

One year ago, second-generation digital camera, a Panasonic PalmCam, had an even smarter computer chip inside it. It was so smart that its On/Off switch had evolved into an Off/Rec/Play switch. It now had modes: it had to put into Rec mode to take pictures and Play mode to view them on its small video display.

The newest camera, a Nikon CoolPix 900, is a third-generation digital camera and the smartest yet. In fact, it has a full-blown computer that displays a Windows-like hourglass while it "boots up". Like some mutant fish with extra heads, its On/Off switch has now grown to have four settings: Off/ARec/MRec/Play. "ARec" means "automatic record" and "MRec" means "manual record." as far as I can figure out how to turn it on without a lengthy explanation.

The new camera is very power-hungry, and its engineers thoughtfully provided it with a sophisticated computer program that manages the consumption of battery power. A typical scenario goes like this: I turn the evil off/etc. switch to "MRec," wait about seven long seconds for the camera to boot up, then point it at my subject. I aim the camera and zoom in to properly frame the image. Just as I'm about to press the shutter button, the camera suddenly realizes that simultaneously running the zoom, charging the flash, and energizing the display has caused it to run out of power. In self-defense, it suspends its ability to actually take pictures. But I don't know that because I'm looking through the viewfinder, waving my arms and saying "Smile" and pressing the shutter button. The computer detects the button press, but it simply cannot obey. In a misguided effort to help out, the power management program instantly takes over and makes an executive decision: shed load. It shuts down the power-greedy LCD video display. I look at the camera quizzically, wondering why it didn't take the picture, shrug my shoulders, and let my arm holding the camera drop to my side. But as soon as the LCD is turned off, there is more battery power available for other systems. The power management program senses this increase and realizes that it now has enough electricity to take pictures. It now returns control to the camera program, which is waiting patiently to process the command it received when I pressed the shutter button, and it takes a nicely auto-focused, well-exposed, high-resolution digital picture of my kneecap.

That old mechanical Pentax had manual focusing, manual exposure, and manual shutter-speed, yet it was far less frustrating to use than the fully computerized modern Nikon CoolPix 900, which has automatic focusing, exposure, and shutter-speed. Camera may still take pictures, but it behaves like a computer instead of a camera.

What do you get when you cross a computer with an alarm clock?



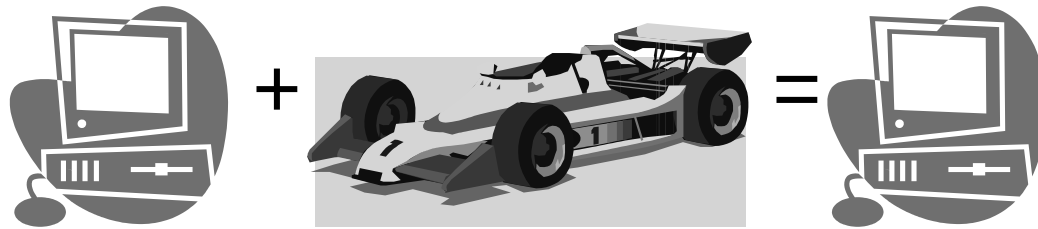
A computer! I just purchased an expensive new clock-radio for my bedroom, a JVC FS-2000. It has a very sophisticated computer brain, and offers high fidelity, digital sound, and lots of features. It wakes me up at a preset time by playing a compact disc, and it has the delicacy and intelligence to slowly fade up the volume when it begins to play at six o'clock in the morning. This feature is really pleasant and quite unique, and it compensates for the fact that I want to hurl the infuriating machine out the window.

It's very hard to tell when the alarm is armed, so it occasionally fails to wake me up on a Monday and rousts me out of bed early on a Saturday. Sure, it has an indicator to show the alarm is set, but that doesn't mean it's useful. The clock has a sophisticated alphanumeric liquid crystal display (LCD) that displays all of its many functions. The presence of a small symbol in the upper left corner of the LCD indicates the alarm is armed, but in a dimly lit bedroom the clock symbol is visible, but the backlight comes on when the CD or radio is explicitly turned on. There's a gotcha, however, as the alarm simply won't ever sound while the CD is explicitly left on, regardless of the setting of the alarm. It is this paradoxical operation that frequently catches me unaware.

It is simple to disarm the alarm: Simply press the "Alarm" button once, and the clock symbol disappears from the display. However to arm it, I must press the "Alarm" button exactly five times. The first time I press it, the display shows me the time of the alarm. On press two, it shows the time when it will turn the sound off. On press three, it shows me whether it will play the radio or the CD. On press four, it shows me the preset volume. On press five, it returns to the normal view, but with the alarm now armed. But with just one additional press, it disarms the alarm. Sleepy, in a dark bedroom, it is quite difficult to perform this little digital ballet correctly. The alarm clock may still wake me up, but it behaves like a computer.

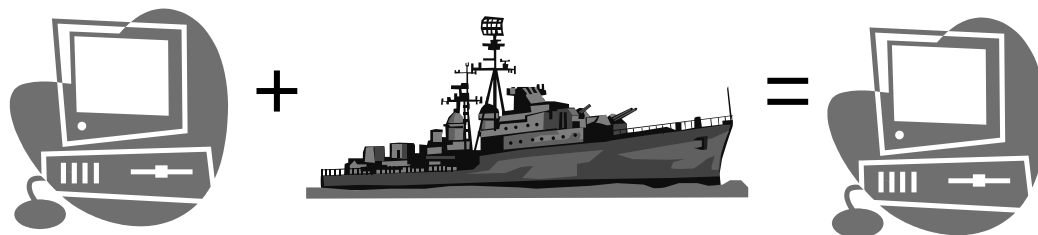
By contrast, my old non-computerized alarm clock woke me up with a sudden, unholy buzzing. When it was armed, a single red light glowed. When it was not armed, the red light was dark. I didn't like this old alarm clock for many reasons, but at least I could tell when it was going to wake me up.

Because it is far cheaper for manufacturers to use computers to control the internal functioning of devices than it is to use older, mechanical methods, it is economically inevitable that computers will insinuate themselves into every product and service in our lives. This means that the behavior of all of our products will be the same as most obnoxious computers, unless we try something different.

What do you get when you cross a computer with a car?

A computer! Porsche's beautiful new high-tech sports car, the Boxster, has seven computers in it to help manage its complex systems. One of them is dedicated to managing the engine. It has special procedures built into it to deal with abnormal situations. Unfortunately, these sometimes backfire. In some early models, if the fuel level in the gas tank got very low---only a gallon or so remaining---the centrifugal force of a sharp turn could cause the fuel to collect in the side of the tank, allowing air to enter the fuel lines. The computer sensed this as a dramatic change in the incoming fuel mixture, and interpreted it as a catastrophic failure of the injection system. To prevent damage, the computer would shut down the ignition and stop the car. Also to prevent damage, the computer would not let the driver restart the engine until the car had been towed to a shop and serviced.

When owners of early Boxsters first discovered this problem, the only solution Porsche could devise was to tell them to open the engine compartment and disconnect the battery for at least five minutes, giving the computer time to forget all knowledge of the hiccup. The sports car may still speed down those two-lane black top roads, but now, in those turns, it behaves like a computer.

What do you get when you cross a computer with a warship?

In September of 1997, while conducting fleet maneuvers in the Atlantic, the USS Yorktown, one of the Navy's new Aegis guided-missile cruisers, stopped dead in the water. A Navy technician, while calibrating an on-board fuel valve, entered a zero into one of the shipboard management computers, a Pentium Pro running Windows NT. The program attempted to divide another number by that zero---a mathematically undefined operation---which resulted in a complete crash of the entire shipboard control system. Without the computers, the engine halted and the ship sat wallowing in the swells for two

hours and fifty-five minutes until it could be towed into port. Good thing it wasn't in a war zone.

What do you get when you cross a computer with a warship? Admiral Nimitz is rolling in his grave! Despite this setback, the Navy is committed to computerizing all of its ships because of the manpower cost savings, and so deflect criticism of this plan, it has blamed the "incident" on human error. Because the software creation process is out of control, the high-tech industry must either bring its process to heel or it will continue to put the blame on ordinary users while ever-bigger machines sit dead in the water

So here you saw the result of integrating computers in our lives. As I said early, computers will annoy us, infuriate us, and even kill a few of us. In turn, we will be tempted to kill our computers, but we won't dare because we are already utterly, irreversibly dependent on these hopeful monsters that make modern life possible. So we will have to think about them. We will have to think how we can make them better. We need to fundamentally rethink how human and machines interact. And rethink the relationship in deep and novel ways, for the fault for our burgeoning problems lies not with our machines, but with us.

1.2 Role of HCI

Here comes the role of HCI. Human designed the interfaces we hate; human continue to use dysfunctional machines even as the awkward interfaces strain their eyes, ache their backs, and ruin their wrist tendons. HCI plays a role to bridge up the gape between the interfaces of machines and human understanding that we have seen in the previous examples.

Definition of HCI

"Human-Computer Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them"
-ACM/IEEE

Lecture 2.

Introduction to Human-Computer Interaction – Part II

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Describe the significance of HCI, particularly adverse impact of computer technology on humans and reasons for these adverse effects
- Describe the nature of humans and computers
- Understand the Paradox of the computing phenomena
- Differentiate between focus of SE and HCI

2.1 Definition of HCI

“Human-Computer Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”
-ACM/IEEE

2.2 Reasons of non-bright Aspects

Airplane + Computer

In last lecture we were discussing the incident of airplane. Today we will look at the reason of such a fatal incident.



The National Transportation Safety Board investigated, and ---as usual---declared the problem human error. The navigational aid the pilots were following was valid but not for the landing procedure at Cali. In the literal definition of the phrase, this was indeed human error, because the pilot selected the wrong fix. However, in the larger picture, it was not the pilot's fault at all.

The front panel of the airplane's navigation computer showed the currently selected navigation fix and a course deviation indicator. When the plane is on course, the needle is centered, but the needle gives no indication whatsoever about the correctness of the selected radio beacon. The gauge looks pretty much the same just before landing as it does just before crashing. The computer told the pilot he was tracking precisely to the beacon he had selected. Unfortunately, it neglected to tell him the beacon the selected was a fatal choice.

The flight computer on Flight 965 could easily have told the pilots that ROMEO was not an appropriate fix for their approach to Cali. Even a simple hint that it was "unusual" or "unfamiliar" could have saved the airplane. Instead, it seemed as though the computer was utterly unconcerned with the actual flight and its passengers. It cared only about its own internal computations

Joke in Computer Industry

There is a widely told joke in the computer industry that goes like this: A man is flying in a small airplane and is lost in the clouds. He descends until he spots an office building and yells to a man in an open window, "Where am I?" The man replies, "You are in an airplane about 100 feet above the ground." The pilot immediately turns to the proper course, spots the airport and lands. His astonished passenger asks how the pilot figured out which way to go. The pilot replies, "The answer the man gave me was completely correct and factual, yet it was no help whatsoever, so I knew immediately he was a software engineer who worked for Microsoft and I know where Microsoft's building is in relation to the airport."

When seen in the light of the tragedy of Flight 965, the humor of the joke is macabre, yet professionals in the digital world tell it gleefully and frequently because it highlights a fundamental truth about computers:

They may tell us facts but they don't inform us.

They may guide us with precision but they don't guide us where we want to go. The flight computer on Flight 965 could easily have told the pilots that ROMEO was not an appropriate fix for their approach to Cali. Even a simple hint that it was "unusual" or "unfamiliar" could have saved the airplane. Instead, it seemed as though the computer was utterly unconcerned with the actual flight and its passengers. It cared only about its own internal computations

Communication can be precise and exacting while still being tragically wrong. This happens all too frequently when we communicate with computers, and computers are invading every aspect of our modern lives. From the planes we fly to just about every consumer product and service, computers are ubiquitous, and so is their characteristically poor way of communicating and behaving.[1]

I-Drive Car Device



It takes automotive computer power to a whole new level. Computer systems provide the car with BMW's most powerful engine, a silky smooth ride and what is supposed to be the simplest in-dash control system available. But what is created for the sake of simplicity can often time creates the most confusion.

Many controls are operated with a single large, multifunction knob located in the console between the front seats. The control consists of a combination rotary and push button for selecting functions. Confirmation of the selected mode is displayed on a dash-mounted screen.

Users can change functions -- from communications to climate control, navigation or entertainment -- by pushing the console knob forward or back, or side-to-side. By twisting the knob, they can scroll through menus. And by clicking a button located in the middle of the knob, they can select functions.

"iDrive" takes into account the fact that comfort, communication and driver assistance functions are only rarely adjusted while driving. The operating unit in the center console gives the driver direct access to many other driving functions and information and communication options. Several hundred functions can be controlled with this device.

A computer-type monitor is positioned directly within the driver's line of vision to the road ahead. The large monitor in the center of the dashboard displays all the information the driver needs, apart from the speedometer and tachometer, which are conventional analog instruments.

The driver slides the dial to choose between multiple control menus displayed on an in-dash LCD screen. The driver rotates the dial to move through lists and pushes the dial axially to select a list item.

After reading that I didn't feel like I had any sort of idea what 'axially' meant, but I suppose this video helps. What concerns me about this is the interaction with this little device requires the driver, hurtling down the road, to look at a screen. They say there is force feedback that indicates the menu, but that's only half the equation, because there are things in the menus. So, I'm guessing the driver needs to memorize the menus, which are sure to be short, so think about the mental modeling here.

To really keep your eyes on the road, you have to be able to do everything by feel and pattern. Is this easier than hot-cold air sliders, vent selection buttons and radio dials?

It takes 15 minutes to change a Radio Channel. The fundamental flaw: you absolutely have to take your eyes off the road to change settings. Result is constant Calls to Help Desk

Feature Shock

Every digital device has more features than its manual counterpart, but manual devices easier to use. Hi-tech companies add more features to improve product. Product becomes complicated

Bad process can't improve product

Computer + Bank

A computer! Whenever I withdraw cash from an automatic teller machine (ATM), I encounter the same sullen and difficult behavior so universal with computers. If I make the slightest mistake, it rejects the entire transaction and kicks me out of the process. I have to pull my card out, reinsert it, reenter my PIN code, and then re-assert my request. Typically, it wasn't my mistake, either, but the ATM computer finesses me into a misstep. It always asks me whether I want to withdraw money from my checking, saving, or money market account, even though I have only checking account. Subsequently, I always forget which type it is, and the question confuses me. About once a month I inadvertently select "savings", and the infernal machine summarily boots me out of the entire transaction to start over the beginning. To reject "savings", the machine has to know that I don't have a saving account, yet it still offers it to me as a choice. The only difference between me selecting "saving" and the pilot of Flight 965 selecting "ROMEO" is the magnitude of the penalty.

The ATM has rules that must be followed, and I am quite willing to follow them, but it is unreasonably computer-like to fail to inform me of them, giving me contradictory indications, and then summarily punish me for innocently transgressing them. This behavior---so typical of computers---is not intrinsic to them. Actually nothing is intrinsic to computers: they merely act on behalf of their software, the program. And programs are as malleable as human speech. A person can speak rudely or politely, helpfully or sullenly. It is as simple for a computer to behave with respect and courtesy as it is for a human to speak that way. All it takes is for someone to describe how. Unfortunately, programmers aren't very good at teaching that to computers.

In order to solve some of these problems, here comes the relatively new and emerging field of Human Computer Interaction (HCI).[1]

2.3 Human verses Computer

Human species

Human beings are the most interesting and fascinating specie on planet. They are the most complex living being on the earth. It has very much diversity in its nature. It is intelligent in its deeds. Human beings think and decide according to their own will. Yes, they are free in nature. They like freedom. They think on a problem dynamically and they can find many solutions that may not exist before. They can invent. They are not only rational but they also have emotions. They also think emotionally. They act emotionally.

And fortunately or unfortunately they make mistakes. They make mistakes which some time become fatal for them and some time they become blessing for them.

Computer species

On contrast, computers are the invention of human being. They are also complex but they are also pretty dumb. It can also think but it can't think on its own will, it thinks how it has been directed to think. No doubt its speed is marvelous. It does not tire. It is emotionless. It has no feelings, no desires. It works how it has been commanded to work. And they do not make mistakes.

Before penetration of computers in our daily life, human beings were performing their tasks at their own responsibility. In a business domain human beings were dealing and interacting with each other's. For example a store manager was dealing with all the workers performing their different duties in the store. Some one was registering the new arrivals of products, some one was numbering the products and many more...and store manager has to interact with all these human beings. If some one was a salesperson, he used to interact with different clients and used to deal with them according to their mood and desire. He could judge their mood with their tone, their attitude and with their body language. He could provide answers relevant to their questions.

But now in this age of information technology we are expecting computers to mimic human behavior e.g. ECommerce systems, now there is no need for a salesperson. Web sites are behaving as a salesperson or as a shopping mal. That is now; a dumb, unintelligent and inanimate object will perform the complex task which was performed by some human being.

2.4 Software Apartheid

Apartheid

Racial segregation; specifically: a policy of segregation and political and economic discrimination against non-European groups in the Republic of South Africa. [Definition of apartheid]

Software Apartheid

Institutionalizing obnoxious behavior and obscure interactions of software-based products. [Definition of software apartheid]

Programmers generally work in high-tech environments, surrounded by their technical peers in enclaves like Silicon Valley. Software engineers constantly encounter their peers when they shop, dine out, take their kids to school and relax, while their contact with frustrated computer users is limited. What's more, the occasional unfocused gripes of the users are offset by the frequent enthusiasm of the knowledgeable elite. We forget how far removed our peers and we are from the frustration and inability of the rest of the country (not to mention the world) to use interactive tools.

We industry insiders toss around the term "computer literacy", assuming that in order to use computers; people must acquire some fundamental level of training. We see this as a

simple demand that is not hard and is only right and proper. We imagine that it is not much to ask of users that they grasp the rudiments of how the machines work in order to enjoy their benefits. But it is too much to ask. Having a computer literate customer base makes the development process much easier—of their can be no doubt—but it hampers the growth and success of the industry and of society. Apologists counter with the argument that you must have training and a license to drive a car, but they overlook the fact that a mistake with software generally does not. If cars were not so deadly, people would train themselves to drive the same way they learn excel.

It has another, more insidious effect. It creates a demarcation line between the haves and have-nots in society. If you must master a computer in order to succeed in America's job Market beyond a burger-flipper's carriers, then the difficulty of mastering interactive systems forces many people into menial jobs rather than allowing them to matriculate into more productive, respected and better-paying jobs.

Users should not have to acquire computer literacy to use computer for common, rudimentary task in everyday life. Users should not have to possess a digital sensitivity to work their VCR, microwave oven, or to get e-mail. What's more, should not have to acquire computer literacy to use computer for enterprise applications, where the user is already trained in the application domain. An accountant for example, who is trained in the general principles of accounting, should not have to become computer literate to use a computer in her accounting practice. Her domain knowledge should be enough to see her through.

As our economy shifts more and more onto information bases, we are inadvertently creating a divided society. The upper class is composed of those who have mastered the nuances of differentiating between "RAM" and "Hard Disk". The lower class is that who treat the difference inconsequential. The irony is that the difference really is inconsequential to any one except a few hard-core engineers. Yet virtually all-contemporary software forces its users to confront a file system, where your success fully dependent on knowing the difference between RAM and disk.

Thus the term "computer literacy" becomes a euphemism for social and economic apartheid. Computer literacy is a key phrase that brutally bifurcates our society.

But about those people who are not inclined to pander to technocrats and who can not or will not become computer literate? These people, many by choice, but most by circumstances, are falling behind in the information revolution. Many high-tech companies, for example, would not even consider for employment any applicant who does not have an e-mail address. I'm sure that there are many otherwise qualified candidates out there who cannot get the hired because they are not yet wired. Despite the claims of the Apologists, using e-mail effectively is difficult and involves a significant level of computer literacy. Therefore, it artificially segregates the work force. It is the model equivalent of the banking technique of "red lining". In this illegal procedure, all houses in a given neighborhood are declared unacceptable as controller for a housing loan. Although the red lines on the map are ostensibly drawn around economic contours, they tend to follow racial lines all too closely bankers protest that they are not racists, but the effect is the same.

When programmers speak of “computer literacy”, they are drawing red lines around ethnic groups, too, yet few have pointed this out. It is too hard to see what is really happening because the issue is obscured by technical mythology. It is easy to see--- regardless of how true---that a banker can make a loan on one house as easily as on another. However, it is not easy to see that a programmer can make interactive products easy enough for people from lower socio-economic backgrounds to use.

“Acceptable levels of quality for software engineers are far lower than are those for traditional engineering disciplines”

“Software-based products not INHERENTLY hard to use Wrong process is used to develop them” [1]

Software Engineering and HCI

There is a basic fundamental difference between the approaches taken by software engineers and human-computer interaction specialists. Human-computer interface specialists are user-centered and software engineers are system-centered.

Software engineering methodologies are good at modeling certain aspects of the problem domain. Formal methods have been developed to represent data, architectural, and procedural aspects of a software system. Software engineering approaches deal with managerial and financial issues well. Software engineering methodologies are useful for specifying and building the functional aspects of a software system.

Human-computer interfaces emphasize developing a deep understanding of user characteristics and a clear awareness of the tasks a user must perform. HCI specialists test design ideas on real users and use formal evaluation techniques to replace intuition in guiding design. This constant reality check improves the final product.

References

- [1] The Inmates are running the asylum by Alan Cooper.
- [2] Human Computer Interaction by Jenny Preece.

Lecture 3.

Introduction to Human-Computer Interaction – Part III

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Discuss the effect of bad tools
- Discuss and argue about why Human computer Interaction (HCI) is important with reference to the way in which technology has developed during past forty years
- Describe requirement of new economy era.

Effect of Bad Tools

Not only are computers taking over the cockpit of jet airliners, they are taking over the passenger cabin, too, behaving in that same obstinate, perverse way that is so easy to recognize and so hard to use. Modern jet planes have in-flight entertainment (IFE) systems that deliver movies and music to airline passengers. These IFEs are merely computers connected with local area network, just like in your office. Advanced IFE systems are generally installed only on larger airplanes flying transoceanic routes.

One airline's IFE was so frustrating for the flight attendants to use that many of them were bidding to fly shorter, local routes to avoid having to learn and use the difficult systems. This is remarkable considering that the time-honored airline route-bidding process is based on seniority, and that those same long-distance routes have always been considered the most desirable plums because of their lengthy layovers in exotic locales like Singapore or Paris. For flight attendants to bid for unglamorous, unromantic yo-yo flights from Denver-to-Dallas or LA-to-San Francisco just to avoid the IFE indicated a serious morale problem. Any airline that inflicted bad tools on its most prized employee—the ones who spent the most time with the customer---was making a foolish decision and was profligately discarding money, customer loyalty, and staff loyalty.

The computer-IFE of another large airline was even worse. The airline had created an in-flight entertainment system that linked movie delivery with the cash collection function. In a sealed et airplane flying at 37,000 feet, cash collection procedures had typically been quite laissez-faire; after all, nobody was going to sneak out the back door. Flight attendants delivered goods and services when it was convenient and collected cash in only a very loosely coupled fashion. This kept them from running unnecessarily up and down the narrow aisles. Sure there were occasional errors, but never more than a few dollars were involved, and the system was quite human and forgiving; everyone was happy and the work was not oppressive.

With cash-collection connected to content delivery by computer, the flight attendant had to first get the cash from the passenger, then walk all the way to the head-end of the cabin, where the attendant's console was, enter an attendant password, then perform a cash register-like transaction. Only when that transaction was completed could the passenger actually view a movie or listen to music. This inane product design forced the flight attendants to walk up and down those narrow aisles hundreds of extra times during a typical trip. Out of sheer frustration, the flight attendants would trip the circuit breaker on the in-flight entertainment system at the beginning of each long flight, shortly after departure. They would then blandly announce to the passengers that, sorry, the system was broken and there would be no movie on this flight.

The airline had spent millions of dollars constructing a system so obnoxious that its users deliberately turned it off to avoid interacting with it. The thousands of bored passengers were merely innocent victims. And this happened on long, overseas trips typically packed with much-sought-after frequent flyers. I cannot put a dollar figure

On the expense this caused the airline, but I can say with conviction that it was catastrophically expensive.

The software inside the IFEs worked with flawless precision, but was a resounding failure because it misbehaved with its human keepers.

3.1 An Industry in Denial

We are a world awash in high-tech tools. Computers dominate the workplace and our homes, and vehicles are filling up with silicon-powered gadgets. All of these computerized devices are wildly sophisticated and powerful, but every one of them is dauntingly difficult and confusing to use.

The high-tech industry is in denial of a simple fact that every person with a cell phone or a word processor can clearly see: our computerized tools are hard to use. The technologists who create software and high-tech gadgets are satisfied with their efforts. The software engineers who create them have tried as hard as they can to make them easy to use and they have made some minor progress. They believe that their products are as easy to use as it is technically possible to make them. As engineers, their belief is in technology, and they have faith that only some new technology, like voice recognition or artificial intelligence, will improve the user's experience.

Ironically, the thing that will likely make the least improvement in the ease of use of software-based products is new technology. There is little difference technically between a complicated, confusing program and a simple, fun, and powerful product. The problem is one of culture, training, and attitude of the people who make them, more than it is one of chips and programming languages. We are deficient in our development process, not in our development tools.

The high-tech industry has inadvertently put programmers and engineers in charge, so their hard-to-use engineering culture dominates. Despite appearances, business executives are simply not the ones in control of the high-tech industry. It is the engineers who are running the show. In our rush to accept the many benefits of the silicon chip, we have abdicated our responsibilities. We have let the inmates run the asylum.

When the inmates run the asylum, it is hard for them to see clearly the nature of the problems that bedevil them. When you look in the mirror, it is all too easy to single out your best features and overlook the warts. When the creators of software-based products examine their handiwork, they see how rich the product is in features and functions. They ignore how excruciatingly difficult it is to use, how many mind-numbing hours it takes to learn, or how it diminishes and degrades the people who must use it in their everyday lives.

3.2 Techno-Rage

An article in a recent issue of the Wall Street Journal described an anonymous video clip circulated widely by email that showed as “...Mustachioed Everyman in a short sleeved shirt hunched over a computer terminal, looking puzzled. Suddenly, he strikes the side of his monitor in frustration. As a curious co-worker peers over his cubicle, the man slams the keyboard into the monitor, knocking it to the floor. Rising from his chair, he goes after the fallen monitor with a final ferocious kick.” The article went on to say that reaction to the clip had been “intense” and that it had apparently tapped into a powerful undercurrent of techno-rage”.

It’s ironic that one needs to be moderately computer savvy to even send or view this video clip. While the man in the video may well be an actor, he touches a widespread, sympathetic chord in our business world. The frustration that difficult and unpleasant software-based products are bringing to our lives is rising rapidly.

Joke email circulates on private lists about “Computer Tourette’s.” This is a play on the disorder known as Tourette’s syndrome, where some sufferers engage in uncontrollable bouts of swearing. The joke is that you can walk down the halls of most modern office buildings and hear otherwise-normal people sitting in front of their monitors, jaws clenched, swearing repeatedly in a rictus of tense fury. Who knows what triggered such an outburst: a misplaced file, an inaccessible image, or a frustrating interaction. Or maybe the program just blandly erased the user’s only copy of a 500-page manuscript because he responded with a “Yes” to a confirmation dialog box, assuming that it had asked him if he wanted to “save your changes?” when it actually asked him if he wanted to “discard your work?”

Novatech survey

One in every four computers has been physically attacked by its owner, according to a survey.

The survey, conducted by British PC manufacturer Novatech, was intended to take a lighthearted look at embarrassing experiences -- those little technical bloopers that happen even to ubergeeks, like forwarding a personal love letter to an entire office mailing list.

But instead, a much darker story emerged in the 4,200 replies. Innocent computers are being beaten on a regular basis, often by technically challenged users who decide to take their frustrations out on their helpless machines.

"We decided to do some research into people's relationships with their computers and we were amazed by some of the results," Novatech managing director David Furby said. "As computers become more and more a part of our daily lives, we obviously share more experiences with them."

Many technical support people from the United States, Canada and parts of Europe have sobering stories of brutalized computers being brought in for repair by sad -- or in some cases, smug -- owners who had smacked, kicked or otherwise deliberately injured their machines.

"The incidences of willful neglect have always been high," said David Tepper, owner of the Village Computer Shop in New York. "We've always had to deal with computers damaged by people who dumped their refreshing beverage on the computer's keyboard, or got tangled up in the cords and bringing (sic) the computer crashing down off their desk."

"But there have also always been a small -- but significant -- number of machines that were obviously intentionally damaged."

"Hopefully as technology improves and computers become ever more user- friendly, these attacks will become less frequent," Furby said.

Computer rage

There is a technology-based scourge afoot...maybe. It's not a virus; it's not a denial of service attack; it's computer rage, and according to the latest reports, it is out to destroy the physical health, the emotional stability, and if left unchallenged, the economic strength of whatever population it strikes.

Security software vendor Symantec, working with Britain's National Opinion Poll, recently found that when confronted with technical problems, more than 40 percent of British users surveyed have sworn at, kicked, or otherwise abused their computers, monitors, and the most victimized of all computer components, their keyboards.

In similar surveys conducted last year, Marlborough, Mass-based Concord Communications discovered that 83 percent of 150 U.S. respondents witnessed such attacks, and the international market research firm Mori found that 40 percent of 1250 British workers had watched as their colleagues leveled verbal and physical abuse at their computers.

Stress related to computer rage, the Symantec study claims, has resulted in a loss of productivity for most respondents.

Robert Edelman, clinical psychologist and author of *Interpersonal Conflicts at Work*, is worried. "Frustration with IT should be taken seriously as a modern malaise," he says. "It is affecting both our work and our home lives to the extent that computer rage is now much more prolific than road rage."

Computers increasingly commonplace in offices

As the reliance on computers in the workplace continues to grow, people in the UK are resorting to violence when their PCs break down, say researchers. When faced with technical problems, most people shouted at colleagues, hit the PC or even threw parts of the computers. The most frustrating hitch was when people lost their work after their computer crashed or froze.

The problems seem to be widespread with more than a quarter of those working with computers experience problems with their PC on a weekly basis.

"Over half of all working days lost to sickness in the UK are related to workplace stress," said Fiona Dennis, a stress management trainer with Priory Healthcare. "Being heavily reliant on IT to speed up our lives means that performance is hampered greatly when it fails, causing an over-reaction and stress."

70% swear at PCs

The study by the National Opinion Poll and the software company Symantec, found that nearly half of all computer users had become angry at some time. Almost a third of people had physically attacked a computer, 67% experienced frustration, exasperation and anger and more than 70% swore at their machines.

Technology rage is the latest rage to emerge in Britain and follows road rage, trolley rage and air rage. There was a dramatic rise in air rage incidents last year, with 174 people detained at Heathrow and Gatwick alone. In 1998 the number of air rage arrests for the whole country was 98.

3.3 Success Criteria in the New Economy

The successful professional for the twenty-first century is either a business savvy technologist or a technology-savvy businessperson.

The technology-savvy businessperson knows that his success is dependent on the quality of the information available to him and the sophistication with which he uses it. The business-savvy technologist, on the other hand, is an entrepreneurial engineer or scientist trained for technology, but possessing a keen business sense and an awareness of the power of information. Both of these new archetypes are coming to dominate contemporary business.

You can divide all businesspeople into two categories: those who will master high technology and those who will soon be going out of business. No longer can executive delegate information processing to specialists. Business is information processing. You differentiate yourself today with the quality of your information-handling systems, not your manufacturing systems. If you manufacture anything, chances are it has a microchip in it. If you offer a service, odds are that offer it with computerized tools. Attempting to identify businesses that depend on high technology is as futile as trying to identify businesses that depend on the telephone. The high-tech revolution has invaded every business, and digital information is the beating heart of your workday.

It has been said, "To err is human; to really screw up, you need a computer."

Inefficient mechanical systems can waste couple cents on every widget you build, but you can lose your entire company to bad information processes. The leverage that software-based products---and the engineers that build them---have on your company is enormous.

Sadly, our digital tools are extremely hard to learn, use, and understand, and they often cause us to fall short of our goals. This wastes money, time, and opportunity. As a business-savvy technologist/ technology-savvy businessperson, you produce software-based products or consume them---probably both. Having better, easier-to-learn, easier-to-use high-tech products is in your personal and professional best interest. Better products don't take longer to create, nor do they cost more to build. The irony is that they don't have to be difficult, but are so only because our process for making them is old-fashioned and needs fixing. Only long-standing traditions rooted in misconceptions keep us from having better products in today.

Consider a scenario: a website is developed of ecommerce system. The site is aesthetically very beautiful, technically it has no flaw and it has wonderful animated content on it. But if user is unable to find its desired information about the products or even he is unable to find the product out of thousands of products, so what of it's use. It is useless from the business point of view.

Here are some facts and figures:

Users can only find information 42% of the time

– Jared Spool

62% of web shoppers give up looking for the item they want to buy online

– Zona Research

–

50% of the potential sales from a site are lost because people cannot find the item they are looking for

– Forrester Research

40% of the users who do not return to a site do so because their first visit resulted in a negative experience

– Forrester Research

80% of software lifecycle costs occur after the product is released, in the maintenance phase - of that work, 80 % is due to unmet or unforeseen user requirements; only 20 % is due to bugs or reliability problems.

- IEEE Software

Around 63% of software projects exceed their cost estimates. The top four reasons for this are:

- Frequent requests for changes from users
- Overlooked tasks
- Users' lack of understanding of their own requirements
- Insufficient user-analyst communication and understanding

- Communications of the ACM

BOO.com, a \$204m startup fails
– BBC News

Poor commercial web sites will kill 80% of Fortune 500 companies within a decade
- Jakob Nielsen

So all above given facts reveals that the product with the bad user experience deserve to die!

The serious financial implications of today's digital products should not in any ways be underestimated.

The table given below depicts two scenarios of potential of sales from an e-commerce web site. In scenario A, users can easily find items they are looking for, so 0% sales are lost, so the actual revenue is \$100 million. In scenario B, users cannot easily find the items they are looking for, therefore, the actual revenue is \$50 million, thus causing a loss of \$50 million.

	Scenario A	Scenario B
Revenue Potential	\$100m	\$100m
User Experience	Good	Bad
Sales Lost	0%	50%
Revenue Lost	\$0m	\$50m
Actual Revenue	\$100m	\$50m

3.4 Computer + Information

What do you get when you cross a computer with information?

In 2000, the Olympic Games were held in Sydney. Before the Olympic games could begin, a lawsuit was filed against the Olympic Committee. The case was called Bruce Lindsay Maguire vs Sydney Organizing Committee for the Olympics Games (SOCOG). On 7 June 1999 the complainant, who is blind, complained to the

Commission that he was unlawfully discriminated against by the respondent in three respects:

- the failure to provide Braille copies of the information required to place orders for Olympic Games tickets;
- the failure to provide Braille copies of the Olympic Games souvenir programmed; and
- The failure to provide a web site which was accessible to the complainant.

It was alleged that the SOCOG was in breach the *Disability Discrimination Act* 1992 by failing to make accessible to him key parts of its web site

According to the law of many European and western countries, organizations with a web site must ensure that their web site is (within certain limits) accessible by disabled persons. However, this was not the case in the matter of the official Olympic Games web site. Could this have been avoided? Certainly: by applying a few very simple techniques, the developers of the web site could have made it accessible to people with vision-impairment. But as is usually the case, this was not done.

Result: the complainant won the case and was awarded a sum of money in damages. This was very embarrassing for both the SOCOG and also the company that developed the web site.

References

1. <http://www.independentliving.org/docs5/sydney-olympics-blind-accessibility-decision.html>
2. <http://comment.cio.com/soundoff/061400.html>

Lecture 4.

Goals & Evolution of Human Computer Interaction

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Describe the goals of HCI
- Define Usability goals
- Define User Experience goals
- Discuss the History and Evolution of HCI

Definition of HCI

“Human-Computer Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”

-ACM/IEEE

4.1 Goals of HCI

The term Human Computer Interaction (HCI) was adopted in the mid-1980s as a means of describing this new field of study. This term acknowledged that the focus of interest was broader than just the design of the interface and was concerned with all those aspects that relate to the interaction between users and computers.

The goals of HCI are to produce usable and safe systems, as well as functional systems. These goals can be summarized as ‘to develop or improve the safety, utility, effectiveness, efficiency and usability of systems that include computers’ (Interacting with computers, 1989). In this context the term ‘system’ derives from systems theory and it refers not just to the hardware and software but to the entire environment---be it organization of people at work at, home or engaged in leisure pursuits---that uses or is affected by the computer technology in question. Utility refers to the functionality of a system or, in other words, the things it can do. Improving effectiveness and efficiency are self-evident and ubiquitous objectives. The promotion of safety in relation to computer systems is of paramount importance in the design of safety-critical systems. Usability, a key concept in HCI, is concerned with making systems easy to learn and easy to use. Poorly designed computer system can be extremely annoying to users, as you can understand from above described incidents. [2]

Part of the process of understanding user’s needs, with respect to designing an interactive system to support them, is to be clear about your primary objective. Is it to design a very efficient system that will allow users to be highly productive to their work, or is to design a system that will be challenging and motivating so that it supports effective learning, or is it some thing else? We call these talk-level concerns usability goals and user experience

goals. The two differ in terms of how they are operationalized, i.e., how they can be met and through what means. Usability goals are concerned with meeting specific usability criteria (e.g., efficiency) and user experience goals are largely concerned with explicating the quality of the user experience (e.g., to be aesthetically pleasing).

Usability goals

To recap, usability is generally regarded as ensuring that interactive products are easy to learn, effective to use, and enjoyable from user perspective.

It involves optimizing the interactions people have with interactive product to enable them to carry out their activities at work, school, and in their everyday life. More specifically, usability is broken down into the following goals:

- Effective to use (effectiveness)
- Efficient to use (efficiency)
- Safe to use (safety)
- Have good utility (utility)
- Easy to learn (learnability)
- Easy to remember how to use (memorability)

For each goal, we describe it in more detail.

Effectiveness

It is a very general goal and refers to how good a system is at doing what it is supposed to do. [1]

Efficiency

It refers to the way a system supports users in carrying out their tasks. [1]

Safety

It involves protecting the users from dangerous conditions and undesirable situations. In relation to the first ergonomics aspect, it refers to the external conditions where people work. For example, where there are hazardous conditions---like x-rays machines or chemical plants---operators should be able to interact with and control computer-based system remotely. The second aspect refers to helping any kind of user in any kind of situation avoid the danger of carrying out unwanted action accidentally. It also refers to the perceived fears users might have of the consequences of making errors and how this affects their behavior to make computer-based system safer in this sense involves:

- Preventing the user from making serious error by reducing the risk of wrong keys/buttons being mistakenly activated (an example is not placing the quit or delete-file command right next to the save command on a menu.) and
- Providing users with various means of recovery should they make errors. Safe interactive systems should engender confidence and allow the users the opportunity to explore the interface to carry out new operations.

Other safety mechanisms include undo facilities and confirmatory dialog boxes that give users another chance to consider their intentions (a well-known used in email application

is the appearance of a dialog box after the user has highlighted the messages to be deleted, saying: “are you sure you want to delete all these messages?”)

Utility

It refers to the extent to which the system provides the right kind of functionality so that user can do what they need or want to do. An example of a system with high utility is an accounting software package providing a powerful computational tool that accountants can use to work out tax returns. An example of a system with low utility is a software drawing tool that does not allow users to draw free hand but forces them to use a mouse to create their drawings, using only polygon shapes. [1]

Learnability

It refers to how easy a system is to learn to use. It is well known that people do not like spending a long time learning how to use a system. They want to get started straight away and become competent at carrying out tasks without too much effort. This is especially so for interactive products intended for everyday use (for example interactive TV, email) and those used only infrequently (for example, video conferencing) to certain extent, people are prepared to spend longer learning more complex system that provide a wider range of functionality (for example web authoring tools, word processors) in these situations, CD ROM and online tutorials can help by providing interactive step by step material with hands-on exercises. However many people find these tedious and often difficult to relate to the tasks they want to accomplish. A key concern is determining how much time users are prepared to spend learning a system. There seems little point in developing a range of functionality if the majority of users are unable or not prepared to spend time learning how to use it. [1]

Memorability

It refers to how easy a system is to remember how to use, once learned. This is especially important for interactive systems that are used infrequently. If users haven't used a system or an operation for a few months or longer, they should be able to remember or at least rapidly be reminded how to use it. Users shouldn't have to keep relearning how to carry out tasks. Unfortunately, this tends to happen when the operation required to be learning are obscure, illogical, or poorly sequenced. Users need to be helped to remember how to do tasks. There are many ways of designing the interaction to support this. For example, users can be helped to remember the sequence of operations at different stages of a task through meaningful icons, command names, and menu options. Also, structuring options and icons so they are placed in relevant categories of options (for example, placing all the drawing tools in the same place on the screen) can help the user remember where to look to find a particular tool at a given stage of a task. [1]

“Don't Make me THINK, is the key to a usable product”

User experience goals

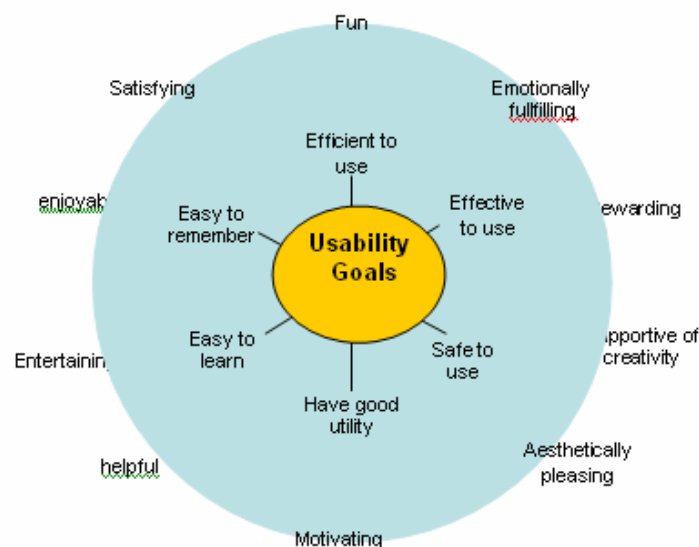
The realization that new technologies are offering increasing opportunity for supporting people in their everyday lives has led researchers and practitioners to consider further goals. The emergence of technologies (for example, virtual reality, the web, mobile computing) in diversity of application areas (e.g., entertainment, education, home, public areas) has brought about a much wider set of concerns. As well as focusing primarily on

improving efficiency and productivity at work, interaction design is increasingly concerning itself with creating systems that are:

- Satisfying
- Enjoyable
- Fun
- Entertaining
- Helpful
- Motivating
- Aesthetically pleasing
- Supportive of creativity
- Rewarding
- Emotionally fulfilling

The goals of designing interactive products to be fun, enjoyable, pleasurable, aesthetically pleasing and so on are concerned primarily with the user experience. By this we mean what the interaction with the system feels like to the users. This involves, explicating the nature of the user experience in subjective terms. For example, a new software package for children to create their own music may be designed with the primary objectives of being fun and entertaining. Hence, user experience goals differs from the more objective usability goals in that they are concerned with how user experience an interactive product from their perspective, rather than assessing how useful or productive a system is from its own perspective. The relationship between two is shown in figure.

Recognizing and understanding the trade-offs, between usability and user experience goals, is important. In particular, this enables designers to become aware of the consequences of pursuing different combinations of them in relation to fulfilling different users' needs. Obviously, not all of the usability goals and user experience goals apply to every interactive product being developed. Some combination will also be incompatible. For example, it may not be possible or desirable to design a process control system that is both safe and fun. [1]

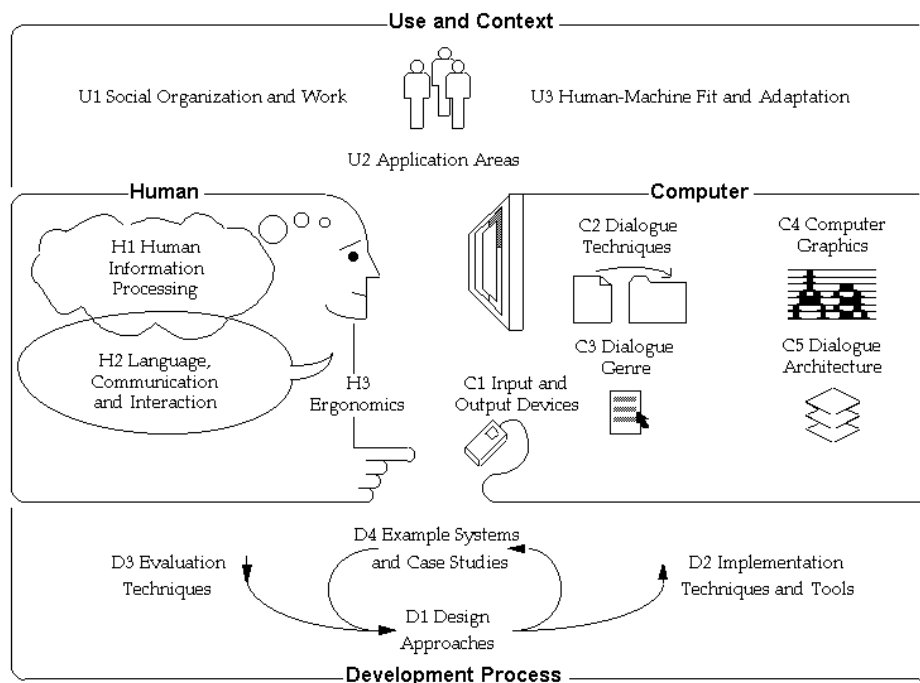


4.2 Evolution of HCI

Figure shows the main topics that make up the discipline of HCI. All HCI takes place within a social and organizational context. Different kinds of applications are required for different purposes and care is needed to divide tasks between humans and machines, making sure that those activities and routine are allocated to machines.

Knowledge of human psychological and physiological abilities and, more important still their limitations is important.

As shown in figure, this involves knowing about such things as human information processing, language, communication, interaction and ergonomics. Similarly it is essential to know about the range of possibilities offered by computer hardware and software so that knowledge about humans can be mapped on to the technology appropriately. The main issues for consideration on the technology side involve input techniques, dialogue technique, dialogue genre or style, computer graphics and dialogue architecture. This knowledge has to be brought together some how into the design and development of computer systems with good HCI, as shown at the bottom of the figure. Tools and techniques are needed to realize systems. Evolution also plays an important role in this process by enabling designers to check that their ideas really are what users want.



Three systems that provide landmarks along this evolutionary path are the Dynabook, the Star and the Apple Lisa, predecessor of today's Apple Macintosh machines. An important unifying theme present in all three computer systems is that they provided a form of interaction that proved effective and easy for novices and experts alike. They were also easy to learn, and provided a visual-spatial interface whereby, in general, objects could be directly manipulated, while the system gave immediate feedback.

Dynabook

Alan Kay designed the first object-oriented programming language in the 1970s. Called Smalltalk, the programs were the basis for what is now known as windows technology—the ability to open more than one program at a time on a personal computer. However, when he first developed the idea, personal computers were only a concept. In fact, the idea of personal computers and laptops also belongs to Kay. He envisioned the Dynabook—a notebook-sized computer, with a keyboard on the bottom and a high-resolution screen at the top.

Star

The Xerox Star was born out of PARC's creative ferment, designing an integrated system that would bring PARC's new hardware and software ideas into a commercially viable product for use in office environments. The Star drew on the ideas that had been developed, and went further in integrating them and in designing for a class of users who were far less technically knowledgeable than the engineers who had been both the creators and the prime users of many PARC systems (one of PARC's favorite mottoes was "Build what you use, use what you build.") The Star designers were challenged to make the personal computer usable for a community that did not have previous computer experience.

From today's perspective, the Star screen looks rather unremarkable, and perhaps a bit clumsy in its graphic design—a boxy model-T when compared to the highly styled look of today's Taurus or Jaguar. What is notable from a historical perspective, of course, is how much the Star does look like current screens and how little it looks like the character-based and vector-drawing screens that preceded it.



The Star (Viewpoint) screen image The Star pioneered the now-familiar constellation of icons, moveable scrollable windows, and inter-mixed text and graphic images. The widely used graphic user interfaces (GUIs) of today are all variants of this original design. (Source: Reprinted by permission from Jeff Johnson et al. Xerox Star, a retrospective. IEEE Computer 22:9 (September, 1989), p. 13.)

The visible mechanisms on the Star display were backed up with a set of design principles that grew out of a user-oriented design methodology and by a great deal of empirical testing. Several principles were central to the Star design:

Direct manipulation

The core concept that distinguished Star (and other Alto programs) from the conventional computer interfaces of their time was the use of a bitmapped screen to present the user with direct visual representations of objects. In the Star's desktop metaphor, documents, printers, folders, collections of folders (file drawers and cabinets), in and out boxes, and other familiar office objects were depicted on the screen. To print a document, for example, the user could point (using the mouse) to the icon for the document and the icon for the printer, while using a key on the keyboard to indicate a Copy operation.

WYSIWYG (what you see is what you get)

In previously available programs for producing sophisticated graphical output—such as drawings or page layout with multiple fonts—the user created and edited a representation that looked like a programming language, and then compiled the resulting program into a visible form. Alto programs pioneered a new style that Star unified, in which the user works directly with the desired form, through direct manipulation. The user makes changes by operating on a direct representation of what will appear on the printed page. The Star user could intermix text, tables, graphs, drawings, and mathematical formulas. In fact, most of the popular microcomputer applications of today have not yet reached the degree of integration that Star offered more than a decade ago.

Consistency of commands

Because a single development group developed all Star applications in a unified way, it was possible to adhere to a coherent and consistent design language. The Star keyboard embodied a set of generic commands, which were used in a consistent way across all applications: Move, Copy, Delete, Open, Show Properties, and Same (copy properties). Evoking one of these commands produced the same behavior whether the object is being moved or copied, for example, was a word of text, a drawing element, or a folder of documents. Through the use of property sheets the user could manipulate the aspects that were specific to each element, such as the font of a text character, or the brush width of a painted line. The Open command was the basis for applying a technique of progressive disclosure—showing the user only the relevant information for a task at hand, and then providing a way to reveal more possibilities, as they were needed.

In addition to these three key concepts, many specific design features made the Star unique, including its attention to the communicative aspects of graphic design, its integration of an end-user scripting language (CUSP), and its underlying mechanisms for internationalization—from the very beginning, Star versions were developed in several languages, including non-European languages with large character sets, non-left-to-right orthography, and so on.

Some of the aspects that led to the Star's design quality may have also hampered its commercial success—in particular Xerox's dependence on development groups within a single company to produce all the applications software.

Lisa by Apple

The GUI (Graphical User Interface) that started it all. If you are sitting in front of a computer with a mouse and pull down menus you owe it to this machine. Windows proponents will tell you that Xerox PARC developed GUIs and Apple stole it from them,

therefore what Mr. Gates has done is okay. Xerox had the core idea, but I've seen video of the early PARC work. It was advanced but it was not nearly what the Lisa (and later the Mac) became.

The first Apple Lisa was equipped with dual 5.25 inch floppy drives in addition to a huge external hard drive (shown here). The Apple Lisa 2/10 moved the hard drive inside the case. It lost one floppy drive and the Macintosh the newer 3.5-inch floppy shared the remaining one.

My Lisa is the later variety. In fact I have no way of knowing how mine was sold but the Lisa was later marketed as the Macintosh XL: a bigger sister to the little Macintosh. Lisa lacked the ROM toolbox built into every Macintosh so it had to do Macintosh emulation through a new operating system known as MacWorks. It allowed Lisa to pretend she was a Macintosh. Why do this when you could just buy a Mac? Lisa offered more RAM (1 meg) a hard drive (10 meg) and some businesses had already bought them.



While giving credit to the workers at Xerox it should also be mentioned that much of the groundwork was done in the 1960s and early 1970s. one influential researcher was Licklider (1960), who visualized a symbiotic relationship between humans and computers. He envisaged computers that would be able to do more than simply handle information: the partnership of computer and human brain would greatly enhance thinking processes and lead to more creative achievements. Another influential development was the pioneering work of Sutherland (1963), who developed the Sketchpad system at MIT. The Sketchpad system introduced a number of powerful new ideas, including the ability to display, manipulate and copy pictures represented on the screen and the use of new input devices such as the light pen.

Alongside developments in interactive graphic interface, interactive text processing systems were also evolving at a rapid rate. Following in the footsteps of line and display editors was the development of systems that allowed users to create and edit documents that were represented fully on the screen. The underlying philosophy of these systems is captured by the term WYSIWYG, which stands for 'what you see is what you get' (pronounced 'whizzee-wig'). In other words, the documents were displayed on the screen exactly as they would look in printed form. This was in stark contrast to earlier document editors, where commands were embedded in the text and it was impossible to see what document would look like without printing it.

Interestingly, difference in research and development interests could be discerned on the two sides of the Atlantic. Pioneers of HCI in the USA were primarily concerned with how the computer could enrich our lives and make them easier. They foresaw it as a tool that could facilitate creativity and problem solving. In Europe, in 1980 researchers began to be more concerned with constructing theories of HCI and developing methods of design which would ensure that the needs of users and their tasks were taken into account. One

of the major contributions from the European side was an attempt to formalize more fully the concept of usability and to show how it could be applied to the design of computer systems (Shackel, 1981).

During the technology explosion of the 1970s the notion of user interface, also known as the Man-Machine Interface (MMI), became a general concern to both system designers and researchers. Moran defined this term as ‘those aspects of the system that the user comes in contact with’ (1981, p.4), which in turn means ‘an input language for the user, an output language for the machine, and a protocol for interaction’ (Chi, 1985, p.671).

Academic researchers were concerned about how the use of computer might enrich the work and personal lives of people. In particular, they focused on the capabilities and limitations of human users, that is, understanding the ‘people side’ of the interaction with computer systems. At that time this primarily meant understanding people’s psychological processes when interacting with computers. However, as the field began to develop it soon became clear that other aspects impinge on users and management and organizational issues and health hazards are all important factors contributing to the success or failure of using the computer systems. [2]

Reference:

- [1] About Face 2.0 the essentials of interaction design by Alan Cooper
- [2] Human Computer Interaction by Jenny Preece

Lecture 5.

Discipline of Human Computer Interaction

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Describe the relationship of Usability and quality
- Understand HCI Discipline

Human-computer Interaction is the kind of discipline, which is neither the study of human, nor the study of technology, but rather the bridging between those two. So you always have to have one eye open to the question: what can the technology do? How can you build it? What are the possibilities? And one eye open to the question: what are people doing and how would this fit in? What they would do with it? If you lose sight of either of those, you fail to design well. And of course they require different ways of thinking. So I think the challenge is to keep knowledge of both the technology and the people playing off against each other in order to develop new things.

If you build something you need to consider not just 'I'm building something because I need to build it', but 'what effect is it going to have on the way people work and the way people live?'

5.1 Quality

Let us firstly look at a general definition of quality.

According to the American Heritage Dictionary "characteristic or attribute of something." As an attribute of an item, quality refers to measurable characteristics---things we are able to compare to know standards such as length, color, electrical properties, malleability, and so on.

Now as we are concerned with software quality so let us look at some other definitions:

According to British Defense Industries Quality Assurance Panel "Quality is conformance to specifications". So, according to this definition quality is the measure of degree to which the design specifications are followed during manufacturing. The greater the degree of conformance, the higher the level of quality is.

Philip Crosby describes, "Quality is conformance to requirements." Here software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.

Juran says, "Quality is fitness for purpose or use"

"Quality is a predictable degree of uniformity and dependability, at low cost and suited to the market", defined by Edward Deming.

By R J Mortiboys "Quality is synonymous with customer needs and expectations." "Quality is meeting the (stated) requirements of the customer- now and in the future." By Mike Robinson.

“Quality is the total composite product and service characteristics of marketing, engineering, manufacturing and maintenance through which the product and service in use will meet the expectations by the customer”

(Armand Feigenbaum)

“Totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.”

All above-mentioned definitions refer quality as a conformance to requirements or conformance to specification or as a synonymous with customer needs and expectations etc. In my point of view or with respect to HCI, quality is something beyond meeting the specifications, requirements or customer expectations. For example, consider a scenario, as you know, there is always a quality assurance department in any software house which checks the final products with reference to their specification or requirements. The products that do not fulfill their specifications or requirements they are considered bugged. In my scenario, what will be the matter if the specifications or requirements, which are being used to measure quality, are not complete? That’s why, I think, quality is beyond the conformance to specifications or requirements or even the customer expectations.

I think quality cannot be measured just by the requirements or specifications described by the customer rather you should approach to that end user who will use this product. The expectations or needs of the end user can be the measure of quality. So, we can say, as much as the product will be useable for end user as much higher will be its quality.

To understand the relationship of quality and usability in a software reference, look at the definition of software quality. “The extent to which a software product exhibits these characteristics”

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

5.2 Interdisciplinary nature of HCI

Organizational Factors Training, job design, politics, roles Work organization		Environmental Factors Noise, heating, ventilation, lighting
Health and Safety Stress, headaches, Muscular-skeleton, disorders	The User Cognitive processes and capabilities Motivation, Enjoyment, Satisfaction, Personality Experience level	Comfort Level Seating Equipment layout
User Interface Input devices, output displays, dialogue structures, User of colour, icons, commands, graphics, natural language		
Task Factors Easy, complex, novel, Task allocation, repetitive, Monitoring, skills, multi-media		
Constraints		
System Functionality Hardware, software, application		
Productivity Factors Increase output, increase quality, decrease costs, decrease errors, decrease labour requirements, and decrease production time, Increase creative and innovative ideas leading to new products		

The main factors that should be taken in account in HCI design are shown in above figure. Primarily, these relate directly to users, such as comfort and health, or are concerned with users' work, the work environment or the technology being used. What makes the analysis even more complex is that many factors inevitably interact with each other. For example, if changes are made to improve productivity, this may have undesirable effects on users' motivations and levels of satisfaction because issues relating to job design and work organization are ignored.

Case Study – Ticketing System

A small travel agency with a number of shops distributed throughout the country decides that, in order to survive in the travel industry, it needs to install an efficient ticketing system. Current practice involves sales staff in a lengthy procedure for issuing tickets to customers. First they have to call an airline to check if there are any vacant seats for the time when the customer wishes to fly. Then they have to check with the customer which of the available seats is suitable before making a reservation with the airline. The ticket is then written out by hand. In addition, the customer needs a receipt and an itinerary, which are also written by hand. One of the biggest problems with this practice is getting a telephone connection to the airline. This means that customers often have to wait while a frustrated sales assistant keeps trying in vain. To overcome this problem it is common practice to ask the customers to come back later in the hope that the sales staff will manage to get through to the airline in the meantime. Another time-consuming job is accounting for each ticket that has been issued, and the sales staff has to do this by hand every two weeks.

Before deciding to get new system the branch manager does some background research into how the agency really functions. She starts by visiting branches in a sister company that is using a computerized ticketing system. After talking to the staff for just a short time she discovers that there are problems. The sales staff complains that the computer is always going wrong and that they don't trust it. Furthermore, they can't understand some of the messages that it produces when they make errors. In fact, they wish they could go back to the old un-computerized way of working. Sales figures since the new system was installed are also disappointing and a large number of staff have left the office. Not surprisingly, the manager is consultants examine the users' needs and how they currently go about their work in detail and also find out exactly what the goals of the company are. They then recommend a system with the following characteristics:

- Immediate ticket booking via a computer connection (alleviating the problem of engaged phone line),
- Automatic print-out of tickets, itineraries and receipts (eliminating the need to write these by hand and thereby reducing the possibility of errors and illegibility while speeding up the process),
- Direct connection between the booking system and accounting (speeding up the process of accounting),
- Elimination of booking forms (reducing overheads as less paper and time are used).

The consultants suggest making the interface to the system mimic the non-computerized task, so menus and forms are used, which means that the sales assistant only has to select options and fill in the resulting forms by typing at a keyboard.

The consultants are optimistic that customer satisfaction will improve because customer will get their tickets on the spot. They point out to the manager, however, that in order to get the most out of the new system the layout of the agency needs to be changed to make it comfortable for the sales staff to operate the compute, while still providing scope for direct contact with customers. Staff will also need training, and some careful changes to existing jobs are needed too—job design. In particular, technology means that they will need support during the period of change. Staff will also need to know how to cope when an airline's computer malfunctions. Changes in employment conditions must also be examined. For instance, if staff is expected to carry out more transactions in less time, are they going to be rewarded for this extra activity? Staff relations with other staff in the company who will not be using the computerized system must also be taken into account. For example, problems associated with technology power such as feelings of elitism among staff that know how to use the new technology, will need to be resolved.

HCI understands the Complex Relationship between Human and Computers, which are two distinct 'Species'. Successful Integration is dependent upon a better understanding of both Species. Hence HCI borrows and establishes its roots in Disciplines concerned with both.

Human

- Cognitive Psychology
- Social Organizational Psychology

- Ergonomics and Human Factors
- Linguistics
- Philosophy
- Sociology
- Anthropology

Machine

- Computer Science
- Artificial Intelligence

Other

- Engineering
- Design

Cognitive Psychology

Psychology is concerned primarily with understanding human behavior and the mental processes that underlie it. To account for human behavior, cognitive psychology has adopted the notion of information processing. Everything we see, feel, touch, taste, smell and do is couched in terms of information processing. The objective cognitive psychology has been to characterize these processes in terms of their capabilities and limitations. [2]

Social and Organizational psychology

Social psychology is concerned with studying the nature and causes of human behavior in a social context. Vaske and Grantham identify the four core concerns of social psychology as:

- The influence of one individual on another person's attitudes and behavior
- The impact of a group on its members' attitude and behavior
- The impact of a member on a group's activities and structure
- The relationship between the structure and activities of different groups.

The role of social and organizational psychology is to inform designers about social and organizational structures and about how the introduction of computers will influence working practices. [2]

Ergonomics or human factor

Ergonomics, or human factor, developed from the interests of a number of different disciplines primarily during World War II. Its purpose is to define and design tools and various artifacts for different work, leisure and domestic environments to suit the capabilities and capacities of users.

The role of ergonomist is to translate information from the above sciences into the context of design, whether for a car seat or a computer system. The objective is to maximize an operator's safety, efficiency and reliability of performance, to make a task easier, and to increase feelings of comfort and satisfaction. [2]

Linguistics

Linguistics is the scientific study of language (Lyons, 1970). From the point of view of HCI there are several issues that may be better understood by applying knowledge and theories from linguistics. For example, in the early days of command languages there was some debate about whether or not the object to which a command applied should come before or after the command itself. When deleting a file called 'xyz', for example, should you type delete 'xyz' or 'xyz' delete. [2]

Philosophy, Sociology and Anthropology

A major concern of these disciplines until relatively recently has been to consider the implication of the introduction of IT to society. More recently, attempts are being made to apply methods developed in the social sciences to the design and evaluation of systems. The reason for applying social science methods of analysis to HCI, it is argued, are that a more accurate description of the interaction between users, their work, the technology that they use and the environment in which they are situated can be obtained. One application of social science methods has been to characterize computer supported cooperative writing (CSCW), which is concerned with sharing software and hardware among groups of people working together. The is to design tools and ways of working which optimize the shared technology so that maximum benefit can be obtained by all those who use or are affected by it. [2]

Artificial Intelligence

Artificial Intelligence (AI) is concerned with the design of intelligent computer programs which simulate different aspects of intelligent human behavior. The relationship of AI to HCI is mainly concerned with user's needs when interacting with an intelligent interface. These include, for example, the use of natural language and speech as a way of communicating with a system and the need for system to explain and justify its advice. [2]

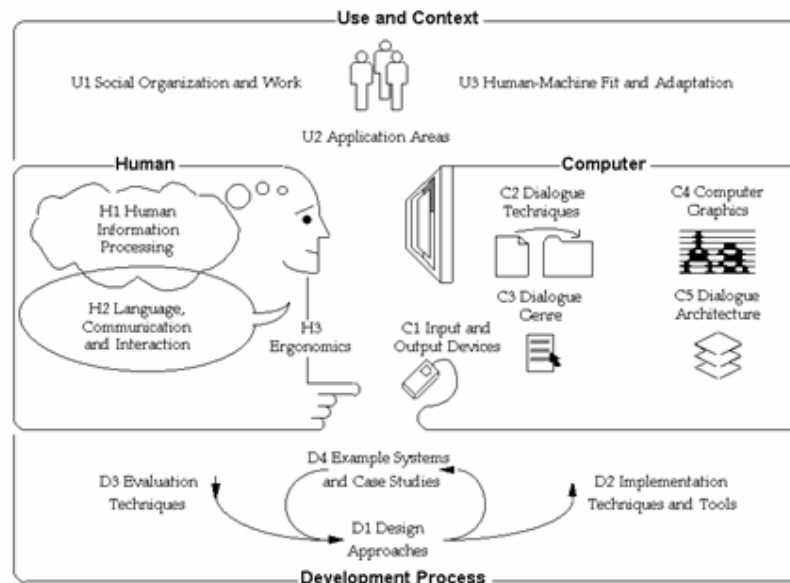
Computer Science

One of the main contributions of computer science to HCI is to provide knowledge about the capabilities of technology and ideas about how this potential can be harnessed. In addition, computer scientists have been concerned about developing various kinds of techniques to support software design, development and maintenance. In particular, there has been a strong interest in automating design and development when feasible. [2]

Engineering and design

Engineering is an applied science, which relies heavily on model building and empirical testing. Design contributes creative skills and knowledge to this process. In many respects the greatest influence of engineering on HCI and subsequently on interface and system development is through software engineering.

Design too is a well-established discipline in its own right, which has potential benefits when applied to HCI problems. An obvious example is graphic design.[2]



Reference:

Human-Computer Interaction by Jenny Preece

Software Engineering A Practitioner's Approach by Roger S. Pressman

Definitions of Quality - Sandeep's Quality Page.htm

Lecture 6.

Cognitive Frameworks

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand the importance of Cognition
- Understand different cognitive frameworks in HCI

6.1 Introduction

Imagine trying to drive a car by using just a computer keyboard. The four arrow keys are used for steering, the space bar for braking, and the return key for accelerating. To indicate left you need to press the F1 key and to indicate right the F2 key. To sound your horn you need to press the F3 key. To switch the headlights on you need to use the F4 key and, to switch the windscreen wipers on, the F5 key. Now imagine as you are driving along a road a ball is suddenly kicked in front of you. What would you do? Bash the arrow keys and the space bar madly while pressing the F4 key? How would rate your chance of missing the ball?

Most of us would balk at the very idea of driving a car this way. Many early video games, however, were designed along these lines: the user had to press an arbitrary combination of function keys to drive or navigate through the game. More recently, computer consoles have been designed with the user's capabilities and demands of the activity in mind. Much better way of controlling and interacting, such as through using joysticks and steering wheels are provided that map much better onto the physical and cognitive aspects of driving and navigating.

We have to understand the limitations of the people to ease them. Let us see what cognitive psychology is and how it helps us.

Cognitive Psychology

Psychology is concerned primarily with understanding human behavior and the mental processes that underlie it. To account for human behavior, cognitive psychology has adopted the notion of information processing. Everything we see, feel, touch, taste, smell and do is couched in terms of information processing. The objective cognitive psychology has been to characterize these processes in terms of their capabilities and limitations. For example, one of the major preoccupations of cognitive psychologists in the 1960s and 1970s was identifying the amount of information that could be processed and remembered at any one time. Recently, alternative psychological frameworks have been sought which more adequately characterize the way people work with each other and with the various artifacts, including computers, that they have use. Cognitive psychology have attempted to apply relevant psychological principles to HCI by using a variety of methods, including development of guidelines, the use of models to predict human performance and the use of empirical methods for testing computer systems.

Cognition

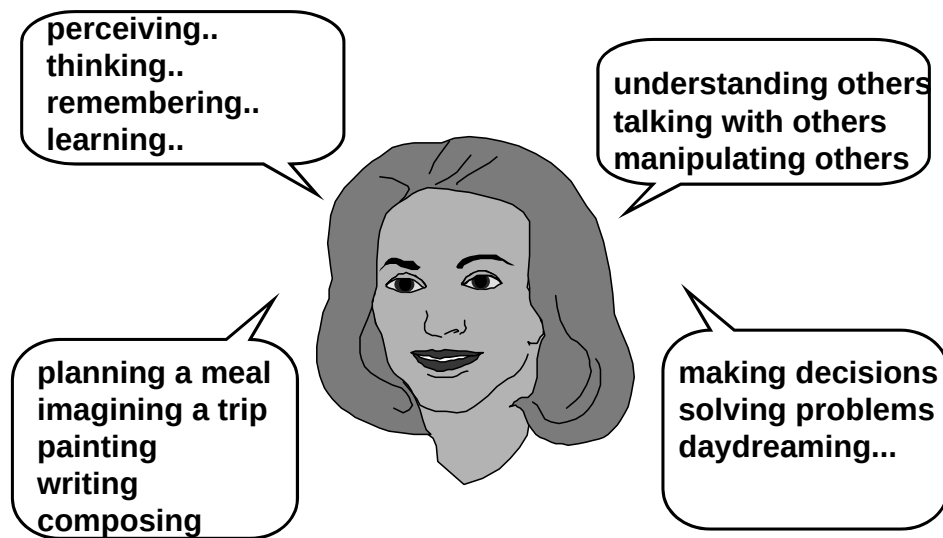
The dominant framework that has characterized HCI has been cognitive. Let us define cognition first:

Cognition is what goes on in our heads when we carry out our everyday activities.

In general, cognition refers to the processes by which we become acquainted with things or, in other words, how we gain knowledge. These include understanding, remembering, reasoning, attending, being aware, acquiring skills and creating new ideas.

As figure indicates there are different kinds of cognition.

What goes on in the mind?



The main objective in HCI has been to understand and represent how human interact with computers in term of how knowledge is transmitted between the two. The theoretical grounding for this approach stems from cognitive psychology: it is to explain how human beings achieve the goals they set.

Cognition has also been described in terms of specific kinds of processes. These include:

- Attention
- Perception and recognition
- Memory
- Learning
- Reading, speaking, and listening
- Problem solving, planning, reasoning, decision-making.

It is important to note that many of these cognitive processes are interdependent: several may be involved for a given activity. For example, when you try to learn material for an exam, you need to attend the material, perceive, and recognize it, read it, think about it,

and try to remember it. Thus cognition typically involves a range of processes. It is rare for one to occur in isolation.

6.2 Modes of Cognition

Norman (1993) distinguishes between two general modes:

1. Experiential cognition
2. Reflective cognition

Experiential cognition

It is the state of mind in which we perceive, act, and react to events around us effectively and effortlessly. It requires reaching a certain level of expertise and engagement. Examples include driving a car, reading a book, having a conversation, and playing a video game.

Reflective cognition

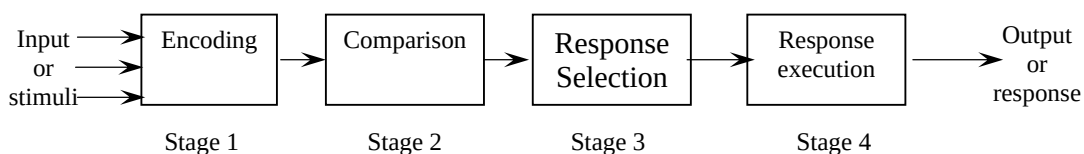
Reflective cognition involves thinking, comparing, and decision-making. This kind of cognition is what leads to new ideas and creativity. Examples include designing, learning, and writing a book.

Norman points out that both modes are essential for everyday life but that each requires different kinds of technological support.

Information processing

One of the many other approaches to conceptualizing how the mind works, has been to use metaphors and analogies. A number of comparisons have been made, including conceptualizing the mind as a reservoir, a telephone network, and a digital computer. One prevalent metaphor from cognitive psychology is the idea that the mind is an information processor.

During the 1960s and 1970s the main paradigm in cognitive psychology was to characterize humans as information processors; everything that is sensed (sight, hearing, touch, smell, and taste) was considered to be information, which the mind processes. Information is thought to enter and exit the mind through a series of ordered processing stages. As shown in figure, within these stages, various processes are assumed to act upon mental representations. Processes include comparing and matching. Mental representations are assumed to comprise images, mental models, rules, and other forms of knowledge.



Stage1 encodes information from the environment into some form of internal representation. In stage 2, the internal representation of the stimulus is compared with memorized representations that are stored in the brain. Stage 3 is concerned with deciding on a response to the encoded stimulus. When an appropriate match is made the process passes on to stage 4, which deals with the organization of the response and the necessary action. The model assumes that information is unidirectional and sequential and that each

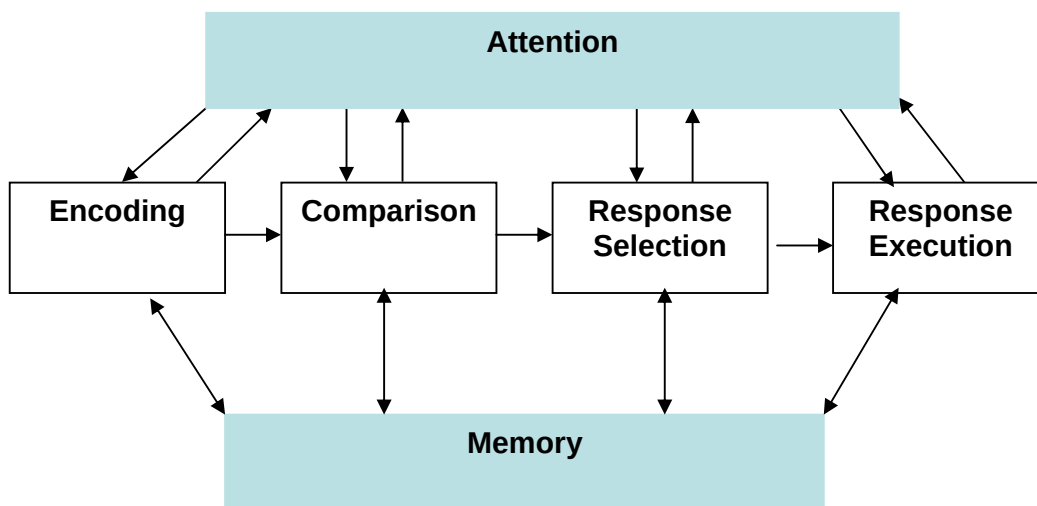
of the stages takes a certain amount of time, generally thought to depend on the complexity of the operation performed.

To illustrate the relationship between the different stages of information processing, consider the sequence involved in sending mail. First, letters are posted in a mailbox. Next, a postman empties the letters from the mailbox and takes them to central sorting office. Letters are then sorted according to area and sent via rail, road, air or ship to their destination. On reaching their destination, the letters are further sorted into particular areas and then into street locations and so on. A major aspect of an information processing analysis, likewise, is tracing the mental operations and their outcomes for a particular cognitive task. For example, let us carry out an information processing analysis for the cognitive task of determining the phone number of a friend.

Firstly, you must identify the words used in the exercise. Then you must retrieve their meaning. Next you must understand the meaning of the set of words given in the exercise. The next stage involves searching your memory for the solution to the problem. When you have retrieved the number in memory, you need to generate a plan and formulate the answer into a representation that can be translated into a verbal form. Then you would need to recite the digits or write them down.

Extending the human information processing model

Two main extensions of the basic information-processing model are the inclusion of the processes of attention and memory. Figure shows the relationship between the different processes. [3]



In the extended model, cognition is viewed in terms of:

1. How information is perceived by processors
2. How that information is attended to, and
3. How that information is processed and stored in memory.

6.3 Human processor model

The information-processing model provides a basis from which to make predictions about human performance. Hypotheses can be made about how long someone will take to

perceive and responds to a stimulus (also known as reaction time) and what bottlenecks occur if a person is overloaded with too much information. The best-known approach is the human processor model, which models the cognitive processes of a user interacting with a computer. Based on the information-processing model, cognition is conceptualized as a series of processing stages where perceptual, cognitive, motor processors are organized in relation to one another. The model predicts which cognitive processes are involved when a user interacts with a computer, enabling calculations to be made how long a user will take to carry out various tasks. This can be very useful when comparing different interfaces. For example, it has been used to compare how well different word processors support a range of editing tasks.

The information processing approach is based on modeling mental activities that happen exclusively inside the head. However, most cognitive activities involve people interacting with external kinds of representations, like books, documents, and computers—not to mention one another. For example, when we go home from wherever we have been we do not need to remember the details of the route because we rely on cues in the environment (e.g., we know to turn left at the red house, right when the road comes to a T-junction, and so on.). Similarly, when we are at home we do not have to remember where everything is because information is “out there.” We decide what to eat and drink by scanning the items in the fridge, find out whether any messages have been left by glancing at the answering machine to see if there is a flashing light, and so on. [2]

6.4 GOMS

Card et al. have abstracted a further family of models, known as GOMS (goals, operations, methods and selection rules) that translate the qualitative descriptions into quantitative measures. The reason for developing a family of models is that it enables various qualitative and quantitative predictions to be made about user performance.

Goals

These are the user’s goals, describing what the user wants to achieve. Further, in GOMS the goals are taken to represent a ‘memory point’ for the user, from which he can evaluate what should be done and to which he may return should any errors occur. [1]

Operators

These are the lowest level of analysis. They are the basic actions that the user must perform in order to use the system. They may affect the system (e.g., press the ‘X’ key) or only the user’s mental state (e.g., read the dialogue box). There is still a degree of flexibility about the granularity of operators; we may take the command level “issue the select command” or be more primitive; “move mouse to menu bar, press center mouse button...” [1]

Methods

As we have already noted, there are typically several ways in which a goal can be split into sub goals. [1]

Selection

Selection means of choosing between competing methods [1]

One of the problems of abstracting a quantitative model from a qualitative description of user performance is ensuring that two are connected. In particular, it has been noted that

the form and contents of GOMS family of models are relatively unrelated to the form and content of the model human processor and it also oversimplified human behavior. More recently, attention has focused on explaining:

- **Knowledge Representation Models**
How knowledge is represented
- **Mental Models**
How mental models (these refer to representation people construct in their mind of themselves, others, objects and the environment to help them know what to do in current and future situations) develop and are used in HCI
- **User Interaction Learning Models**
How user learn to interact and become experienced in using computer system.

With respect to applying this knowledge to HCI design, there has been considerable research in developing:

Conceptual Models

Conceptual models are (these are the various ways in which systems are understood by different people) to help designers develop appropriate interfaces.

Interface Metaphor

Interface metaphors are (these are GUIs that consists of electronic counterparts to physical objects in the real world) to match the knowledge requirements of users.

6.5 Recent development in cognitive psychology

With the development of computing, the activity of brain has been characterized as a series of programmed steps using the computer as a metaphor. Concept such as buffers, memory stores and storage systems, together with the type of process that act upon them (such as parallel verses serial, top-down verses down-up) provided psychologist with a mean of developing more advanced models of information processing, which was appealing because such models could be tested. However, since the 1980s there has been a more away from the information-processing framework with in cognitive psychology. This has occurred in parallel with the reduced importance of the model human processor with in HCI and the development other theoretical approaches. Primarily, these are the computational and the connectionist approaches. More recently other alternative approaches have been developed that has situated cognitive activity in the context in which they occur. [3]

Computational Approaches

Computational approaches continue to adopt the computer metaphor as a theoretical framework, but they no longer adhere to the information-processing framework. Instead, the emphasis is on modeling human performance in terms of what is involved when information is processed rather than when and how much. Primarily, computational models conceptualize the cognitive system in terms of the goals, planning and action that are involved in task performance. These aspects include modeling: how information is organized and classified, how relevant stored information is retrieved, what decisions are made and how this information is reassemble. Thus tasks are analyzed not in terms of the

amount of information processed per se in the various stages but in terms of how the system deals with new information. [3]

Connectionist Approaches

Connectionist approaches, otherwise known as neural networks or parallel distributed processing, simulate behavior through using programming models. However, they differ from conceptual approaches in that they reject the computer metaphor as a theoretical framework. Instead, they adopt the brain metaphor, in which cognition is represented at the level of neural networks consisting of interconnected nodes. Hence all cognitive processes are viewed as activations of the nodes in the network and the connections between them rather than the processing and manipulation of information. [3]

6.6 External Cognition

External cognition is concerned with explaining the cognitive processes involved when we interact with different external representations. A main goal is to explicate the cognitive benefits of using different representations for different cognitive activities and the processes involved. The main one includes:

1. externalizing to reduce memory load
2. computational offloading
3. Annotating and cognitive tracing.

Externalizing to reduce memory load

A number of strategies have been developed for transforming knowledge into external representations to reduce memory load. One such strategy is externalizing things we find difficult to remember, such as birthdays, appointments and addresses.

Externalizing, therefore, can help reduce people's memory burden by:

- reminding them to do something (e.g., to get something for their mother's birthday)
- reminding them of what to do (e.g., to buy a card)
- reminding them of when to do something (send it by a certain date)

Computational offloading

Computational offloading occurs when we use a tool or device in conjunction with an external representation to help us carry out a computation. An example is using pen or paper to solve a math problem.[2]

Annotating and cognitive tracing

Another way in which we externalize our cognitions is by modifying representations to reflect changes that are taking place that we wish to mark. For example, people often cross thinks off in to-do list to show that they have been completed. They may also reorder objects in the environment; say by creating different piles as the nature of the work to be done changes. These two kinds of modification are called annotating and cognitive tracing:

- Annotating involves modifying external representations, such as crossing off underlining items.
- Cognitive tracing involves externally manipulating items different orders or structures.

Information Visualization

A general cognitive principle for interaction design based on the external cognition approach is to provide external representations at the interface that reduce memory load and facilitates computational offloading. Different kinds of information visualizations can be developed that reduce the amount of effort required to make inferences about a given topic (e.g., financial forecasting, identifying programming bugs). In so doing, they can extend or amplify cognition, allowing people to perceive and do activities tat they couldn't do otherwise. [2]

6.7 Distributed cognition

Distributed cognition is an emerging theoretical framework whose goal is to provide an explanation that goes beyond the individual, to conceptualizing cognitive activities as embodied and situated within the work context in which they occur. Primarily, this involves describing cognition as it is distributed across individuals and the setting in which it takes place. The collection of actors (more generally referred to just as 'people' in other parts of the text), computer systems and other technology and their relations to each other in environmental setting in which they are situated are referred to as functional systems. The functional systems that have been studied include ship navigation, air traffic control, computer programmer teams and civil engineering practices.

A main goal of the distributed cognition approach is to analyze how the different components of the functional system are coordinated. This involves analyzing how information is propagated through the functional system in terms of technological cognitive, social and organizational aspects. To achieve this, the analysis focuses on the way information moves and transforms between different representational states of the objects in the functional system and the consequences of these for subsequent actions.[3]

References:

- [1] Human Computer Interaction by Alan Dix
- [2] Interaction Design by Jenny Preece
- [3] Human Computer Interaction by Jenny Preece

Lecture 7.

Human Input-Output Channels – Part I

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand role of input-output channels
- Describe human eye physiology and
- Discuss the visual perception

7.1 Input Output channels

A person's interaction with the outside world occurs through information being received and sent: input and output. In an interaction with a computer the user receives information that is output by the computer, and responds by providing input to the computer – the user's output become the computer's input and vice versa. Consequently the use of the terms input and output may lead to confusion so we shall blur the distinction somewhat and concentrate on the channels involved. This blurring is appropriate since, although a particular channel may have a primary role as input or output in the interaction, it is more than likely that it is also used in the other role. For example, sight may be used primarily in receiving information from the computer, but it can also be used to provide information to the computer, for example by fixating on a particular screen point when using an eye gaze system.

Input in human is mainly through the senses and output through the motor control of the effectors. There are five major senses:

- Sight
- Hearing
- Touch
- Taste
- Smell

Of these first three are the most important to HCI. Taste and smell do not currently play a significant role in HCI, and it is not clear whether they could be exploited at all in general computer systems, although they could have a role to play in more specialized systems or in augmented reality systems. However, vision hearing and touch are central.

Similarly there are a number of effectors:

- Limbs
- Fingers
- Eyes
- Head

- Vocal system.

In the interaction with computer, the fingers play the primary role, through typing or mouse control, with some use of voice, and eye, head and body position.

Imagine using a personal computer with a mouse and a keyboard. The application you are using has a graphical interface, with menus, icons and windows. In your interaction with this system you receive information primarily by sight, from what appears on the screen. However, you may also receive information by ear: for example, the computer may 'beep' at you if you make a mistake or to draw attention to something, or there may be a voice commentary in a multimedia presentation. Touch plays a part too in that you will feel the keys moving (also hearing the 'click') or the orientation of the mouse, which provides vital feedback about what you have done. You yourself send information to the computer using your hands either by hitting keys or moving the mouse. Sight and hearing do not play a direct role in sending information in this example, although they may be used to receive information from a third source (e.g., a book or the words of another person) which is then transmitted to the computer.

7.2 Vision

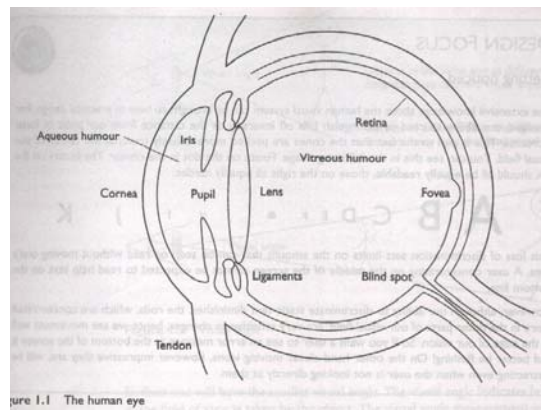
Human vision is a highly complex activity with range of physical and perceptual limitations, yet it is the primary source of information for the average person. We can roughly divide visual perception into two stages:

- the physical reception of the stimulus from outside world, and
- The processing and interpretation of that stimulus.

On the one hand the physical properties of the eye and the visual system mean that there are certain things that cannot be seen by the human; on the other interpretative capabilities of visual processing allow images to be constructed from incomplete information. We need to understand both stages as both influence what can and can not be perceived visually by a human being, which in turn directly affect the way that we design computer system. We will begin by looking at the eye as a physical receptor, and then go onto consider the processing involved in basic vision.

The human eye

Vision begins with light. The eye is a mechanism for receiving light and transforming it into electrical energy. Light is reflected from objects in the world and their image is focused upside down on the back of the eye. The receptors in the eye transform it into electrical signals, which are passed to brain.



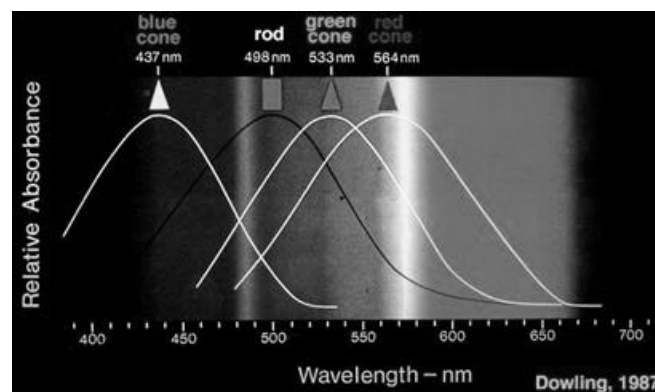
The eye has a number of important components as you can see in the figure. Let us take a deeper look. The cornea and lens at the front of eye focus the light into a sharp image on the back of the eye, the retina. The retina is light sensitive and contains two types of photoreceptor: rods and cones.

Rods

Rods are highly sensitive to light and therefore allow us to see under a low level of illumination. However, they are unable to resolve fine detail and are subject to light saturation. This is the reason for the temporary blindness we get when moving from a darkened room into sunlight: the rods have been active and are saturated by the sudden light. The cones do not operate either as they are suppressed by the rods. We are therefore temporarily unable to see at all. There are approximately 120 million rods per eye, which are mainly situated towards the edges of the retina. Rods therefore dominate peripheral vision.

Cones

Cones are the second type of receptor in the eye. They are less sensitive to light than the rods and can therefore tolerate more light. There are three types of cone, each sensitive to a different wavelength of light. This allows color vision. The eye has approximately 6 million cones, mainly concentrated on the fovea.



Fovea

Fovea is a small area of the retina on which images are fixated.

Blind spot

Blind spot is also situated at retina. Although the retina is mainly covered with photoreceptors there is one blind spot where the optic nerve enter the eye. The blind spot has no rods or cones, yet our visual system compensates for this so that in normal circumstances we are unaware of it.

Nerve cells

The retina also has specialized nerve cells called ganglion cells. There are two types:

X-cells

These are concentrated in the fovea and are responsible for the early detection of pattern.

Y-cells

These are more widely distributed in the retina and are responsible for the early detection of movement. The distribution of these cells means that, while we may not be able to detect changes in pattern in peripheral vision, we can perceive movement.

7.3 Visual perception

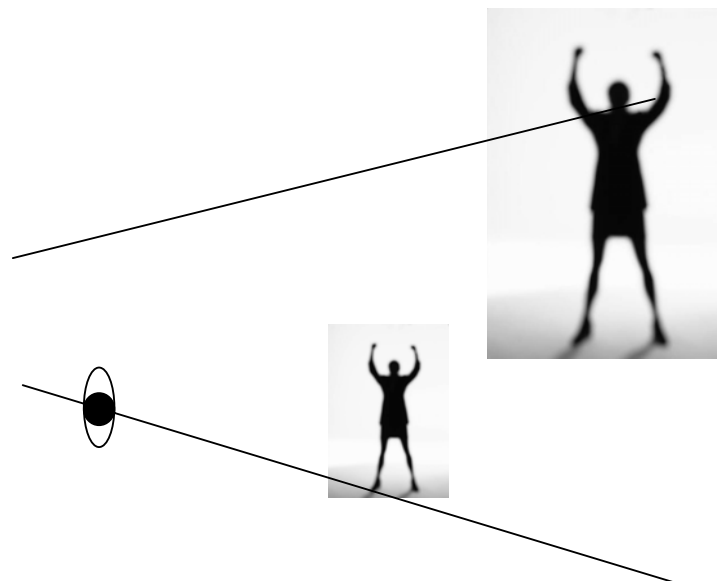
Understanding the basic construction of the eye goes some way to explaining the physical mechanism of vision but visual perception is more than this. The information received by the visual apparatus must be filtered and passed to processing elements which allow us to recognize coherent scenes, disambiguate relative distances and differentiate color. Let us see how we perceive size and depth, brightness and color, each of which is crucial to the design of effective visual interfaces.

Perceiving size and depth

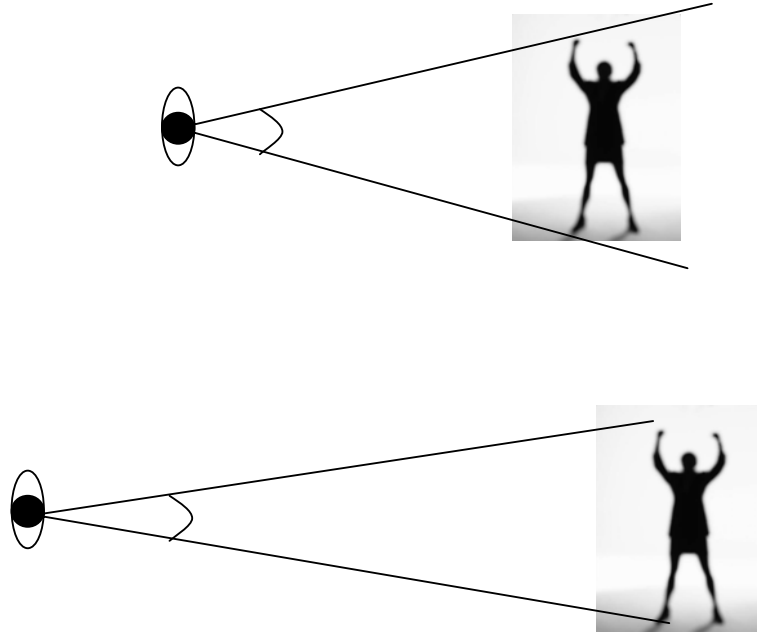
Imagine you are standing on a hilltop. Beside you on the summit you can see rocks, sheep and a small tree. On the hillside is a farmhouse with outbuilding and farm vehicles. Someone is on the track, walking toward the summit. Below in the valley is a small market town.

Even in describing such a scene the notions of size and distance predominate. Our visual system is easily able to interpret the images, which it receives to take account of these things. We can identify similar objects regardless of the fact that they appear to us to be vastly different sizes. In fact, we can use this information to judge distance.

So how does the eye perceive size, depth and relative distances? To understand this we must consider how the image appears on the retina. As we mentioned, reflected light from the object forms an upside-down image on the retina. The size of that image is specified as visual angle. Figure illustrates how the visual angle is calculated.



If we were to draw a line from the top of the object to a central point on the front of the eye and a second line from the bottom of the object to the same point, the visual angle of the object is the angle between these two lines. Visual angle is affected by both the size of the object and its distance from the eye. Therefore if two objects are at the same distance, the larger one will have the larger visual angle. Similarly, if two objects of the same size are placed at different distances from the eye, the furthest one will have the smaller visual angle, as shown in figure.



Visual angle indicates how much of the field of view is taken by the object. The visual angle measurement is given in either degrees or minutes of arc, where 1 degree is equivalent to 60 minutes of arc, and 1 minute of arc to 60 seconds of arc.

Visual acuity

So how does an object's visual angle affect our perception of its size? First, if the visual angle of an object is too small we will be unable to perceive it at all. Visual acuity is the ability of a person to perceive fine detail. A number of measurements have been established to test visual acuity, most of which are included in standard eye tests. For example, a person with normal vision can detect a single line if it has a visual angle of 0.5 seconds of arc. Spaces between lines can be detected at 30 seconds to 1 minute of visual arc. These represent the limits of human visual perception.

Law of size constancy

Assuming that we can perceive the object, does its visual angle affect our perception of its size? Given that the visual angle of an object is reduced, as it gets further away, we might expect that we would perceive the object as smaller. In fact, our perception of an object's size remains constant even if its visual angle changes. So a person's height is perceived as constant even if they move further from you. This is the law of size constancy, and it indicates that our perception of size relies on factors other than the visual angle.

One of these factors is our perception of depth. If we return to the hilltop scene there are a number of cues, which can use to determine the relative positions and distances of the objects, which we see. If objects overlap, the object that is partially covered is perceived to be in the background, and therefore further away. Similarly, the size and height of the object in our field of view provides a cue to its distance. A third cue is familiarity: if we expect an object to be of a certain size then we can judge its distance accordingly.

Perceiving brightness

A second step of visual perception is the perception of brightness. Brightness is in fact a subjective reaction to level of light. It is affected by luminance, which is the amount of light emitted by an object. The luminance of an object is dependent on the amount of light falling on the object's surface and its reflective prosperities. Contrast is related to luminance: it is a function of the luminance of an object and the luminance of its background.

Although brightness is a subjective response, it can be described in terms of the amount of luminance that gives a just noticeable difference in brightness. However, the visual system itself also compensates for changes in brightness. In dim lighting, the rods predominate vision. Since there are fewer rods on the fovea, object in low lighting can be seen easily when fixated upon, and are more visible in peripheral vision. In normal lighting, the cones take over.

Visual acuity increases with increased luminance. This may be an argument for using high display luminance. However, as luminance increases, flicker also increases. The eye will perceive a light switched on and off rapidly as constantly on. But if the speed of switching is less than 50 Hz then the light is perceived to flicker. In high luminance flicker can be perceived at over 50 Hz. Flicker is also more noticeable in peripheral vision. This means that the larger the display, the more it will appear to flicker.

Perceiving color

A third factor that we need to consider is perception of color. Color is usually regarded as being made up of three components:

- hue
- intensity
- saturation

Hue

Hue is determined by the spectral wavelength of the light. Blues have short wavelength, greens medium and reds long. Approximately 150 different hues can be discriminated by the average person.

Intensity

Intensity is the brightness of the color.

Saturation

Saturation is the amount of whiteness in the colors.

By varying these two, we can perceive in the region of 7 million different colors. However, the number of colors that can be identified by an individual without training is far fewer.

The eye perceives color because the cones are sensitive to light of different wavelengths. There are three different types of cone, each sensitive to a different color (blue, green and red). Color vision is best in the fovea, and worst at the periphery where rods predominate. It should also be noted that only 3-4 % of the fovea is occupied by cones which are sensitive to blue light, making blue acuity lower.

Finally, we should remember that around 8% of males and 1% of females suffer from color blindness, most commonly being unable to discriminate between red and green. The capabilities and limitations of visual processing

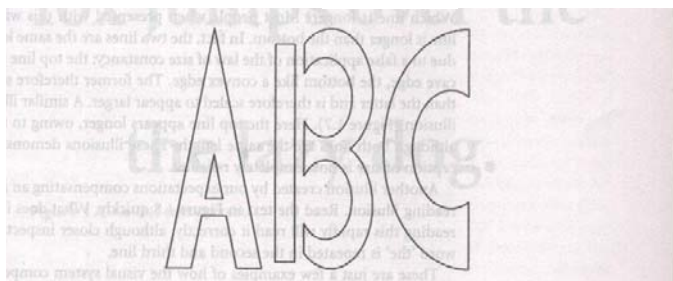
In considering the way in which we perceive images we have already encountered some of the capabilities and limitations of the human visual processing system. However, we have concentrated largely on low-level perception. Visual processing involves the transformation and interpretation of a complete image, from the light that is thrown onto the retina. As we have already noted, our expectations affect the way an image is perceived. For example, if we know that an object is a particular size, we will perceive it as that size no matter how far it is from us.

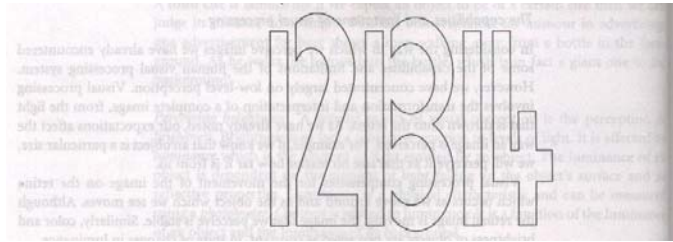
Visual processing compensates for the movement of the image on the retina which occurs as we around and as the object which we see moves. Although the retinal image is moving, the image that we perceive is stable. Similarly, color and brightness of objects are perceived as constant, in spite of changes in luminance.

This ability to interpret and exploit our expectations can be used to resolve ambiguity. For example consider the image shown in figure 'a'. What do you perceive? Now consider figure 'b' and 'c'. the context in which the object appears allow our expectations to clearly disambiguate the interpretation of the object, as either a B or 13.

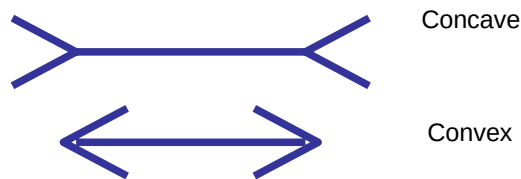


Figure a





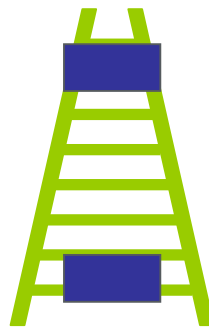
However, it can also create optical illusions. Consider figure 'd'. Which line is longer?



the Muller Lyer illusion

Figure d

A similar illusion is the Ponzo illusion as shown in figure



Another illusion created by our expectations compensating an image is the proofreading illusion. Example is shown below

The quick brown
fox jumps over the
lazy dog.

The way that objects are composed together will affect the way we perceive them, and we do not perceive geometric shapes exactly as they are drawn. For example, we tend to magnify horizontal lines and reduce vertical. So a square needs to be slightly increased in height to appear square and line will appear thicker if horizontal rather than vertical. Optical illusions also affect page symmetry. We tend to see the center of a page as being a little above the actual center – so if a page is arranged symmetrically around the actual center, we will see it as too low down. In graphic design this is known as the optical center.

Lecture 8.

Human Input-Output Channels Part II

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand role of color theory in design
- Discuss hearing perception
- Discuss haptic perception
- Understand movement

8.1 Color Theory

Color theory encompasses a multitude of definitions, concepts and design applications. All the information would fill several encyclopedias. As an introduction, here are a few basic concepts.

The Color Wheel



A color circle, based on red, yellow and blue, is traditional in the field of art. Sir Isaac Newton developed the first circular diagram of colors in 1666. Since then scientists and artists have studied and designed numerous variations of this concept. Differences of opinion about the validity of one format over another continue to provoke debate. In reality, any color circle or color wheel, which presents a logically arranged sequence of pure hues, has merit.

Primary Colors

In traditional color theory, these are the 3 pigment colors that cannot be mixed or formed by any combination of other colors. All other colors are derived from these 3 hues



PRIMARY
Red, yellow and blue

COLORS

Secondary Colors

These are the colors formed by mixing the primary colors.



SECONDARY COLORS
Green, orange and purple

Tertiary colors

These are the colors formed by mixing one primary and one secondary color.



TERTIARY COLORS
Yellow-orange, red-orange, red-purple, blue-purple, blue-green and yellow-green.

Color Harmony

Harmony can be defined as a pleasing arrangement of parts, whether it be music, poetry, color, or even an ice cream sundae.

In visual experiences, harmony is something that is pleasing to the eye. It engages the viewer and it creates an inner sense of order, a balance in the visual experience. When something is not harmonious, it's either boring or chaotic. At one extreme is a visual experience that is so bland that the viewer is not engaged. The human brain will reject under-stimulating information. At the other extreme is a visual experience that is so overdone, so chaotic that the viewer can't stand to look at it. The human brain rejects what it cannot organize, what it cannot understand? The visual task requires that we present a logical structure. Color harmony delivers visual interest and a sense of order.

In summary, extreme unity leads to under-stimulation, extreme complexity leads to over-stimulation. Harmony is a dynamic equilibrium.

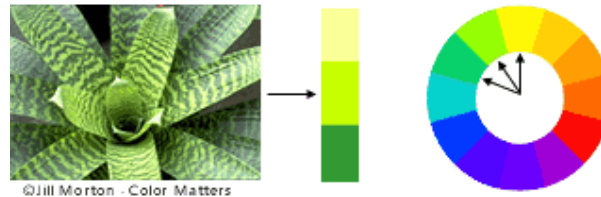
Some Formulas for Color Harmony

There are many theories for harmony. The following illustrations and descriptions present some basic formulas.

Analogous colors

Analogous colors are any three colors, which are side by side on a 12 part color wheel, such as yellow-green, yellow, and yellow-orange. Usually one of the three colors predominates.

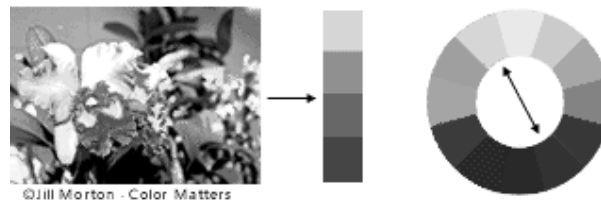
A color scheme based on analogous colors



Complementary colors

Complementary colors are any two colors, which are directly opposite each other, such as red and green and red-purple and yellow-green. In the illustration above, there are several variations of yellow-green in the leaves and several variations of red-purple in the orchid. These opposing colors create maximum contrast and maximum stability.

A color scheme based on complementary colors



Natural harmony

Nature provides a perfect departure point for color harmony. In the illustration above, red yellow and green create a harmonious design, regardless of whether this combination fits into a technical formula for color harmony.

A color scheme based on nature



Color Context

How color behaves in relation to other colors and shapes is a complex area of color theory. Compare the contrast effects of different color backgrounds for the same red square.



Red appears more brilliant against a black background and somewhat duller against the white background. In contrast with orange, the red appears lifeless; in contrast with blue-green, it exhibits brilliance. Notice that the red square appears larger on black than on other background colors.

Different readings of the same color



As we age, the color of lens in eye changes. It becomes yellow and absorb shorter wavelengths so the colors with shorter wavelength will not be visible as we aged. So, do not use blue for text or small objects. As we age, the fluid between lens and retina absorbs more light due to which eye perceive lower level of brightness. Therefore older people need brighter colors.

Different wavelengths of light focused at different distances behind eye's lens this require constant refocusing which causes fatigue. So, be careful about color combinations. Pure (saturated) colors require more focusing then less pure. Therefore do not use saturated colors in User interface unless you really need something to stand out (danger sign).

Guidelines

- Opponent colors go well together (red & green) or (yellow & blue)
- Pick non-adjacent colors on the hue circle
- Size of detectable changes in color varies. For example, it is hard to detect changes in reds, purples, & greens and easier to detect changes in yellows & blue-greens
- Older users need higher brightness levels to distinguish colors
- Hard to focus on edges created by color alone, therefore, use both brightness & color differences
- Avoid red & green in the periphery due to lack of RG cones there, as yellows & blues work in periphery
- Avoid pure blue for text, lines, & small shapes.

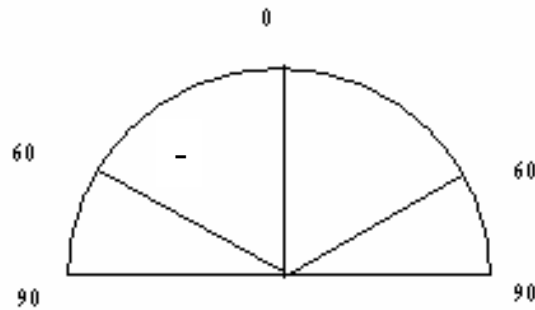
- Blue makes a fine background color
- Avoid adjacent colors that differ only in blue
- Avoid single-color distinctions but mixtures of colors should differ in 2 or 3 colors., 2 colors shouldn't differ only by amount of red

Accurate color discrimination is at ± 60 degree of straight head position.

Limit of color awareness is ± 90 degree of straight head position

8.2 Stereopsis

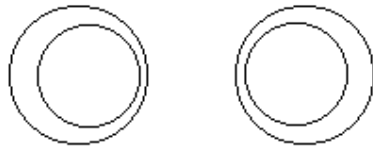
Introduction



3D vision, *binocular vision* and *stereopsis* all mean the same thing: That remarkable power of the visual sense to give an immediate perception of depth on the basis of the difference in points of view of the two eyes. It exists in those animals with overlapping optical fields, acting as a range finder for objects within reach. There are many clues to depth, but stereopsis is the most reliable and overrides all others. The sensation can be excited by presenting a different, properly prepared, view to each eye. The pair of views is called a *stereopair* or *stereogram*, and many different ways have been devised to present them to the eye. The appearance of depth in miniature views has fascinated the public since the 1840's, and still appears now and then at the present time. There was a brief, but strong, revival in the 1990's with the invention of the autostereogram. Stereopsis also has technical applications, having been used in aerial photograph interpretation and the study of earth movements, where it makes small or slow changes visible.

The word stereopsis was coined from the Greek στερεος, solid or firm, and οψις, look or appearance. Since terms derived from Greek are often used in this field, it may be useful to have a brief discussion. Single and double vision are called *haplopia* and *diplopia*, respectively, from 'απλους (haplous) and διπλους (diplous), which mean "single" and "double". Haplopia is the happy case; with diplopia we are seeing double. The use of a Greek term removes the connotations that may attach to a common English word, and sounds much more scientific. Note that the "opia" part of these words refers to "appearance", and does not come from a word for "eye". The -s- has been dropped for euphony. Otherwise, the closest Greek to "opia" means a cheese from milk curdled with fig juice. "Ops", for that matter is more usually associated with cooked meat or evenings. In fact, words like "optic" come from οπτικός, meaning "thing seen", from the future οψομαι of οραω, (horao) "to see", not from a reference to the eye. The Latin *oculus* does mean "eye" and is used in many technical terms, like *binocular*, which combines Greek and Latin.

Stereopsis



If you fuse this stereopair, you will see a frustum of a right cone in three dimensions.

A Simple Stereopair

Stereopsis is the direct sensing of the distance of an object by comparing the images received by the two eyes. This is possible only when the eyes of a creature look in the same direction, and have overlapping fields. The placing of the two eyes this way gives up the opportunity of a wide field of view obtained with eyes on the sides of the head. Predators find it best to have eyes in front, prey to have them on the sides. Stereopsis yields benefits for close work, such as fighting for cats and hand work for humans. Note that

normal binocular vision is *single*, so that the two images have been *fused* by the brain. There is no evidence that the image resulting from the many simple eyes of an insect is not also fused in a similar way.

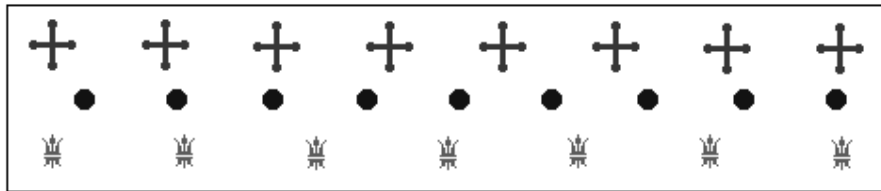
Visual perception makes use of a large number of distance clues to create its three-dimensional picture from the two-dimensional retinal images. Strong clues are the apparent sizes of objects of known size, overlapping and parallax, shadows and perspective. Weaker clues are atmospheric perspective (haze and scattering), speed of movement, and observed detail. The strongest clue of all, however, is stereopsis, which overrides all other evidence save touch itself. The convergence of the optic axes of the two eyes, and their distance accommodation, when fixated on an object, do not seem to be strong clues, though some have believed them to be. Although we have two eyes, we usually have only one visual world, which is a remarkable and important fact calling for explanation. Stereopsis gives a reliable distance clue as far away as 450 metres, Helmholtz estimated. The fineness of the comparison that must be made by the visual system is remarkable.

The interpretation of retinal images to produce stereopsis is entirely mental, and must be learned. When the images on the eyes are consistent with the observation of a single object, the two flat images *fuse* to form a vivid three-dimensional image. With practice, fusion can be achieved with two pictures side by side and the eyes voluntarily diverged so that each eye sees its picture straight ahead, though accommodated for the actual distance. Both the original pictures remain in view, but a third, fused, image appears before them when the concentration is diverted to it that appears strikingly solid. The brain regards this fused image as the real one, the others as mere ghosts. This skill is called *free fusion*, and requires considerable practice to acquire. In free fusion, both the convergence of the eyes, and their distance accommodation, are inconsistent with the actual location of the image, and must be overridden by stereopsis. It shows, incidentally, that convergence of the optic axes is not a strong depth clue. By the use of a *stereoscope*, one can achieve fusion without diverging the eyes, or focusing on a close object with the eyes so diverged, so no practice or skill is required. A stereoscope mainly changes the directions in which the two images are seen so that they can both be fixated by normally converged eyes. The two images are called a *stereo pair*.

When the images on the retinas are too different to be views of the same object, *rivalry* occurs, and either one image is favored and the other suppressed, or a patchwork of parts

of the two images is seen. When everything corresponds except the illumination or colour, the fused image exhibits *lustre*.

The fundamentals of stereopsis were discovered by Charles Wheatstone in 1836, when stereopairs had to be created by drawing (this could be aided with the camera obscura, but was very difficult except for stick images). The methods of descriptive geometry can be used to create stereopairs. He designed the mirror stereoscope, which directs the view of the images into the eyes with plane mirrors and reflection at about 45°. David Brewster invented the prism stereoscope, which used prisms to deviate the light, which made a more compact and convenient apparatus. Lenses can also be used to decrease the viewing distance and make fixation easier. Photography was the natural way to create stereopairs. A stereopair can also be drawn in two colours with the views superimposed. When this *anaglyph* is viewed through coloured filters that present one image to each eye, fusion is easy. A similar method is to project the two images in orthogonal polarizations, and to view them through polarizing filters. Both of these methods have been used to project 3D films and transparencies before an audience. A small fraction of people, perhaps 4%, have defective stereopsis.



The pattern above demonstrates the stereoscopic *wallpaper* illusion, which was first discovered by H. Meyer in 1842, and also noted by Brewster. When viewed with the eyes parallel, a strong stereoscopic effect is seen. The green fleurs-de-lis are farthest away, the blue discs closest, and the red crosses at an intermediate distance. This is an *autostereogram*, a single figure that gives stereoscopic images to the two eyes. Since the figures in a line are identical, when the eyes are turned for free fusion, two different figures are assumed to be parallax views of the same object. The eye finds it preferable to fuse the images rather than report double vision. It is easier to fuse this autostereogram than a normal stereopair, so it is good practice for developing the useful skill of free fusion.

The mind does not have to recognize the object in a stereopair for fusion to occur. The pattern can be random, but the stereopair must represent the same random pattern as seen from the different positions of the eyes (Julesz, 1960). Even more strikingly, a single apparently random pattern can be fused auto stereographically to give a three-dimensional image. No image is seen until fusion occurs. Each point on the image must be capable of interpretation as two different points of a stereopair. This random-dot auto stereogram was widely enjoyed in the 1980's. An auto stereogram requires free fusion, which must be learned in order to appreciate them. Many people found this difficult, so the auto stereogram was usually presented as a kind of puzzle.

Psychologists have argued about stereopsis for many years, but most of their musings are not worth repeating. A widely held theory was that the two retinas were somehow mapped point-by-point, and differing image positions with respect to this reference frame was interpreted stereoptically. It seems more likely to me that the images are compared by the visual sense for differences, than by their absolute locations on the retina. In the past, psychologists have preferred mechanical explanations, where the brain and retina

are created with built-in specializations and functions, spatially localized, rather than regarding the organs as canvases, which the cognitive powers organize as necessary.

I have not discussed the broad and interesting field of optical illusions here, since they tell us nothing definite about the inner workings of the visual sense, only give examples of its operation, and also because the 'reasons' for them are controversial, and the arguments are not especially enlightening. Illusions are discussed at length in another article on this website. The oldest and most widely known illusion is the horizon illusion, in which the moon appears larger on the horizon than at the zenith. This illusion was known and discussed in antiquity, and is still the subject of much study. Its explanation is not known. For the application of the visual sense to investigation and appreciation of the world around us, Minaret's book is outstanding.

8.3 Reading

So far we have concentrated on the perception of images in general. However, the perception and processing of text is a special case that is important to interface design, which inevitably requires some textual display.

There are several stages in the reading process. First the visual pattern of the word on the page is perceived. It is then decoded with reference to an internal representation of language. The final stages of language processing include syntactic and semantic analysis and operate on phrases or sentences.

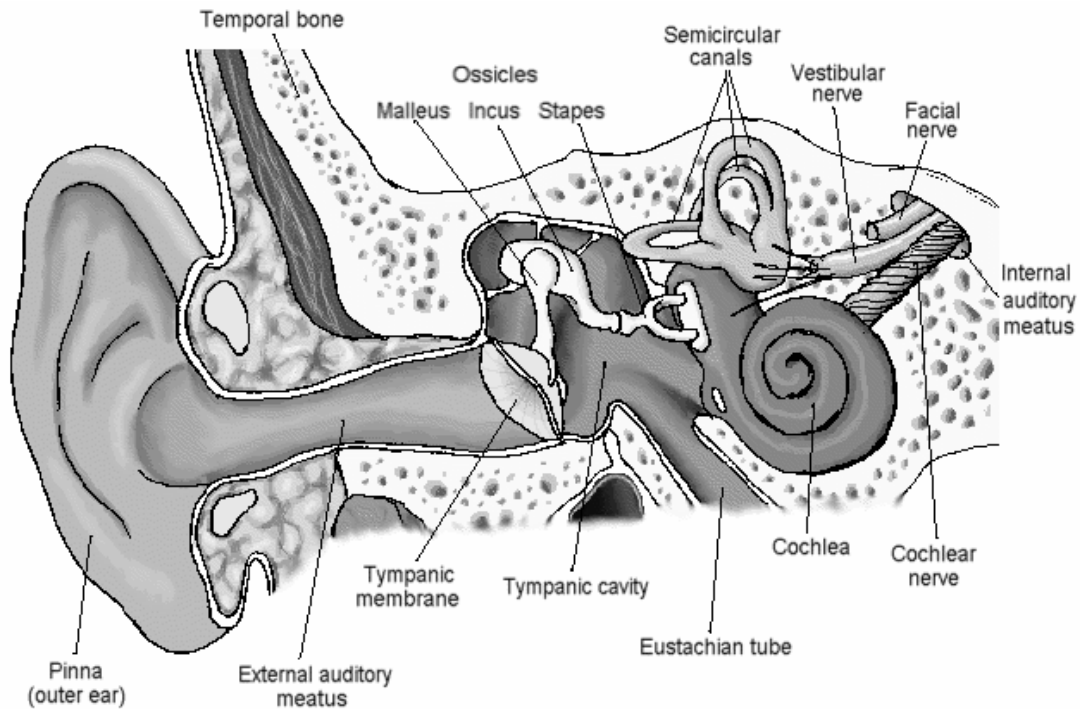
We are most interested with the first two stages of this process and how they influence interface design. During reading, the eye makes jerky movement called saccades followed by fixations. Perception occurs during the fixation periods, which account for approximately 94% of the time elapsed. The eye moves backwards over the text as well as forwards, in what are known as regressions. If the text is complex there will be more regressions.

8.4 Hearing

The sense of hearing is often considered secondary to sight, but we tend to underestimate the amount of information that we receive through our ears.

The human ear

Hearing begins with vibrations in the air or sound waves. The ear receives these vibrations and transmits them, through various stages, to the auditory nerves. The ear comprises three sections commonly known as the outer ear, middle ear and inner ear.



The outer ear is the visible part of the ear. It has two parts: the pinna, which is the structure that is attached to the sides of the head, and the auditory canal, along which sound waves are passed to the middle ear. The outer ear serves two purposes. First, it protects the sensitive middle ear from damage. The auditory canal contains wax, which prevents dust, dirt and over-inquisitive insects reaching the middle ear. It also maintains the middle ear at a constant temperature. Secondly, the pinna and auditory canal serve to amplify some sounds.

The middle ear is a small cavity connected to the outer ear by the tympanic membrane, or eardrum, and to the inner ear by the cochlea. Within the cavity are the ossicles, the smallest bones in the body. Sound waves pass along the auditory canal and vibrate the ear drum which in turn vibrates the ossicles, which transmit the vibrations to the cochlea, and so into the inner ear.

The waves are passed into the liquid-filled cochlea in the inner ear. Within the cochlea are delicate hair cells or cilia that bend because of the vibrations in the cochlear liquid and release a chemical transmitter, which causes impulses in the auditory nerve.

Processing sound

Sound has a number of characteristics, which we can differentiate.

Pitch

Pitch is the frequency of the sound. A low frequency produces a low pitch, a high frequency, a high pitch.

Loudness

Loudness is proportional to the amplitude of the sound; the frequency remains constant.

Timber

Timber related to the type of the sound

Sounds may have the same pitch and loudness but be made by different instruments and so vary in timber.

Sound characteristics

Audible range is 20 Hz to 15 KHz. Human ear can distinguish between changes less than 1.5 Hz but less accurate at higher frequencies. Different frequencies trigger neuron activity causing nerve impulses. Auditory system filters sounds e.g., Cocktail Party Effect

8.5 Touch

The third sense is touch or haptic perception. Although this sense is often viewed as less important than sight or hearing, imagine life without it. Touch provides us with vital information about our environment. It tells us when we touch something hot or cold, and can therefore act as a warning. It also provides us with feedback when we attempt to lift and object.

Haptic perception involves sensors in the skin as well as the hand and arm. The movement that accompanies hands-on exploration involves different types of mechanoreceptors in the skin (involving deformation, thermo reception, and vibration of the skin), as well as receptors in the muscles, tendons, and joints involved in movement of the object (Verry, 1998). These different receptors contribute to a neural synthesis that interprets position, movement, and mechanical skin inputs. Druryan (1997) argues that this combination of kinesthetic and sensory perception creates particularly strong neural pathways in the brain.

Haptics vs. Visual

For the science learner, kinesthetic allows the individual to explore concepts related to location, range, speed, acceleration, tension, and friction. Haptics enables the learner to identify hardness, density, size, outline, shape, texture, oiliness, wetness, and dampness (involving both temperature and pressure sensations) (Druryan, 1997; Schiffman, 1976).

When haptics is compared to vision in the perception of objects, vision typically is superior with a number of important exceptions. Visual perception is rapid and more holistic—allowing the learner to take in a great deal of information at one time. Alternatively, haptics involves sensory exploration over time and space. If you give a student an object to observe and feel, the student can make much more rapid observations than if you only gave the student the object to feel without the benefit of sight. But of interest to science educators is the question of determining what a haptic experience adds to a visual experience. Researchers have shown that haptics is superior to vision in helping a learner detect properties of texture (roughness/ smoothness, hardness/ softness, wetness/ dryness, stickiness, and slipperiness) as well as micro spatial properties of pattern, compliance, elasticity, viscosity, and temperature (Lederman, 1983; Zangaladze, et al., 1999). Vision dominates when the goal is the perception of macro geometry (shape) but haptics is superior in the perception of micro geometry (texture) (Sathian et al., 1997; Verry, 1998). Haptics and vision together are superior to either alone for many learning contexts.

While vision provides information about an object geometric feature, touch is unparalleled in its ability to extract information about materials. For a surgeon trying to decide where to begin excising a patch of cancerous tissue, it might be helpful to feel the texture and compliance, and not just rely on the shape.

Haptic Learning

Haptic learning plays an important role in a number of different learning environments. Students with visual impairments depend on haptics for learning through the use of Braille as well as other strategies.

8.6 Movement

Before leaving this section on the human's input-output channels, we need to consider motor control and how the way we move affects our interaction with computers. A simple action such as hitting a button in response to a question involves a number of processing stages. The stimulus is received through the sensory receptors and transmitted to the brain. The question is processed and a valid response generated. The brain then tells the appropriate muscles to respond. Each of these stages takes time, which can be roughly divided into reaction time and movement time.

Movement time is dependent largely on the physical characteristics of the subjects: their age and fitness, for example. Reaction time varies according to the sensory channel through which the stimulus is received. A person can react to an auditory signal in approximately 150ms, to a visual signal in 200ms and to pain in 700ms.

Movement perception

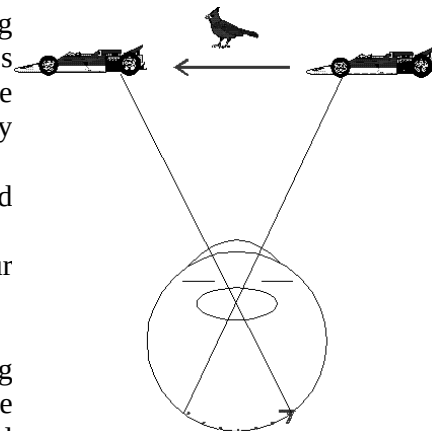
Assume that while you are staring at the bird, a racing car zooms by. The image of the car will travel across your retina as indicated by the dotted line with the arrow. This image movement will cause you to say that the car moves from your right to your left.

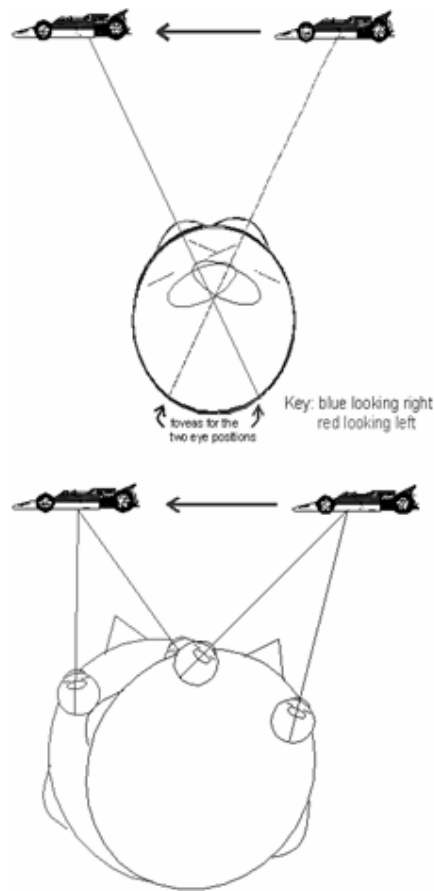
Now suppose you were looking at the car and followed its movement as it passes in front of you.

This time you are following the car by moving your eyes from right to left.

Just as before, your percept is that of the car moving from right to left. This is true even though the image remains on the fovea during the motion of the car and your eyes.

Third illustration shows that another way to follow the racing car is to keep the eyes steady and to move just the head. This causes the image to project to exactly the same retinal location at each instant (assuming you move your head at precisely the correct angular velocity) as the car moves from right to left.





Once again, the percept is of the car moving from right to left. This percept will be the same as the two previous illustrations. How the brain distinguishes these different ways of following moving objects is the subject of much research. One more thing, although I have presented three distinct ways of following moving objects, these illustrations are gross simplifications. In point of fact, when we follow moving objects we use various combinations of head and eye movements.

The illustrations that, undoubtedly you have been looking at demonstrate that motion perception is very complex. Recall that we perceive motion if we hold our heads and eyes still as a moving object passes in front of us. If we decide to hold our heads still and let our eyes follow the object we still see it move. Finally, we could even decide to hold our eyes steady and move only our head to follow an object. The interesting thing is all three modes of viewing a moving object result in about the same perception.

So far we have been concerned with perceiving real movement. By real movement I mean that the physical stimulus is actually moving and we perceive it as moving. It is possible to perceive motion when the stimulus is not moving. An example is the motion after effect (MAE) demonstration that was loaned to me by Dr. Ben Bauer, Trent University.

Here is a demonstration you can observe for yourself. If you have the opportunity to view a waterfall, (e.g., Niagara Falls) look at the falling water for about a minute and then allow your gaze to fall on any stationary object. A building would be excellent. If you do this, the texture of the building, perhaps even the windows will appear to move up.

Waterfalls usually are not readily available. However, you can easily build your own MAE apparatus. Take a round paper plate. Draw a dozen or so heavy lines radiating out from the middle of the plate. Then with a pin attach the plate through its center to the eraser end of a pencil. Now spin the plate at a moderate speed. Don't spin it so fast that the lines become an indistinct blur. After viewing the spinning plate for about a minute stop it and continue to look at the radiating lines. What do you suppose you will see? If you see what most people notice the radiating lines, which are actually stationary, will appear to rotate in the direction opposite to that which you spun the plate originally. If that is way you saw you witnessed the MAE. It is useful to try this demonstration with the paper plate because it will convince you that there are no special tricks involved with the MAE demo I mentioned above.

The phenomenon of Motion after Effects (MAE) has been studied intensively by visual scientists for many years. One explanation of how the MAE works is the following. The visual system has motion detectors that, like most neurons, undergo spontaneous activity.

You normally do not see motion when there is none because the spontaneous activity is in balance. However, when you viewed the downward motion of the black bars you adapted the motion detectors for motion in the downward direction. When the real motion stopped, the spontaneous activity was no longer in balance, the upward spontaneous activity being slightly stronger and thus the black bars appear to drift upward. The adaptation effect lasts for a short time, the motion detection system quickly becomes balanced again and the apparent movement stops.

Another example of motion being seen, when there is no physical motion, is the phi phenomenon. To those unacquainted with the field of vision research this phenomenon is probably unknown. However, all of you have seen it. The simplest demonstration of the phi phenomenon is to have two illuminated spots of light about 6 to 8 inches apart. When these lights alternately go on and off one usually see a single spot of light moving back and forth.

This principle is used in many movie marquees where one sees a pattern of lights moving around the display. In fact, there is no physical motion, only a series of lights going on and off. Then, of course there are the movies. Movies are a series of single frames presented in rapid succession. No one would doubt the perception of movement seen in the cinema. Yet, if you analyze the strips of film that yield these images all you would see is a series of frames each with a slightly different image. When they are rapidly projected on to the viewing screen motion is seen.

A similar technique is used with cartoons. The illustrator actually draws a series of pictures. When they are rapidly presented to the viewer motion of the cartoon characters is seen.

There are two other instances when movement is perceived. Have you ever sat in a train or bus station patiently waiting to get moving? Then all of a sudden, low and behold there you go. Or are you? You feel no vibration, something feels wrong. Then you notice that it is the vehicle (train or bus) right next to you that is moving and it just felt as if you were moving. This is called induced motion.

Finally, (and this is an experiment you can try at home) view a small very dim light in an otherwise completely dark room. Make sure that the light is in a fixed position and not

moving. After sometime in the dark, the small light will appear to move somewhat randomly. This is called auto kinetic movement.

Here is another little experiment you can try. Look around your surroundings freely moving your eyes. As you move your eyes around are the stationary objects moving? Probably not. Now look at some object and with your finger rapidly press against your eyeball by pushing on your eyelid. (Don't push directly against the white (sclera) area). As you force your eye to move you will probably notice that whatever you are looking at starts to jump around. So you can see that it makes a difference whether you move your eyes normally or cause them to move in an unusual manner.

Electro physiologists are scientists who insert tiny electrode into the brain of experimental subjects. They have discovered that there are cortical neurons which are specialized for movement. In fact, these neurons often are so specialized that they will respond best when the motion is in a specific direction. E. Bruce Goldstein presents a neural model in his textbook, which shows how the early retinal neural processing could occur which results in a signal being sent to the brain which says that movement has occurred in a specific direction.

How to use MAE

Fixate the red square in the center of the diagram as the black bars move down. When the black bars stop moving down, continue to fixate the red square and pay attention to the black bars. What if anything do the black bars appear to be doing? If they do not appear to do anything, try running the demonstration again by clicking on the refresh icon at the top of your screen. If the black bars appeared to be drifting upwards you witnessed the motion after effect. If you have a slow computer, a 486 machine or older, this demo may not work very well and you won't experience the MAE.

Lecture 9.

Cognitive Process - Part I

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

Understand attention

Describe memory models

To day is first lecture on cognitive processes out series of two lectures on the same topic. In our previous lectures we have in detail talked about the cognitive psychology and cognitive frame works. Now is this lecture and in next coming lectures we will talk about a detail about the cognitive processes. As we have already discussed that cognition can be described in terms of specific kinds of processes. These include:

Attention

Memory

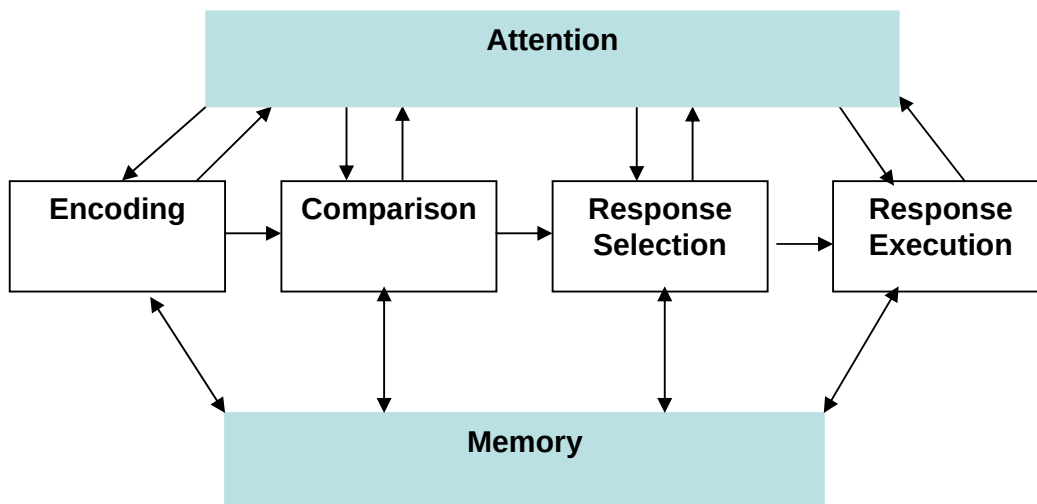
Perception and recognition

Learning

Reading, speaking and listening

Problem solving, planning, reasoning, decision-making.

Here in this lecture we will study the first two cognitive processes named attention and memory. The importance of these two you have seen in the Extended Human Processing model, studied in Lecture No. 6, as shown in figure



9.1 Attention

Attention is the process of selecting things to concentrate on, at a point in time, from the range of possibilities available.

A famous psychologist, Williams James says, “ Everyone knows what attention is. It is the taking possession of mind, in clear and vivid form, of one out of what seem several

simultaneously possible objects or trains of thought... it requires withdrawal from some things in order to deal effectively with others.”

Attention involves our auditory and/or visual senses an example of auditory attention is waiting in the dentist’s waiting room for our name to be called out to know when it is our time to go in. auditory attention is based on pitch, timber and intensity. An example of attention involving the visual senses in scanning the football results in a newspaper to attend to information about how our team has done. Visual attention is based on color and location.

Attention allows us to focus on information that is relevant to what we are doing. The extent to which this process is easy or difficult depends on

Whether we have clear goals and

Whether the information we need is salient in the environment.

Our goals

If we know exactly what we want to find out, we try to match this with the information that is available. For example, if we have just landed at an airport after a long flight and want to find out who has won the World Cup, we might scan the headlines at the newspaper stand, check the web, call a friend, or ask someone in the street.

When we are not sure exactly what we are looking for we may browse through information, allowing it to guide our attention to interesting or salient items. For example, when we go to restaurant we may have the general goal of eating a meal but only a vague idea of what we want to eat. We peruse the menu to find things that whet our appetite, letting our attention be drawn to the imaginative descriptions of various dishes. After scanning through the possibilities and imagining what each dish might be like (plus taking into account other factors, such as cost, who we are with, what the specials are, what the waiter recommends, whether we want a two-or- three-course meal, and so on.), we may then make a decision.

Information presentation

The way information is displayed can also greatly influence how easy or difficult it is to attend to appropriate pieces of information. Look at the figure below, two different ways of structuring the same information at the interface: one makes it much easier to find information than the other. Look at the top screen and (i) find the price for a double room at the Holiday Inn in Lahore; (ii) find the phone number of the Sheraton in the Karachi. Then look at the bottom screen and (i) find the price of for a double room at the Pearl Continental in Faisalabad; (ii) find the phone number of the Holiday Inn in the Islamabad. Which took longer to do? Experiments showed that the two screens produced quite different results: it took an average of 3.2 seconds to search the top screen and 5.5 seconds to find the same kind of information in the bottom screen. Why is this so, considering that the both displays have the same density of information? The primary reason is the way the characters are grouped in the display; in the top they are grouped into vertical categories of information that have columns of space between them. In the bottom screen the information is bunched up together, making it much harder to search through.

City	Hotel	Area Code	Phone	Rates	
				Single	Double
Lahore	Holiday Inn	042	6300634	2000	4000
Lahore	Sheraton	042	5456322	1800	3500
Lahore	Perl Continental	042	4565654	2500	5000
Marachi	Holiday Inn	021	1645656	2000	4000
Marachi	Sheraton	021	6545646	1800	3500
Marachi	Perl Continental	021	9545656	2500	5000

```

Islamabad: Holiday Inn
(051) 4544556 $:2000 D:4000
Islamabad: Sheraton
(051) 2135466 $:1500 D:3500
Islamabad: Pearl Continental
(051) 6565322 $:3000 D:6000
Faisalabad: Holiday Inn
(041) 4544556 $:2000 D:4000
Faisalabad: Sheraton
(041) 2135466 $:1500 D:3500
Faisalabad: Pearl Continental
(041) 6565322 $:3000 D:6000

```

Models of Attention

There are two models of attention:

- Focused attention
- Divided attention

Focused Attention

Our ability to attend to one event from what amounts to a mass of competing stimuli in the environment have been psychologically termed as focused attention. The streams of information we choose to attend to will tend to be relevant to the activities and intentions that we have at that time. For example, when engaged in a conversation it is usual to attend to what the other person is saying. If something catches our eye in the periphery to our vision, for example, another person we want to talk to suddenly appear, we may divert our attention to what she is doing. We may then get distracted from the conversation we are having and as a consequence have to ask the person we are conversing with to repeat them. On the other hand, we may be skilled at carrying on the conversation while intermittently observing what the person we want to talk to is doing.

Divided Attention

As we said, we may be skilled at carrying on the conversation while intermittently observing what the person we want to talk to is doing. When we attempt to attend to more than one thing at a time, as in the above example, it is called divided attention. Another example that is often used to illustrate this intentional phenomenon is being able to drive while holding a conversation with a passenger.

Voluntary attention

A further property of attention is that can be voluntary, as when we make a conscious effort to change our attention.

Involuntary attention

Attention may also be involuntary, as when the salient characteristics of the competing stimuli grab our attention. An everyday example of an involuntary act is being distracted from working when we can hear music or voices in the next room.

Another thing is that frequent actions become automatic actions, that is, they do not need any conscious attention and they require no conscious decisions.

Focusing attention at the interface

What is the significance of attention for HCI? How can an understanding of intentional phenomena be usefully applied to interface design? Clearly, the manner in which we deploy our attention has a tremendous bearing on how effectively we can interact with a system. If we know that people are distracted, often involuntarily, how is it possible to get their attention again without allowing them to miss the ‘window of opportunity’? Moreover, how can we focus people’s attention on what they need to be looking at or listening to for any given stage of task? How can we guide their attention to the relevant information on display?

Structuring Information

One way in which interfaces can be designed to help users find the information they need is to structure the interface so that it is easy to navigate through. Firstly. This requires presenting not too much information and not too little on a screen, as in both cases the user will have to spend considerable time scanning through either a cluttered screen or numerous screens of information. Secondly, instead of arbitrarily presenting data on the screen it should be grouped and ordered into meaningful parts capitalizing on the perceptual laws of grouping, information can be meaningfully structured so that it is easier to perceive and able to guide attention readily to the appropriate information.

Help user to.

Attend his/her task not the interface.

Decide what to focus on, based on their tasks, interest, etc.

Stay focused, do not provide unnecessary distractions.

Structure his/her task, e.g. help

Create distraction, when really necessary!

Use alerts (only) when appropriate!

Some other considerations are as under:

Make information salient when it needs attending to

Use techniques that make things stand out like colour, ordering, spacing, underlining, sequencing and animation

Avoid cluttering the interface - follow the google.com example of crisp, simple design

Avoid using too much because the software allows it

9.2 Memory

Indeed, much of our everyday activities rely on memory. As well as storing all our factual knowledge, our memory contains our knowledge of actions or procedures. It allows us to repeat actions, to use language, and to use new information received via our senses. It also gives us our sense of identity, by preserving information from our past experiences. If want to understand the working of our memory, it is necessary to understand the structure of memory. Let us look at a memory model.

Memory Model

It is generally agreed that there are three types of memory or memory functions sensory buffers, short-term memory or working memory and long-term memory. It is also called the multi-store model of memory. The main characteristics of the multi-store model of memory are the various types of memory stores. These are:

Sensory store modality-specific, hold information for a very brief period of time (a few tenth of a second),

Short-term memory store holds limited information for a short period of time (a few seconds),

Permanent long-term memory store hold information indefinitely.
Let's us have a detailed look of this model.

Sensory memory

The sensory memories act as buffer for stimuli received through the senses. A sensory memory exists for each sensory channel: iconic memory for visual stimuli, echoic memory for aural stimuli and haptic memory for touch. These memories are constantly overwritten by new information coming in on these channels.

We can demonstrate the existence of iconic memory by moving a finger in front of the eye. Can you see it in more than one place at once? This indicates a persistence of the image after the stimulus has been removed. A similar effect is noticed most vividly at firework displays where moving sparklers leave a persistent image. Information remains in iconic memory very briefly, in the order of 0.5 seconds.

Similarly, the existence of echoic memory is evidenced by our ability to ascertain the direction from which a sound originates. This is due to information being received by both ears. However, since this information is received at different times, we must store the stimulus in the meantime. Echoic memory allows brief playback of information. Have you ever had someone ask you a question when you are reading? You ask them to repeat the question only to realise that you know what was asked after all. This experience, too, is evidence of the existence of echoic memory. Information is passed from sensory memory into short-term memory by attention, thereby filtering the stimuli to only those, which are of interest at a given time. Attention is the concentration of the mind on one out of a number of competing stimuli or thoughts. It is clear that we are able to focus our attention selectively, choosing to attend to one thing rather than another. This is due to the limited capacity of our sensory and mental processes. If we did not selectively attend to the stimuli coming into our senses, we would be overloaded. We can choose which stimuli to attend to, and the choice is governed to an extent by our arousal, our level of interest or need. This explains the cocktail party phenomenon. According to cocktail party effect we can attend to one conversation over the background noise, but we may choose to switch our attention to a conversation across the room if we hear our name mentioned. Information received by sensory memories is quickly passed into a more permanent memory store, or overwritten and lost.

Short term memory

Short-term memory or working memory acts as a scratch pad for temporary recall of information. It is used to store information which is only required fleetingly. For example, calculate the multiplication $35 * 6$ in your head. The chances are that you will have done this calculation in stages, perhaps $5 * 6$ and then $3 * 6$ and added the results. To perform calculations such as this we need to store the intermediate stages for use later. Or consider reading. In order to comprehend this sentence you need to hold in your mind the beginning of the sentence as you read the rest. Both of these tasks use short-term memory.

Short-term memory can be accessed rapidly, in the order of 70ms. However, it also decays rapidly, meaning that information can only be held there temporarily, in the order of 200ms.

Short-term memory also has a limited capacity. There are two basic methods for measuring memory capacity. The first involves determining the length of a sequence, which can be remembered in order. The second allows items to be freely recalled in any order. Using the first measure, the average person can remember 7 ± 2 digits. This was established in experiments by Miller. Try it look at the following number sequence:

54988319814237

Now write down as much of the sequence as you can remember. Did you get it all right? If not, how many digits could you remember? If remembered between five and nine digits your digit span is average.

Now try the following sequence:

22 55 36 8998 30

Did you recall that more easily? Here the digits are grouped or chunked. A generalization of the 7 ± 2 rule is that we can remember 7 ± 2 chunks of information. Therefore chunking information can increase the short-term memory capacity. The limited capacity of short-term memory produces a subconscious desire to create chunks, and so optimise the use of the memory. The successful formation of a chunk is known as closure. This process can be generalized to account for the desire to complete or close tasks held in short-term memory. If a subject fails to do this or is prevented from doing so by interference, the subject is liable to lose track of what she is doing and make consequent errors.

Recency effect

In experiments where subjects were able to recall words freely, evidence shows that recall of the last words presented is better than recall of those in the middle. This is known as recency effect. Recency effect can be defined as: 'better recall for items at the end of the list because these items are still active in STM (and possibly SM) at time of recall'.

However, if the subject is asked to perform another task between presentation and recall the recency effect is eliminated. The recall of other words is unaffected. This suggests that short-term memory recall is damaged by interference of other information.

Primacy effect

'Better recall for items at the beginning of the list (because these items have been rehearsed more frequently than other items and thus have a greater chance of being placed in LTM).'

Long Term Memory

If short-term memory is our working memory or 'scratch-pad', long-term memory is our main resource. Here we store factual information, experiential knowledge, and procedural

rules of behavior- in fact, everything that we know. It differs from short-term memory in a number of significant ways. First, it has a huge, if not unlimited, capacity. Secondly, it has a relatively slow access time of approximately a tenth of a second. Thirdly, forgetting occurs more slowly in long-term memory, if at all.

Long-term memory is intended for the long-term storage of information. Information is placed there from working memory through rehearsal. Unlike working memory there is little decay: long-term recall after minutes is the same as that after hours or days.

Long-term memory structure

There are two types of long-term memory: episodic memory and semantic memory.

Episodic memory

Episodic memory represents our memory of events and experiences in a serial form. It is from this memory that we can reconstruct the actual events that took place at a given period of our lives.

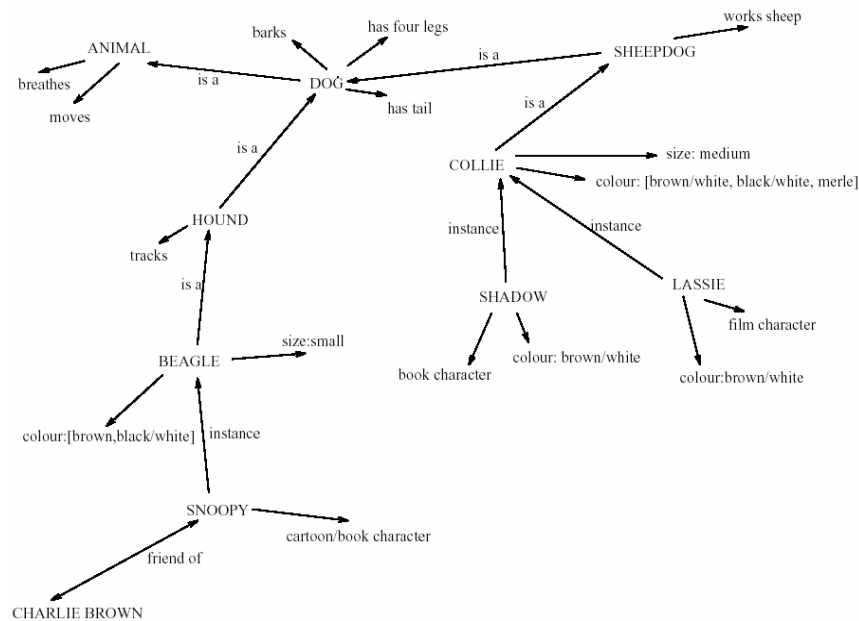
Semantic memory

Semantic memory is structured record of facts, concepts and skills that we have acquired. The information in semantic memory is derived from that in our episodic memory, such that we can learn new facts or concepts from our experience.

Semantic memory is structured in some way to allow access to information, representation of relationships between pieces of information, and inference. One model for the way in which semantic memory is structured is as a network. Items are associated to each other in classes, and may inherit attributes from parent classes. This model is known as a semantic network. As an example, knowledge about dogs may be stored in a network such as that shown in figure.

Specific breed attributes may be stored with each given breed; yet general dog information is stored at a higher level. This allows us to generalize about specific cases. For instance, we may not have been told that the sheepdog Shadow has four legs and a tail, but we can infer this information from our general knowledge about sheepdogs and dogs in general. Note also that there are connections within the network which link into other domains of knowledge, for example cartoon characters

A number of other memory structures have been proposed to explain how we represent and store different types of knowledge. Each of these represents a different aspect of knowledge and, as such, the models can be viewed as complementary rather than mutually exclusive. Semantic networks represent the associations and relationship between single items in memory. However, they do not allow us to model the representation of more complex objects or events, which are perhaps composed of a number of items or activities. Structured representations such as frames and scripts organize information into data structures. Slots in these structures allow attribute values to be added. Frame slots may contain default, fixed or variable information. A frame is instantiated when the slots are filled with appropriate values. Frame and scripts can be linked together in networks to represent hierarchical structured knowledge.

**DOG**

Fixed
legs: 4

Default
diet: carniverous
sound: bark

Variable
size:

COLLIE

Fixed
breed of: DOG
type: sheepdog

Default
size: 65 cm

Variable
colour

Script for a visit to the vet

Entry conditions: dog ill
vet open
owner has money

Result: dog better
owner poorer
vet richer

Props: examination table
medicine
instruments

Roles: vet examines
diagnoses
treats
owner brings dog in
pays
takes dog out

Scenes: arriving at reception
waiting in room
examination
paying

Tracks: dog needs medicine
dog needs operation

9.3 Revised Memory Model

According to revised memory model working memory is a subset of LTM.

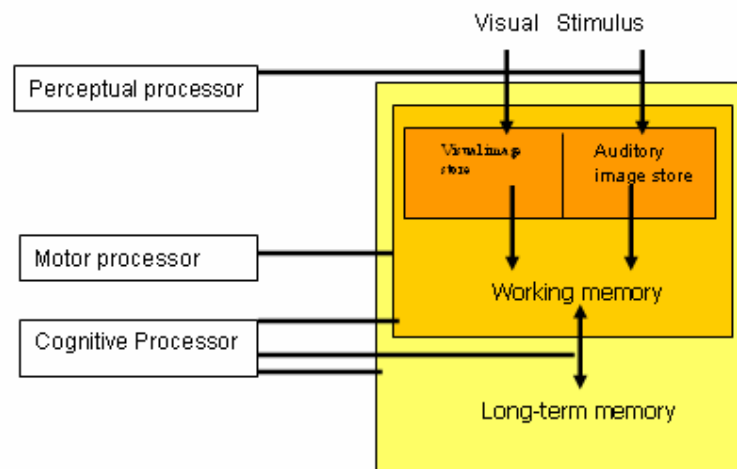
Some other characteristics are as under:

Items are semantically linked.

Items in working memory are activated.

Activation is supplied from other linked chunks and from sensory input.

As we know human processor consists of three interacting systems: the perceptual system, the motor system and the cognitive system. Each has its own memory and process as shown in figure. Similar to the notions of human information processing, human performance is viewed as a series of processing stages, whereby the different processors and memories are organized in a particular way.



Lecture 10.

Cognitive Processes - Part II

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand learning
- Discuss planning, reasoning, decision making
- Understand problem solving

Today is part second of our two parts series lecture on Cognitive Process. As we have earlier seen that cognition involves following processes:

- Attention
- Memory
- Perception and recognition
- Learning
- Reading, speaking and listening
- Problem solving, planning, reasoning, decision-making.

Today we will learn about learning and thinking. Let us first look at learning.

10.1 Learning

Learning can be considered in two terms:

- Procedural
- Declarative

Procedural

According to procedural learning we come to any object with questions like how to use it? How to do something? For example, how to use a computer-based application?

Declarative

According to declarative learning we try to find the facts about something. For example, using a computer-based application to understand a given topic.

Jack Carroll and his colleagues have written extensively about how to design interfaces to help learners develop computer-based skills. A main observation is that people find it very hard to learn by following sets of instructions in a manual. For example, when people encounter a computer for the first time their most common reaction is one of fear

and trepidation. In contrast, when we sit behind the steering wheel of a car for the first time most of us are highly motivated and very excited with the prospect of learning to drive. Why, then, is there such a discrepancy between our attitudes to learning these different skills? One of the main differences between the two domains is the way they are taught. At the end of the first driving lesson, a pupil will have usually learned how to drive through actually doing. This includes performing a number of complex tasks such as clutch control, gear changing, learning to use the controls and knowing what they are. Furthermore, the instructors are keen to let their pupils try things out and get started. Verbal instruction initially is kept to a minimum and usually interjected only when necessary. In contrast, someone who sits in front of a computer system for the first time may only have a very large manual, which may be difficult to understand and poorly presented. Often training and reference materials are written as a series of ordered explanations together with step by step exercises, which may cause the learner to feel overloaded with information or frustrated at not being able to find information that she wants. One of the main developing usable training materials and helps facilities. There is a general assumption that having read something in the manual users can immediately match it to what is happening at the interface and respond accordingly. But as you may have experienced, trying to put into action even simple descriptions can sometimes be difficult.

Experienced users also appear to be reluctant to learn new methods and operations from manuals. When new situations arise that could be handled more effectively by new procedures, experienced users are more likely to continue to use the procedures they already know rather than try to follow the advanced procedures outlined in a manual, even if the former course takes much longer and is less effective.

So, people prefer to learn through doing. GUI and direct manipulation interface are good environments for supporting this kind of learning by supporting exploratory interaction and importantly allowing users to 'undo' their actions, i.e., return to a previous state if they make a mistake by clicking on the wrong option.

Carroll has also suggested that another way of helping learners is by using a 'training wheels' approach. This involves restricting the possible functions that can be carried out by a novice to the basics and then extending these as the novice becomes more experienced. The underlying rationale is to make initial learning more tractable, helping the learner focus on simple operations before moving on to more complex ones.

There have also been numerous attempts to harness the capabilities of different technologies, such as web-based, multimedia, and virtual reality, is that they provide alternative ways of representing and interacting with information that are not possible with traditional technologies. In so doing, they have the potential of offering learners the ability to explore ideas and concepts different ways.

People often have problems learning the difficult stuff---by this we mean mathematical formulae, notations, laws of physics, and other abstract concepts. One of the main reasons is that they find it difficult to relate their concrete experiences of the physical world with these higher-level abstractions. Research has shown, however, that it is possible to facilitate this kind of learning through the use of interactive multimedia.

Dynalinking

The process of linking and manipulating multimedia representations at the interface is called dynalinking. It is helpful in learning. An example where dynalinking have been found beneficial is in helping children and students learn ecological concepts. During experiment a simple ecosystem of a pond was built using multimedia. The concrete simulation showed various organisms swimming and moving around and occasionally an event where one would eat another. When an organism was clicked on, it would say what it was and what it ate.

The simulation was dynalinked with other abstract representations of the pond ecosystem. One of these was a food web diagram. People were encouraged to interact with the interlinked diagrams in various ways and to observe what happened in the concrete simulation when something was changed in the diagram and vice versa.

Dynalinking is a powerful form of interaction and can be used in a range of domains to explicitly show relationships among multiple dimensions, especially when the information to be understood or learned is complex.

10.2 Reading, Speaking and Listening

These three forms of language processing have both similar and different properties. One similarity is that the meaning of sentences or phrases is the same regardless of the mode in which it is conveyed. For example, the sentence “Computer are a wonderful invention” essentially has the same meaning whether one reads it, speaks it, or hears it. However, the ease with which people can read, listen, or speak differs depending on the person, task, and context. For example, many people find listening much easier than reading. Specific differences between the three modes include:

- Written language is permanent while listening is transient. It is possible to reread information if not understood the first time round. This is not possible with spoken information that is being broadcast.
- Reading can be quicker than speaking or listening, as written text can be rapidly scanned in ways not possible when listening to serially presented spoken works.
- Listening require less cognitive effort than reading or speaking. Children, especially, often prefer to listen to narratives provided in multimedia or web-based learning material than to read the equivalent text online.
- Written language tends to be grammatical while spoken language is often ungrammatical. For example, people often start and stop in mid-sentence, letting someone also start speaking.
- There are marked differences between people in their ability to use language. Some people prefer reading to listening, while others prefer listening. Likewise, some people prefer speaking to writing and vice versa.
- Dyslexics have difficulties understanding and recognizing written words, making it hard for them to write grammatical sentences and spell correctly.
- People who are hard of hearing or hart of seeing are also restricted in the way they can process language.

Incorporating Language processing in applications

Many applications have been developed either to capitalize on people's reading writing and listening skills, or to support or replace them where they lack or have difficulty with them. These include:

- Interactive books and web-based material that help people to read or learning foreign languages.
- Speech-recognition systems that allow users to provide instructions via spoken commands.
- Speech-output systems that use artificially generated speech
- Natural-language systems that enable users to type in questions and give text-based responses.
- Cognitive aids that help people who find it difficult to read, write, and speak. A number of special interfaces have been developed for people who have problems with reading, writing, and speaking.
- Various input and output devices that allow people with various disabilities to have access to the web and use word processors and other software packages.

Design Implications

- Keep the length of speech-based menus and instructions to a minimum. Research has shown that people find it hard to follow spoken menu with more than three or four options. Likewise, they are bad at remembering sets of instructions and directions that have more than a few parts.
- Accentuate the intonation of artificially generated speech voices, as they are harder to understand than human voices.
- Provide opportunities for making text large on a screen, without affecting the formatting, for people who find it hard to read small text.

10.3 Problem Solving, Planning, Reasoning and Decision-making

Problem solving, planning, reasoning and decision-making are all cognitive processes involving reflective cognition. They include thinking about what to do, what the options are, and what the consequences might be of carrying out a given action. They often involve conscious processing (being aware of what one is thinking about), discussion with others, and the use of various kinds of artifacts, (e.g., maps, books, and pen and paper). For example, when planning the best route to get somewhere, say a foreign city, we may ask others use a map, get instructions from the web, or a combination of these.

Reasoning also involves working through different scenarios and deciding which is the best option or solution to a given problem. In the route-planning activity we may be aware of alternative routes and reason through the advantages and disadvantages of each route before deciding on the best one. Many family arguments have come about because one member thinks he or she knows the best route while another thinks otherwise.

Comparing different sources of information is also common practice when seeking information on the web. For example, just as people will phone around for a range of

quotes, so too, will they use different search engines to find sites that give the best deal or best information. If people have knowledge of the pros and cons of different search engines, they may also select different ones for different kinds of queries. For example, a student may use a more academically oriented one when looking for information for writing an essay, and a more commercially based one when trying to find out what's happening in town.

The extent to which people engage in the various forms of reflective cognition depends on their level of experience with a domain; applications about what to do using other knowledge about similar situations. They tend to act by trial and error, exploring and experimenting with ways of doing things. As a result they may start off being slow, making errors and generally being inefficient. They may also act irrationally, following their superstitions and not thinking ahead to the consequences of their actions. In contrast experts have much more knowledge and experience and are able to select optimal strategies for carrying out their tasks. They are likely to be able to think ahead more, considering what the consequences might be of opting for a particular move or solution.

Reasoning

Reasoning is the process by which we use the knowledge we have to draw conclusions or infer something new about the domain of interest. There are a number of different types of reasoning:

- Deductive reasoning
- Inductive reasoning
- Abductive reasoning

Deductive reasoning

Deductive reasoning derives the logically necessary conclusion from the given premises. For example,

It is Friday then she will go to work

It is Friday

Therefore she will go to work

It is important to note that this is the logical conclusion from the premises; it does not necessarily have to correspond to our notion of truth. So, for example,

If it is raining then the ground is dry

It is raining

Therefore the ground is dry.

Is a perfectly valid deduction, even though it conflicts with our knowledge of what is true in the world?

Inductive reasoning

Induction is generalizing from cases we have seen to infer information about cases we have not seen. For example, if every elephant we have ever seen has a trunk, we infer that all elephants have trunks. Of course, this inference is unreliable and cannot be proved to be true; it can only be proved to be false. We can disprove the inference simply by producing an elephant without a trunk. However, we can never prove it true because, no matter how many elephants with trunks we have seen or are known to exist, the next one

we see may be trunkless. The best that we can do is gather evidence to support our inductive inference.

In spite of its unreliability, induction is a useful process, which we use constantly in learning about our environment. We can never see all the elephants that have ever lived or will ever live, but we have certain knowledge about elephants, which we are prepared to trust for all practical purposes, which has largely been inferred by induction. Even if we saw an elephant without a trunk, we would be unlikely to move from our position that 'All elephants have trunk', since we are better at using positive than negative evidence.

Abdicative reasoning

The third type of reasoning is abduction. Abduction reasons from a fact to the action or state that caused it. This is the method we use to derive explanations for the events we observe. For example, suppose we know that Sam always drives too fast when she has been drinking. If we see Sam driving too fast we may infer that she has been drinking. Of course, this too is unreliable since there may be another reason why she is driving fast: she may have been called to an emergency, for example.

In spite of its unreliability, it is clear that people do infer explanations in this way and hold onto them until they have evidence to support an alternative theory or explanation. This can lead to problems in using interactive systems. If an event always follows an action, the user will infer that the event is caused by the action unless evidence to the contrary is made available. If, in fact, the event and the action are unrelated, confusion and even error often result.

Problem solving

If reasoning is a means of inferring new information from what is already known, problem solving is the process of finding a solution to an unfamiliar task, using the knowledge we have. Human problem solving is characterized by the ability to adapt the information we have to deal with new situations. However, often solutions seem to be original and creative. There are a number of different views of how people solve problems. The earliest, dating back to the first half of the twentieth century is the Gestalt view that problem solving involves both reuse of knowledge and insight. This has been largely superseded but the questions it was trying to address remain and its influence can

be seen in later research. A second major theory, proposed in the 1970s by Newell and Simon, was the problem space theory, which takes the view that the mind is a limited information processor. Later variations on this drew on the earlier theory and attempted to reinterpret Gestalt theory in terms of information-processing theories. Let us look at these theories.

Gestalt theory

Gestalt psychologists were answering the claim, made by behaviorists, that problem solving is a matter of reproducing known responses or trial and error. This explanation was considered by the Gestalt school to be insufficient to account for human problem-solving behavior. Instead, they claimed, problem solving is both productive and reproductive. Reproductive problem solving draws on previous experience as the behaviorist claimed, but productive problem solving involves insight and restructuring of the problem. Indeed, reproductive problem solving could be hindrance to finding a

solution, since a person may 'fixate' on the known aspects of the problem and so be unable to see novel interpretations that might lead to a solution.

Although Gestalt theory is attractive in terms of its description of human problem solving, it does not provide sufficient evidence or structure to support its theories. It does not explain when restructuring occurs or what insight is, for example.

Problem space theory

Newell and Simon proposed that problem solving centers on the problem space. The problem space comprises problem states, and problem solving involves generating these states using legal state transition operators. The problem has an initial state and a goal state and people use the operator to move from the former to the latter. Such problem spaces may be huge, and so heuristics are employed to select appropriate operators to reach the goal. One such heuristic is means-ends analysis. In means-ends analysis the initial state is compared with the goal state and an operator chosen to reduce the difference between the two. For example, imagine you are recognizing your office and you want to move your desk from the north wall of the room to the window. Your initial state is that the desk is at the north wall. The goal state is that the desk is by the window. The main difference between these two is the location of your desk. You have a number of operators, which you can apply to moving things: you can carry them or push them or drag them, etc. however, you know that to carry something it must be light and that your desk is heavy. You therefore have a new sub-goal: to make the desk light. Your operators for this may involve removing drawers, and so on.

An important feature of Newell and Simon's model is that it operates within the constraints of the human processing system, and so searching the problem space is limited by capacity of short-term memory, and the speed at which information can be retrieved. Within the problem space framework, experience allows us to solve problems more easily since we can structure the problem space appropriately and choose operators efficiently.

Analogy in problem solving

A third element of problem solving is the use of analogy. Here we are interested in how people solve novel problems. One suggestion is that this is done by mapping knowledge relating to a similar known domain to the new problem-called analogical mapping. Similarities between the known domain and the new one are noted and operators from the known domain are transferred to the new one.

This process has been investigated using analogous stories. Gick and Holyoak gave subjects the following problem:

A doctor is treating a malignant tumor. In order to destroy it he needs to blast it with high-intensity rays. However, these will also destroy the healthy tissue, surrounding tumor. If he lessens the ray's intensity the tumor will remain. How does he destroy the tumor?

The solution to this problem is to fire low-intensity rays from different directions converging on the tumor. That way, the healthy tissue receives harmless low-intensity rays while the tumor receives the rays combined, making a high- intensity does. The investigators found that only 10% of subjects reached this solution without help.

However, this rose to 80% when they were given this analogous story and told that it may help them:

A general is attacking a fortress. He can't send all his men in together as the roads are mined to explode if large numbers of men cross them. He therefore splits his men into small groups and sends them in on separate roads. In spite of this, it seems that people often miss analogous information, unless it is semantically close to the problem domain.

Skill acquisition

All of the problem solving that we have considered so far has concentrated on handling unfamiliar problems. However, for much of the time, the problems that we face are not completely new. Instead, we gradually acquire skill in a particular domain area. But how is such skill acquired and what difference does it make to our problem-solving performance? We can gain insight into how skilled behavior works, and how skills are acquired, by considering the difference between novice and expert behavior in given domains.

A commonly studied domain is chess playing. It is particularly suitable since it lends itself easily to representation in terms of problem space theory. The initial state is the opening board position; the goal state is one player checkmating the other; operators to move states are legal moves of chess. It is therefore possible to examine skilled behavior within the context of the problem space theory of problem solving.

In all experiments the behavior of chess masters was compared with less experienced chess players. The first observation was that players did not consider large number of moves in choosing their move, nor did they look ahead more than six moves. Masters considered no more alternatives than the less experienced, but they took less time to make decision and produced better moves.

It appears that chess masters remember board configurations and good moves associated with them. When given actual board positions to remember, masters are much better at reconstructing the board than the less experienced. However, when given random configurations, the groups of players were equally bad at reconstructing the positions. It seems therefore that expert players 'chunk' the board configuration in order to hold it in short-term memory. Expert players use larger chunks than the less experienced and can therefore remember more detail.

Another observed difference between skilled and less skilled problem solving is in the way that different problems are grouped. Novices tend to group problems according to superficial characteristics such as the objects or features common to both. Experts, on the other hand, demonstrate a deeper understanding of the problems and group them according to underlying conceptual similarities, which may not be at all obvious from the problem descriptions.

Lecture 11.

The Psychology of Actions

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand mental models
- Understand psychology of actions
- Discuss errors.

11.1 Mental model

The concept of mental model has manifested itself in psychology theorizing and HCI research in a multitude of ways. It is difficult to provide a definitive description, because different assumption and constraints are brought to bear on the different phenomena it has been used to explain. A well-known definition, in the context of HCI, is provided by Donald Norman: ‘the model people have of themselves, others, the environment, and the things with which they interact. People form mental models through experience, training and instruction’.

It should be noted that in fact the term mental model was first developed in the early 1640s by Kenneth Craik. He proposed that thinking ‘...models, or parallels reality’: ‘If the organism carries a “small-scale model” of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to emergencies which face it.’

Just as an engineer will build scale models of a bridge, in order to test out certain stresses prior to building the real thing, so, too, do we build mental models of the world in order to make predictions about an external event before carrying out an action? Although our construction and use of mental models may not be as extensive or as complete as Craik’s hypothesis suggests, it is likely that most of us can probably recall using a form of mental simulation at some time or other. An important observation of these types of mental models is that they are invariably incomplete, unstable, and easily confusable and are often based on superstition rather than scientific fact.

Within cognitive psychology the term mental model has since been explicated by Johnson-Laird (1983, 1988) with respect to its structure and function in human reasoning and language understanding. In terms of structure of mental models, he argues that mental models are either analogical representations or a combination of analogical and prepositional representations. They are distinct from, but related to images. A mental model represents the relative position of a set of objects in an analogical manner that

parallels the structure of the state of objects in the world. An image also does this, but more specifically in terms of view of a particular model.

An important difference between images and mental models is in terms of their function. Mental models are usually constructed when we are required to make an inference or a prediction about a particular state of affairs. In constructing the mental model a conscious mental simulation may be 'run' from which conclusions about the predicted state of affairs can be deduced. An image, on the other hand, is considered to be a one-off representation. A simplified analogy is to consider an image to be like a frame in a movie while a mental model is more like a short snippet of a movie.

So, after this discussion we can say that while learning and using a system, people develop knowledge of how to use the system and, to a lesser extent, how the system works. These two kinds of knowledge are often referred to as a user's mental model.

Having developed a mental model of an interactive product, it is assumed that people will use it to make inferences about how to carry out tasks when using the interactive product. Mental models are also used to fathom what to do when something unexpected happens with a system and when encountering unfamiliar systems. The more someone learns about a system and how it functions, the more their mental model develops. For example, TV engineers have a deep mental model of how TVs work that allows them to work out how to fix them. In contrast, an average citizen is likely to have a reasonably good mental model of how to operate a TV but a shallow mental model of how it worked.

To illustrate how we use mental models in our everyday reasoning, imagine the following scenario:

You arrive home from a holiday on a cold winter's night to a cold house. You have small baby and you need to get the house warm as quickly as possible. Your house is centrally heated. Do you set the thermostat as high as possible or turn it to the desired temperature (e.g., 70F)

Most people when asked the questions imagine the scenario in terms of what they would do in their own house they choose the first option. When asked why, a typical explanation that is given is that setting the temperature to be as high as possible increases the rate at which the room warms up. While many people may believe this, it is incorrect.

There are two commonly held folk theories about thermostats: the timer theory and the valve theory. The timer theory proposes that the thermostat simply controls the relative proportion of time that the device stays on. Set the thermostat midway, and the device is on about half the time; set it all the way up and the device is on all the time; hence, to heat or cool something most quickly, set the thermostat so that the device is on all the time. The valve theory proposes that the thermostat controls how much heat comes out of the device. Turn the thermostat all the way up, and you get maximum heating or cooling.

Thermostats work by switching on the heat and keeping it going at a constant speed until the desired temperature set is reached, at which point they cut out. They cannot control the rate at which heat is given out from a heating system. Left a given setting, thermostats will turn the heat on an off as necessary to maintain the desired temperature. It treats the

heater, oven, and air conditioner as all-or-nothing devices that can be either fully on or fully off, with no in-between states. The thermostat turns the heater, oven, or air conditioner completely on—at full power—until the temperature setting on the thermostat is reached. Then it turns the unit completely off. Setting the thermostat at one extreme cannot affect how long it takes to reach the desired temperature.

The real point of the example is not that some people have erroneous theories; it is that everyone forms theories (mental models) to explain what they have observed. In the case of the thermostat the design gives absolutely no hint as to the correct answer. In the absence of external information, people are free to let their imaginations run free as long as the mental models they develop account for the facts as they perceive them.

Why do people use erroneous mental models?

It seems that in the above scenario, they are running a mental model based on general valve theory of the way something works. This assumes the underlying principle of “more is more”: the more you turn or push something, the more it causes the desired effect. This principle holds for a range of physical devices, such as taps and radio controls, where the more you turn them, the more water or volume is given. However, it does not hold for thermostats, which instead function based on the principle of an on-off switch. What seems to happen is that in everyday life people develop a core set of abstractions about how things work, and apply these to a range of devices, irrespective of whether they are appropriate.

Using incorrect mental models to guide behavior is surprisingly common. Just watch people at a pedestrian crossing or waiting for an elevator (lift). How many times do they press the button? A lot of people will press it at least twice. When asked why, a common reason given is that they think it will make it lights change faster or ensure the elevator arrives. This seems to do another example of following the “more is more” philosophy: it is believed that the more times you press the button; the more likely it is to result in the desired effect.

Another common example of an erroneous mental model is what people do when the cursor freezes on their computer screen. Most people will bash away at all manner of keys in the vain hope that this will make it work again. However, ask them how this will help and their explanations are rather vague. The same is true when the TV starts acting up: a typical response is to hit the top of the box repeatedly with a bare hand or a rolled-up newspaper. Again, as people why and their reasoning about how this behavior will help solve the problem is rather lacking.

Indeed, research has shown that people’s mental models of the way interactive devices work is poor, often being incomplete, easily confusable, based on inappropriate analogies, and superstition. Not having appropriate mental models available to guide their behavior is what caused people to become very frustrated—often resulting in stereotypical “venting” behavior like those described above.

On the other hand, if people could develop better mental models of interactive systems, they would be in a better position to know how to carry out their tasks efficiently and what to do if the system started acting up. Ideally, they should be able to develop a mental model that matches the conceptual; modal developed by the designer. But how can you

help users to accomplish this? One suggestion is to educate them better, however, many people are resistant to spending much time learning about how things work, especially if it involves reading manuals and other documentation. An alternative proposal is to design systems to be more transparent, so that they are easier to understand.

People do tend to find causes for events, and just what they assign as the cause varies. In part people tend to assign a causal relation whenever two things occur in succession. If I do some action A just prior to some result R, then I conclude that A must have caused R, even if, there really was no relationship between the two.

Self-blaming

Suppose I try to use an everyday thing, but I can't: where is the fault, in my action or in the thing? We are apt to blame ourselves. If we believe that others are able to use the device and if we believe that it is not very complex, then we conclude that any difficulties must be our own fault. Suppose the fault really lies in the device, so that lots of people have the same problems. Because everyone perceives the fault to be his or own, nobody wants to admit to having trouble. This creates a conspiracy of silence, maintaining the feeling of guilt and helplessness among users.

Interestingly enough, the common tendency to blame ourselves for failures with everyday objects goes against the normal attributions people make. In general, it has been found that normal attribute their own problems to the environment, those of other people to their personalities.

It seems natural for people to blame their own misfortunes on the environment. It seems equally natural to blame other people's misfortunes on their personalities. Just the opposite attribution, by the way, is made when things go well. When things go right, people credit their own forceful personalities and intelligence. The onlookers do the reverse. When they see things go well for someone else, they credit the environment.

In all cases, whether a person is inappropriately accepting blame for the inability to work simple objects or attributing behavior to environment or personality, a faulty mental model is at work.

Reason for self-blaming

Learned helplessness

The phenomenon called learned helplessness might help explain the self-blame. It refers to the situation in which people experience failure at a task, often numerous times. As a result, they decide that the task cannot be done, at least not by them: they are helpless. They stop trying. If this feeling covers a group of tasks, the result can be severe difficulties coping with life. In the extreme case, such learned helplessness leads to depression and to a belief that the person cannot cope with everyday life at all. Some times all that it takes to get such a feeling of helplessness is a few experiences that accidentally turn out bad. The phenomenon has been most frequently studied as a precursor to the clinical problem of depression, but it might easily arise with a few bad experiences with everyday life.

Taught helplessness

Do the common technology and mathematics phobias results from a kind of learned helplessness? Could a few instances of failure in what appear to be straightforward situations generalize to every technological object, every mathematics problem? Perhaps. In fact, the design of everyday things seems almost guaranteed to cause this. We could call this phenomenon taught helplessness.

With badly designed objects—constructed so as to lead to misunderstanding—faulty mental models, and poor feedback, no wonder people feel guilty when they have trouble using objects, especially when they perceive that nobody else is having the same problems. The problem is that once failure starts, it soon generalizes by self-blame to all technology. The vicious cycle starts: if you fail at something, you think it is your fault. Therefore you think you can't do that task. As a result, next time you have to do the task, you believe you can't so you don't even try. The result is that you can't, just as you thought. You are trapped in a self-fulfilling prophecy.

The nature of human thought and explanation

It isn't always easy to tell just where the blame for problem should be placed. A number of dramatic accidents have come about, in part, from the false assessment of blame in a situation. Highly skilled, well-trained people are using complex equipment when suddenly something goes wrong. They have to figure out what the problem is. Most industrial equipment is pretty reliable. When the instruments indicate that something is wrong, one has to consider the possibility that the instruments themselves are wrong. Often this is the correct assessment. When operators mistakenly blame the instruments for an actual equipment failure, the situation is ripe for a major accident.

It is spectacularly easy to find examples of false assessment in industrial accidents. Analysts come in well after the fact, knowing what actually did happen; with hindsight, it is almost impossible to understand how the people involved could have made the mistake. But from the point of view of the person making decisions at time, the sequence of events is quite natural.

Three Mile Island Nuclear Power Plant

At the Three Mile Island nuclear power plant, operators pushed a button to close a valve; the valve had been opened (properly) to allow excess water to escape from the nuclear core. In fact, the valve was deficient, so it didn't close. But a light on the control panel indicated that the valve position was closed. The light actually didn't monitor the valve, only the electrical signal to the valve, a fact known by the operators. Still, why suspect a problem? The operators did look at the temperature in the pipe leading from the valve: it was high, indicating that fluid was still flowing through the closed valve. Ah, but the operators knew that the valve had been leaky, so the leak would explain the high temperature; but the leak was known to be small, and operators assumed that it wouldn't affect the main operation. They were wrong, and the water that was able to escape from the core added significantly to the problems of that nuclear disaster. Norman says that the operators' assessment was perfectly reasonable: the fault was in the design of the lights and in the equipment that gave false evidence of a closed valve.

Lockheed L-1011

Similarly many airline accidents happened just due to misinterpretations. Consider flight crew of the Lockheed L-1011 flying from Miami, Florida, to Nassau, Bahamas. The plane was over the Atlantic Ocean, about 110 miles from Miami, when the low oil pressure light for one of the three engines went on. The crew turned off the engine and turned around to go back to Miami. Eight minutes later, the low-pressure lights for the remaining two engines also went on, and the instruments showed zero oil pressure and quantity in all three engines. What did the crew do now? They didn't believe it! After all, the pilot correctly said later, the likelihood of simultaneous oil exhaustion in all three engines was "one in millions I would think." At the time, sitting in the airplane, simultaneous failure did seem most unlikely. Even the National Transportation Safety Board declared, "The analysis of the situation by the flight crew was logical, and was what most pilots probably would have done if confronted by the same situation."

What happened? The second and third engines were indeed out of oil, and they failed. So there were no operating engines: one had been turned off when its gauge registered low, the other two had failed. The pilots prepared the plane for an emergency landing on the water. The pilots were too busy to instruct the flight crew properly, so the passengers were not prepared. There was semi-hysteria in the passenger cabin. At the last minute, just as the plane was about to ditch in the ocean, the pilots managed to restart the first engine and land safely to Miami. Then that engine failed at the end of the runway.

Why did all three engine fail? Three missing O-rings, one missing from each of three oil plugs, allowed all the oil to seep out. The O-rings were put in by two different people who worked on the three engines (one for the two plugs on the wings, the other of the plug on the tail). How did both workers make the same mistake? Because the normal method by which they got the oil plugs had been changed that day. The whole tale is very instructive, for there were four major failures of different sorts, from the omission of the O-rings, to the inadequacy of the maintenance procedures, to the false assessment of the problem, to the poor handling of the passengers. Fortunately nobody was injured. The analysts of the National Transportation Safety Board got to write a fascinating report.

Find an explanation, and we are happy. But our explanations are based on analogy with past experience, experience that may not apply in the current situation. In the Three Mile Island incident, past experience with the leaky valve explained away the discrepant temperature reading; on the flight from Miami to Nassau, the pilots' lack of experience with simultaneous oil pressure failure triggered their belief that the instruments must be faulty. Once we have an explanation—correct or incorrect—for otherwise discrepant or puzzling events, there is no more puzzle, no more discrepancy. As a result, we are complacent, at least for a while.

How people do things

To get something done, you have to start with some notion of what is wanted—the goal that is to be achieved. Then, you have to do some thing to the world, that is, take action to move yourself or manipulate someone or something. Finally, you check to see that your goal was made. So there are four different things to consider: the goal, what is done to the world, the world itself, and the check of the world. The action itself has two major aspects: doing something and checking. Call these executions and evaluation. Goals do not state precisely what to do—where and how to move, what to pick up. To lead to

actions goals must be transformed into specific statements of what is to be done, statements that are called intentions. A goal is some thing to be achieved, often vaguely stated. An intention is specific action taken to get to the goal. Yet even intentions are not specific enough to control actions.

Suppose I am sitting in my armchair, reading a book. It is dust, and the light has gotten dimmer and dimmer. I decide to need more light (that is the goal: get more light). My goal has to be translated into the intention that states the appropriate action in the world: push the switch button on the lamp. There's more: I need to specify how to move my body, how to stretch to reach the light switch, how to extend my finger to push the button (without knocking over the lamp). The goal has to be translated into an intention, which in turn has to make into a specific action sequence, one that can control my muscles. Note that I could satisfy my goal with other action sequences, other intentions. If some one walked into the room and passed by the lamp, I might alter my intention from pushing the switch button to asking the other person to do it for me. The goal hasn't changed, but the intention and resulting action sequence have.

Action Cycle

Human action has two aspects, execution and evaluation. Execution involves doing something. Evaluation is the comparison of what happened in the world with what we wanted to happen

Stages of Execution

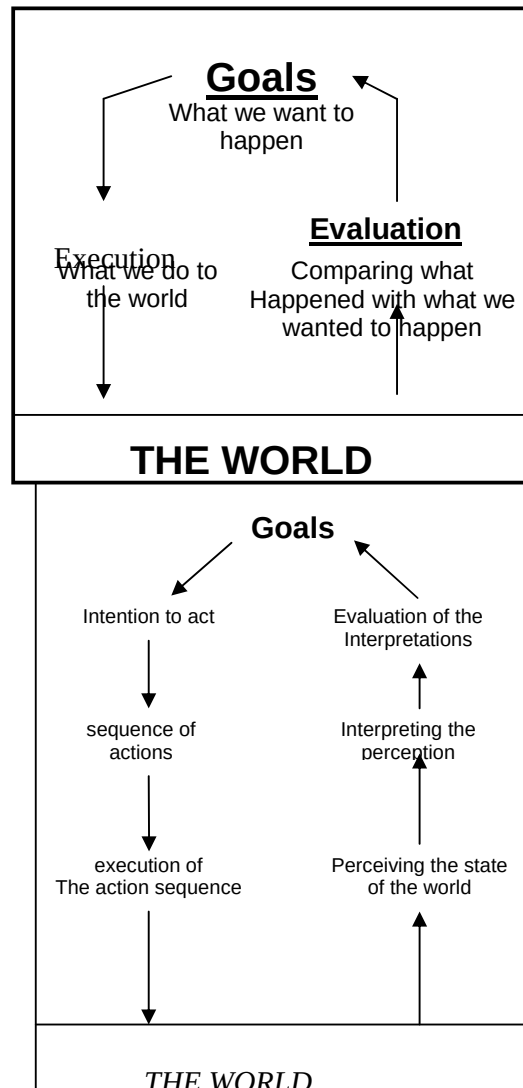
Start at the top with the goal, the state that is to be achieved. The goal is translated into an intention to do some action. The intention must be translated into a set of internal commands, an action sequence that can be performed to satisfy the intention. The action sequence is still a mental event: noting happens until it is executed, performed upon the world.

Stages of Evaluation

Evaluation starts with our perception of the world. This perception must then be interpreted according to our expectations and then compared with respect to both our intentions and our goals

Seven stages of action

The stages of execution (intentions, action sequence, and execution) are coupled with the stages of evaluation (perception, interpretation, and evaluation), with goals common to both stages.



11.2 Errors

Human capability for interpreting and manipulating information is quite impressive. However, we do make mistake. Whenever we try to learn a new skill, be it skiing, typing, cooking or playing chess, we are bound to make mistakes. Some are trivial, resulting in no more than temporary inconvenience or annoyance. Other may be more serious, requiring substantial effort to correct. In most situations it is not such a bad thing because the feedback from making errors can help us to learn and understand an activity. When learning to use a computer system, however, learners are often frightened of making errors because, as well as making them feel stupid; they think it can result in catastrophe. Hence, the anticipation of making an error and its consequences can hinder a user's interaction with a system.

Why do we make mistakes and can we avoid them? In order to answer the latter part of the question we must first look at what is going on when we make an error. There are several different types of errors. Some errors result from changes in the context of skilled behavior. If a pattern of behavior has become automatic and we change some aspect of it, the more familiar pattern may break through and cause an error. A familiar example of this is where we intend to stop at the shop on the way home from work but in fact drive past. Here, the activity of driving home is the more familiar and overrides the less familiar intention.

Other errors result from an incorrect understanding, or model, of a situation or system. People build their own theories to understand the casual behavior of systems. These have been termed mental models. They have a number of characteristics. Mental models are often partial: the person does not have a full understanding of the working of the whole system. They are unstable and are subject to change. They can be internally inconsistent, since the person may not have worked through the logical consequences of their beliefs. They are often unscientific and may be based on superstition rather than evidence. Often they are based on an incorrect interpretation of the evidence.

A classification of errors

There are various types of errors. Norman has categorized them into two main types, slips and mistakes:

Mistakes

Mistakes occur through conscious deliberation. An incorrect action is taken based on an incorrect decision. For example, trying to throw the icon of the hard disk into the wastebasket, in the desktop metaphor, as a way of removing all existing files from the disk is a mistake. A menu option to erase the disk is appropriate action.

Slips

Slips are unintentional. They happen by accident, such as making typos by pressing the wrong key or selecting wrong menu item by overshooting. The most frequent errors are slips, especially in well-learned behavior.

Lecture 12.

Design Principles

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand conceptual models
- Discuss design principles

Conceptual Model

“The most important thing to design is the user’s conceptual model. Every thing else should be subordinated to making that model clear, obvious, and substantial. That is almost exactly the opposite of how most software is designed.” (David Liddle)

By a conceptual model is meant:

A description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended.

To develop a conceptual model involves envisioning the proposed product, based on the user’s needs and other requirements identified. To ensure that it is designed to be understandable in the manner intended requires doing iterative testing of the product as it is developed.

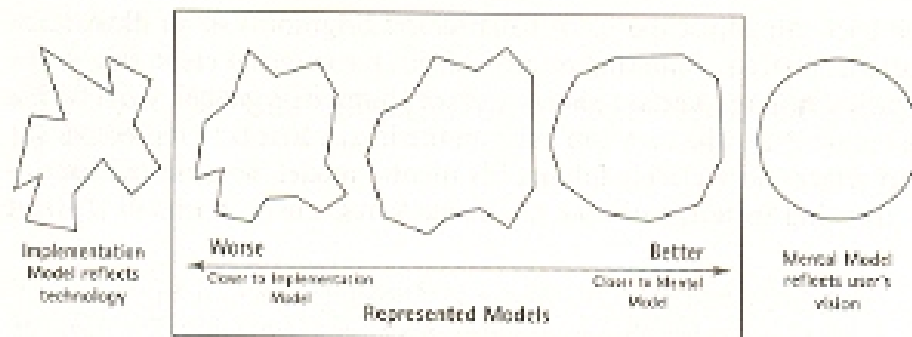
A key aspect of this design process is initially to decide what the user will be doing when carrying out their tasks. For example, will they be primarily searching for information, creating documents, communicating with other users, recording events, or some other activity? At this stage, the interaction mode that would best supports this need to be considered. For example, would allowing the users to browse be appropriate, or would allowing them to ask questions directly to the system in their native language be more affective? Decision about which kind of interaction style use (e.g., whether to use a menu-based system, speech inputs, commands) should be made in relation to the interaction mode. Thus, decision about which mode of interaction to support differ from those made about which style of interaction to have; the former being at a higher level of abstraction. The former are also concerned with determining the nature of the users’ activities to support, while the later are concerned with the selection of specific kinds of interface.

Once a set of possible ways of interacting with interactive system has been identified, the design of the conceptual modal then needs to be thought through in term of actual concrete solution. This entail working out the behavior of the inter face, the particular interaction style that will be used, and the “look and feel” of the interface. At this stage of

“fleshing out,” it is always a good idea to explore a number of possible designs and to assess the merits and problems of each one.

Another way of designing an appropriate conceptual model is to interface metaphor this can provide a basic structure for the conceptual model that is couched in knowledge users are familiar with. Examples of well-known interface metaphors are the desktop and search engines Software has a behavioral face it shows to the world that is created by the programmer or designer. This representation is not necessarily an accurate description of what is really going on inside the computer, although unfortunately, it frequently is. This ability to represent the computer functioning independent of its true actions is far more pronounced in software than in any other medium. It allows a clever designer to hide some of the more unsavory facts of how the software is really getting the job done. This disconnection between what is implemented and what it offered as explanation gives rise to a third model in the digital world, the designer’s represented model—the way the designer chooses to represent a program’s functioning to the user. Donald Norman refers to this simply as the designer’s model.

In the world of software, a program’s represented model can be quite different from the actual processing structure of the program. For example, an operating system can make a network file server look as though it were a local disk. The model does not represent the fact that the physical disk drive may be miles away. This concept of the represented model has no widespread counterpart in the mechanical world. The representation between the three models is shown in Figure.



The closer the represented model comes to the user’s mental model, the easier he will find the program to use and to understand. Generally, offering a represented model that follows the implementation model too closely significantly reduces the user’s ability to learn and use the program, assuming that the user’s mental model of his tasks differs from the implementation model of the software.

We tend to form mental models that are simpler than reality; so if we create represented models that are simpler than the actual implementation model, we help the user achieve a better understanding. Pressing the brake pedal in your car, for example, may conjure a mental image of pushing a lever that rubs against the wheels to slow you down. The actual mechanism includes hydraulic cylinders, tubing, and metal pads that squeeze on a perforated disk, but we simplify all that out of our minds, creating a more effective, albeit less accurate, mental model. In software, we imagine that a spreadsheet scrolls now cells into view when we click on the scrollbar. Nothing of the sort actually happens. There is

no sheet of cells out there, but a tightly packed data structure of values, with various pointers between them, from which the program synthesizes a new image to display in real-time.

Another important thing is that there are several gulfs that separate mental states from physical ones. Each gulf reflects one aspect of the distance between the mental representation of the person and the physical components and states of the environment. And these gulfs present major problems for users.

The Gulf of Execution

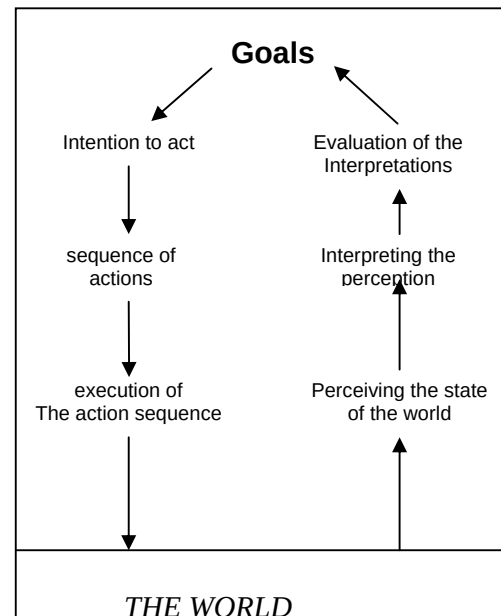
Does the system provide actions that correspond to the intentions of the person? The difference between the intentions and allowable actions is the gulf of execution. One measure of this gulf is how well the system allows the person to do the intended actions directly, without extra effort: do the action provided by the system match those intended by the person?

The Gulf of Evaluation

Does the system provide a physical representation that can be directly perceived and that is directly interpretable in terms of the intentions and expectations of the person? The Gulf of evaluation reflects the amount of effort that the person must exert to interpret the physical state of the system and to determine how well the expectations and intentions have been met. The gulf is small when the system provides information about its state in a form that is easy to get, is easy to interpret, and matches the way the person thinks of the system.

The Seven Stages of Action as Design aids

The seven-stage structure can be a valuable design aid, for it provides a basic checklist of questions to ask to ensure that the Gulfs of evaluation and execution are bridged. In general each stage of action requires its own special design strategies and, in turn, provides its own opportunity for disaster. It would be fun were it not also so frustrating, to look over the world and gleefully analyze each deficiency. On the whole, as you can see in figure the questions for each stage are relatively simple. And these, in turn, boil down to the principles of good design. Principles of good design are discussed below.



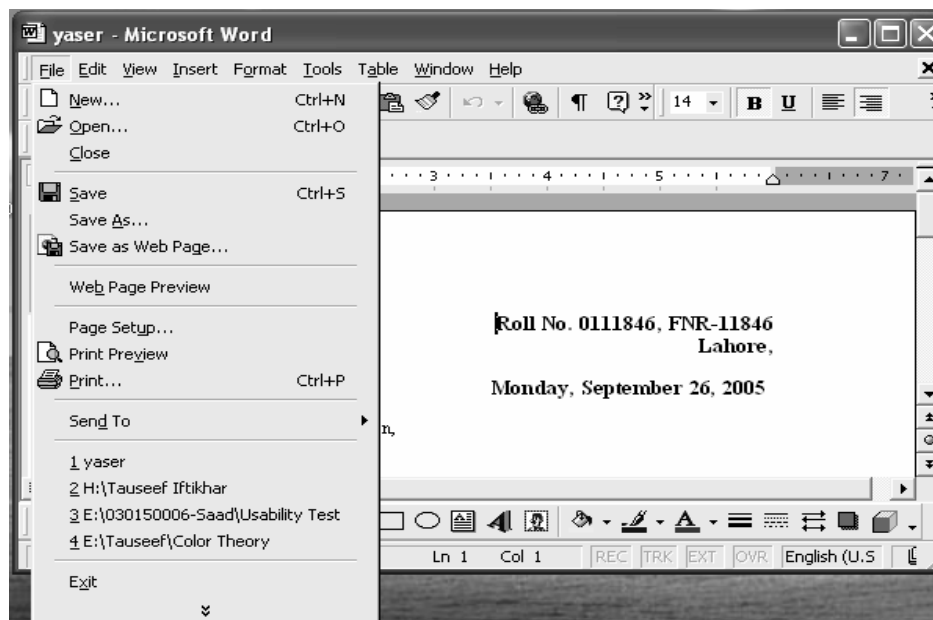
12.1 Design Principles

A number of design principles have been promoted. The best known are concerned with how to determine what users should see and do when carrying out their tasks using an interactive product. Here we briefly describe the most common ones

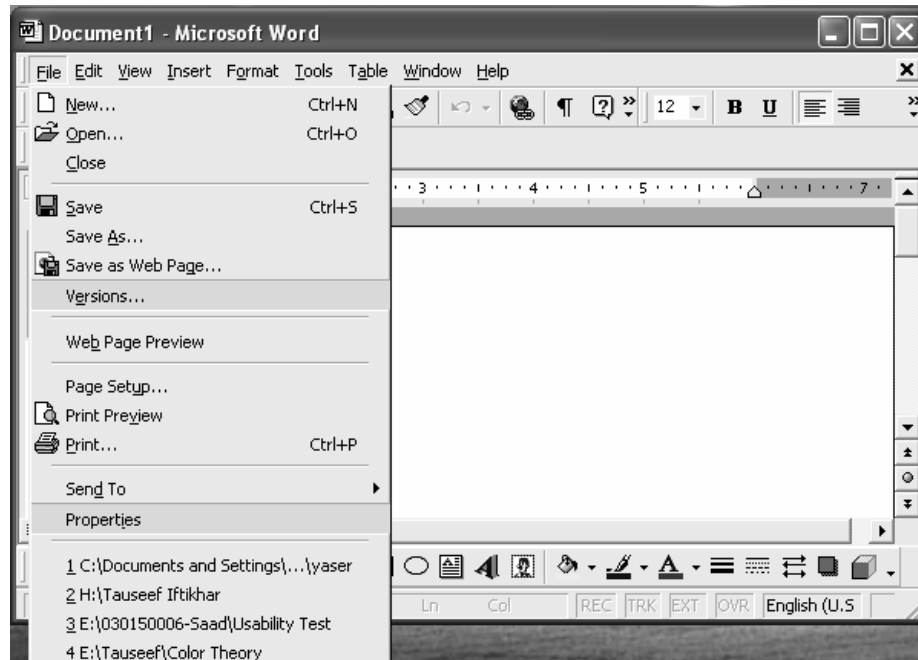
- Visibility
- Affordance
- Constraints
- Mapping
- Consistency
- Feedback

Visibility

The more visible functions are, the more likely users will be able to know what to do next. In contrast, when functions are “out of sight,” it makes them more difficult to find and know how to use. Norman describes the controls of a car to emphasize this point. The controls for different operations are clearly visible (e.g., indicator, headlights, horn, hazard warning lights), indicating what can be done. The relationship between the way the controls have been positioned in the car and what they do makes it easy for the driver to find the appropriate control for the task at hand. For example, one problem that I often encounter, in word processing software I often needed to set the properties of a word document. For this logically option of properties should be in the File menu, and I have often seen it there. But once, I opened the file menu I could not find it there, I was confused. Look at the figure



In confusion, I looked through all the menus but in vain. At last, surprisingly I was again looking at the file menu when I noticed the arrow at the bottom of the menu, when I clicked on that I was able to see that option again on the menu. Look at the figure bellow.



Affordance

Affordance is a term used to refer to an attribute of an object that allows people to know how to use it. For example, a mouse button invites pushing by the way it is physically constrained in its plastic shell. At a very simple level, to afford means “to give a clue.” When the affordances of a physical object are perceptually obvious it is easy to know how to interact with it. For example, a door handle affords pulling, a cup handle affords grasping, and a mouse button affords pushing. Norman introduced this concept in the late 80s in his discussion of the design of everyday objects. Since then, it has been much popularized, being what can be done to them. For example, graphical elements like button, icon, links, and scroll bars are talked about with respect to how to make it appear obvious how they should be used: icons should be designed to afford clicking, scroll bars to afford moving up and down, buttons to afford pushing.

There are two kind of affordance:

- Perceived
- Real

Real

Physical objects are said to have real affordances, like grasping, that are perceptually obvious and do not have to be learned.

Perceived

User interfaces that are screen-based are virtual and do not make sense to try to design for real affordances at the interface---except when designing physical devices, like control consoles, where affordance like pulling and pressing are helpful in guiding the user to

know what to do. Alternatively screen based interfaces are better conceptualized as perceived affordances, which are essentially learned conventions.

Constraints

The design concept of constraining refers to determining ways of restricting the kind of user interaction that can take place at a given moment. There are various ways this can be achieved. A common design practice in graphical user interfaces is to deactivate certain menu options by shading them, thereby restricting the user to only actions permissible at that stage of the activity. One of the advantages of this form of constraining is it prevents the user from selecting incorrect options and thereby refuses the chances of making a mistake. The use of different kinds of graphical representations can also constrain a person's interpretation of a problem or information space. For example flow chart diagram show which objects are related to which thereby constraining the way the information can be perceived. Norman classified constraints into three categories: physical, logical, and cultural.

Physical constraints

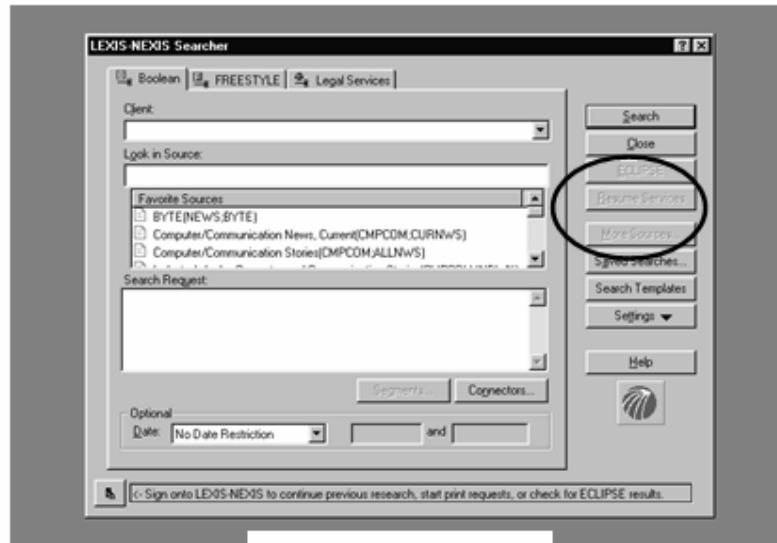
Physical constraints refer to the way physical objects restrict the movement of things. For example, the way a external disk can be placed into a disk drive is physically constrained by its shape and size, so that it can be inserted in only one way. Likewise, keys on a pad can usually be pressed in only one way.

Logical constraints

Logical constraints rely on people's understanding of the way the world works. They rely on people's common-sense reasoning about actions and their consequences. Picking up a physical marble and placing it in another location on the phone would be expected by most people to trigger something else to happen. Making actions and their effects obvious enables people to logically deduce what further actions are required. Disabling menu options when not appropriate for the task in hand provides logical constraining. It allows users to reason why (or why not) they have been designed this way and what options are available.

Culture constraints

Culture constraints rely on learned conventions, like the use of red for warning, the use of certain kinds of signals for danger, and the use of the smiley face to represent happy emotions. Most cultural constraints are arbitrary in the sense that their relationship with what is being represented is abstract, and could have equally evolved to be represented in another form (e.g., the use of yellow instead of red for warning). Accordingly, they have to be learned. Once learned and accepted by a cultural group, they become universally accepted conventions. Two universally accepted interface conventions are the use of windowing for displaying information and the use icons on the desktop to represent operations and documents.



Example of logical constraints

Mapping

This refers to the relationship between controls and their effects in the world. Nearly all artifacts need some kind of mapping between controls and effects, whether it is a flashlight, car, power plant, or cockpit. An example of a good mapping between controls and effects is the up and down arrows used to represent the up and down movement of the cursor, respectively, on a computer keyboard. The mapping of the relative position of controls and their effects is also important. Consider the various musical playing devices. How are the controls of playing rewinding and fast forward mapped onto the desired effects? They usually follow a common convention of providing a sequence of buttons, with the play button in the middle, the rewind button on the left and the fast-forward on the right. This configuration maps directly onto the directionality of the actions.

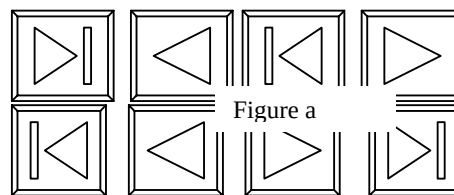


Figure b

Imagine how difficult it would be if the mapping in figure (a) were used.

Consistency

This refers to designing interfaces to have similar operations and use similar elements for achieving similar tasks. In particular, a consistent interface is one that follows rules, such as using the same operation to select all objects. For example, a consistent operation is using the same input action to highlight any graphical object at the interfaces, such as always clicking the left mouse button. Inconsistent interfaces, on the other hand, allow

exceptions to a rule. An example of this is where certain graphical objects (e.g., email messages presented in a table) can be highlighted using the right mouse button, while all other operations are highlighted using the left button. A problem with this kind of inconsistency is that is quite arbitrary, making it difficult for users to remember and making the users more prone to mistakes.

One of the benefits of consistent interfaces, therefore, is that they are easier to learn and use. Users have to learn only a single mode of operation that is applicable to all objects. This principle worked well for simple interfaces with limited operations, like mini CD player with small number of operations mapped onto separate buttons. Here all the user has to do is learn what each button represents and select accordingly. However, it can be more problematic to apply the concept of consistency to more complex interfaces, especially when many different operations need to be designed for. For example, consider how to design an interface for an application that offers hundreds of operations. There is simply not enough space for a thousand buttons, each of which maps onto an individual operation. Even if there were, it would be extremely difficult and time consuming for the user to search through them all to find the desired operation.

A much more effective design solution is to create categories of commands that can be mapped into subsets of operations. For the word-processing application, the hundreds of operation available are categorized into subsets of different menus. All commands that are concerned with file operations are placed together in the same file menu.

Another problem with consistency is determining what aspect of an interface to make consistent with what else. There are often many choices, some of which can be inconsistent with other aspects of the interface or ways of carrying out actions. Consider the design problem of developing a mechanism to let users lock their files on a shared server. Should the designer try to design it to be consistent with the way people lock things in the outside world (called external consistency) or with the way they lock objects in the existing system (called internal consistency)? However, there are many different ways of locking objects in the physical world (e.g., placing in a safe, using a padlock, using a key, using a child safety lock), just as there are different ways of locking electronically. The problem facing designer knows which one to be consistent with.

Feedback

Related to the concept of visibility is feedback. This is best illustrated by an analogy to what everyday life would be like without it. Imagine trying to play a guitar, slice bread using knife, or write a pen if none of the actions produced any effect for several seconds. There would be an unbearable delay before the music was produced, the bread was cut, or the words appeared on the paper, making it almost impossible for the person to continue with the next strum, saw, or stroke.

Feedback is about sending back information about what action has been done and what has been accomplished, allowing the person to continue with the activity. Various kinds of feedback are available for interaction design—audio, tactile, verbal, visual, and combinations of these. Deciding which combinations are appropriate for different kinds of activities and interactivities is central. Using feedback in the right way can also provide the necessary visibility for user interaction.

Lecture 13.

The Computer

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

Describe the advantages and disadvantages of different input output devices keeping in view different aspects of HCI

In previous lectures our topics of discussion were covering the human aspects. From now we will pay some attention towards computers. We will study some computer aspects. You may have studied many of them before in any other course, but that are also part of our discussion, as at one side of our subject is human and at the other side computer lies. Today will look at some input and output devices of computer. Let us first look at input devices.

13.1 Input devices

Input is concerned with recording and entering data into computer system and issuing instruction to the computer. In order to interact with computer systems effectively, users must be able to communicate their interaction in such a way that the machine can interpret them. Therefore, input devices can be defined as: a device that, together with appropriate software, transforms information from the user into data that a computer application can process.

One of the key aims in selecting an input device and deciding how it will be used to control events in the system is to help users to carry out their work safely, effectively, efficiently and, if possible, to also make it enjoyable. The choice of input device should contribute as positively as possible to the usability of the system. In general, the most appropriate input device will be the one that:

- Matches the physiology and psychological characteristics of users, their training and their expertise. For example, older adults may be hampered by conditions such as arthritis and may be unable to type; inexperienced users may be unfamiliar with keyboard layout.
- Is appropriate for the tasks that are to be performed. For example, a drawing task from a list requires an input device that allows continuous movement; selecting an option from a list requires an input device that permits discrete movement.
- Is suitable for the intended work and environment. For example, speech input is useful where there is no surface on which to put a keyboard but is unsuitable in noisy condition; automatic scanning is suitable if there is a large amount of data to be generated.

Frequently the demands of the input device are conflicting, and no single optimal device can be identified: trade-offs usually have to be made between desirable and undesirable features in any given situation. Furthermore, many systems will use two or more input

devices together, such as a keyboard and a mouse, so the devices must be complementary and well coordinated. This means that not only must an input device be easy to use and the form of input be straightforward, there must also be adequate and appropriate system feedback to guide, reassure, inform and if necessary, correct user's errors. This feedback can take various forms. It can be a visual display screen: a piece of text appears, an icon expands into a window, a cursor moves across the screen or a complete change of screen presentation occurs. It can be auditory: an alarm warning, a spoken comment or some other audible clue such as the sound of keys clicking when hit. It can be tactile: using a joystick. In many cases feedback from input can be a combination of visual, auditory and tactile responses. For example, when selecting an icon on a screen, the tactile feedback from the mouse button or function keys will tell users that they instructed the system to activate the icon. Simultaneously, visual feedback will show the icon changing shape on the screen. This is coordinated with the sound of the button clicking or the feel of the key resisting further pressure. Let us now discuss various types of devices in terms of their common characteristics and the factors that need to be considered when selecting an input device. We will discuss text entry devices first.

13.2 Text entry devices

There are many text entry devices as given below:

Keyboard

The most common method of entering information into the computer is through a keyboard. Since you have probably used them a lot without perhaps thinking about the related design issue, thinking about keyboards is a convenient starting point for considering input design issue. Broadly defined, a keyboard is a group of on—off push button, which are used either in combination or separately. Such a device is a discrete entry device. These devices involve sensing essentially one of two or more discrete positions (for example, keys on keyboards, touch-sensitive switches and buttons), which are either on or off, whereas others (for example, pens with digitizing tablets, moving joysticks, roller balls and sliders) involve sensing in a continuous range. Devices in this second category are therefore, known as continuous entry devices.

When considering the design of keyboards, both individual keys and grouping arrangements need to be considered. The physical design of keys is obviously important. For example, if keys are too small this may cause difficulty in locating and hitting chosen keys accurately. Some calculators seeking extreme miniaturization and some modern telephones suffer from this. Some keyboards use electro mechanical switches, while others use sealed, flat membrane keyboards. When pressing a key on a membrane keyboard, unless appropriate feedback is given on screen, or using sound it may be difficult to tell which key, if any, has been pressed. On the other hand, membrane keyboards can typically withstand grease, dirt and liquids that would soon clog up typical electromechanical switches. This can be an important consideration in environments such as production floors, farm and public places.

Alterations in the arrangement of the keys can affect a user's speed and accuracy. Various studies have shown that typing involves a great deal of analyses of trained typists suggest that typing is not a sequential act, with each key being sought out and pressed as the letters occur in the words to be typed. Rather, the typist looks ahead, processes text in chunks, and then types it in chunks. For alphabetic text these chunks are about two to

three world long for numerical material they are three to four characters long. The effect is to increase the typing speed significantly.

QWERTY keyboard

Most people are quite familiar with the layout of the standard alphanumeric keyboard, often called the qwerty keyboard, the name being derived from the first letters in the upper most row from left to center. This design first became a commercial success when used for typewriters in the USA in 1874, after many different prototypes had been tested. The arrangement of keys was chosen in order to reduce the incidence of keys jamming in the manual typewriters of the time rather than because of any optimal arrangement for typing. For example, the letters 's', 't', and 'h' are far apart even though they are far apart even though they are frequently used together.

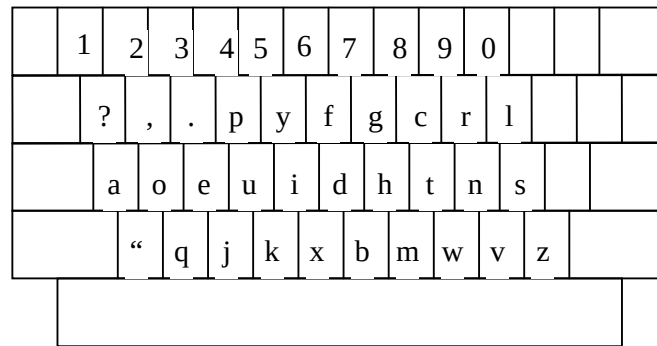
Alphabetic keyboard

One of the most obvious layouts to be produced is the alphabetic keyboard, in which the letters are arranged alphabetically across the keyboard. It might be expected that such a layout would make it quicker for untrained typists to use, but this is not the case. Studies have shown that this keyboard is not faster for properly trained typists, as we may expect, since there is no inherent advantage to this layout. And even for novice or occasional users, the alphabetic layout appears to make very little difference to the speed of typing. These keyboards are used in some pocket electronic personal organizers, perhaps because the layout looks simpler to use than the QWERTY one. Also, it dissuades people from attempting to use their touch-typing skills on a very small keyboard and hence avoids criticisms of difficulty of use.

Dvorak Keyboard

With the advent of electric and electronic keyboards and the elimination of levered hammers such considerations are no longer necessary. Attempts at designing alternative keyboards that are more efficient and quicker to use have produced, among others, the Dvorak and Alphabetic boards. The Dvorak board, first patented in 1932, was designed using the following principles:

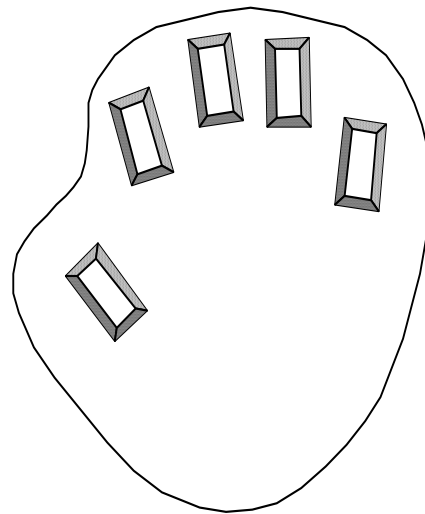
- Layout is arranged on the basis of frequency of usage of letters and the frequency of letter pattern and sequences in the English language.
- All vowels and the most frequently used consonants are on the second or home row, so that something like 70% of common words are typed on this row alone.
- Faster operation is made possible by tapping with fingers on alternate hands (particularly the index fingers) rather than by repetitive tapping with one finger and having the majority of keying assigned to one hand, as in the QWERTY keyboard, which favors left-handers. Since the probability of vowels and consonants altering is very high, all vowels are typed with the left hand and frequent home row consonants with the right.



The improvements made by such as ergonomic design are a significant reduction in finger travel and consequent fatigue and a probable increase in accuracy. Dvorak also claimed that this arrangement reduces the between-row movement by 90% and allows 35% of all words normally used to be typed on the home row. Despite its significant benefits, the Dvorak layout, shown in figure, has never been commercially successful. The possible gain in input speed has to be weighed against the cost of replacing existing keyboards and retraining millions of people who have learned the QWERTY keyboard.

Chord keyboards

In chord keyboards several keys must be pressed at once in order to enter a single character. This is a bit like playing a flute, where several keys must be pressed to produce with a small number of keys, few keys are required, so chord keyboards can be very small, and many can be operated with just one hand. Training is required to learn the finger combination required to use a chord keyboard. They can be very useful where space is very limited, or where one hand is involved in some other task. Training is required to learn the finger combinations required to use a chord keyboard. They can be very useful where space is very limited, or where one hand is involved in some other task. Chord keyboards are also used for mail sorting and a form of keyboard is used for recording transcripts of proceedings in law courts.



Some keyboards are even made of touch-sensitive buttons, which require a light touch and practically no travel; they often appear as a sheet of plastic with the buttons printed on them. Such keyboards are often found on shop till, though the keys are not QWERTY, but specific to the task. Being fully sealed, they have the advantage of being easily cleaned and resistant to dirty environment, but have little feel, and are not popular with trained touch-typists. Feedback is important even at this level of human-computer interaction! With the recent increase of repetitive strain injury (RSI) to users' finger, and the increased responsibilities of employers in these circumstances, it may be that such

designs will enjoy resurgence in the near future. The tendons that control the movement of the fingers becoming inflamed owing to overuse cause RSI in fingers and making repeated unnatural movement.

There are very varieties of specially shaped keyboards to relieve the strain of typing or to allow people to type with some injury or disability. These may slope the keys towards the hands to improve the ergonomics position, be designed for single-handed use, or for no hands at all. Some use bespoke key layouts to reduce strain of finger movements. The keyboard illustrated is produced by PCD Maltron Ltd. for left-handed use.

Phone pad and T9 entry

With mobile phones being used for SMS text messaging and WAP, the phone keypad has become an important form of text input. Unfortunately a phone only has digits 0-9, not a full alphanumeric keyboard.

To overcome this for text input the numeric keys are usually pressed several times. Figure shows a typical mapping of digits to letters. For example, the 3 keys have 'def' on it. If you press the key once you get a 'd', if you press 3 twice you get an 'e', and if you press it three times you get an 'f'. The main number-to-letter mapping is standard, but punctuation and accented letters differ between phones. Also there needs to be a way for the phone to distinguish, say, the 'dd' from 'e'. on some phones you need to pause for short period between successive letters using the same key, for others you press an additional key (e.g. '#').

Most phones have at least two modes for the numeric buttons: one where the keys mean the digits (for example when entering a phone number) and one where they mean letters (for example when typing an SMS message). Some have additional modes to make entering accented characters easier. Also a special mode or setting is needed for capital letters although many phones use rules to reduce this, for example automatically capitalizing the initial letter in a message and letters following full stops, question marks and exclamation marks.

This is all very laborious but you can see experienced mobile users make use of highly developed shorthand to reduce the number of keystrokes. If you watch a teenager or other experienced txt-er, you will see they often develop great typing speed holding the phone in one hand and using only their thumb. As these skills spread through society it may be that future devices use this as a means of small format text input. For those who never develop this physical dexterity some phones have tiny plug-in keyboards, or come with foldout keyboards.

Another technical solution to the problem is the T9 algorithm. This uses a large dictionary to disambiguate words by simply typing the relevant letters once. For example, '3926753' becomes 'example' as there is only one word with letters that match (alternative like 'ewbosld' that also match is not real words). Where there are ambiguities such as '26', which could be an 'am' or an 'an', the phone gives a series of options to choose from.

Handwriting recognition

Handwriting is a common and familiar activity, and is therefore attractive as a method of text entry. If we were able to write as we would when we use paper, but with the

computer taking this form of input and converting it to text, we can see that it is an intuitive and simple way of interacting with the computer. However, there are a number of disadvantages with hand writing recognition. Current technology is still fairly inaccurate and so makes a significant number of mistakes in recognizing letters, though it has improved rapidly. Moreover, individual differences in handwriting are enormous, and make the recognition process even more difficult. The most significant information in handwriting is not in the letter shape itself but in the stroke information— the way in which the letter is drawn. This means that devices which support handwriting recognition must capture the stroke information, not just the final character shape. Because of this, online recognitions are far easier than reading handwritten text on paper. Further complications arise because letters within words are shaped and often drawn very differently depending on the actual wide enough information. More serious in many ways is the limitation on speed; it is difficult to write at more than 25 words a minute, which is no more than half the speed of a decent typist.

The different nature of handwriting means that we may find it more useful in situation where a keyboard-based approach would have its own problems. Such situation will invariably result in completely new systems being designed around the handwriting recognizer as the predominant mode of textual input, and these may bear very little resemblance to the typical system. Pen-based systems that use handwriting recognition are actively marketed in the mobile computing market, especially for smaller pocket organizers. Such machines are typically used for taking notes and jotting down and sketching ideas, as well as acting as a diary, address book and organizer. Using handwriting recognition has many advantages over using a keyboard. A pen-based system can be small and yet still accurate and easy to use, whereas small keys become very tiring, or even impossible, to use accurately. Also the pen-based approach does not have to be altered when we move from jotting down text to sketching diagrams; pen-based input is highly appropriate for this also.

Some organizer designs have dispensed with a keyboard completely. With such systems one must consider all sorts of other ways to interact with the system that are not character based. For example, we may decide to use gesture recognition, rather than commands, to tell the system what to do, for example, drawing a line through a word in order to delete it. The important point is that a different input device that was initially considered simply as an alternative to the keyboard opens up a whole host of alternative designs and different possibilities for interaction.

Speech recognition

Speech recognition is a promising area of text entry, but it has been promising for a number of years and is still only used in very limited situations. However, speech input suggests a number of advantages over other input methods:

- Since speech is a natural form of communication, training new users is much easier than with other input devices.
- Since speech input does not require the use of hands or other limbs, it enables operators to carry out other actions and to move around more freely.
- Speech input offers disabled people such as the blind and those with severe motor impairment the opportunities to use new technology.

However, speech input suffers from a number of problems:

- Speech input has been applied only in very specialized and highly constrained tasks.
- Speech recognizers have severe limitations whereas a human would have a little problem distinguishing between similar sounding words or phrases; speech recognition systems are likely to make mistakes.
- Speech recognizers are also subject to interference from background noise, although the use of a telephone-style handset or a headset may overcome this.
- Even if the speech can be recognized, the natural form of language used by people is very difficult for a computer to interpret.

The development of speech input systems can be regarded as a continuum, with device that have a limited vocabulary and recognize only single words at one end of the spectrum and systems that attempt to understand natural speech at the other. Isolated word recognition systems typically require pauses between words to be longer than in natural speech and they also tend to be quite careful about how she speaks. Continuous speech recognition systems are capable, up to a point, of problems and system complexity. Although these systems still operate by recognizing a restricted number of words, the advantage is that they allow much faster data entry and are more natural to use. One way of reducing the possible confusion between words is to reduce the number of people who use the system. This can overcome some of the problem caused by variations in accent and intonation. Speaker-dependent systems require each user to train a system to recognize her voice by repeating all the words in the desired vocabulary one or more times. However, individual variability in voice can be a problem, particularly when a user has a cold. It is not uncommon for such systems to confuse words like three and repeat. Speaker-independent systems, as the name suggests, do not have this training requirement; they attempt to accommodate a large range of speaking characteristics and vocabulary. However, the problem of individual variability means that these types of system are less reliable, or have a smaller vocabulary than speaker-dependent systems.

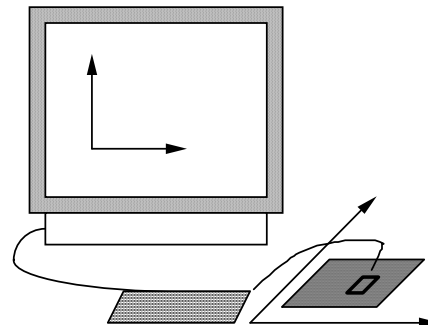
The perfect system would be one that would understand natural speech to such extent that it could not only distinguish differences in speech presentation but also have the intelligence to resolve any conflicts in meaning by interpreting speech in relation to the context of the conversation, as a human being does. This is a deep unsolved problem in Artificial Intelligence, and progress is likely to be slow.

13.3 Positioning, Pointing and Drawing

Pointing devices are input devices that can be used to specify a point or path in a one-, two- or three- dimensional space and, like keyboards, their characteristics have to be considering in relation to design needs.

Pointing devices are as follow:

- Mouse
- Touch pad
- Track ball
- Joystick
- Touch screen
- Eye gaze



Mouse

The mouse has become a major component of the majority of desktop computer systems sold today, and is the little box with the tail connecting it to the machine in our basic computer system picture. It is a small, palm-sized box housing a weighted ball- as the box is moved on the tabletop; the ball is rolled by the table and so rotates inside the housing. This rotation is detected by small rollers that are in contact with the ball, and these adjust the values of potentiometers.

The mouse operates in a planar fashion, moving around the desktop, and is an indirect input device, since a transformation is required to map from the horizontal nature of desktop to the vertical alignment of the screen. Left-right motion is directly mapped, whilst up-down on the screen is achieved by moving the mouse away-towards the user.

Foot mouse

Although most mice are hand operated, not all are there have been experiments with a device called the foot mouse. As the name implies, it is foot-operated device, although more akin to an isometric joysticks than a mouse. The cursor is moved by foot pressure on one side or the other of pad. This allows one to dedicate hands to the keyboard. A rare device, the foot mouse has not found common acceptance.

Touch pad

Touchpad's are touch-sensitive tablets usually around 2-3 inches square. They were first used extensively in Apple PowerBooks portable computers but are now used in many other notebook computers and can be obtained separately to replace the mouse on the desktop. They are operated by stroking a finger over their surface, rather like using a simulated trackball. The feel is very different from other input devices, but as with all devices users quickly get used to the action and become proficient.



Because they are small it may require several strokes to move the cursor across the screen. This can be improved by using acceleration settings in the software linking the trackpad movement to the screen movement. Rather than having a fixed ratio of pad distance to screen distance, this varies with the speed of movement. If the finger moves slowly over the pad then the pad movements map to small distances on the screen. If the finger is moving quickly the same distance on the touchpad moves the cursor a long distance.

Trackball and thumbwheel

Trackball is really just an upside-down mouse. A weighted ball faces upwards and is rotated inside a static housing, the motion being detected in the same way as for a mechanical mouse, and the relative motion of the ball moves the cursor. It is a very compact device, as it requires no additional space in which to operate. It is an indirect device, and requires separate buttons for selection. It is fairly accurate, but is hard to draw with, as long movements are difficult. Trackball now appear in a wide variety of sizes, the most usual being about the same as golf ball, with a number of larger and smaller devices available.

Thumbwheels are different in that they have two orthogonal dials to control the cursor position. Such a device is very cheap, but slow, and it is difficult to manipulate the cursor in any way other than horizontally or vertically. This limitation can sometimes be a useful constraint in the right application.

Although two-axis thumbwheels are not heavily used in mainstream applications, single thumbwheels are often included on a standard mouse in order to offer an alternative mean to scroll documents. Normally scrolling requires you to grab the scroll bar with the mouse cursor and drag it down. For large documents it is hard to be accurate and in addition the mouse dragging is done holding a finger down which adds to hand strain. In contrast the small scroll wheel allows comparatively intuitive and fast scrolling, simply rotating the wheel to move the page.

Joystick and track point

The joystick is an indirect input device, taking up very little space. Consisting of a small palm-sized box with a stick or shaped grip sticking up from it, the joystick is a simple device with which movements of the stick cause a corresponding movement of the screen cursor. There are two type of joystick: the absolute and the isometric.

In absolute joystick, movement is the important characteristic, since the position of the joystick in the base corresponds to the position of the cursor on the screen. In the isometric joystick, the pressure on the stick corresponds to the velocity of the cursor, and when released, the stick returns to its usual upright centered position.

Track point is a smaller device but with the same basic characteristics is used on many laptop computers to control the cursor. Some older systems had a variant of this called the key mouse, which was a single key. More commonly a small rubber nipple projects in the center of keyboard and acts as a tiny isometric joystick. It is usually difficult for novice to use, but this seems to be related to fine adjustment of the speed settings.

Touch screens

Touch displays allow the user to input information into the computer simply by touching an appropriate part of the screen or a touch-sensitive pad near to the screen. In this way the screen of the computer becomes a bi-directional instrument in that it both receives information from a user and displays output from a system. Using appropriate software different parts of a screen can represent different responses as different displays are presented to a user. For example, a system giving directions to visitors at a large exhibition may first present an overview of the exhibition layout in the form of general map. A user may then be requested to touch the hall that he wishes to visit and the system will present a list of exhibits. Having selected the exhibit of his choice by touching it, the user may then be presented with a more detailed map of the chosen hall.

The advantages of touch screens are that they are easy to learn, require no extra workplace, have no moving parts and are durable. They can provide a very direct interaction. Ease of learning makes them ideal for domains in which use by a particular user may occur only once or twice, and users cannot be expected to spend a time learning to use the system.

They suffer from a number of disadvantages, however. Using the finger to point is not always suitable, as it can leave greasy marks on screens and, being a fairly blunt instrument, it is quite inaccurate. This means that the selection of small regions is very difficult, as is accurate drawing. Moreover, lifting the arm to point a vertical screen is very tiring, and also means that the screen has to be within about a meter of the user to enable to be reached, which can make it too close for comfort.

Stylus and light pen

For more accurate positioning, systems with touch-sensitive surface often employ a stylus. Instead of pointing at the screen directly, small pen-like plastic stick is used to point and draw on the screen. This is particularly popular in PDAs, but they are also being used in some laptop computers.

An old technology that is used in the same way is the light pen. The pen is connected to the screen by a cable and, in operation, is held to the screen and detects a burst of light from the screen phosphor during the display scan. The light pen can therefore address individual pixels and so is much more accurate than the touch screen.

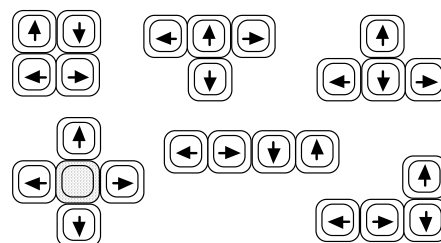
Eyegaze

Eyegaze systems allow you to control the computer by simply looking at it. Some systems require you to wear special glasses or a small head-mounted box, others are built into the screen or sit as a small box below the screen. A low-power laser is shone into the eye and is reflected off the retinal. The reflection changes as the angle of the eye alters, and by tracking the reflected beam the eyegaze system can determine the direction in which the eye is looking. The system needs to be calibrated, typically by staring at a series of dots on the screen, but thereafter can be used to move the screen cursor or for other more specialized uses. Eyegaze is a very fast and accurate device, but the more accurate versions can be expensive. It is fine for selection but not for drawing since the eye does not move in smooth lines. Also in real application it can be difficult to distinguish deliberately gazing at some thing and accidentally glancing it.

Cursor keys

Cursor keys are available on most keyboards. Four keys on the keyboard are used to control the cursor, one each for up, down, left and right. There is no standardized layout for the keys. Some layouts are shown in figure but the most common now is the inverted 'T'.

Cursor keys used to be more heavily used in character-based systems before windows and mice were the norm. However, when logging into remote machines such as web servers, the interface is often a virtual character-based terminal within a telnet window.

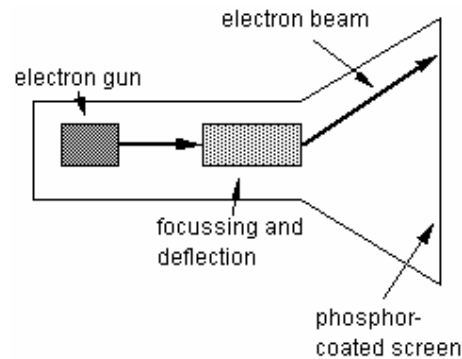


13.4 Display devices

Cathode ray tube

The cathode ray tube is the television-like computer screen still most common as we write this, but rapidly being displaced by flat LCD screens. It works in a similar way to a

standard television screen. A stream of electrons is emitted from an electron gun, which is then focused and directed by magnetic fields. As the beam hits the phosphor-coated screen, the phosphor is excited by the electrons and glows. The electron beam is scanned from left to right, and then flicked back to rescan the next line, from top to bottom.



Black and white screens are able to display grayscale by varying the intensity of the electron beam; color is achieved using more complex means. Three electron guns are used, one each to hit red, green and blue phosphors. Combining these colors can produce many others, including white, when they are all fully on. These three phosphor dots are focused to make a single point using a shadow mask, which is imprecise and gives color screens a lower resolution than equivalent monochrome screens.

The CRT is a cheap display device and has fast enough response times for rapid animation coupled with a high color capability. Note that animation does not necessarily means little creatures and figures running about on the screen, but refers in a more general sense to the use of motion in displays: moving the cursor, opening windows, indicating processor-intensive calculations, or whatever. As screen resolution increased, however, the price rises. Because of the electron gun and focusing components behind the screen, CRTs are fairly bulky, though recent innovations have led to flatter displays in which the electron gun is not placed so that it fires directly at the screen, but fires parallel to the screen plane with the resulting beam bent through 90 degrees to hit the screen.

Liquid Crystal Display

Liquid Crystal Displays are mostly used in personal organizer or laptop computers. It is a light, flat plastic screen. These displays utilize liquid crystal technology and are smaller, lighter and consume far less power than traditional CRTs. These are also commonly referred to as flat-panel displays. They have no radiation problems associated with them, and are matrix addressable, which means that individual pixels can be accessed without the need for scanning.

This different technology can be used to replace the standard screen on a desktop computer, and this is now common. However, the particular characteristics of compactness, lightweight, and low power consumption have meant that these screens have created a large niche in the computer market by monopolizing the notebook and portable computer systems side.

Digital paper

A new form of display that is still in its infancy is the various forms of digital papers. These are thin flexible materials that can be written to electronically, just like a computer screen, but which keep their contents even when removed from any electrical supply.

Physical controls and sensors**Sound output**

Another mode of output that we should consider is that of auditory signals. Often designed to be used in conjunction with screen displays, auditory outputs are poorly understood: we do not yet know how to utilize sound in a sensible way to achieve maximum effect and information transference. Sounds like beeps, bongs, clanks, whistles and whirrs are all used for varying effect. As well as conveying system output, sounds offer an important level of feedback in interactive systems. Keyboards can be set to emit a click each time a key is pressed, and this appears to speed up interactive performance. Telephone keypads often sound different tones when the keys are pressed; a noise occurring signifies that the key has been successfully pressed, whilst the actual tone provides some information about the particular key that was pressed.

13.5 Touch, feel and smell

Sense of touch and feel is also used for feedback; tactile feedback has its own importance and is being used in many interactive devices. We usually feel textures when we move our fingers over a surface. Technology for this is just beginning to become available.

13.6 Physical controls

A desktop computer has to serve many functions and do has generic keys and controls that can be used for a variety of purpose. In contrast, these dedicated controls panes have been designed for a particular device and for a single use. This is why they differ so much.

Usually microwave a flat plastic control panel. The reason is this, the microwave is used in the kitchen whilst cooking, with hands that may be greasy or have food on them. The smooth controls have no gaps where food can accumulate and clog buttons, so it can easily be kept clean an hygienic.

When using the washing machine you are handling dirty clothes, which may be grubby, but not to the same extent, so the smooth easy-clean panel is less important. It has several major settings and the large buttons act both as control and display.

13.7 Environment and bio sensing

Although we are not always conscious of them, there are many sensors in our environment—controlling automatic doors, energy saving lights, etc. and devices monitoring our behavior such as security tags in shops. The vision of ubiquitous computing suggests that our world will be filled with such devices.

Lecture 14.

Interaction

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Define interaction
- Discuss interaction styles keeping in view different aspects of HCI

In the previous lectures we have studied the detailed introduction of human side and computer side. These are two participant of our course Human Computer Interaction. As the name of the course reveals that both of these complex entities are not in isolation rather they come in contact with each other. Human communicate with computers.

There are a number of ways in which human can communicate with the system. If we look at the beginning, batch input system was used, in which the user provides all the information to the computer in form of batch. Now a day it is the age of virtual reality and ubiquitous computing. Here user constantly interacts with computers in his surroundings. Today there is richer interaction.

14.1 The terms of Interaction

Domain

A domain defines an area of expertise and knowledge in some real-world activity. Some examples of domains are graphic design, authoring and process control in a factory. A domain consists of concepts that highlight its important aspects. In a graphic design domain, some of the important concepts are geometric shapes, a drawing surface and a drawing utensil.

Task

Tasks are the operation to manipulate the concepts of a domain. A goal is the desired output from a performed task. For example, one task within the graphic design domain is the construction of a specific geometric shape with particular attributes on the drawing surface.

Goal

A related goal would be to produce a solid red triangle centered on the canvas. So, goal is ultimate result, which you want to achieve after performing some specific tasks.

14.2 Donald Norman's Model

We have already studied Donald Norman's Model of interaction. In which user chooses a goal, formulate a plan of action, which is then executed at the computer interface. When the plan, or part of the plan has been executed, the user observes the computer interface to evaluate the result of the execution plan, and to determine further actions.

The two major parts, execution and evaluation, of interactive cycle are further subdivided into seven stages, where each stage is an activity of the user. Seven stages of action are shown in figure. To understand these we see an example, which was also used by Norman.

Imagine you are sitting reading as evening falls. You decide you need more light; that is you establish the goal to get lighter. From there you form an intention to switch on the desk lamp, and you specify the actions required to reach over and press the lamp switch. If some one else is closer, the intention may be different-you may ask them to switch on the light for you. Your goal is the same but the intention and actions are different. When you have executed the action you perceive the result, either the light is on or it isn't and you interpret this, based on your knowledge of the world. For example, if the light does not come on you may interpret this as indicating the bulb has blown or the lamp is not plugged into the mains, you will formulate the new state according to the original goals – is there is now enough light? If so, the cycle is completed. If not, you may formulate a new intention to switch on the main ceiling light as well.

Gulf of execution and evaluation

Norman also describes the two gulfs, which represent the problems that are caused by some interfaces to their users.

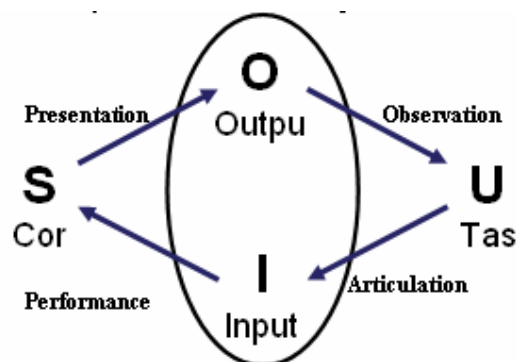
Gulf of execution

Gulf of execution is the difference between the user's formulation of the actions to reach the goal and the actions allowed by the system. If the action allowed by the system correspond to those intended by the user, the interaction will be effective. The interface should therefore aim to reduce this gulf of execution.

Gulf of evaluation

The gulf of evaluation is the distance between the physical presentation of the system state and the expectation of the user. If the user can readily evaluate the presentation in terms of his goal, the gulf of evaluation is small. The more effort that is required on the part of the user to interpret the presentation, the less effective the interaction.

14.3 The interaction framework

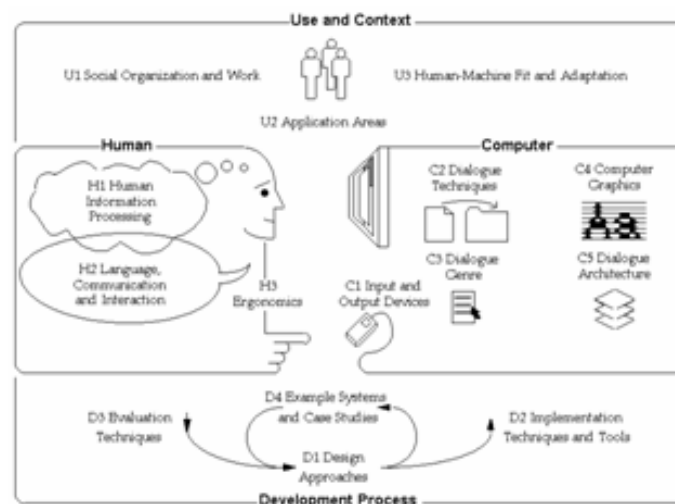


The interaction framework breaks the system into four main components as shown in figure. The nodes represent the four major components in an interactive system – the

System, the User, the Input and the Output. Each component has its own language. The system and user are each described by means of a language that can express concepts relevant in the domain of the application. The system's language is referred as the core language and the user's language is referred as the task language. The core language describes computational attributes of the domain relevant to the system state, whereas the task language describes psychological attributes of the domain relevant to the user state. There are also languages for both the input and output components. Input and output together form the interface.

As the interface sits between the user and the system, there are four steps in the interactive cycle, each corresponding to a translation from one component to another, as shown by the labeled arcs in figure. The user begins the interactive cycle with the formulation of a goal and a task to achieve that goal. The only way the user can manipulate the machine is through the input, and so the task must be articulated within the input language, the input language is translated into the core language as operations to be performed by the system. The system then transforms itself as described by the operations; the execution phase of the cycle is complete and the evaluation phase now begins. The system is in a new state, which must now be communicated to the user. The current values of system attributes are rendered as concepts or features of the output. It is then up to the user to observe the output and assess the results of the interaction relative to the original goal, ending the evaluation phase and, hence, the interactive cycle. There are four main translations involved in the interaction: articulation, performance, presentation and observation. The user's formulation of the desired task to achieve some goal needs to be articulated in the input language. The tasks are responses of the user and they need to be translated to stimuli for the input. As pointed out above, this articulation is judged in terms of the coverage from tasks to input and the relative ease with which the translation can be accomplished. The task is phrased in terms of certain psychological attributes that highlight the important features of the domain for the user. If these psychological attributes map clearly onto the input language, then articulation of the task will be made much simpler.

14.4 Frameworks and HCI



The ACM SIGCHI Curriculum Development Group presents a framework and uses it to place different areas that relate to HCI. As you can see in the figure, the field of ergonomics addresses issues on the user side of the interface, covering input and output, as well as the user's immediate context. Dialog design and interface styles can be placed particularly along the input branch of the framework, addressing both articulation and performance. However, dialog is most usually associated with the computer and so is biased to that side of the framework. Presentation and screen design relates to the output branch of the framework. The entire framework can be placed within a social and organizational context that also affects the interaction. Each of these areas has important implications for the design of interactive systems and the performance of the user. Let us first take a brief look.

Ergonomics

Ergonomics (or human factors) is traditionally the study of the physical characteristic of the interaction: how the controls are designed, the physical environment in which the interaction takes place and the layout and physical qualities of the screen. A primary focus is on user performance and how the interface enhances or detracts from this. In seeking to evaluate these aspects of the interaction, ergonomics will certainly also touch upon human psychology and system constraints. It is a large and established field, which is closely related to but distinct from HCI.

Physical aspects of Interface are as follow:

- Arrangement of controls and displays
- The physical environment
- Health issues
- Use of colors

Arrangement of controls and displays

We already have discussed in previous lectures the perceptual and cognitive issues that affect the way we present information on a screen and provide control mechanisms to the user. In addition to these cognitive aspects of design, physical aspects are also important. The user should group sets of controls and parts of the display logically to allow rapid access. This may not seem so important when we are considering a single user of a spreadsheet on a PC, but it becomes vital we turn to safety-critical applications such as plant control, aviation and air traffic control. In each of these contexts, users are under pressure and are faced with a huge range of displays and controls. Here it is crucial that the physical layout of these be appropriate. Indeed, returning to the less critical PC application, inappropriate placement of controls and displays can lead to inefficiency and frustration.

Industrial Interface

The interfaces to office systems have changed dramatically since the 1980s. However, some care is needed in transferring the idioms of office-based systems into the industrial domain. Office information is primarily textual and slow varying, whereas industrial interfaces may require the rapid assimilation of multiple numeric displays; each of which is varying in response to the environment. Furthermore, the environment conditions may rule out certain interaction styles. Consequently, industrial interfaces raise some additional design issues rarely encountered in the office.

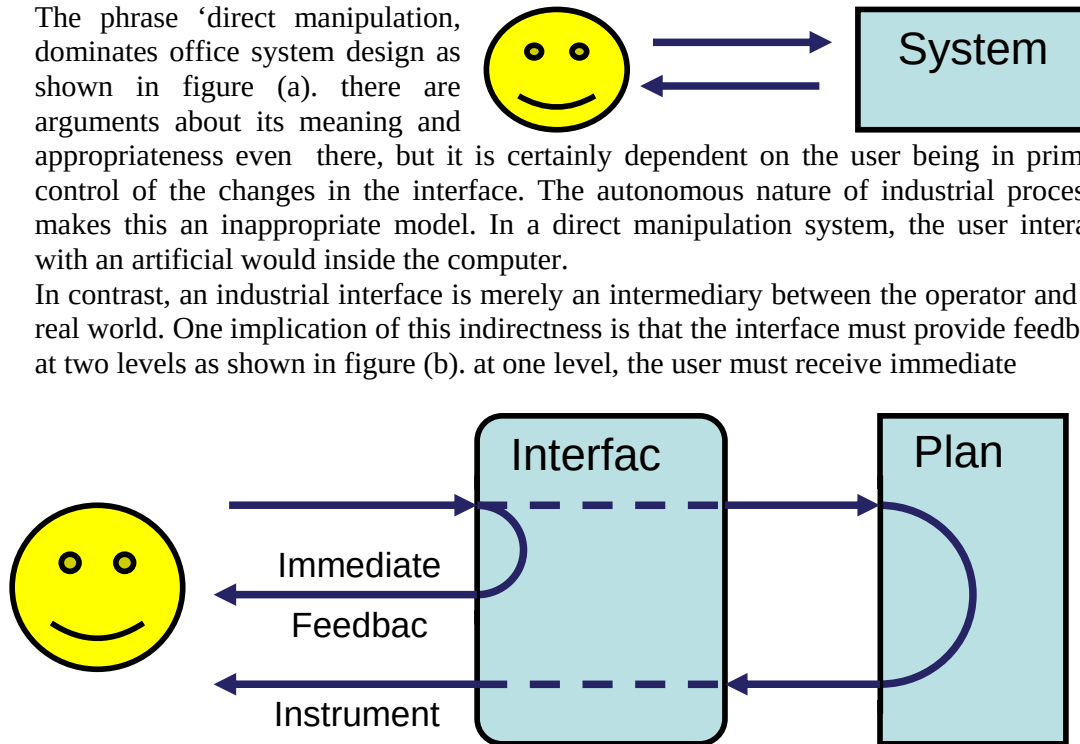
Glass interfaces vs. dials and knobs

The traditional machine interface consists of dials and knobs directly wired or piped to the equipment. Increasingly, some or all of the controls are replaced with a glass interface, a computer screen through which the equipment is monitored and controlled. Many of the issues are similar for the two kinds of interface, but glass interfaces do have some special advantages and problems. For a complex system, a glass interface can be both cheaper and more flexible, and it is easy to show the same information in multiple forms.

Indirect manipulation

The phrase 'direct manipulation, dominates office system design as shown in figure (a). there are arguments about its meaning and appropriateness even there, but it is certainly dependent on the user being in primary control of the changes in the interface. The autonomous nature of industrial processes makes this an inappropriate model. In a direct manipulation system, the user interacts with an artificial world inside the computer.

In contrast, an industrial interface is merely an intermediary between the operator and the real world. One implication of this indirectness is that the interface must provide feedback at two levels as shown in figure (b). at one level, the user must receive immediate



feedback, generated by the interface, that keystrokes and other actions have been received. In addition, the user's action will have some effect on the equipment controlled by the interface and adequate monitoring must be provided for this.

The indirectness also causes problems with simple monitoring tasks. Delays due to periodic sampling, slow communication and digital processing often mean that the data displayed are somewhat out of date. If the operator is not aware of these delays, diagnoses of system state may be wrong. These problems are compounded if the interface produces summary information displays. If the data comprising such a display are of different timeliness the result may be misleading.

The physical environment of the interaction

As well as addressing physical issues in the layout and arrangement of the machine interface, ergonomics is concerned with the design of the work environment itself. Where will the system be used? By whom will it be used? Will users be sitting, standing or moving about? Again, this will depend largely on the domain and will be more critical in

specific control and operational settings than in general computer use. However, the physical environment in which the system is used may influence how well it is accepted and even the health and safety of its users. It should therefore be considered in all design.

Health issues

Perhaps we do not immediately think of computer use as a hazardous activity but we should bear in mind possible consequences of our designs on the health and safety of users. Leaving aside the obvious safety risks of poorly designed safety-critical systems. There are a number of factors in that may affect the use of more general computers. Again these are factors in the physical environment that directly affect the quality of the interaction and the user's performance:

Physical position

As we discussed earlier users should be able to reach all controls comfortably and see all displays. Users should not be expected to stand for long periods and, if sitting, should be provided with back support.

Temperature

Although most users can adapt to slight changes in temperature without adverse effect, extremes of hot or cold will affect performance and, in excessive cases, health.

Lighting

The lighting level will again depend on the work environment. However, adequate lighting should be provided to allow users to see the computer screen without discomfort or eyestrain. The light source should also be positioned to avoid glare affecting the display.

Noise

Excessive noise can be harmful to health, causing the user pain, and in acute cases, loss of hearing. Noise level should be maintained at a comfortable level in the work environment.

Time

The time users spend using the system should also be controlled.

The use of color

Ergonomics has a close relationship to human psychology in that it is also concerned with the perceptual limitations of humans. For example, the use of color in displays is an ergonomics issue. The human visual system has some limitations with regard to color, including the number of colors that are distinguishable and the relatively low blue acuity. Color used in display should be as distinct as possible and the distinction should not be affected by changes in contrast. The colors used should also correspond to common conventions and user expectation. However, we should remember that color conventions are culturally determined.

14.5 Interaction styles

Interaction is communication between computer and human (user). For a successful enjoyable communication interface style has its own importance.

There are a number of common interface styles including

- Command line interface
- Menus
- Natural language

Question/answer and query dialog

- Form fills and spreadsheets
- WIMP
- Point and click
- Three-dimensional interfaces.

Command line interface

Command line interface was the first interactive dialog style to be commonly used and, in spite of the availability of menu-driven interface, it is still widely used. It provides a means of expressing instructions to the computer directly, using some function keys, single characters, abbreviations or whole-word commands.

Command line interface are powerful in that they offer direct access to system functionality, and can be combined to apply a number of tools to the same data. They are also flexible: the command often has a number of options or parameters that will vary its behavior in some way, and it can be applied to many objects at once, making it useful for repetitive tasks.

Menu

In the menu-driven interface, the set of options available to the user is displayed on the screen and selected using the mouse, or numeric or alphabetic keys. Since the options are visible they are less demanding of the user, relying on recognition rather than recall. However, menu options still need to be meaningful and logically grouped to aid recognition. Often menus are hierarchically ordered and the option required is not available at the top layer of the hierarchy. The grouping and naming of menu options then provides the only cue for the user to find the required option. Such systems either can be purely text based, with the menu options being presented as numbered choices, or may have a graphical component in which the menu appears within a rectangular box and choices are made, perhaps by typing the initial letter of the desired selection, or by entering the associated number, or by moving around the menu with the arrow keys. This is restricted form of a full WIMP system.



Natural Language

Perhaps the most attractive means of communicating with computers, at least at first glance, is by natural language. Users unable to remember a command or lost in a hierarchy of menus, may long for the computer that is able to understand instructions expressed in everyday words. Unfortunately, however, the ambiguity of natural language makes it very difficult for a machine to understand.

Question/answer and query dialog

Question and answer dialog is a simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions and so is led through the interaction step by step. These interfaces are easy to learn and use, but are limited in functionality and power. As such, they are appropriate for restricted domains and for novice or casual users.

Query languages, on the other hand, are used to construct queries to retrieve information from a database. They use natural-language-style phrases, but in fact require specific syntax, as well as knowledge of database structure. Queries usually require the user to specify an attribute or attributes for which to search the database, as well as the attributes of interest to be displayed. This is straightforward where there is a single attribute, but becomes complex when multiple attributes are involved, particularly of the user is interested in attribute A or attribute B, or attribute A and not attribute B, or where values of attributes are to be compared. Most query language do not provide direct confirmation of what was requested, so that the only validation the user has is the result of the search. The effective use of query languages therefore requires some experience.

Form-fills and spreadsheets

Form-filling interfaces are used primarily for data entry but can be useful in data retrieval applications. The user is presented with a display resembling a paper form, with slots to fill in as shown in figure. Most form-filling interfaces allow easy movement around the form and allow some fields to be left blank. They also require correction facilities, as users may change their minds or make a mistake about the value that belongs in each field.

Spreadsheets are sophisticated variation of form filling. The spreadsheet comprises a grid of cells, each of which can contain a value or a formula. The formula can involve the value of other cells. Now days MS Excel is used widely. In past VISICALC and Lotus 123 had been used.

D13 /F\$ <U> +D3-B13+C13 C 58

	A	B	C	D	E	F	G	H
1	Payee	Cheques	Deposits	Balance				
2								
3				545.20				
4								
5	Electric	14.95						
6	Oil	102.15						
7	Phone	36.80						
8	Dentist	40.00						
9	Salary		395.00					
10	Rent	350.00						
11	Gas Card	12.93						
12								
13	TOTALS	556.83	395.00	383.37				
14								
15								
16								
17								
18								
19								
20								
21								

VISICALC

The WIMP Interfaces

Currently many common environments for interactive computing are examples of the WIMP interface style, often simply called windowing systems. WIMP stands for windows, icons, menus, and pointers, and is default interface style for the majority of interactive computer systems in use today, especially in the PC and desktop workstation arena.

Point and Click interface

In most multimedia systems and in web browsers, virtually all actions take only a single click of the mouse button. You may point at a city on a map and when you click a window opens, showing you tourist information about the city. You may point at a word in some text and when you click you see a definition of the word. You may point at a recognizable iconic button and when you click some action is performed.

Three-dimensional interfaces

There is an increasing use of three-dimensional effects in user interfaces. The most obvious example is virtual reality, but VR is only part of a range of 3D techniques available to the interface designer.

The simplest technique is where ordinary WIMP elements, buttons, scroll bars, etc., are given a 3D appearance using shading, giving the appearance of being sculpted out of stone.

Lecture 15.

Interaction Paradigms

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Describe WIMP interfaces in detail
- Discuss different interaction paradigms

We have briefly discussed about the WIMP interfaces in last lecture. Today we will discuss WIMP interfaces in detail.

15.1 The WIMP Interfaces

In our last lecture we have already discussed the four key features of the WIMP interface that give it its name – windows, icons, pointers and menus – and today we will discuss these in greater detail. There are also many additional interaction objects and techniques commonly used in WIMP interfaces, some designed for specific purposes and others more general. Our discussion will cover the toolbars, menus, buttons, palettes and dialog boxes. Together, these elements of the WIMP interfaces are called widgets, and they comprise the toolkit for interaction between user and system.

Windows

Windows are areas of the screen that behave as if they were independent terminals in their own right. A window can usually contain text or graphics, and can be moved or resized. More than one window can be on a screen at once, allowing separate tasks to be visible at the same time. Users can direct their attention to the different windows as they switch from one thread of work to another.

If one window overlaps the other, the back window is partially obscured, and then refreshed when exposed again. Overlapping windows can cause problems by obscuring vital information, so windows may also be tiled, when they adjoin but do not overlap each other. Alternatively, windows may be placed in a cascading fashion, where each new window is placed slightly to the left and below the previous window. In some systems this layout policy is fixed, in others the user can select it.

Usually windows have various things associated with them that increase their usefulness. Scrollbars are one such attachment, allowing the user to move the contents of the window up and down, or from side to side. This makes the window behave as if it were a real window onto a much larger world, where new information is brought into view by manipulating the scrollbars.

There is usually a title bar attached to the top of a window, identifying it to the user, and there may be special boxes in the corners of the window to aid resizing, closing, or making as large as possible. Each of these can be seen in the figure.

In addition, some systems allow windows within windows. For example, in Microsoft Office applications, such as Excel and Word, each application has its own window and then within this each document has a window. It is often possible to have different layout policies within the different application windows.

Icons

Windows can be closed and lost forever, or they can be shrunk to some much reduced representation. A small picture is used to represent a closed window, and this representation is known as an icon. By allowing icons, many windows can be available on the screen at the same time, ready to be expanded to their full size by clicking on the icon. Shrinking a window to its icon is known as iconifying the window. When a user temporarily does not want to follow a particular thread of dialog, he can suspend that dialog by iconifying the window containing the dialog. The icon saves space on the screen and serves as a remainder to the user that he can subsequently resume the dialog by opening up the window. Figure shows a few examples of icons used in a typical windowing system (Microsoft).

Icons can also be used to represent other aspects of the system, such as a wastebasket for throwing unwanted files into, or various disks, programs or functions, that are accessible to the user. Icon can take many forms: they can be realistic representation of the objects that they stand for, or they can be highly stylized. They can even be arbitrary symbols, but these can be difficult for users to interpret.

Pointers

The Pointer is an important component of the WIMP interface, since the interaction style required by WIMP relies very much on pointing and selecting things such as icons. The mouse provides an input device capable of such tasks, although joysticks and trackballs are other alternatives. The user is presented with a cursor on the screen that is controlled by the input device.

The different shapes of cursor are often used to distinguish modes, for example the normal pointer cursor maybe an arrow, but change to change to cross-hairs when drawing a line. Cursors are also used to tell the user about system activity, for example a watch or hourglass cursor may be displayed when the system s busy reading a file. Pointer cursors are like icons, being small bitmap images, but in addition all cursors have a hot-spot, the location to which they point.

Menus

The last main feature of the windowing system is the menu, an interaction technique that is common across many non-windowing systems as well. A menu presents a choice of operations or services that can be performed by the system at a given time. As we discussed our ability to recall information is inferior to our ability to recognize it from some visual cue. Menus provide information cues in the form of an ordered list of operations that can be scanned. This implies that the names used for the commands in the menu should be meaningful and informative.

The pointing device is used to indicate the desired option. As the pointer moves to the position of a menu item, the item is usually highlighted to indicate that it is the potential candidate for selection. Selection usually requires some additional user action, such as

pressing a button on the mouse that controls the pointer cursor on the screen or pressing some special key on the keyboard. Menus are inefficient when they have too many items, and so cascading menus are utilized, in which item selection opens up another menu adjacent to the item, allowing refinement of the selection. Several layers of cascading menus can be used.

The main menu can be visible to the user all the time, as a menu bar and submenus can be pulled down or across from it upon request. Menu bars are often placed at the top of the screen or at the top of each window. Alternative includes menu bars along one side of the screen, or even placed amongst the windows in the main 'desktop' area. Websites use a variety of menu bar locations, including top, bottom and either side of the screen. Alternatively, the main menu can be hidden and upon request it will pop up onto the screen. These pop-up menus are often used to present context-sensitive options, for example allowing one to examine properties of particular on-screen objects. In some systems they are also used to access more global actions when the mouse is depressed over the screen background.

Pull-down menus are dragged down from the title bar at the top of the screen, by moving the mouse pointer into the title bar area and pressing the button. Pull-down menus are similar, except that the menu automatically appears when the mouse pointer enters the title bar, without the user having to press the button. Some menus explicitly ask to go away. Pop up menus appear when a particular region of the screen, may be designated by an icon, is selected, but they only stay as long as the mouse button is depressed.

Another approach to menu selection is to arrange the options in a circular fashion. The pointer appears in the center of the circle, and so there is the same distance to travel to any of the selections. This has the advantages that it is easier to select items, since they can each have a larger target area, and that the selection time for each item is the same, since the pointer is equidistant from them all. However, these pie menus take up more screen space and are therefore less common in interface.

The major problems with menus in general are deciding what items to include and how to group those items. Including too many items makes menus too long or creates too many of them, whereas grouping causes problems in that items that relate to the same topic need to come under the same heading, yet many items could be grouped under more than one heading. In pull-down menus the menu label should be chosen to reflect the function of the menu items, and items grouped within menus by function. These groupings should be consistent across applications so that the user can transfer learning to new applications. Menu items should be ordered in the menu according to importance and frequency of use, and appropriate functionalities should be kept apart to prevent accidental selection of the wrong function, with potentially disastrous consequences.

Keyboard accelerators

Menus often offer keyboard accelerators, key combinations that have the same effect as selecting the menu item. This allows more expert users, familiar with the system, to manipulate things without moving off the keyboard, which is often faster. The accelerators are often displayed alongside the menu item so that frequent use makes them familiar.

Buttons

Buttons are individual and isolated regions within display that can be selected by the user to invoke specific operations. These regions are referred to as buttons because they are purposely made to resemble the push buttons you would find on a control panel. 'Pushing' the button invokes a command, the meaning of which is usually indicated by a textual label or a small icon.

Radio Buttons

Buttons can also be used to toggle between two states, displaying status information such as whether the current font is italicized or not in a word processor or selecting options on a web form. Such toggle buttons can be grouped together to allow a user to select one feature from a set of mutually exclusive options, such as the size in points of the current font. These are called radio buttons.

Check boxes

If a set of options is not mutually exclusive, such as font characteristics like bold, italic and underlining, and then a set of toggle buttons can be used to indicate the on/off status of the options. This type of collection of buttons is sometimes referred to as check boxes

Toolbars

Many systems have a collection of small buttons, each with icons, placed at the top or side of the window and offering commonly used functions. The function of this toolbar is similar to a menu bar, but as the icons are smaller than the equivalent text more functions can be simultaneously displayed. Sometimes the content of the toolbar is fixed, but often users can customize it, either changing which functions are made available, or choosing which of several predefined toolbars is displayed

Palettes

In many application programs, instructions can either one of several modes. The defining characteristic of modes is that the interpretation of actions, such as keystrokes or gestures with the mouse, changes as the mode change. For example, using the standard UNIX text editor vi, keystrokes can be interpreted either as operations to insert characters in the document or as operations to perform file manipulation. Problems occur if the user is not aware of the current mode. Palettes are a mechanism for making the set of possible modes and the active mode visible to the user. A palette is usually a collection of icons that are reminiscent of the purpose of the various modes. An example in a drawing package would be a collection of icons to indicate the pixel color or pattern that is used to fill in objects, much like an artist's palette for paint.

Some systems allow the user to create palettes from menus or toolbars. In the case of pull-down menus, the user may be able 'tear off' the menu, turning it into a palette showing the menu items. In the case of toolbars, he may be able to drag the toolbar away from its normal position and place it anywhere on the screen. Tear-off menus are usually those that are heavily graphical anyway, for example line style of color selection in a drawing package.

Dialog boxes

Dialog boxes are information windows used by the system to bring the user's attention to some important information, possibly an error or a warning used to prevent a possible error. Alternatively, they are used to invoke a sub dialog between user and system for a very specific task that will normally be embedded within some larger task. For example, most interactive applications result in the user creating some file that will have to be named and stored within the filing system. When the user or the file and indicate where it is to be located within the filing system. When the save sub dialog is complete, the dialog box will disappear. Just as windows are used to separate the different threads of user-system dialog, so too are dialog boxes used to factor out auxiliary task threads from the main task dialog.

15.2 Interaction Paradigms

We believe that we now build interactive systems that are more usable than those built in the past. We also believe that there is considerable room for improvement in designing more usable systems in the future. The great advances in computer technology have increased the power of machines and enhanced the bandwidth of communication between human and computer. The impact of the technology alone, however, is not sufficient to enhance its usability. As our machines have become more powerful, the key to increased usability has come from the creative and considered application of the technology to accommodate and augment the power of the human. Paradigms for interaction have for the most part been dependent upon technological advances and their creative application to enhance interaction.

By interaction paradigm, it is meant a particular philosophy or way of thinking about interaction design. It is intended to orient designers to the kinds of questions they need to ask. For many years the prevailing paradigm in interaction design was to develop application for the desktop – intended to be used by single user sitting in front of a CPU, monitor, keyboard and mouse. A dominant part of this approach was to design software applications that would run using a GUI or WIMP interface.

Recent trend has been to promote paradigms that move beyond the desktop. With the advent of wireless, mobile, and handheld technologies, developers started designing applications that could be used in a diversity of ways besides running only on an individual's desktop machine.

Time sharing

In the 1940s and 1950s, the significant advances in computing consisted of new hardware technologies. Mechanical relays were replaced by vacuum electron tubes. Tubes were replaced by transistors, and transistors by integrated chips, all of which meant that the amount of sheer computing power was increasing by orders of magnitude. By the 1960s it was becoming apparent that the explosion of growth in computing power would be wasted if there were not an equivalent explosion of ideas about how to channel that power. One of the leading advocates of research into human-centered applications of computer technology was J.C.R Licklider, who became the director of the Information Processing Techniques Office of the US Department of Defense's Advanced Research Agency (ARPA). It was Licklider's goal to finance various research centers across the United States in order to encourage new ideas about how best to apply the burgeoning computing technology.

One of the major contributions to come out of this new emphasis in research was the concept of time-sharing, in which a single computer could support multiple users. Previously, the human was restricted to batch sessions, in which complete jobs were submitted on punched cards or paper tape to an operator who would then run them individually on the computer. Time-sharing systems of the 1960s made programming a truly interactive venture and brought about a subculture of programmers known as ‘hackers’ – single-minded masters of detail who took pleasure in understanding complexity. Though the purpose of the first interactive time-sharing systems was simply to augment the programming capabilities of the early hackers, it marked a significant stage in computer applications for human use.

Video display units

As early as the mid-1950s researchers were experimenting with the possibility of presenting and manipulating information from a computer in the form of images on a video display unit (VDU). These display screens could provide a more suitable medium than a paper printout for presenting vast quantities of strategic information for rapid assimilation. It was not until 1962, however, when a young graduate student at the Massachusetts Institute of Technology (MIT), Ivan Sutherland, astonished the established computer science community with the Sketchpad program, that the capabilities of visual images were realized.

Sketchpad demonstrated two important ideas. First, computers could be used for more than just data processing. They could extend the user’s ability to abstract away from some levels of detail, visualizing and manipulating different representations of the same information. Those abstractions did not have to be limited to representations in terms of bit sequences deep within the recesses of computer memory. Rather, the abstraction could be made truly visual. To enhance human interaction, the information within the computer was made more amenable to human consumption. The computer was made to speak a more human language, instead of the human being forced to speak more like a computer. Secondly, Sutherland’s efforts demonstrated how important the contribution of one creative mind could be to the entire history of computing.

Programming toolkits

Douglas Engelbart’s ambition since the early 1950s was to use computer technology as a means of complementing human problem-solving activity. Engelbart’s idea as a graduate student at the University of California at Berkeley was to use the computer to teach humans. This dream of naïve human users actually learning from a computer was a stark contrast to the prevailing attitude of his contemporaries that computers were purposely complex technology that only the intellectually privileged were capable of manipulating.

Personal computing

Programming toolkits provide a means for those with substantial computing skills to increase their productivity greatly. But Engelbart’s vision was not exclusive to the computer literate. The decade of the 1970s saw the emergence of computing power aimed at the masses, computer literate or not. One of the first demonstrations that the powerful tools of the hacker could be made accessible to the computer novice was a graphics programming language for children called LOGO. The inventor, Seymour Papert, wanted to develop a language that was easy for children to use. He and his colleagues from MIT and elsewhere designed a computer-controlled mechanical turtle that dragged a pen along

a surface to trace its path. In the early 1970s Alan Kay view of the future of computing was embodied in small, powerful machines, which were dedicated to single users, that is personal computers. Together with the founding team of researchers at the Xerox Palo Alto Research Center, Kay worked on incorporating a powerful and simple visually based programming environment, Smalltalk, for the personal computing hardware that was just becoming feasible. As technology progresses, it is now becoming more difficult to distinguish between what constitutes a personal computer, or workstation, and what constitutes a mainframe.

Window systems and the WIMP interface

With the advent and immense commercial success of personal computing, the emphasis for increasing the usability of computing technology focused on addressing the single user who engaged in a dialog with the computer in order to complete some work. Humans are able to think about more than one thing at a time, and in accomplishing some piece of work, they frequently interrupt their current train of thought to pursue some other related piece of work. A personal computer system which forces the user to progress in order through all of the tasks needed to achieve some objective, from beginning to end without any diversions, does not correspond to that standard working pattern. If the personal computer is to be an effective dialog partner, to must be as flexible in its ability to change the topic as the human is.

But the ability to address the needs of a different user task is not the only requirement. Computer systems for the most part react to stimuli provided by the user, so they are quite amenable to a wandering dialog initiated by the user. As the user engages in more than one plan of activity over a stretch of time, it becomes difficult for him to maintain the status of the overlapping threads of activity.

Interaction based on windows, icons, menus, and pointers—the WIMP interface—is now commonplace. These interaction devices first appeared in the commercial marketplace in April 1981, when Xerox Corporation introduced the 8010 Star Information System.

The metaphor

Metaphor is used quite successfully to teach new concepts in terms of ones, which are already understood. It is no surprise that this general teaching mechanism has been successful in introducing computer novices to relatively foreign interaction techniques. Metaphor is used to describe the functionality of many interaction widgets, such as windows, menus, buttons and palettes. Tremendous commercial successes in computing have arisen directly from a judicious choice of metaphor. The Xerox Alto and Star were the first workstations based on the metaphor of the office desktop. The majority of the management tasks on a standard workstation have to do with the file manipulation. Linking the set of tasks associated with file manipulation to the filing tasks in a typical office environment makes the actual computerized tasks easier to understand at first. The success of the desktop metaphor is unquestionable. Another good example in the personal computing domain is the widespread use of the spreadsheet for accounting and financial modeling.

Very few will debate the value of a good metaphor for increasing the initial familiarity between user and computer application. The danger of a metaphor is usually realized after the initial honeymoon period. When word processors were first introduced, they relied

heavily on the typewriter metaphor. The keyboard of a computer closely resembles that of a standard typewriter, so it seems like a good metaphor from any typewriter. For example, the space key on a typewriter is passive, producing nothing on the piece of paper and just moving the guide further along the current line. For a typewriter, a space is not a character. However, for a word processor, the blank space is a character, which much be inserted within a text just as any other character is inserted. So an experienced typist is not going to be able to predict his experience with a preliminary understanding of a word processor.

Another problem with a metaphor is the cultural bias that it portrays. With the growing internationalization of software, it should not be assumed that a metaphor will apply across national boundaries. A meaningless metaphor will only add another layer of complexity between the user and the system.

Direct Manipulation

In the early 1980s as the price of fast and high-quality graphics hardware was steadily decreasing, designers were beginning to see that their products were gaining popularity as their visual content increased. As long as the user-system command line prompt computing was going to stay within the minority population of the hackers who reveled in the challenge of complexity. In a standard command line interface, the only way to get any feedback on the results of previous interaction is to know that you only have to ask for it and to know how to ask for it. Rapid visual and audio feedback on a high-resolution display screen or through a high-quality sound system makes it possible to provide evaluative information for every executed user action.

Rapid feedback is just one feature of the interaction technique known as direct manipulation. Ben Shneiderman is attributed with coining this phrase in 1982 to describe the appeal of graphics-based interactive systems such as Sketchpad and the Xerox Alto and Star. He highlights the following features of a direct manipulation interface.

- Visibility of the objects of interest
- Incremental action at the interface with rapid feedback on all actions
- Reversibility of all actions, so that users are encouraged to explore without severe penalties
- Syntactic correctness of all actions, so that every user action is a legal operation
- Replacement of complex command language with actions to manipulate directly the visible objects.

The first real commercial success which demonstrated the inherent usability of direct manipulation interfaces for the general public was the Macintosh personal computer, introduced by Apple Computer, Inc. in 1984 after the relatively unsuccessful marketing attempt in the business community of the similar but more pricey Lisa computer. The direct manipulation interface for the desktop metaphor requires that the documents and folders are made visible to the user as icons, which represent the underlying files and directories. An operation such as moving a file from one directory to another is mirrored as an action on the visible document, which is picked and dragged along the desktop from one folder to the next.

Language versus action

Whereas it is true that direct manipulation interface make some tasks easier to perform correctly, it is equally true that some tasks are more difficult, if not impossible. Contrary to popular wisdom, it is not generally true that action speak louder than words. The image, projected for direct manipulation was of the interface as a replacement for the underlying system as the world of interest to the user. Actions performed at the interface replace any need to understand their meaning at any deeper, system level. Another image is of the interface as the interlocutor or mediator between the user and the system. The user gives the interface instructions and it is then the responsibility of the interface to see that those instructions are carried out. The user-system communication is by means of indirect language instead of direct actions.

We can attach two meaningful interpretations to this language paradigm. The first requires that the user understands how the underlying system functions and the interface as interlocutor need not perform much translation. In fact, this interpretation of the language paradigm is similar to the kind of interaction, which existed before direct manipulation interfaces were around. In a way, we have come full circle.

The second interpretation does not require the user to understand the underlying system's structure. The interface serve a more active role, as it must interpret between the intended operation as requested by the user and the possible system operations that must be invoked to satisfy that intent. Because it is more active, some people refer to the interface as an agent in these circumstances. This kind of language paradigm can be seen in some internal system database, but you would not know how that information is organized.

Whatever interpretation is attached to the language paradigm, it is clear that it has advantages and disadvantages when compared with the action paradigm implied by direct manipulation interfaces. In the action paradigm, it is often much easier to perform simple tasks without risk of certain classes or error. For example, recognizing and pointing to an object reduces the difficulty of identification and the possibility of misidentification. On the other hand, more complicated tasks are often rather tedious to perform in the action paradigm, as they require repeated execution of the same procedure with only minor modification. In the language paradigm, there is the possibility describing a generic procedure once and then leaving it to be executed without further user intervention.

The action and language paradigms need not be completely separate. In the above example two different paradigms are distinguished by saying that generic and repeatable procedures can be described in the language paradigm and not in the action paradigm. An interesting combination of the two occurs in programming by example when a user can perform some routine tasks in the action paradigm and the system records this as a generic procedure. In a sense, the system is interpreting the user's actions as a language script that it can then follow.

Hypertext

In 1945, Vannevar Bush, then the highest-ranking scientific administrator in the US war effort, published an article entitled 'As We May Think' in The Atlantic Monthly. Bush was in charge of over 6000 scientists who had greatly pushed back the frontiers of scientific knowledge during the Second World War. He recognized that a major drawback of these prolific research efforts was that it was becoming increasingly difficult to keep in

touch with the growing body of scientific knowledge in the literature. In his opinion, the greatest advantages of this scientific revolution were to be gained by those individuals who were able to keep abreast of an ever-increasing flow of information. To that end, he described an innovative and futuristic information storage and retrieval apparatus – the memex –, which was constructed with technology wholly existing in 1945 and aimed at increasing the human capacity to store and retrieve, connected pieces of knowledge by mimicking our ability to create random associative links.

An unsuccessful attempt to create a machine language equivalent of the memex on early 1960s computer hardware led Nelson on a lifelong quest to produce Xanadu, a potentially revolutionary worldwide publishing and information retrieval system based on the idea of interconnected, non-linear text and other media forms. A traditional paper is read from beginning to end, in a linear fashion. But within that text, there are often ideas or footnotes that urge the reader to digress into richer topic. The linear format for information does not provide much support for this random and associated browsing task. What Bush's memex suggested was to preserve the non-linear browsing structure in the actual documentation? Nelson coined the phrase hypertext in the mid 1960s to reflect this non-linear text structure.

Multi-modality

The majority of interactive systems still uses the traditional keyboard and a pointing device, such as a mouse, for input and is restricted to a color display screen with some sound capabilities for output. Each of these input and output devices can be considered as communication channels for the system and they correspond to certain human communication channels. A multi-modal interactive system is a system that relies on the use of multiple human communication channels. Each different channel for the user is referred to as a modality of interaction. In this sense, all interactive systems can be considered multi-modal, for human have always used their visual and haptic channels in manipulating a computer. In fact, we often use our audio channel to hear whether the computer is actually running properly.

However, genuine multi-modal systems rely to an extent on simultaneous use of multiple communication channels for both input and output. Humans quite naturally process information by simultaneous use of different channels.

Computer-supported cooperative work

Another development in computing in the 1960s was the establishment of the first computer networks, which allowed communication between separate machines. Personal computing was all about providing individuals with enough computing power so that they were liberated from dumb terminals, which operated on time-sharing systems. It is interesting to note that as computer networks become widespread, individuals retained their powerful workstations but now wanted to reconnect themselves to the rest of the workstations in their immediate working environment, and even throughout the world. One result of this reconnection was the emergence of collaboration between individuals via the computer – called computer – supported cooperative work, or CSCW.

The main distinction between CSCW systems and interactive systems designed for a single user is that designer can no longer neglect the society within which any single user operates. CSCW systems are built to allow interaction between humans via the computer

and so the needs of the many must be represented in the one product. A fine example of a CSCW system is electronic mail – email – yet another metaphor by which individuals at physically separate locations can communicate via electronic messages that work in a similar way to conventional postal systems.

The World Wide Web

Probably the most significant recent development interactive computing is the World Wide Web, often referred to as just the web, or WWW. The web is built on top of the Internet, and offers an easy to use, predominantly graphical interface to information, hiding the underlying complexities of transmission protocols, addresses and remote access to data.

The Internet is simply a collection of computers, each linked by any sort of data connections, whether it be slow telephone line and modem or high-bandwidth optical connection. The computers of the Internet all communicate using common data transmission protocols and addressing systems. This makes it possible for anyone to read anything from anywhere, in theory, if it conforms to the protocol. The web builds on this with its own layer of network protocol, a standard markup notation for laying out pages of information and a global naming scheme. Web pages can contain text, color images, movies, sound and, most important, hypertext links to other web pages. Hypermedia documents can therefore be published by anyone who has access to a computer connected to the Internet.

Ubiquitous computing

In the late 1980s, a group of researchers at Xerox PARC led by Mark Weiser, initiated a research program with the goal of moving human-computer interaction away from the desktop and out into our everyday lives. Weiser observed.

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. These words have inspired a new generation of researchers in the area of ubiquitous computing. Another popular term for this emerging paradigm is pervasive computing, first coined by IBM. The intention is to create a computing infrastructure that permeates our physical environment so much that we do not notice the computer may longer. A good analogy for the vision of ubiquitous computing is the electric motor. When the electric motor was first introduced, it was large, loud and very noticeable. Today, the average household contains so many electric motors that we hardly ever notice them anymore. Their utility led to ubiquity and, hence, invisibility.

Sensor-based and context-aware interaction

The yard-scale, foot-scale and inch-scale computers are all still clearly embodied devices with which we interact, whether or not we consider them ‘computers’. There are an increasing number of proposed and existing technologies that embed computation even deeper, but unobtrusively, into day-to-day life. Weiser’s dream was computers anymore’, and the term ubiquitous computing encompasses a wide range from mobile devices to more pervasive environments.

Lecture 16.

HCI Process and Models

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand the need of new software development models
- Understand the importance of user experience and usability in design

It has been said, “to err is human; to really screw up, you need a computer.”

Inefficient mechanical systems can waste couple cents on every widget you build, but you can lose your entire company to bad information processes. The leverage that software-based products---and the engineers that build them---have on your company is enormous. Sadly, our digital tools are extremely hard to learn, use, and understand, and they often cause us to fall short of our goals. This wastes money, time, and opportunity. As a business-savvy technologist/ technology-savvy businessperson, you produce software-based products or consume them---probably both. Having better, easier-to-learn, easier-to-use high-tech products is in your personal and professional best interest. Better products don’t take longer to create, nor do they cost more to build. The irony is that they don’t have to be difficult, but are so only because our process for making them is old-fashioned and needs fixing. Only long-standing traditions rooted in misconceptions keep us from having better products in today.

Consider a scenario: a website is developed of commerce system. The site is aesthetically very beautiful, technically it has no flaw and it has wonderful animated content on it. But if user is unable to find its desired information about the products or even he is unable to find the product out of thousands of products, so what of it’s use. It is useless from the business point of view.

Here are some facts and figures:

Users can only find information 42% of the time

– Jared Spool

62% of web shoppers give up looking for the item they want to buy online

– Zona Research

50% of the potential sales from a site are lost because people cannot find the item they are looking for

– Forrester Research

40% of the users who do not return to a site do so because their first visit resulted in a negative experience

– Forrester Research

80% of software lifecycle costs occur after the product is released, in the maintenance phase - of that work, 80 % is due to unmet or unforeseen user requirements; only 20 % is due to bugs or reliability problems.

- IEEE Software

Around 63% of software projects exceed their cost estimates. The top four reasons for this are:

- Frequent requests for changes from users
- Overlooked tasks
- Users' lack of understanding of their own requirements
- Insufficient user-analyst communication and understanding

- Communications of the ACM

BOO.com, a \$204m startup fails

– BBC News

Poor commercial web sites will kill 80% of Fortune 500 companies within a decade

Jakob Nielsen

So all above given facts reveals that the product with the bad user experience deserve to die!

	Scenario A	Scenario B
Revenue Potential	\$100m	\$100m
User Experience	Good	Bad
Sales Lost	0%	50%
Revenue Lost	\$0m	\$50m
Actual Revenue	\$100m	\$50m

All you can see that the real importance is of good interaction design. In above example you saw that whole the systems which were developed technically very well but failed just because of bad user experience, bad usability.

So, what if we design a system with good usability and good user experience. The result will be satisfaction and happiness. People will be happy with your product and they will buy your product, not only they will buy but also they will recommend others for your

product. And other people will also buy your product and this will result a chain reaction. So, as you saw how your product can be successful if it provides a good user experience. Our discussion has a simple premise: if achieving the user's goal is the basis of our design process, the user will be satisfied and happy. If the user is happy, he will gladly pay us money (and recommend that others do the same), and then we will be successful as a business.

On the surface, this premise sounds quite obvious and straightforward: make the user happy, and your products will be a success. Why then are so many digital products so difficult and unpleasant to use? Why aren't we all either happy or successful—or both? Most digital products today emerge from the development process like a monster emerging from a bubbling tank. Developers, instead of planning and executing with their users in mind, end up creating technological solutions over which they ultimately have little control. Like a mad scientists, they fail because they have not imbued their creations with humanity. We are mainly concerned with software, so, as we have stated earlier the definition of software quality:

The extent to which a software product exhibits these characteristics

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

As you can see there are many characteristics to measure the quality of software and the usability is one of them. In fact, we can understand the nature of usability in two aspects

- Strategic
- Tactical

Strategic aspect guides us to think about user interface idioms – in other words, the way in which the user and the idiom interact. Tactical aspects give us hints and tips about using and creating user interface idioms, like dialog boxes and pushbuttons.

Integrating the strategic and tactical approaches is the key to designing effective user interaction and interface. For example, there is no such thing as an objectively good dialog box—the quality depends on the situation: who the user is and what his background and goals are. Merely applying a set of tactical dictums makes user interface creation easier, but it doesn't make the end result better. Deep thoughts about how users should interact with your system won't improve the software, either. What does work is maintaining a strategic sensitivity for how users actually interact with specific software and having at your command a tactical toolbox to apply in any particular situation.

Ignorance about users

It is a sad truth that the digital technology industry doesn't have a good understanding of what it takes to make users happy. In fact, most technology products get built without much understanding of the users. We might know what market segment our users are in., how much money they make, how much money they like to spend on weekends, and what sort of cars they buy. May be we even having a vague idea what kind of jobs they have and some of the major tasks that they regularly perform? But does any of this tell us how to make them happy? Does it tell us how they will actually use the product we are building? Does it tell us why they are doing whatever it is they might need our product for, why they might want to choose our product over our competitors, or how we can make sure they do? Unfortunately, it does not.

Conflict of Interest

A second problem affects the ability of vendors and manufacturers to make users happy. There is an important conflict of interest in the world of digital product development: the people who build the products—programmers—are usually also the people who design them. Programmers are often required to choose between ease of coding and ease of use. Because programmers' performance is typically judged by their ability to code efficiently and meet incredibly tight deadlines, it isn't difficult to figure out what direction most software-enabled products take. Just as we would never permit the prosecutor in a legal trial to also adjudicate the case, we should make sure that those designing a product are not the same people building it. It simply isn't possible for a programmer to advocate for the user, the business, and the technology at the same time.

How can you achieve success?

The third reason that digital technology industry isn't cranking out successful products is that it has no reliable process for doing so. Or, to be more accurate, it doesn't have a complete process for doing so. Engineering departments follow—or should follow—rigorous engineering methods that ensure the feasibility and quality of the technology. Similarly, marketing, sales, and other business units follow their own well-established methods for ensuring the commercial viability of new products. What's left out is a repeatable, analytical process for transforming and understanding of users into products that both meet their needs and excite their imaginations.

While thinking about complex mechanical devices, we take for granted that they have been carefully designed for use, in addition to being engineered. Most manufactured objects are quite simple, and even complex mechanical products are quite simple when compared to most software and software-enabled products that can sport in excess of one million lines of code. Yet more software have never been undergone a rigorous design process from a user-centered perspective. The current process of determining what software will do and how it will communicate with the user is today closely intertwined with its construction. Programmers, deep in their thoughts of algorithms and code, “design” user interface the same way that miners “design” the landscape with their cavernous pits and enormous tailing piles. The software interface design process alternates between the accidental and the non-existent.

Many programmers today embrace the notion that integrating users directly into the programming process on a frequent basis—weekly or sometimes even daily—can solve design problems. Although this has the salutary effect of sharing the responsibility for

design with the user, it ignores a serious methodological flaw: a confusion of domain knowledge with design knowledge. Users although they might be able to articulate the problems with an interaction, are not often capable of visualizing the solutions to those problems. Design is a specialized skill, just like programming. Programmer would never ask users to help them code; design problems should be treated no differently.

Evolution of Software Development Process

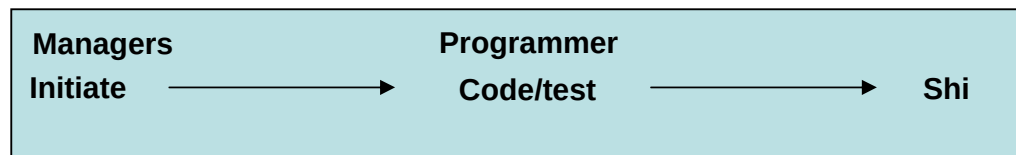
Originally programmers did it all

In the early days of software industry smart programmers dreamed up useful software, wrote and even tested it on their own. But as their businesses grew, the software business and software business and software products became more complicated.



Managers brought order

Inevitably, professional managers were brought in. Good product managers understand the market and competitors. They define software product by creating requirements documents. Often, however, requirements are little more than a list of features and managers find themselves having to give up features in order to meet schedule.

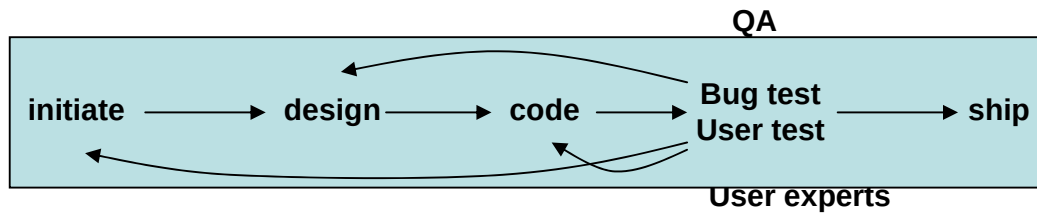


Testing and design became separate steps

As the industry matured, testing became a separate discipline and separate step in the process. In the move from command-line to graphical user interface, design and usability also became involved in the process, though often only at the end, and often only affecting visual presentation. Today common practice includes simultaneous coding and design followed by bug and user testing and then revision.

Design must precede the programming effort

A goal directed design approach to software development means that all decisions proceed from a format definition of the user and his or her goals. Definition of the user and user goals is the responsibility of the designer, thus design must precede programming.



Design

Design, according to industrial designer Victor Papanek, is the conscious and intuitive effort to impose meaningful order. Cooper proposes a somewhat detailed definition:

- Understanding the user's wants, needs, motivations, and contexts
- Understanding business, technical, and domain requirements and constraints
- Translating this knowledge into plans for artifacts whose form content, and behavior is useful, usable and desirable, as well as economically viable and technically feasible.

This definition applies across all design disciplines, although the precise focus on form versus content versus behavior varies by design discipline.

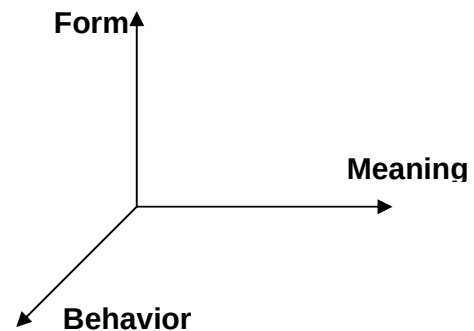
When performed using the appropriate methods, design can provide the missing human connection in technological product. But clearly, the currently approach to the design of digital products either isn't working or isn't happening as advertised.

Three dimensions of designs

Interaction design focuses on an area that traditional design disciplines do not often explore: the design of behavior.

All design affects human behavior: architecture is about how people use spaces as much as it is about form and light. And what would be the point of a poster if no one acted on the information it presented?

However, it is only with the introduction of interactive technologies – courtesy of the computer – that the design of the behavior of artifacts and how this behavior affects and



supports human goals and desires, has become a discipline worthy of attention.

One way of making sense of the difference is focus between interaction design and more traditional design is through a historical lens. In the first half of the twentieth century, designers focused primarily on form.

Later designers became increasingly concerned with meaning; for example, product designers and architects introduced vernacular and retro forms in the 70s. The trend continues today with retro-styled automobiles such as the PT Cruiser. Today, information designers continue the focus on meaning to include the design of usable content.

Within the last fifteen years a growing group of designers have begun to talk about behavior: the dynamic ways that software-enabled products interact directly with users. These concerns (form, meaning and behavior) are not exclusive. Interactive products must have each in some measure; software applications focus more on behavior and form, with less demand on content; web sites and kiosks focus more on content and form, with less sophisticated behaviors.

Definition interaction design

Simply put, interaction design is the definition and design of the behavior of artifacts, environment, and systems, as well as the formal elements that communicate that behavior. Unlike, traditional design disciplines, whose focus has historically been on form and, more recently, on content and meaning, interaction design seeks first to plan and describe how things behave and then, as necessary, to describe the most effective form to communicate those behaviors.

Lecture 17.

HCI Process and Methodologies

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Describe the advantages and disadvantages of different Software Development Lifecycles
- In our last lecture we ended with the definition of design, today we will look at different design models used for development of software and we also consider their flaws. We start from where we left in last lecture, design.

The term design, even used in the context of designing a computer system, can have different meanings. In a review of general design philosophy, Jones (1981) found a number of definitions including:

‘Finding the right physical components of a physical structure.’

‘A goal-directed problem-solving activity.’

Simulating what we want to make (or do) before we make (or do) it as many times as may be necessary to feel confident in the final result.’

‘The imaginative jump from present facts to future possibilities.’

‘A creative activity – it involves bringing into being something new and useful that has not existed previously.’

These descriptions focus on the process of design in general. On the topic of engineering design, Jones found this: ‘Engineering design is the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform pre-specified functions with the maximum economy and efficiency.’ Webster, on the other hand, stresses the relationship between design representation and design process: ‘A design is an information base that describes elaborations of representations, such as adding more information or even backtracking and exploring alternatives’ (Webster, 1988).

Thus, ‘design’ refers to both the process of developing a product, artifact or system and to the various representations (simulations or models) of the product that are many produced during the design process. There are many representations, which are more or less appropriate in different circumstances. Designers need not only to be able to understand user’ requirements, but also to be able to represent this understanding in different ways at different stages of the design. Selecting suitable representations is, therefore, important for exploring, testing, recording and communicating design ideas and decisions, both within the design team and with users.

In order to develop any product, two major activities have to be undertaken: the designer must understand the requirements of the product, and must develop the product.

Understanding requirements involves looking at similar products, discussing the needs of the people who will use the product, and analyzing any existing systems to discover the problems with current designs. Development may include producing a variety of representations until a suitable artifact is produced.

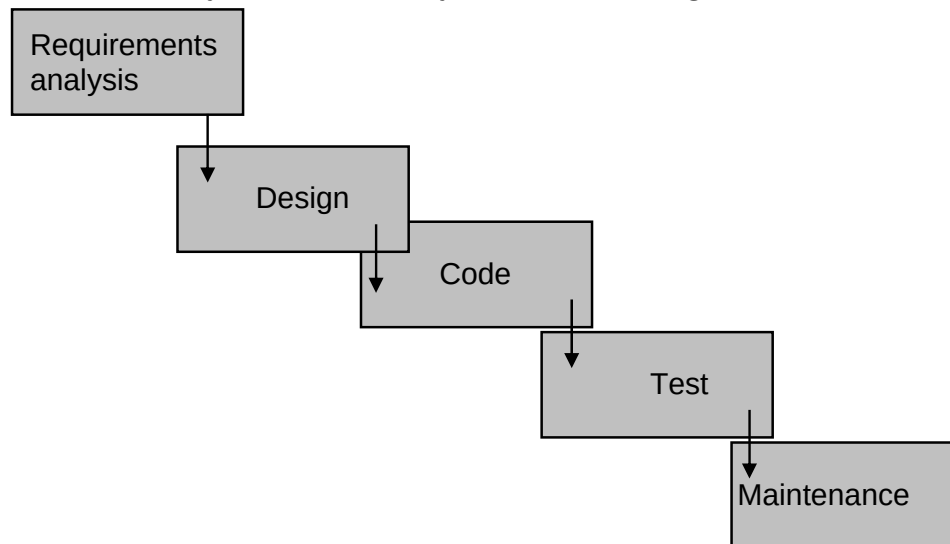
17.1 Lifecycle models

Understanding what activities are involved in interaction design is the first step to being able to do it, but it is also important to consider how the activities are related to one another so that the full development process can be seen. The term lifecycle model is used to represent a model that captures a set of activities and how they are related. Sophisticated models also incorporate a description of when and how to move from one activity to the next and a description of the deliverables for each activity. The reason such models are popular is that they allow developers, and particularly managers, to get an overall view of the development effort so that progress can be tracked, deliverables specified, resources allocated, targets set, and so on.

Existing models have varying levels of sophistication and complexity. For projects involving only a few experienced developers, a simple process would probably be adequate. However, for larger systems involving tens or hundreds of developers with hundreds or thousands of users, a simple process just isn't enough to provide the management structure and discipline necessary to engineer a usable product. So something is needed that will provide more formality and more discipline. Note that this does not mean that innovation is lost or that creativity is stifled. It just means that structured process is used to provide a more stable framework for creativity.

The design of software systems

The traditional view of software engineering characterizes the development of software as consisting of number of processes and representations that are produced in an essentially linear fashion. This is often called waterfall model, because the output of each process 'tumbles down' neatly to the next activity, as illustrated in Figure.



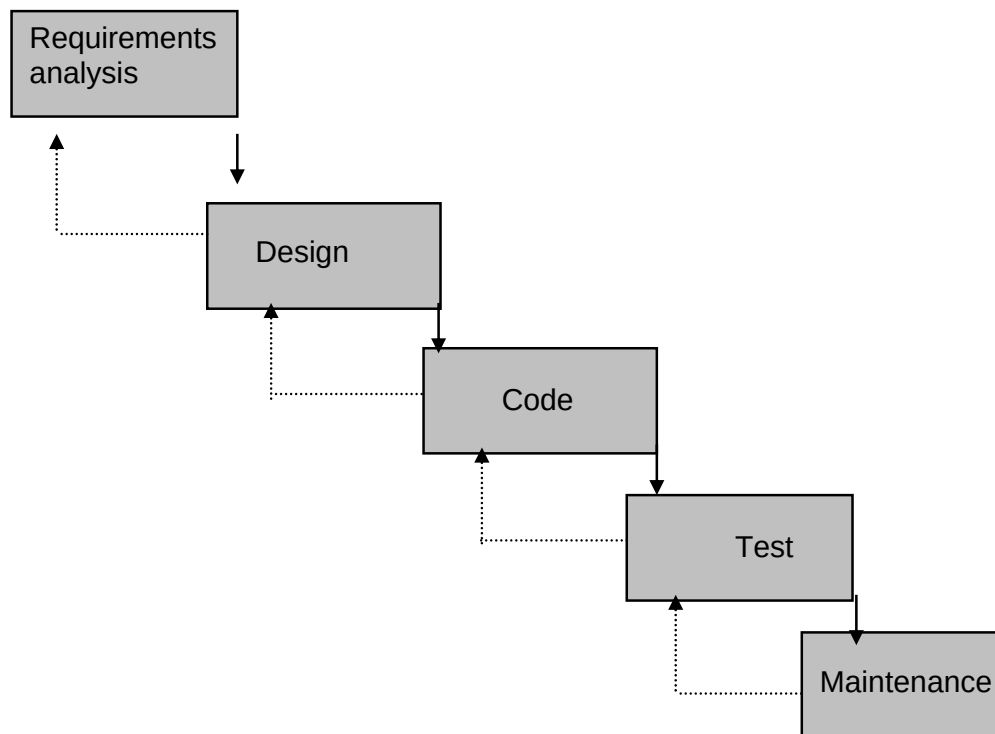
The waterfall lifecycle was the first model generally known in software engineering and forms the basis of many lifecycle in use today. This is basically a linear model in which each step must be completed before the next step can be started. For example,

requirements analysis has to be completed before design begin. The name given to these steps varies, as does the precise definition of each one, but basically, the lifecycle starts with some requirements analysis, moves into design, then coding, then implementation, testing, and finally maintenance.

Flaws of waterfall model

One of the main flaws with this approach is that requirements change over time, as businesses and the environment in which they operate change rapidly. This means that it does not make sense to freeze requirements for months, or maybe years. While design and implementation are completed.

Some feedback to earlier stages was acknowledged as desirable and indeed practical soon after this lifecycle became widely used as shown in figure below. But the idea of iteration was not embedded in the waterfall's philosophy, and review sessions among developers are commonplace. However, the opportunity to review and evaluation with users was not built into this model.

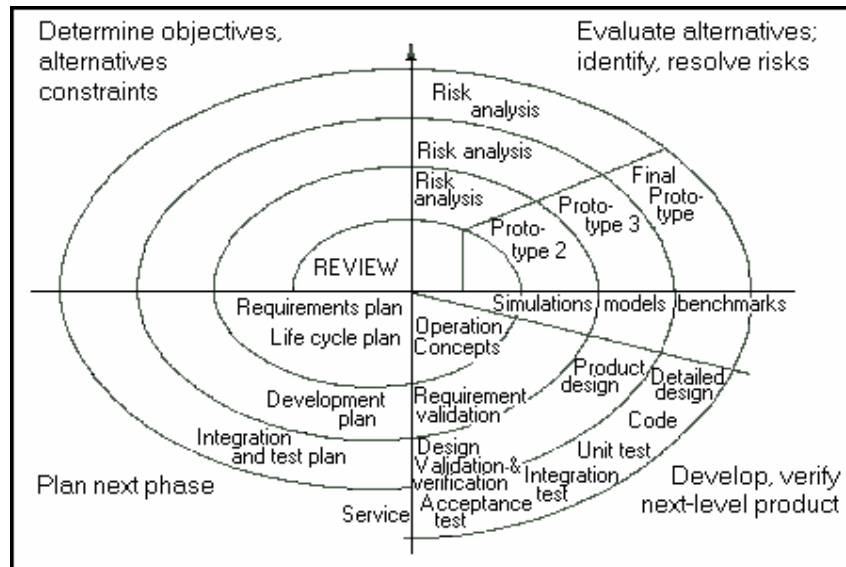


The spiral lifecycle model

For many years, the waterfall formed the basis of most software developments, but in 1988 Barry Boehm suggested the spiral model of software development. Two features of the spiral model are immediately clear from figure: risk analysis and prototyping.

The spiral model incorporates them in an iterative framework that allows ideas and progress to be repeatedly checked and evaluated. Each iteration around the spiral may be based on a different lifecycle model and may have different activities.

In the spiral's case, it was not the need for user involvement that insured the introduction of iteration but the need to identify and control risks. In Boehm's approach, development plans and specifications that are focused on the risks involved in developing the system drive development rather than the intended functionality, as was the case with the waterfall. Unlike the waterfall, the spiral explicitly encourages alternatives to be considered, and steps in which problems or potential problems are encountered to be re-addressed.



The spiral idea has been used by others for interactive devices. A more recent version of the spiral, called the Win Win spiral model, explicitly incorporates the identification of key stakeholders and their respective “win” conditions, i.e., what will be regarded as a satisfactory outcome for each stakeholder group. A period of stakeholder negotiation to ensure a “win-win” result is included.

Rapid Application Development (RAD)

During the 1990s the drive to focus upon users became stronger and resulted in a number of new approaches to development. The Rapid Application Development (RAD) approach attempts to take a user-centered view and to minimize the risk caused by requirements changing during the course of the project. The ideas behind RAD began to emerge in the early 1990s, also in response to the inappropriate nature of the linear lifecycle models based on the waterfall. Two key features of a RAD project are:

- Time-limited cycles of approximately six months, at the end of which a system or partial system must be delivered. This is called time-boxing. In effect, this breaks down a large project into many smaller projects that can deliver product incrementally, and enhance flexibility in terms of the development techniques used and the maintainability of the final system.
- JAD (Joint Application Development) workshops in which users and developers come together to thrash out the requirements of the system. These are intensive requirements-gathering sessions in which difficult issues are faced and decisions

are made. Representatives from each identified stakeholder group should be involved in each workshop so that all the relevant views can be heard.

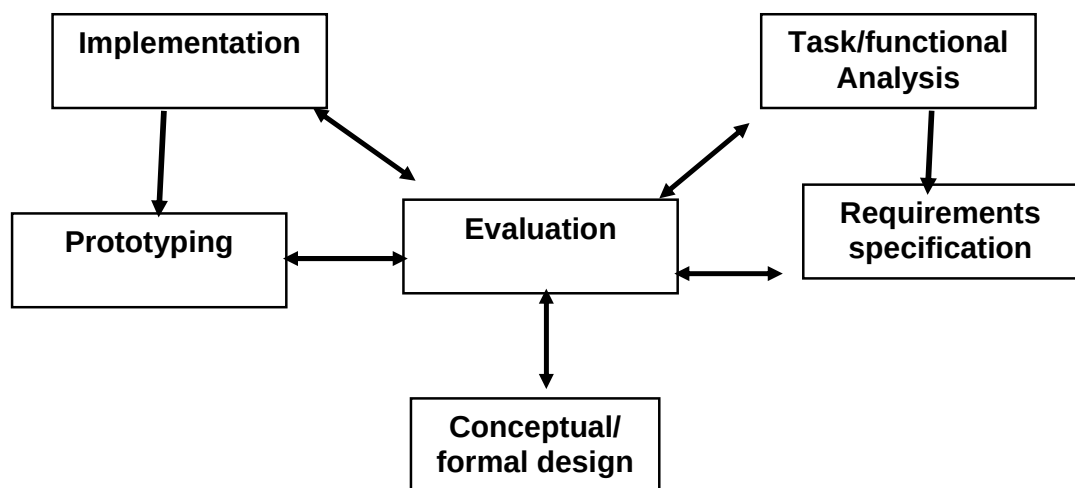
A basic RAD lifecycle has five phases (as shown in figure): project set-up, JAD workshops, iterative design and build, engineer and test final prototype, implementation review. The popularity of RAD has led to the emergence of an industry-standard RAD-based method called DSDM (Dynamic Systems Development Method). This was developed by a non-profit-making DSDM consortium made up of a group of companies that recognized the need for some standardization in the field. The first of nine principles stated as underlying DSDM is that “active user involvement is imperative.” The DSDM lifecycle is more complicated than the one which shown here. It involves five phases: feasibility study, business study, functional model iteration, design and build iteration, and implementation. This is only a generic process and must be tailored for a particular organization.

17.2 Lifecycle models in HCI

Another of the traditions from which interaction design has emerged is the field of HCI. Fewer lifecycle models have arisen from this field than from software engineering and, as you would expect, they have a stronger tradition of user focus. We will discuss two of them in our lecture. The first one, the Star, was derived from empirical work on understanding how designers tackled HCI design problems. This represents a very flexible process with evaluation at its core. In contrast, the second one, the usability engineering lifecycle, shows a more structured approach and hails from the usability engineering tradition.

The Star Lifecycle model

About the same time that those involved in software engineering were looking for alternatives to the waterfall lifecycle, so too were people involved in HCI looking for alternative ways to support the design of interfaces. In 1989, the Star lifecycle model was proposed by Hartson and Hix as shown in figure.

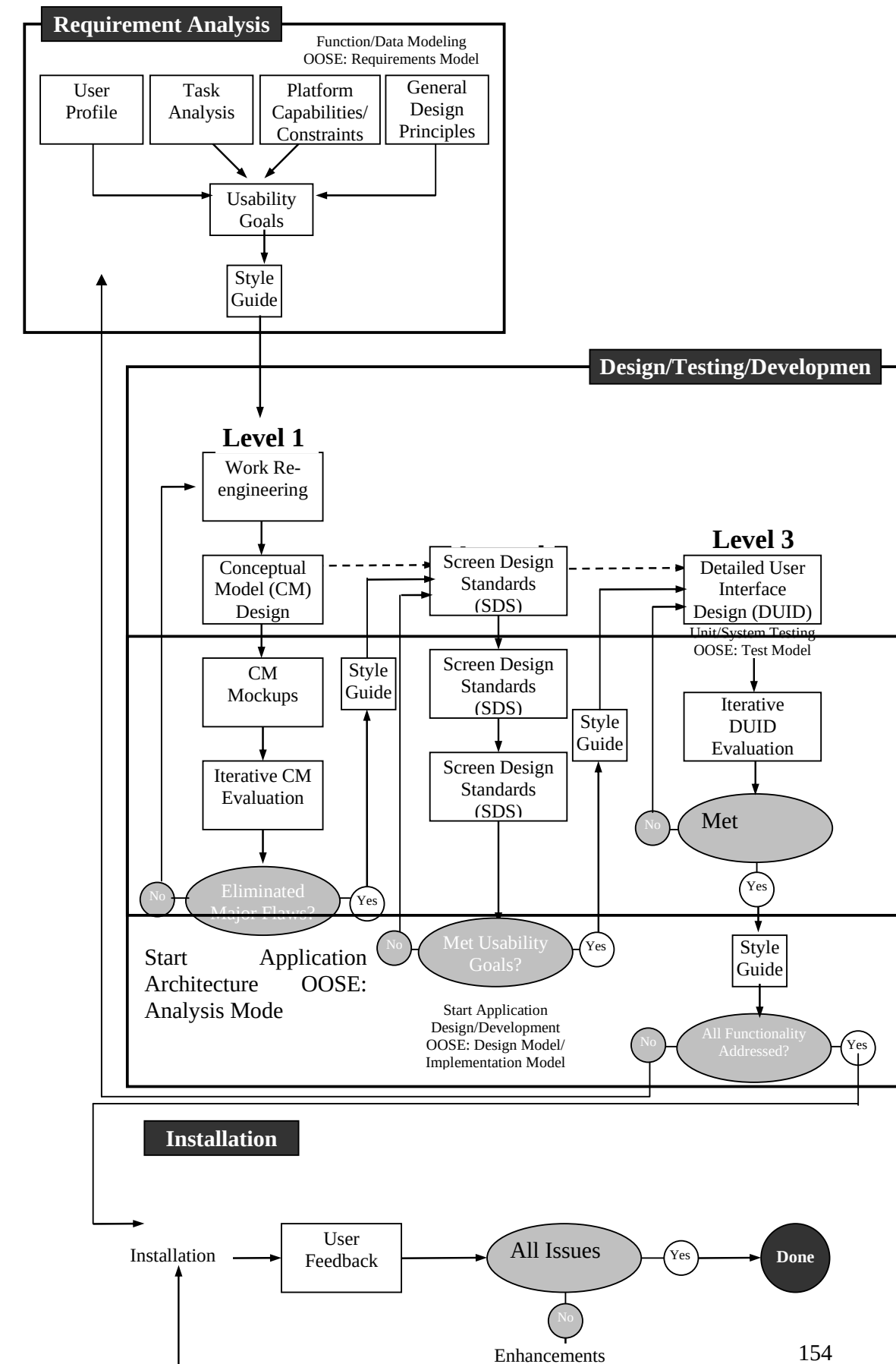


This emerged from some empirical work they did looking at how interface designers went about their work. They identified two different modes of activity: analytic mode and synthetic mode. The former is characterized by such notions as top-down, organizing, judicial, and formal, working from the systems view towards the user's view; the latter is characterized by such notions as bottom-up, free-thinking, creative and adhoc, working from the user's view towards the systems view. Interface designers served in software designers.

Unlike the lifecycle models introduced above, the Star lifecycle does not specify any ordering of activities. In fact, the activities are highly interconnected: you can move from any activity to any other, provided you first go through the evaluation activity. This reflects the findings of the empirical studies. Evaluation is central to this model, and whenever an activity is completed, its result(s) must be evaluated. So a project may start with requirements gathering, or it may start with evaluating an existing situation, or by analyzing existing tasks, and so on.

The Usability engineering lifecycle

The Usability Engineering lifecycle was proposed by Deborah Mayhew in 1999. Many people have written about usability engineering, and as Mayhew herself says, "I did not invent the concept of a Usability Engineering Lifecycle. Nor did I invent any of the Usability Engineering tasks included in lifecycle....".



However, what her lifecycle does provide is a holistic view of usability engineering and a detailed description of how to perform usability tasks, and it specifies how usability tasks can be integrated into traditional software development lifecycle. It is therefore particularly helpful for those with little or no expertise in usability to see how the tasks may be performed alongside more traditional software engineering activities. For example, Mayhew has linked the stages with an oriented software engineering that have arisen from software engineering. The lifecycle itself has essentially three tasks: requirements analysis, design/testing/development, and installation, with the middle stage being the largest and involving many subtasks as shown in figure. Note the production of a set of usability goals in the first task. Mayhew suggests that these goals be captured in a style guide that is then used throughout the project to help ensure that the usability goals are adhered to.

This lifecycle follows a similar thread to our interaction design models but includes considerably more detail. It includes stages of identifying requirements, designing, evaluating, and building prototypes. It also explicitly includes the style guide as a mechanism for capturing and disseminating the usability goals of the project. Recognizing that some projects will not require the level of structure presented in the full lifecycle, Mayhew suggests that some sub steps can be skipped if they are unnecessarily complex for the system being developed.

The Goal-Directed Design Process

Most technology-focused companies don't have an adequate process or user-centered design, if they have a process at all. But even the more enlightened organizations, ones that can boast of an established process, come up against some critical issues that result from traditional ways of approaching the problems of research and design.

In recent years, the business community has come to recognize that user research is necessary to create good products, but the proper nature of that research is still in question in many organizations. Quantitative market research and market segmentation is quite useful for selling products, but falls short of providing critical information about how people actually use products—especially products with complex behaviors. A second problem occurs after the results have been analyzed: most traditional methods don't provide a means of translating research results into design solutions. A hundred pages of user survey data don't easily translate into a set of product requirements, and they say even less about how those requirements should be expressed in terms of a logical and appropriate interface structure. Design remains a black box: "a miracle happens here." This gap between research results and the ultimate design solution is the result of a process that doesn't connect the dots from user to final product.

Bridging the gap

As it is already discussed that the role of design in the development process needs to be change. We need to start thinking about design in new ways, and start thinking differently about how product decisions are made.

Design as product definition

Design has, unfortunately, become a limiting term in the technology industry. Design, when properly deployed identifies user requirements and defines a detailed plan for the

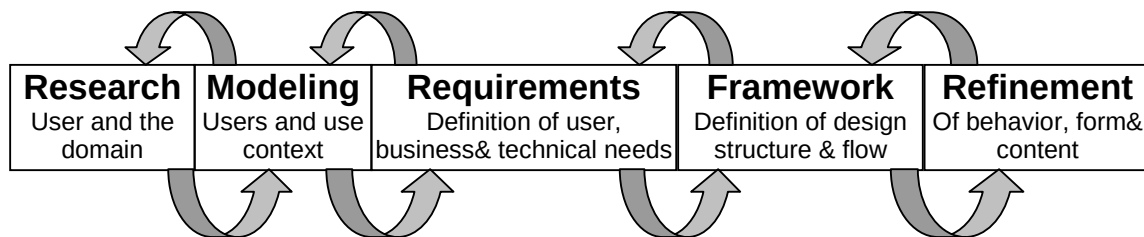
behavior and appearance of products. In other words, design provides true product definition, based on goals of users, needs of business, and constraints of technology.

Designers as researchers

If design is to become product definition, designers need to take on broader roles than that assumed in traditional design, particularly when the object of this design is complex, interactive systems. One of the problems with the current development process is that roles in the process are overspecialized: Researchers perform research, and designers perform design. The results of user and market research are analyzed by the usability and market researchers and then thrown over the transom to designers or programmers. What is missing in this model is a systematic means of translating and synthesizing the research into design solutions. One of the ways to address this problem is for designers to learn to be researchers. There is a compelling reason for involving designers in the research process. One of the most powerful tools designers bring to the table is empathy: the ability to feel what others are feeling. The direct and extensive exposure to users that proper user research entails immerses designers in the users' world, and gets them thinking about users long before they propose solutions. One of the most dangerous practices in product development is isolating designers from the users because doing so eliminates empathic knowledge. Additionally, it is often difficult for pure researchers to know what user information is really important from a design perspective. Involving designers directly in research addresses both. Although research practiced by designers takes us part of the way to Goal-Directed Design solutions, there is still a translation gap between research results and design details.

Between research and design: models, requirements, and frameworks

Few design methods in common use today incorporate a means of effectively and systematically translating the knowledge gathered during research into a detailed design specification. Part of the reason for this has already been identified. Designers have historically been out of the research loop and have had to rely on third-person accounts of user behaviors and desires.



The other reason, however, is that few methods capture user behaviors in a manner that appropriately directs the definition of a product. Rather than providing information about user goals, most methods provide information at the task level. This type of information is useful for defining layout, workflow, and translation of functions into interface controls, but less useful for defining the basic framework of what a product is, what it does, and how it should meet the broad needs of the user. Instead we need explicit, systematic processes for defining user models, establishing design requirements, and translating those into a high level interaction framework.

Goal-Directed Design seeks to bridge the gap that currently exists in the digital product development process, the gap between user research and design, through a combination of new techniques and known methods brought together in more effective ways.

Lecture 18.

Goal-Directed Design Methodologies

Learning Goals

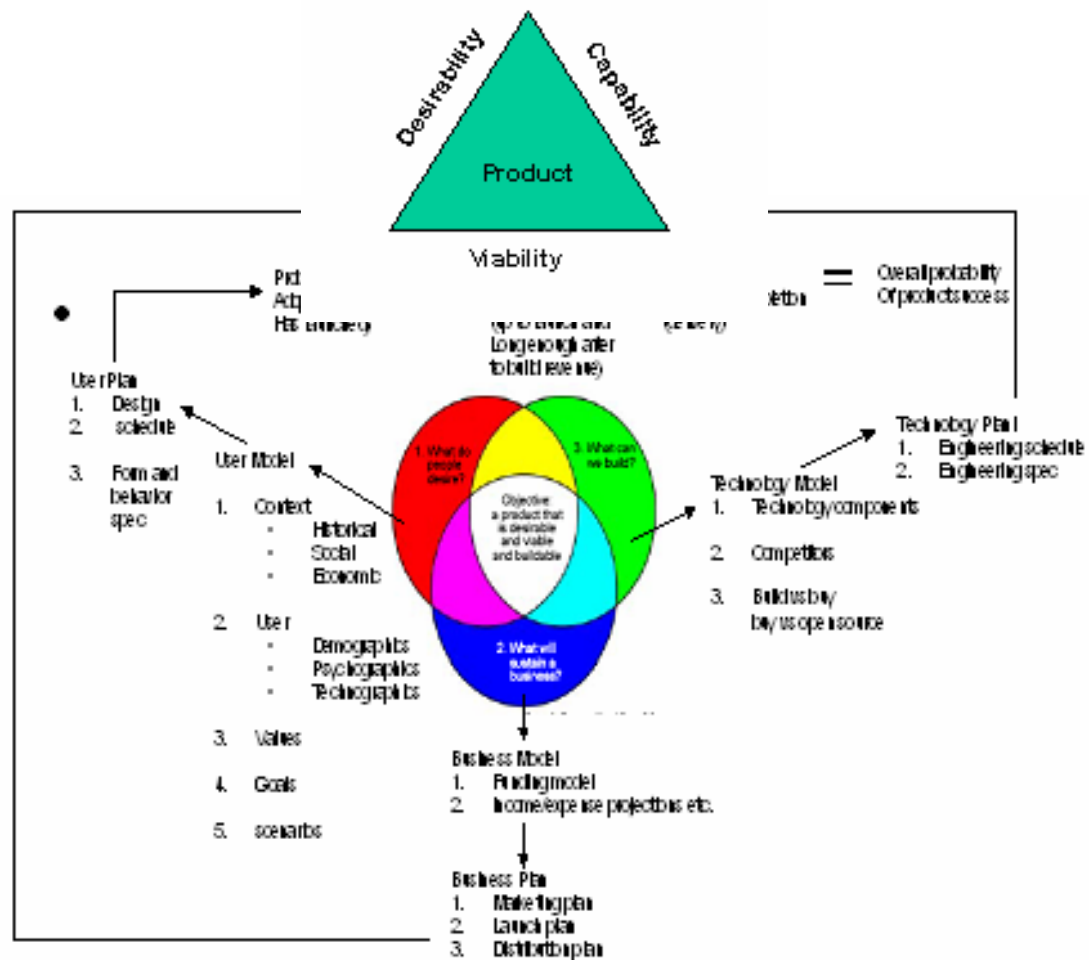
As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

Understand different phases of Goal-Directed Design Approach

18.1 Goal-Directed Design Model

Underlying the goal-directed approach to design is the premise that product must balance business and engineering concerns with user concerns.

You begin by asking, “What do people desire?” then you ask, “of the things people desire, what will sustain a business.” And finally you ask, “Of the things people desire, that will also sustain the business, what can we build?” a common trap is to focus on technology while losing the sight of viability and desirability.



Understanding the importance of each dimension is only the beginning, which understanding must also be acted upon. We are familiar with this process along the business and technology dimension; you create a business model and then develop a business plan, and similarly create an engineering model and specification. The goal-directed design process is an analog to these planning processes. It results in a solid user model and a comprehensive interaction plan.

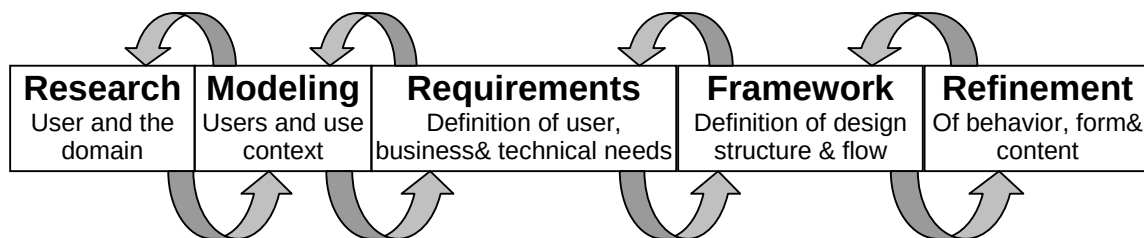
The user plan determines the probability that customer will adopt a product. The business plan determines the probability that business can sustain itself up to and through launch—and that sale will actually support growth thereafter. And technology plan determines the probability that the product can be made to work and actually deliverable.

Multiplying these three factors determines the overall probability that a product will be successful.

18.2 A process overview

Goal-Directed Design combines techniques of ethnography, stakeholder interviews, market research, product/literature reviews, detailed user model, scenario-based design, and a core set of interaction principles and patterns. It provides solutions that meet the needs and goals of users, while also addressing business/organizational and technical imperatives. This process can be roughly divided into five phases:

- Research
- Modeling
- Requirements Definition
- Framework Definition
- Refinement



These phases follow the five component activities of interaction design identified by Gillian Crampton Smith and Philip Tabor—understanding, abstracting, structuring, representing, and detailing—with a greater emphasis on modeling user behaviors and defining system behaviors.

Research

The research phase employs ethnographic field study techniques (observation and contextual interviews) to provide qualitative data about potential and/or actual users of the product. It also includes competitive product audits, reviews of market research and technology white papers, as well as one-on-one interviews with stakeholders, developers, subject matter experts (SMEs), and technology experts as suits the particular domain.

One of the principles out comes of field observation and user interviews are an emergent set of usage patterns—identifiable behaviors that help categorize modes of use of a

potential or existing product. These patterns suggest goals and motivations (specific and general desired outcomes of using the product). In business and technical domains, these behavior patterns tend to map to professional roles; for consumer product, they tend to correspond to lifestyle choices. Usage patterns and the goals associated with them drive the creation of personas in the modeling phase. Market search helps select and filter for valid persons that fit corporate business models. Stakeholder interviews, literature reviews, and product audits deepen the designers' understanding of the domain and elucidate business goals and technical constraints that the design must support.

Modeling

During the modeling phase, usage and workflow patterns discovered through analysis of the field research and interviews are synthesized into domain and user models. Domain models can include information flow and workflow diagrams. User models, or personas, are detailed composite user archetypes that represent distinct groupings of behavior patterns, goals, and motivations observed and identified during the research phase.

Personas serves as the main characters in a narrative scenario-based approach to design that iterative generates design concepts in the framework definition phase, provides feedback that enforces design coherence and appropriateness in the refinement phase, and represents a powerful communication tool that helps developers and managers to understand design rationale and to prioritize features based on user needs. In the modeling phase, designers employ a variety of methodological tools to synthesize, differentiate, and prioritize personas, exploring different types of goals and mapping personas across ranges of behavior to ensure there are no gaps or duplications.

Specific design targets are chosen from the cast of personas through a process of comparing goals and assigning a hierarchy of priority based on how broadly each persona's goals encompass the goals of other personas. A process of designating persona types determines the amount of influence each persona has on the eventual form and behavior of the design.

Possible user persona type designations include:

- Primary: the persona's needs are sufficiently unique to require a distinct interface form and behavior
- Secondary: primary interface serves the needs of the persona with a minor modification or addition
- Supplement: the persona's needs are fully satisfied by a primary interface
- Served: the persona is not an actual user of the product, but is indirectly affected by it and its use
- Negative: the persona is created as an explicit, rhetorical example of whom not to design for

Requirements definition

Design methods employed by teams during the requirements definition phase provides the much-needed connection between user and other models and the framework of the design. This phase employs scenario-based design methods, with the important innovation of focusing the scenarios not on user tasks in the abstract, but first and foremost of meeting the goals and needs of specific user personas. Personas provide and

understanding of which tasks are truly important and why, leading to an interface that minimize necessary tasks while maximizing return. Personas become the main characters of these scenarios, and the designers explore the design space via a form of role-playing. For each interface/primary persona the process of design in the requirements definition phase involves an analysis of persona data and functional needs, prioritized and informed by persona goals, behaviors, and interactions with other personas in various contexts.

This analysis is accomplished through an iteratively refined context scenario that start with a “day in the life” of the persona using the product, describing high-level product touch points, and thereafter successively defining detail at ever-deepening levels. As this iteration occurs, both business goals and technical constraints are also considered and balanced with personas goals and needs. The output of this process is a requirements definition that balances user, business, and technical requirements of the design to follow.

Framework definition

In the framework definition phase, teams synthesize an interaction framework by employing two other critical methodological tools in conjunction with context scenarios. The first is a set of general interaction design principles that, like their visual design counterparts, provide guidance in determining appropriate system behavior in a variety of contexts

The second critical methodological tool is a set of interaction design patterns that encode general solutions (with variations dependent on context) to classes of previously analyzed problems. These patterns bear close resemblance to the concept of architectural design patterns developed by Christopher Alexander. Interaction design patterns are hierarchically organized and continuously evolve as new contexts arise. Rather than stifling designer creativity, they often provide needed leverage to approach difficult problems with proven design knowledge.

After data and functional needs are described at this high level, they are translated into design elements according to interaction principles and then organized using patterns and principles into design sketches and behavior descriptions. The output of this process is an interaction framework definition, a stable design concept that provides the logical and gross formal structure for the detail to come. Successive iterations of more narrowly focused scenarios provide this detail in the refinement phase. The approach is often a balance of top-down design and bottom-up design.

Refinement

The refinement phase proceeds similarly to the framework definition phase, but with greater focus on task coherence, using key path and validation scenarios focused on storyboarding paths through the interface in high detail. The culmination of the refinement phase is the detailed documentation of the design, a form and behavior specification, delivered in either paper or interactive media as context dictates.

Goals, not features, are key to product success

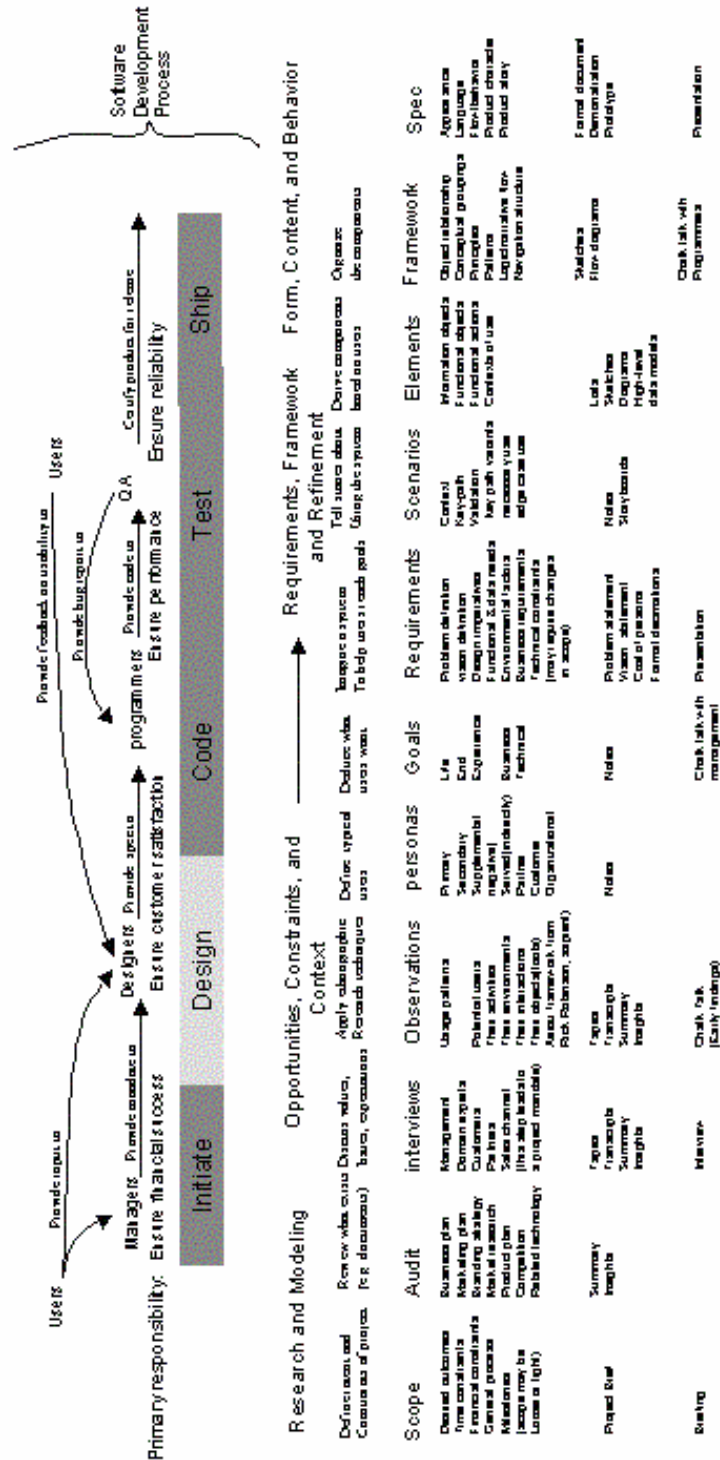
Programmers and engineers—people who are intrigued by technology—share a strong tendency to think about products in terms of functions and features. This is only natural, as this is how developers build software: function-by-function. The problem is that this is not how users want to use it.

The decision about whether a feature should be included in a product shouldn't rest on its technological understandings. The driving force behind the decision should never be simply that we have the technical capability to do it. The deciding factor should be whether that feature directly, or indirectly, helps to achieve the goals of the user while still meeting the needs of the business.

The successful interaction designer must be sensitive to user's goals amid the pressures and chaos of the product-development cycle. The Goal-Directed process, with its clear rationale for design decisions, makes persuading engineers easier, keeps marketing and management stakeholders in the loop, and ensures that the design in question isn't just guesswork, or a reflection of the team members' personal preferences.

Most computer users know all too well that opening the shrink wrap on a new software product augurs several days of frustration and disappointment spent learning the new interface. On the other hand, many experienced users of a program may find themselves continually frustrated because the program always treats them like rank beginners. It seems impossible to find the right balance between catering to the needs of the first-timer and the needs of the expert.

One of the eternal conundrums of interaction and interface design is deciding how to address the needs of both beginning users and expert users with a single interface. Some programmers and designers choose to abandon this idea completely, choosing instead to create software with a beginner mode and an expert mode, the former usually being an oversimplified and underpowered subset of the latter. Of course, nobody wants to be caught dead using software in beginner mode, but the leap from there to expert mode is usually off a rather tall cliff into a shark-infested moat of implementation-model design. What, then, is the answer? The solution to this predicament lies in a different understanding of the way user's master new concepts and tasks.



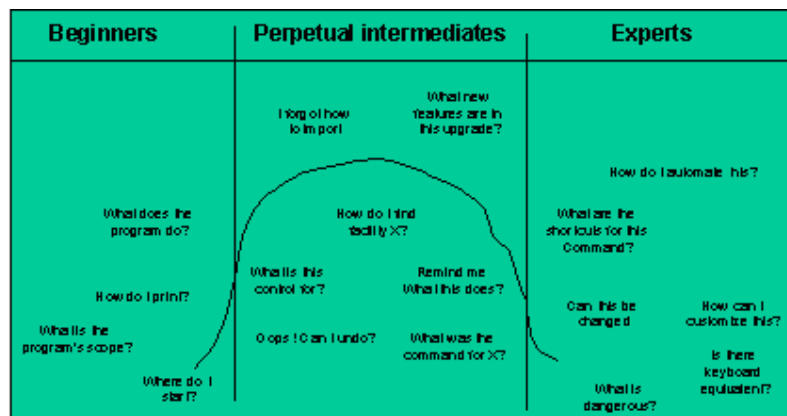
18.3 Types of users

Most users are neither beginners nor experts; instead they are intermediates. The experience level of people performing an activity requiring knowledge or skill, if we graph number of people against skill level, a relatively small number of beginners are on

the left side, a few experts are on the right, and the majority—intermediate users—are in the center.

Statistics don't tell the whole story, however, the bell curve is a snapshot in time, and although most intermediate tend to stay in that category, the beginners do not remain beginners for very long. The difficulty of maintaining a high level of expertise also means that experts come and go rapidly, but beginners change even more rapidly. Both beginners and experts tend over time to gravitate towards intermediacy.

Although everybody spends some minimum time as a beginner, nobody remains in that state for long. People don't like to be incompetent; and beginners, by definition, are incompetent. Conversely, learning and improving is rewarding, so beginners become intermediates very quickly—or they drop out altogether. All skiers, for example, spend time as beginners, but those who find they don't rapidly progress beyond more-falling-than-skiing quickly abandon the sport. The rest soon move off of the bunny slopes onto the regular runs. Only a few ever make it onto the double-black diamond runs for experts.



The occupants of the beginner end of the curve will either migrate into the center bulge of intermediates, or they will drop off the graph altogether and find some product or activity in which they can migrate into intermediacy. Most users thus remain in a perpetual state of adequacy striving for fluency, with their skills ebbing and flowing like the tides depending on how frequently they use the program. Larry Constantine first identified the importance of designing for intermediates, and in his book *Software for Use*; he refers to such users as improving intermediates. The term perpetual intermediates are preferred, because although beginners quickly improve to become intermediates, they seldom go on to become experts.

A good ski resort has a gentle slope for learning and a few expert runs to really challenge the serious skiers. But if the resort wants to stay in business, it will cater to the perpetual intermediate skier, without scaring off the beginner or insulting the expert. The beginner must find it easy to matriculate into the world of intermediacy, and the expert must not find his vertical runs obstructed by aids for bewildered perpetual intermediates.

A well-balanced user interface takes the same approach. It doesn't cater to the beginner or to the experts, but rather devotes the bulk of its efforts to satisfying the perpetual intermediate. At the same time, it avoids offending either of its smaller constituencies, recognizing that they are both vital.

Most users in this middle state would like to learn more about the program but usually don't have the time. Occasionally, the opportunity to do so will surface. Sometimes these intermediates use the product extensively for weeks at a time to complete a big project. During this time, they learn new things about the program. Their knowledge grows beyond its previous boundaries.

Sometimes, however, they do not use the program for months at a time and forget significant portions of what they knew. When they return to the program, they are not beginners, but they will need reminders to jog their memory back to its former state.

If a user finds himself not satisfactorily progressing beyond the beginner stage after only a few hours, he will often abandon the program altogether and find another to take its place. No one is willing to remain incompetent at a task for long.

Optimizing for intermediates

Now let's contrast our bell curve of intermediates with the way that software is developed. Programmers qualify as experts in the software they code because they have to explore every possible use case, no matter how obscure and unlikely, to create program code to handle it. Their natural tendency is to design implementation model software with every possible option given equal emphasis in the interaction, which they, as experts, have no problem understanding.

All the same, time, sales, marketing, and management—none of whom are likely to be expert users or even intermediates—demonstrate the product to customers, reporters, partners, and investors who are themselves unfamiliar with the product. Because of their constant exposure to beginners, these professionals have a strongly biased view of the user community. Therefore, it comes as no surprise that sales and marketing folks lobby for bending the interface to serve beginners. They demand that training wheels be attached to the product to help out the struggling beginner.

Programmers create interaction suitable only for experts, while the marketers demand interactions suitable only for beginners, but the largest, most stable, and most important group of users is the intermediate group.

It's amazing to think that the majority of real users are typically ignored, but more often than not that is the case. You can see it in many enterprise and commercial software-based products. The overall design biases them towards expert users, while at the same time, cumbersome tools like wizards and Clippy are rafted on the meet the marketing department's perception of new users. Experts rarely use them, and beginners soon desire these embarrassing reminders of their ignorance. But the perpetual intermediate majority is perpetually stuck with them.

Our goal should be neither to pander to beginners nor to rush intermediates into expertise. Our goal is threefold: to rapidly and painlessly get beginners into intermediacy; to avoid putting obstacles in the way of those intermediates who want to become experts; and most of all, to keep perpetual intermediates happy as they stay firmly in the middle of the skill spectrum.

It is required to spend more time making our programs powerful and easy to use for perpetual intermediate users. Beginners and experts are also accommodated but not to the discomfort of the largest segment of users.

What perpetual intermediates need

Perpetual intermediates need access to tools. They don't need scope and purpose explained to them because they already know these things. Tooltips are the perfect perpetual intermediate idiom. Tooltips say nothing about scope and purpose and meaning; they only state function in the briefest of idioms, consuming the least amount of video space in the process.

Perpetual intermediates know how to use reference materials. They are motivated to dig deeper and learn, as long as they don't have to tackle too much at once. This means that online help is a perpetual intermediate tool. They use it by way of the index, so that part of help must be very comprehensive.

Perpetual intermediates will be establishing the functions that they use with regularity and those that they only use rarely. The user may experiment with obscure features, but he will soon identify—probability subconsciously—his frequently used working set. The user will demand that the tools in his working set are placed front-and-center in the user interface, easy to find and to remember.

Perpetual intermediates usually know that advanced features exist, even though they may not need them or know how to use them. But the knowledge that they are there is reassuring to the perpetual intermediate, convincing him that he made the right choice investing in this program. The average skier may find it reassuring to know that there is a really scary black diamond expert runs just beyond those trees, even if she never intends to use it. It gives her something to aspire to and dream about.

Your program's code must provide for both rank amateurs and all the possible cases an expert might encounter. Don't let this technical requirement influence your design thinking. Yes, you must apply the bulk of your talents, time, and resources to designing the best interaction possible for your most representative users: the perpetual intermediates.

Lecture 19.

User Research Part-I

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand the difference in qualitative and quantitative research
- Discuss in detail qualitative research technique

The outcome of any design effort must ultimately be judged by how successfully it meets the requirements of both the user and the organization that commissioned it. No matter how skillful and creative the designer. If she does not have a clear and detailed knowledge of the users she is designing for, what the constraints of the problem are, and what business or organizational goals the design is hoping to achieve, she will have little chance of success.

What and how questions like these are best answered by qualitative research, not metrics or demographics (though these also have their purpose). There are many types of qualitative research, each of which plays an important role in filling in a picture of the design landscape of a product.

Qualitative versus Quantitative Research

Research is a word that most people associate with science and objectivity. This association isn't incorrect, but it biases many people towards the notion that the only valid sort of research is the kind that yields the supposed ultimate in objectivity: quantitative data. The notion that numbers don't lie is prevalent in the business and engineering communities, even though we all rationally understand that numbers—especially numbers ascribed to human activities—can be manipulated or reinterpreted at least as dramatically as words.

Data gathered by the hard sciences like physics is simply different from that gathered on human activities: electrons don't have moods that vary from minute to minute, and the tight controls physicists place on their experiments to isolate observed behaviors are next to impossible in the social sciences. Any attempt to reduce human behavior to statistics is likely to overlook important nuances, which though they might not directly affect business plans, do make an enormous difference to the design of products. Quantitative research can only answer questions about how much or how many along a few reductive axes. Qualitative research can tell you about what, how and why in rich, multivariate detail.

Social scientists have long realized that human behaviors are too complex and subject to too many variables to rely solely on quantitative data to understand them. Usability practitioners, borrowing techniques from anthropology and other social sciences, have

developed many alternative methods for gathering useful data on user behaviors to a more pragmatic end: to help create products that better serve user need.

The value of qualitative research

Qualitative research helps us understand the domain, context and constraints of a product in different, more useful ways than quantitative research do. It also quickly helps us identify patterns of behavior among users and potential users of a product much more quickly and easily than would be possible with quantitative approaches. In particular, qualitative research helps us understand:

- Existing products and how they are used.
- Potential users of new or existing products, and how they currently approach activities and problems the new product design hopes to address
- Technical, business, and environmental contexts—the domain—of the product to be designed
- Vocabulary and other social aspects of the domain in question

Qualitative research can also help the progress of design projects by:

- Providing credibility and authority to the design team, because design decisions can be traced to research results
- Uniting the team with a common understanding of domain issues and user concerns
- Empowering management to make more informed decisions about product design issues that would otherwise be based on guesswork or personal preference

It is the experience that qualitative method, in addition to the benefits described above, tend to be faster, less expensive, more flexible, and more likely than their quantitative counterparts to provide useful answers to important questions that leads to superior design:

- What problems are people encountering with their current ways of doing what the product hopes to do?
- Into what broader contexts in people's lives does the product fit and how?
- What are the basic goals people have in using the product, and what basic tasks help people accomplish them?

19.1 Types of qualitative research

Social science and usability texts are full of methods and techniques for conducting qualitative research. We will discuss following qualitative research techniques:

- Stakeholder interviews
- Subject matter expert (SME) interviews
- User and customer interviews
- User observation/ethnographic field studies
- Literature review
- Product/prototype and competitive audits

Stakeholder interviews

Research for any new product design, through it must end with understanding the user, should start by understanding the business and technical context in which the product will be built. This is necessary not only to ensure a viable and feasible end result, but also to provide a common language and understanding among the design team, management, and engineering teams.

Stakeholders are any key members of the organization commissioning the design work, and typically include managers and key contributors from engineering, sales, product marketing, marketing communications, customer support, and usability. They may also include similar people from other organizations in business partnership with the commissioning organization, and executives. Interviews with stakeholders should occur before any user research begins.

It is usually most effective to interview each stakeholder one-on-one, rather than in a larger, cross-departmental group. A one-on-one setting promotes candor on the part of the stakeholder, and ensure that individual views are not lost in a crowd. Interviews need not last longer than about an hour, though follow-up meetings may be called for if a particular stakeholder is identified as an exceptionally valuable source of information.

The type of information that is important to gather from stakeholders includes:

- What is the preliminary vision of the product from each stakeholder perspective? As in the fable of the blind men and the elephant, you may find that each business department has a slightly different and slightly incomplete perspective on the product to be designed. Part of the design approach must therefore involve harmonizing these perspectives with those of users and customers.
- What is the budget and schedule? The answer to this question often provides a reality check on the scope of the design effort and provides a decision point for management if user research indicates a greater scope is required.
- What are the technical constraints? Another important determinant of design scope is a firm understanding of what is technically feasible given budget, time, and technology.
- What are the business drivers? It is important for the design team to understand what the business is trying to accomplish. This again leads to a decision point, should user research indicate a conflict between business and user needs. The design must, as much as possible, create a win-win situation for users, customers, and providers of the product.
- What are the stakeholders' perceptions of the user? Stakeholders who have relationships with users (such as customer support representative) may have important insights on users that will help you to formulate your user research plan. You may also find that there are significant disconnects between some stakeholders' perceptions of their users and what you discover in your research. This information can become an important decision point for management later in the process.

Understanding these issues and their impact on design solutions helps you as a designer to better serve your customer, as well as users of the product. Building consensus internally will help you to articulate issues that the business as a whole may not identified, build

internal consensus that is critical for decision making later in the design process, and build credibility for your design team.

Subject matter expert (SME) interviews

Some stakeholders may also be subject matter experts (SMEs): experts on the domain within which the product you are designing will operate. Most SMEs were users of the product or its predecessors at one time, and may now be trainers, managers, or consultants. Often they are experts hired by stakeholders, rather than stakeholders themselves. Similar to stakeholders, SMEs can provide valuable perspective on a product and its users, but designers should be careful to recognize that SMEs represent a somewhat skewed perspective. Some points to consider about using SMEs are:

- SMEs are expert users. Their long experience with a product or its domain mean that they may have grown accustomed to current interactions. They may also lean towards expert controls rather than interactions designed for perpetual intermediate perspective. SMEs are often not current users of the product, and may have more of a management perspective.
- SMEs are knowledgeable, but they aren't designers. They may have many ideas on how to improve a product. Some of these may be valid and valuable, but the most useful pieces of information to glean from these suggestions are the causative problems that lead to their proposed solutions.
- SMEs are necessary in complex or specialized domains such as medical, scientific, or financial services. If you are designing for a technical or otherwise specialized domain, you will likely need some guidance from SMEs, unless you are one yourself. Use SMEs to get information on complex regulations and industry best practices. SME knowledge of user roles and characteristics is critical for planning user research in complex domains.
- You will want access to SMEs throughout the design process. If your product domain requires use of SMEs, you should be able to bring them in at different stages of the design to help perform reality checks on design details. Make sure that you secure this access in your early interviews.

User and customer interviews

It is easy to confuse users with customers. For consumer products, customers are often the same as users, but in corporate or technical domain, users and customers rarely describe the same sets of people. Although both groups should be interviewed, each has its own perspective on the product that needs to be factored quite differently into an eventual design.

Customers of a product are those people who make the decision to purchase it. For consumer product, customers are frequently users of the product; although for products aimed at children or teens, the customers are parents or other adult supervisors of children. In the case of most enterprise or technical products, the customer is someone very different from the user—often an IT manager—with distinct goals and needs. It's important to understand customers and their goals in order to make a product viable. It is also important to realize that customers seldom actually use the product themselves, and when they do, they use it quite differently than the way their users do.

When interviewing customers, you will want to understand:

- Their goals in purchasing the product
- Their frustrations with current solutions
- Their decision process for purchasing a product of the type you're designing
- Their role in installation, maintenance, and management of the product
- Domain related issues and vocabulary

Like SMEs, customers may have many opinions about how to improve the design of the product. It is important to analyze these suggestions, as in the case of SMEs, to determine what issues or problems underline the ideas offered, because better, more integrated solutions become evident later in the design process.

Users of a product should be the main focus of the design effort. They are the people (not their managers or support team) who are personally trying to accomplish something with the product. Potential users are people who do not currently use the product, but who are good candidates for using it in the future. A good set of user interviews includes both current users (if the product already exists and is being revised) and potential users (users of competitive products and non-automated systems of appropriate). Information we are interested in learning from users includes:

- Problems and frustrations with the product (or analogous system if they are potential users)
- The context of how the product fits into their lives or workflow: when, why, and how the product is used, that is, patterns of user behavior with the product.
- Domain knowledge from a user perspective: what do users need to know to accomplish their jobs
- A basic understanding of the users' current tasks: both those the product requires and those it doesn't support
- A clear understanding of user goals: their motivations and expectations concerning use of the product

User observation

Most people are incapable of accurately assessing their own behaviors, especially outside of the context of their activities. It then follows that interviews performed outside the context of the situations the designer hopes to document will yield less complete and less accurate data. Basically, you can talk to users about how they think they behave, or you can observe it first hand. The latter route provides superior results.

Many usability professionals make use of technological aids such as audio or video recorders to capture what users say and do. Care must be taken not to make these technologies too obtrusive; otherwise the users will be distracted and behave differently than they would off-tape.

Perhaps the most effective technique for gathering qualitative user data combines interviews and observation, allowing the designer to ask clarifying questions and direct inquiries about situations and behaviors they observe in real-time.

Literature review

In parallel with stakeholder interviews, the design team should review any literature pertaining to the product or its domain. This can and should include product marketing plans, market research, technology specifications and white papers, business and technical journal articles in the domain, competitive studies. Web searches for related and competing products and news, usability study results and metrics, and customer support data such as call center statistics.

The design team should collect this literature, use it as a basis for developing questions to ask stakeholders and SMEs, and later use it to supply additional domain knowledge and vocabulary, and to check against compiled user data.

Product and competitive audits

Also in parallel to stakeholder and SME interviews, it is often quite helpful for the design team to examine any existing version or prototype of the product, as well as its chief competitors. Doing so gives the design team a sense of the state of the art, and provides fuel for questions during the interviews. The design team, ideally, should engage in an informal heuristic or expert review of both the current and competitive interfaces, comparing each against interaction and visual design principles. This procedure both familiarizes the team with the strengths and limitations of what is currently available to users, and provides a general idea of the current functional scope of the product.

Lecture 20.

User Research Part-II

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand the User-Centered approach
- Discuss in detail the ethnographic interviews
- Understand how to prepare for ethnographic interviews

In our last lecture we were studying the qualitative research techniques. Today we will discuss last technique, ethnographic field study. But, first let us look at the user-centered design approach.

20.1 User-Centered Approach

The user-centered approach means that the real users and their goals, not just technology, should be the driving force behind development of a product. As a consequence, a well-designed system should make the most of human skill and judgment, should be directly relevant to the work in hand, and should support rather than constrain the user. This is less technique and more a philosophy.

In 1985, Gould and Lewis laid down three principles they believed would lead to a “useful and easy to use computer system.” These are very similar to the three key characteristics of interaction design.

- Early focus on users and tasks: This means first understanding who the users will be by directly studying their cognitive, behavioral, anthropomorphic, and attitudinal characteristics. This required observing users doing their normal tasks, studying the nature of those tasks, and then involving users in the design process.
- Empirical measurement: early in development, the reactions and performance of intended users to printed scenarios, manuals, etc, is observed and measured. Later on, users interact with simulations and prototypes and their performance and reactions are observed, recorded and analyzed.
- Iterative design: when problems are found in user testing, they are fixed and then more tests and observations are carried out to see the effects of the fixes. This means that design and development is iterative, with cycles of “design, test, measure, and redesign” being repeated as often as necessary.

Iteration is something, which is emphasized in user-centered design and is now widely accepted that iteration is required. When Gould and Lewis wrote their paper, however, the iterative nature of design was not accepted by most developers. In fact, they comment in their paper how “obvious” these principles are, and remark that when they started

recommending these to designers, the designers' reactions implied that these principles were indeed obvious.

Applying ethnography in design

Ethnography is a method that comes originally from anthropology and literally means "writing the culture". It has been used in the social sciences to display the social organization of activities, and hence to understand work. It aims to find the order within an activity rather than impose any framework of interpretation on it. It is a broad-based approach in which users are observed as they go about their normal activities. The observers immerse themselves in the users' environment and participate in their day-to-day work, joining in conversations, attending meetings, reading documents, and so on. The aim of an ethnographic study is to make the implicit explicit. Those in the situation, the users in this case, are so familiar with their surroundings and their daily tasks that they often don't see the importance of familiar actions or happenings, and hence don't remark upon them in interviews or other data-gathering sessions.

There are different ways in which this method can be associated with design. Beynon-Davies has suggested that ethnography can be associated with the development as "ethnography of", "ethnography for", and "ethnography within." Ethnography of development refers to studies of developers themselves and their workplace, with the aim of understanding the practices of development (e.g. Button and Sharrock). Ethnography for development yields ethnographic studies that can be used as a resource for development, e.g., studies of organizational work. Ethnography within software development is the most common form of study; here the techniques associated with ethnography are integrated into methods and approaches for development.

Because of the very nature of the ethnography experience, it is very difficult to describe explicitly what data is collected through such an exercise. It is an experience rather than a data-collection exercise. However, the experience must be shared with other team members, and therefore needs to be documented and rationalized.

Studying the context of work and watching work being done reveals information that might be missed by other methods that concentrate on asking about work away from its natural setting. For example, it can shed light on how people do the "real" work as opposed to the formal procedures that you 'd find in documentation; the nature and purpose of collaboration, awareness of other's work, and implicit goals that may not even be recognized by the workers themselves. For example, Heath et al. has been exploring the implications of ethnographic studies of real-world setting for the design of cooperative systems. They studied medical centers, architects' practices, and TV and radio studios.

In one of their studies Heath et al. looked at how dealers in a stock exchange work together. A main motivation was to see whether proposed technological support for market trading was indeed suitable for that particular setting. One of the tasks examined in detail was the process of writing tickets to record deals. It had been commented upon earlier by others that this process of deal capture, using "old-fashioned" paper and pencil technology, was currently time-consuming and prone to error. Based on this finding, it had been further suggested that the existing way of making deals could be improved by

introducing new technologies, including touch screens to input the details of transactions, and headphones to eliminate distracting external noise.

However, when Heath et al. began observing the deal capture in practice, they quickly discovered that these proposals were misguided. In particular, they warned that these new technologies would destroy the very means by which the traders currently communicate and keep informed of what others are up to. The touch screens would reduce the availability of information to others on how deals were progressing; while headphones would impede the dealers' ability to inadvertently monitoring of other dealers' actions was central to the way deals are done. Moreover, if any dealers failed to keep up with what the other dealers were doing by continuously monitoring them, it was likely to affect their position in the market, which ultimately could prove very costly to the bank they were working for.

Hence, the ethnographic study proved to be very useful in warning against attempts to integrate new technologies into a workplace without thinking through the implications for the work practice. As an alternative, Heath et al. suggested pen-based mobile systems with gesture recognition that could allow deals to be made efficiently while also allowing the other dealers to continue to monitor one another unobtrusively. Hughes et al state that "doing" ethnography is about being reasonable, courteous and unthreatening, and interested in what's happening. Training and practice are required to produce good ethnographies.

Collecting ethnographic data is not hard although it may seem a little bewildering to those accustomed to using a frame of reference to focus the data collection rather than letting the frame of reference arise from the available data. You collect what is available, what is "ordinary", what it is that people do, say, how they work. The data collected therefore has many forms: documents, notes of your own, pictures, room layouts.

In some way, the goals of design and the goals of ethnography are at opposite ends of a spectrum. Design is concerned with abstraction and rationalization. Ethnography, on the other hand, is about detail. An ethnographer's account will be concerned with the minutiae of observation, while a designer is looking for useful abstractions that can be used to inform design. One of the difficulties faced by those wishing to use this very powerful technique is how to harness the data gathered in a form that can be used in design.

20.2 Ethnography framework

Ethnographic framework has been developed specifically to help structure the presentation of ethnographies in a way that enables designers to use them. This framework has three dimensions

1. Distributed co-ordination
2. Plans and procedures
3. Awareness of work

1. Distributed co-ordination

The distributed co-ordination dimension focuses on the distributed nature of the tasks and activities, and the means and mechanisms by which they are coordinated. This has implications for the kind of automated support required.

2. Plans and procedures

The plans and procedures dimension focuses on the organizational support for the work, such as workflow models and organizational charts, and how these are used to support the work. Understanding this aspect impacts on how the system is designed to utilize this kind of support.

3. Awareness of work

The awareness of work dimension focuses on how people keep themselves aware of others' work. No one works in isolation, and it has been shown that being aware of others' actions and work activities can be a crucial element of doing a good job. In the stock market example this was one aspect that ethnographers identified. Implications here relate to the sharing of information.

Rather than taking data from ethnographers and interpreting this in design, an alternative approach is to train developers to collect ethnographic data themselves. This has the advantage of giving the designers first-hand experience of the situation. Telling someone how to perform a task, or explaining what an experience is like is very difficult from showing him or her or even gaining the experience themselves. Finding people with the skills of ethnographers and interaction designers may be difficult, but it is possible to provide notational and procedural mechanisms to allow designers to gain some of the insights first-hand. Two methods described below give such support.

- Coherence
- Contextual design

Coherence

The coherence method combines experiences of using ethnography to inform design with developments in requirements engineering. Specifically, it is intended to integrate social analysis with object-oriented analysis from software engineering. Coherence does not prescribe how to move from the social analysis to use cases, but claims that presenting the data from an ethnographic study based around a set of "viewpoints" and "concerns" facilitated the identification of the product's most important use cases.

Viewpoints and concerns

Coherence builds upon the framework introduced above and provides a set of focus questions for each of the three dimensions, here called "viewpoints". The focus questions are intended to guide the observer to particular aspects of the workplace. They can be used as a starting point to which other questions may be added as experience in the domain and the method increase.

In addition to viewpoints, Coherence has a set of concerns and associated questions. Concerns are a kind of goal, and they represent criteria that guide the requirements activity. These concerns are addressed within each appropriate viewpoint. One of the first tasks is to determine whether the concern is indeed relevant to the viewpoint. If it is relevant, then a set of elaboration questions is used to explore the concern further. The concerns, which have arisen from experience of using ethnography in systems design, are:

- Paper work and computer work
- Skill and the use of local knowledge

- Spatial and temporal organization
- Organizational memory

Paperwork and computer work

These are embodiments of plans and procedures, and at the same time are a mechanism for developing and sharing an awareness of work.

Skill and the use of local knowledge

This refers to the “workarounds” that are developed in organizations and are at the heart of how the real work gets done.

Spatial and temporal organization

This concern looks at the physical layout of the workplace and areas where time is important.

Organizational memory

Formal documents are not the only way in which things are remembered within an organization. Individuals may keep their own records, or there maybe local gurus.

Contextual design

Contextual design was another technique that was developed to handle the collection and interpretation of data from fieldwork with the intention of building a software-based product. It provides a structured approach to gathering and representing information from fieldwork such as ethnography, with the purpose of feeding it into design.

Contextual design has seven parts:

- Contextual inquiry
- Work modeling, consolidation
- Work redesign
- User environment design
- Mockup
- Test with customers
- Putting it into practice

Contextual inquiry

Contextual inquiry, according to Beyer and Holtzblatt, is based on a master-apprentice model of learning: observing and asking questions of the users as if she is the master craftsman and he interviews the new apprentice. Beyer and Holtzblatt also enumerate four basic principles for engaging in ethnographic interview:

Context:

Rather than interviewing the user in a clean white room, it is important to interact with and observe the user in their normal work environment, or whatever physical context is appropriate for the product. Observing users as they perform activities and questioning them in their own environment, filled with the artifacts they use each day, can bring the all-important details of their behaviors to light.

Partnership:

The interview and observation should take the tone of a collaborative exploration with the user, alternating between observation of and discussion of its structure and details.

Interpretation:

Much of the work of the designer is reading between the lines of facts gathered about user's behaviors, their environment, and what they say. These facts must be taken together as a whole, and analyzed by the designer to uncover the design implications. Interviewers must be careful, however, to avoid assumptions based on their own interpretation of the facts without verifying these assumptions with users.

Focus:

Rather than coming to interviews with a set questionnaire or letting the interview wander aimlessly, the designer needs to subtly direct the interview so as to capture data relevant to design issues.

Improving on contextual inquiry

Contextual inquiry forms a solid theoretical foundation for quantitative research, but as a specific method it has some limitations and inefficiencies. The following process improvements result in a more highly leveraged research phase that better sets the stage for successful design:

- **Shortening the interview process:** contextual inquiry assumes full day interviews with users. The authors have found that interviews as short as one hour in duration are sufficient to gather the necessary user data, provided that a sufficient number of interviews (about six well-selected users for each hypothesized role or type) are scheduled. It is much easier and more effective to find a diverse set of users who will consent to an hour with a designer than it is to find users who will agree to spend an entire day.
- **Using smaller design teams:** Contextual inquiry assumes a large design team that conducts multiple interviews in parallel, followed by debriefing sessions in which the full team participates. Experiments show that it is more effective to conduct interviews sequentially with the same designers in each interview. This allows the design team to remain small (two or three designers), but even more important, it means that the entire team interacts with all interviewed users directly; allowing the members to most effectively analyze and synthesize the user data.
- **Identifying goals first:** Contextual inquiry, as described by Beyer and Holtzblatt, feeds a design process that is fundamentally task-focused. It is proposed that ethnographic interviews first identify and prioritize user goals before determining the tasks that relate to these goals.
- **Looking beyond business contexts:** the vocabulary of contextual inquiry assumes a business product and a corporate environment. Ethnographic interviews are also possible in consumer domains, though the focus of questioning is somewhat different.

20.3 Preparing for ethnographic interviews

As we discussed ethnography is term borrowed from anthropology, meaning the systematic and immersive study of human cultures. In anthropology, ethnographic researchers spend years living immersed in the cultures they study and record. Ethnographic interviews take the spirit of this type of research and apply it on a micro level. Rather than trying to understand behaviors and social ritual of an entire culture, the goal is understand the behaviors and rituals of people interacting with individual products.

Identifying candidates

Because the designer must capture an entire range of user behaviors regarding a product, it is critical that the designers identify and appropriately diverse sample of users and user types when planning a series of interviews. Based on information gleaned from stakeholders, SMEs, and literature reviews, designers need to create a hypothesis that serves as a starting point in determining what sorts of users and potential users to interview.

Kim Goodwin has coined this the persona hypothesis, because it is the first step towards identifying and synthesizing personas. The persona hypothesis is based on likely behavioral differences, not demographics, but takes into consideration identified target markets and demographics. The nature of the product's domain makes a significant difference in how a persona hypothesis is constructed. Business users are often quite different than consumer users in their behavior patterns and motivations, and different techniques are used to build the persona hypothesis in each case.

The personal hypothesis

The persona hypothesis is a first cut at defining the different kinds of users (and sometimes customers) for a product in a particular domain. The hypothesis serves as a basis for an initial set of interviews; as interviews proceed, new interviews may be required if the data indicates the existence of user types not originally identified.

The persona hypothesis attempts to address, at a high level, these three questions:

- What different sorts of people might use this product?
- How might their needs and behaviors vary?
- What ranges of behavior and types of environments need to be explored?

Roles in business and customer domains

Patterns of needs and behavior, and therefore types of users, vary significantly between business and technical, and consumer products. For business products, roles—common sets of tasks and information needs related to distinct classes of users—provide an important initial organizing principle. For example, in an enterprise portal, these search roles can be found:

- People who search for content on the portal
- People who upload and update content on the portal
- People who technically administer the portal

In business and technical context, roles often map roughly to job descriptions, so it is relatively easy to get a reasonable first cut of user types to interview by understanding the kind of jobs held by users of the system.

Unlike business users, consumers don't have concrete job descriptions, and their use of products tends to cross multiple contexts. Their roles map more closely to lifestyle choices, and it is possible for consumer users to assume multiple roles even for a single product in this sense. For consumers, roles can usually better be expressed by behavioral variables

Behavioral and demographic variables

Beyond roles, a persona hypothesis seeks to identify variables that might distinguish users based on their needs and behaviors. The most useful, but most difficult to anticipate without research, are behavioral variables: types of behavior that behavior concerning shopping that we might identify:

- Frequency of shopping (frequent--infrequent)
- Desire to shop (loves to shop—hates to shop)
- Motivation to shop (bargain hunting—searching for just the right item)

Although consumer user types can often be roughly defined by the combination of behavioral variables they map to, behavioral variables are also important for identifying types of business and technical users. People within a single business-role definition may have different motivations for being there and aspirations for what they plan to do in the future. Behavioral variables can capture this; through usually not until user data has been gathered.

Given the difficulty in accurately anticipating behavioral variables before user data is gathered, another helpful approach in building a persona hypothesis is making use of demographic variables. When planning your interviews, you can use market research to identify ages, locations, gender, and incomes of the target markets for the product. Interviews should be distributed across these demographic ranges.

Domain expertise versus technical expertise

One important type of behavioral distinction to note is the difference between technical expertise (knowledge of digital technology) and domain expertise (knowledge of a specialized subject area pertaining to a product). Different users will have varying amount of technical expertise; similarly, some users of a product may be less expert in their knowledge of the product's domain (for example, accounting knowledge in the case of a general ledger application). Thus, depending on who the design target of the product is, domain support may be a necessary part of the product's design, as well as technical ease of use.

Environmental variables

A final consideration, especially in the case of business products, is the cultural differences between organizations in which the users are employed. Small companies, for example, tend to have more interpersonal contact between workers: huge companies have layers of bureaucracy. These environmental variables also fall into ranges:

- Company size (small—multinational)
- IT presence (ad hoc—draconian)
- Security level (lax--tight)

Like behavioral variables, these may be difficult to identify without some domain research, because patterns do vary significantly by industry and geographic region.

20.4 Putting a plan together

After you have created a persona hypothesis, complete with potential roles, behavioral, demographic, and environmental variables, you then need to create an interview plan that can be communicated to the person in charge of providing access to users.

Each identified role, behavioral variable, demographic variable, and environmental variable identified in the persona hypothesis should be explored in four to six interviews (some time more if a domain is particularly complex). However, these interviews can overlap: it is perfectly acceptable to interview a female in her twenties who loves to shop; this would count as an interview for each of three different variables: gender, age group, and desire to shop. By being clever about mapping variables to interviewee screening profiles, you can keep the number of interviews to a manageable number.

Lecture 21.

User Research Part-III

Learning Goals

The aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand how to conduct ethnographic interviews
- Discuss briefly other research techniques

21.1 How to conducting ethnographic interviews

Securing interviews

Use the followings to get access

1. Stakeholders
2. Market or usability research firm
3. Friends and relatives

Interview teams and timings

-2 interviewees, 1-hr per interview, 6 per day

21.2 Phases of ethnographic interviews

Begin	→	End
Structural issues	→	Specific issues
Goal-oriented issues	→	Task-oriented issues

Early-phase

- Exploratory
- Focused on domain knowledge
- Open-ended questions

Mid-phase

- Identify patterns of use
- Clarifying questions
- More focused questions

Late-phase

- Confirm patterns of use
- Clarify user roles and behaviors
- Closed-ended questions

21.3 Basic interview methods

- Interview where the action happens
- Avoid a fixed set of questions
- Focus on goals first, tasks second

- Avoid making the user a designer
- Avoid discussions of technology
- Encouraging storytelling
- Ask for a show-and-tell
- Avoid leading questions

Basic interview methods

1. Goal-oriented questions
2. System-oriented questions
3. Workflow-oriented question
4. Attitude-oriented questions

1. Goal-oriented questions

Opportunity

What activities currently waste your time?

Goals

What makes a good day?

A bad day.

Priorities

What is the most important to you?

Information

What helps you make decisions?

2. System-oriented questions

Function

What are the most common things you do with the product?

Frequency

What parts of the product do you use most?

Preference

What are your favorite aspects of the product? What drives you crazy?

Failure

How do you work around problems?

Expertise

What shortcuts do you employ?

3. Workflow-oriented questions

Process

What did you do when you first came into today? And after that?

Occurrence and recurrence

How often do you do this?

What things do you do weekly, monthly but not every day?

Exceptions

What constitutes a typical day?

What would be an unusual event?

4. Attitude-oriented questions

Aspiration

What do you see yourself doing five years from now?

Avoidance

What would you prefer not to do?

What do you procrastinate on?

Motivation

What do you enjoy most about your job (or lifestyle)?

What do you always tackle first?

21.4 Types of qualitative research

Stakeholder interview

Subject matter experts (SME) interviews

User and customer interviews

Literature review

Product/prototype and competitive audits

User observation/ethnographic field studies

21.5 Others types of research

1. Focus group
2. Market demographics and segments
3. Usability and user testing

1. Focus group

- Used by marketing organizations
- Used in traditional product marketing
- Representative users gathered in room
- Shown a product and reactions gauged
- Reactions recorded by audio/video

Limitations**2. Market demographics and segments**

What motivates people to buy?

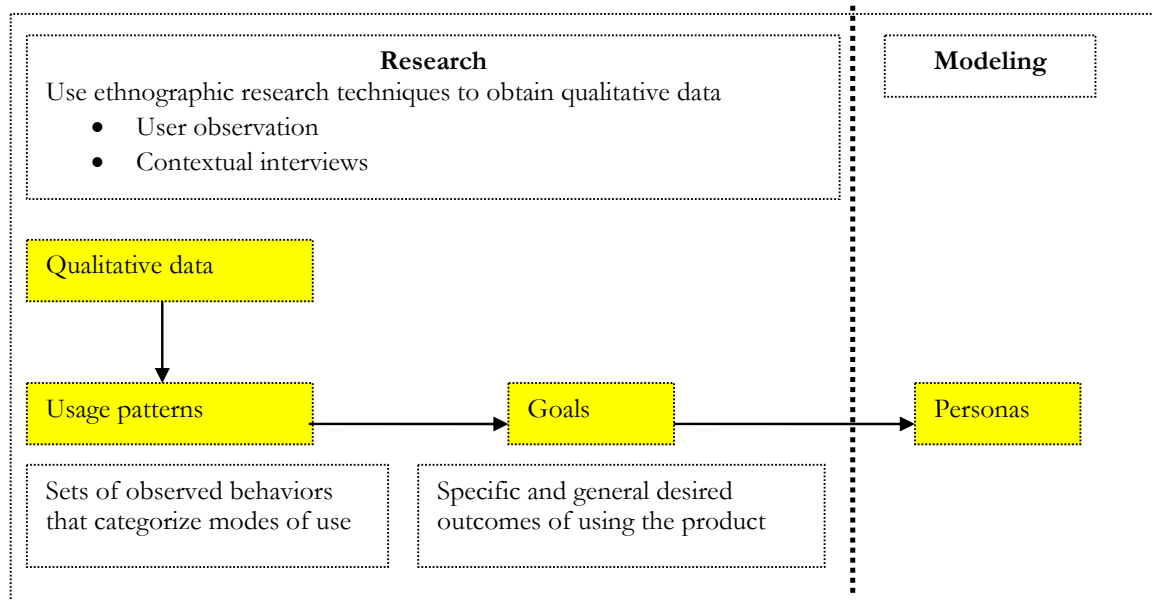
- Determined by market segmentation that group people by distinct needs
- Determines who will be receptive to what marketing message or a particular product
- Demographic data
Race, education, income, location etc.
- Psychographic data

Attitude, lifestyle, values, ideology etc.

3. Usability and user testing

21.6 Comparison of different techniques

Techniques	Good for	Kind of data	Advantages	Disadvantages
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee if necessary. Encourages contact between developers and users	Time consuming Artificial environment may intimidate interviewee
Studying documentation	Learning about procedures, regulations and standards	Quantitative	No time commitment from users required	Day-to-day working will differ from documented procedures
Naturalistic observation	Understanding context of user activity	Quantitative	Observing actual work gives insights that other techniques can't give	Very time consuming. Huge amounts of data
Focus groups and workshops	Collecting multiple view points	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of dominant characters



Lecture 22.

User Modeling

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand how to conduct ethnographic interviews
- Discuss briefly other research techniques

The most powerful tools are simple in concept, but must be applied with some sophistication. The most powerful interaction design tool used is a precise descriptive model of the user, what he wishes to accomplish, and why. The sophistication becomes apparent in the way we construct and use that model.

These user models, which we call personas, are not real people, but they are based on the behaviors and motivations of real people and represent them throughout the design process. They are composite archetypes based on behavioral data gathered from many actual users through ethnographic interviews. We discover our personas during the course of the Research phase and formalize them in the Modeling phase, by understanding our personas, we achieve an understanding of our users' goals in specific context—a critical tool for translating user data into design framework.

There are many useful models that can serve as tools for the interaction designer, but it is felt that personas are among the strongest.

22.1 Why Model?

Models are used extensively in design, development, and the sciences. They are powerful tools for representing complex structures and relationships for the purpose of better understanding or visualizing them. Without models, we are left to make sense of unstructured, raw data, without the benefit of the pig picture or any organizing principle. Good models emphasize the salient features of the structures or relationships they represent and de-emphasize the less significant details.

Because we are designing for users, it is important that we can understand and visualize the salient aspects of their relationships with each other, with their social and physical environment and of course, with the products we hope to design.

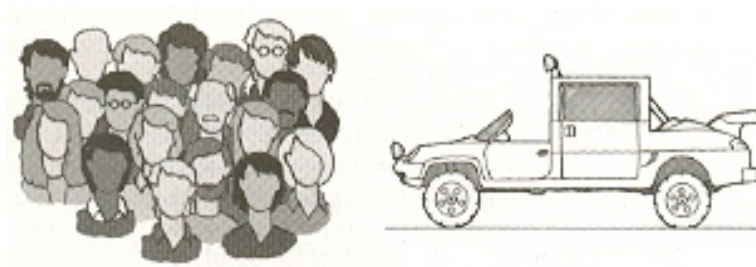
Just as physicists create models of the atom based on raw, observed data and intuitive synthesis of the patterns in their data, so must designers create models of users based on raw, observed behaviors and intuitive synthesis of patterns in the data. Only after we formalize such patterns can we hope to systematically construct patterns of interactions that smoothly match the behaviors, mental models and goals of users. Personas provide this formalization.

22.2 Personas

To create a product that must satisfy a broad audience of users, logic tells you to make it as broad in its functionality as possible to accommodate the most people. This logic, however, is flawed. The best way to successfully accommodate a variety of users is to design for specific types of individuals with specific needs.

When you broadly and arbitrarily extend a product's functionality to include many constituencies, you increase the cognitive load and navigational overhead for all users. Facilities that map please some users will likely interfere with the satisfaction of other.

A simple example of how personas are useful is shown in figure below, if you try to design an automobile that pleases every possible driver, you end up with a car with every possible feature, but which pleases nobody. Software today is too often designed to please to many users, resulting in low user satisfaction



But by designing different cars for different people with different specific goals, as shown in figure below, we are able to create designs that other people with similar needs to our target drivers also find satisfying. The same hold true for the design of digital products and software.



The key is in choosing the right individuals to design for, ones whose needs represent the needs of a larger set of key constituents, and knowing how to prioritize design elements to

address the needs of the most important users without significantly inconveniencing secondary users. Personas provide a powerful tool for understanding user needs, differentiating between different types of users, and prioritizing which users are the most important to target in the design of function and behavior.

Personas were introduced as a tool for user modeling, they have gained great popularity in the usability community, but they have also been the subjects of some misunderstandings.

Strengths of personas as a design tool

The persona is a powerful, multipurpose design tool that helps overcome several problems that currently plague the development of digital products. Personas help designers”

- Determine what a product should do and how it should behave. Persona goals and tasks provide the basis for the design effort.
- Communicate with stakeholders, developers, and other designers. Personas provide a common language for discussing design decisions, and also help keep the design centered on users at every step in the process.
- Build consensus and commitment to the design. With a common language comes a common understanding. Personas reduce the need for elaborate diagrammatic models because, as it is found, it is easier to understand the many nuances of user behavior through the narrative structures that personas employ.
- Measure the design’s effectiveness. Design choices can be tested on a persona in the same way that they can be show to a real user during the formative process. Although this doesn’t replace the need to test on real users. It provides a powerful reality check tool for designers trying to solve design problems. This allows design iteration to occur rapidly and inexpensively at the whiteboard, and it results in a far stronger design baseline when the time comes to test with real users.
- Contribute to other product-related efforts such as marketing and sales plan. It has been seen that clients repurpose personas across their organization, informing marketing campaigns, organizational structure, and other strategic planning activities. Business units outside of product development desire sophisticated knowledge of a product’s users and typically view personas with great interest.

Personas and user-centered design

Personas also resolve three User-Centered design issues that arise during product development:

- The elastic user
- Self-referential design
- Design edge cases

The elastic user

Although satisfying the user is goal, the term user causes trouble when applied to specific design problems and contexts. Its imprecision makes it unusable as a design tool—every person on a product team has his own conceptions of the user and what the user needs. When it comes time to make a product decisions, this “user” becomes elastic, bending and stretching to fit the opinions and presuppositions of whoever has the floor.

If programmers find it convenient to simply drop a user into a confusing file system of nested hierarchical folders to find the information she needs, they define the elastic user as an accommodating, computer-literate power user. Other times, when they find it more convenient to step the user through a difficult process with a wizard, they define the elastic user as an unsophisticated first-time user. Designing for the elastic user gives the developer license to code as he pleases while still apparently serving “the user”. However, our goal is to design software that properly meets real user needs. Real users—and the personas representing them—are not elastic, but rather have specific requirements based on their goals, capabilities, and contexts.

Self-referential design

Self-referential design occurs when designers or developers project their own goals, motivations, skills, and mental models onto a product’s design. Most “cool” product designs fall into this category: the audience doesn’t extend beyond people like the designer, which is fine for a narrow range of products and completely inappropriate for most others. Similarly, programmers apply self-referential design when they create implementation-model products. They understand perfectly how it works and are comfortable with such products. Few non-programmers would concur.

Design edge cases

Another syndrome that personas help prevent is designing for edge cases—those situations that might possibly happen, but usually won’t for the target personas. Naturally, edge cases must be programmed for, but they should never be the design focus. Personas provide a reality check for the design.

Personas are based on research

Personas must, like any model, be based on real-world observation. The primary source of data used to synthesize personas must be from ethnographic interviews, contextual inquiry, or other similar dialogues with and observation of actual and potential users. Other data that can support and supplement the creation of personas include, in rough order of efficacy:

- Interviews with users outside of their use contexts
- Information about users supplied by stakeholders and subject matter experts
- Market research data such as focus groups and surveys
- Market segmentation models
- Data gathered from literature reviews and previous studies

However, none of this supplemental data can take the place of direct interaction with and observation of users in their native environments. Almost every word in a well-developed persona’s description can be traced back to user quotes or observed behaviors.

Personas are represented as individuals

Personas are user models that are represented as specific, individual humans. They are not actual people, but are synthesized directly from observations of real people. One of the key elements that allow personas to be successful as user models is that they are personifications. They are represented as specific individuals. This is appropriate and effective because of the unique aspects of personas as user models: they engage the

empathy of the development team toward the human target of design. Empathy is critical for the designers, who will be making their decisions for design frameworks and details based on both the cognitive and emotional dimensions of the persona, as typified by the persona's goals. However, the power of empathy should not be quickly discounted for other team members.

Personas represent classes of users in context

Although personas are represented as specific individuals, at the same time they represent a class or type of user of a particular interactive product. Specifically, persona encapsulates a distinct set of usage patterns, behavior patterns regarding the use of a particular product. These patterns are identified through an analysis of ethnographic interviews, supported by supplemental data if necessary or appropriate. These patterns, along with work or lifestyle-related roles define personas as user archetype. Personas are also referred as composite user archetypes because personas are in sense composites assembled by clustering related usage patterns observed across individuals in similar roles during the research phase.

Personas and reuse

Organizations with more than one product often want to reuse the same personas. However, to be effective, personas must be context-specific—they should be focused on the behaviors and goals related to the specific domain of a particular product. Personas, because they are constructed from specific observations of users interacting with specific products in specific contexts, cannot easily be reused across products even when those products form a closely linked suite. Even then, the focus of behaviors may be quite different in one product than in another, so researchers must take care to perform supplemental user research.

Archetypes versus stereotype

Don't confuse persona archetype with stereotypes. Stereotypes are, in most respects, the antithesis of well-developed personas. Stereotypes represent designer or researcher biases and assumptions, rather than factual data. Personas developed drawing on inadequate research run the risk of degrading to stereotypical caricatures. Personas must be developed and treated with dignity and respect for the people whom they represent. Personas also bring to the forefront issues of social and political consciousness. Because personas provide a precise design target and also serve as a communication tool to the development team, the designer must choose particular demographic characteristics with care. Personas should be typical and believable, but not stereotypical.

Personas explore ranges of behavior

The target market for a product describes demographics as well as lifestyle and sometimes job roles. What it does not describe are the ranges of different behaviors that members of that target market exhibit regarding the product itself and product-related contexts. Ranges are distinct from averages: personas do not seek to establish an average user, but rather to identify exemplary types of behaviors along identified ranges.

Personas fill the need to understand how users behave within given product domain—how they think about it and what they do with it—as well as how they behave in other contexts that may affect the scope and definition of the product. Because personas must describe ranges of behavior to capture the various possible ways people behave with the

product, designers must identify a collection or cast of personas associated with any given product.

Personas must have motivations

All humans have motivations that drive their behaviors; some are obvious, and many are subtle. It is critical that personas capture these motivations in the form of goals. The goals we enumerate for our personas are shorthand notation for motivations that not only point at specific usage patterns, but also provide a reason why those behaviors exist. Understanding why a user performs certain tasks gives designers great power to improve or even eliminate those tasks, yet still accomplish the same goals.

Personas versus user roles

User roles and user profiles each share similarities with personas; that is, they both seek to describe relationships of users to products. But persona and the methods by which they are employed as a design tool differ significantly from roles and profiles in several key aspects.

User roles or role models, are an abstraction, a defined relationship between a class of users and their problems, including needs, interests, expectations, and patterns of behavior. Holtzblatt and Beyer's use of roles in consolidated flow, cultural, physical, and sequence models is similar in that it attempts to isolate various relationships abstracted from the people possessing these relationships.

Problem with user role

There are some problems with user roles:

- It is more difficult to properly identify relationships in the abstract, isolated from people who possess them—the human power of empathy cannot easily be brought to bear on abstract classes of people.
- Both methods focus on tasks almost exclusively and neglect the use of goals as an organizing principle for design thinking and synthesis.
- Holtzblatt and Beyer's consolidated models, although useful and encyclopedic in scope, are difficult to bring together as a coherent tool for developing, communicating, and measuring design decisions.

Personas address each of these problems. Well-developed personas incorporate the same type of relationships as user roles do, but express them in terms of goals and examples in narrative.

Personas versus user profile

Many usability practitioners use the terms persona and user profile synonymously. There is no problem with this if the profile is truly generated from ethnographic data and encapsulates the depth of information. Unfortunately, all too often, it has been seen that user profile =s that reflect Webster's definition of profile as a 'brief biographical sketch.' In other words, user profiles are often a name attached to brief, usually demographic data, along with a short, fictional paragraph describing the kind of car this person drives, how many kids he has, where he lives, and what he does for a living. This kind of user profile is likely to be a user stereotype and is not useful as a design tool. Personas, although has names and sometimes even cars and family members, these are employed sparingly as narrative tools to help better communicate the real data and are not ends in themselves.

Personas versus market segments

Marketing professionals may be familiar with a process similar to persona development because it shares some process similarities with market definition. The main difference between market segments and design personas are that the former are based on demographics and distributed channels, whereas the latter are based on user behaviors and goals. The two are not the same and don't serve the same purpose. The marketing personas shed light on the sales process, whereas the design personas shed light on the development process. This said, market segments play role in personas development.

User personas versus non-user personas

A frequent product definition error is to target people who review, purchase, or administer the product, but who are not end users. Many products are designed for columnists who review the product in consumer publications. IT managers who purchase enterprise products are, typically, not the users the products. Designing for the purchaser is a frequent mistake in the development of digital products. In certain cases, such as for enterprise systems that require maintenance and administrator interface; it is appropriate to create non-user personas. This requires that research be expanded to include these types of people.

22.3 Goals

If personas provide the context for sets of observed behaviors, goals are the drivers behind those behaviors. A persona without goals can still serve as a useful communication tool, but it remains useless as a design tool. User goals serve as a lens through which designers must consider the functions of a product. The function and behavior of the product must address goals via tasks—typically as few tasks as absolutely necessary.

Goals motivate usage patterns

People's or personas' goals motivate them to behave the way they do. Thus, goals provide not only answer to why and how personas desire to use a product, but can also serve as a shorthand in the designer's mind for the sometimes complex behaviors in which a persona engages and, therefore, for the tasks as well.

Goals must be inferred from qualitative data

You can't ask a person what his goals are directly: Either he won't be able to articulate them, or he won't be accurate or even perfectly honest. People simply aren't well prepared to answer such questions accurately. Therefore, designers and researchers need to carefully reconstruct goals from observed behaviors, answers to other questions, non-verbal cues, and clues from the environment such as book titles on shelves. One of the most critical tasks in the modeling of personas is identifying goals and expressing them succinctly: each goal should be expressed as a simple sentence.

22.4 Types of goals

Goals come in many different varieties. The most important goals from a user-centered design standpoint are the goals of users. These are, generally, first priority in a design, especially in the design of consumer products. Non-user goals can also come into play, especially in enterprise environments. The goals of organizations, employers, customers,

and partners all need to be acknowledged, if not addressed directly, by the product's design.

User goals

User personas have user goals. These range from broad aspirations to highly pragmatic product expectations. User goals fall into three basic categories

- Life goals
- Experience goals
- End goals

Life goals

Life goals represent personal aspirations of the user that typically go beyond the context of the product being designed. These goals represent deep drives and motivations that help explain why the user is trying to accomplish the end goals he seeks to accomplish. These can be useful in understanding the broader context or relationships the user may have with others and her expectations of the product from a brand perspective.

Examples:

- Be the best at what I do
- Get onto the fast track and win that big promotion
- Learn all there is to know about this field
- Be a paragon of ethics, modesty and trust

Life goals rarely figure directly into the design of specific elements of an interface. However, they are very much worth keeping in mind.

Experience goals

Experience goals are simple, universal, and personal. Paradoxically, this makes them difficult for many people to talk about, especially in the context of impersonal business. Experience goals express how someone wants to feel while using a product or the quality of their interaction with the product.

Examples

- Don't make mistakes
- Feel competent and confident
- Have fun

Experience goals represent the unconscious goals that people bring to any software product. They bring these goals to the context without consciously realizing it and without necessarily even being able to articulate the goals.

End goals

End goals represent the user's expectations of the tangible outcomes of using specific product. When you pick up a cell phone, you likely have an outcome in mind. Similarly, when you search the web for a particular item or piece of information, you have some

clear end goals to accomplish. End goals must be met for users to think that a product is worth their time and money, most of the goals a product needs to concern itself with are, therefore, end goals such as the following:

- Find the best price
- Finalize the press release
- Process the customer's order
- Create a numerical model of the business

Non-user goals

Customer goals, corporate goals, and technical goals are all non-user goals. Typically, these goals must be acknowledged and considered, but they do not form the basis for the design direction. Although these goals need to be addressed, they must not be addressed at the expense of the user.

Types of non-user goals

- Customer goals
- Corporate goals
- Technical goals

Customer goals

Customers, as already discussed, have different goals than users. The exact nature of these goals varies quite a bit between consumer and enterprise products. Consumer customers are often parents, relatives, or friends who often have concerns about the safety and happiness of the persons for whom they are purchasing the product. Enterprise customers are typically IT managers, and they often have concerns about security, ease of maintenance, and ease of customization.

Corporate goals

Business and other organizations have their own requirements for software, and they are as high level as the personal goals of the individual. "To increase our profit" is pretty fundamental to the board of directors or the stockholders. The designers use these goals to stay focused on the bigger issues and to avoid getting distracted by tasks or other false goals.

Examples

- Increase profit
- Increase market share
- Defeat the competition
- Use resources more efficiently
- Offer more products or services

Technical goals

Most of the software-based products we use everyday are created with technical goals in mind. Many of these goals ease the task of software creation, which is a programmer's goal. This is why they take precedence at the expense of the users' goals.

Example:

- Save money
- Run in a browser
- Safeguard data integrity
- Increase program execution efficiency

22.5 Constructing personas

Creating believable and useful personas requires an equal measure of detailed analysis and creative synthesis. A standardized process aids both of these activities significantly. Process of constructing personas involve following steps:

1. Revisit the persona hypothesis
2. Map interview subjects to behavioral variables
3. Identify significant behavior patterns
4. Synthesize characteristics and relevant goals.
5. Check for completeness.
6. Develop narratives
7. Designate persona types

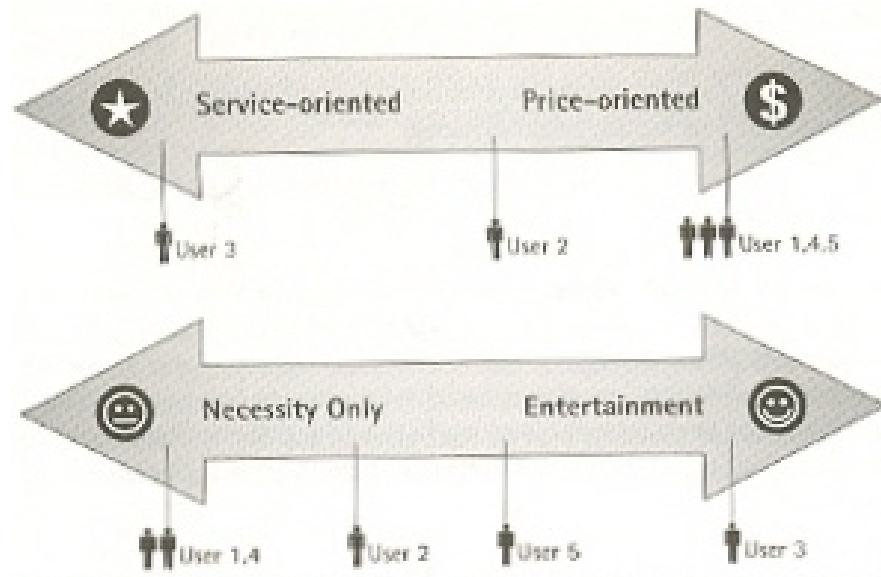
Revisit the persona hypothesis

After you have completed your research and performed a cursory organization of the data, you next compare patterns identified in the data to the assumptions made in the persona hypothesis. Were the possible roles that you identified truly distinct? Were the behavioral variables you identified valid? Were there additional, unanticipated ones, or ones you anticipated that weren't supported by data?

If your data is at variance with your assumptions, you need to add, subtract, or modify the roles and behaviors you anticipated. If the variance is significant enough, you may consider additional interviews to cover any gaps in the new behavioral ranges that you've discovered.

Map interview subjects to behavioral variables

After you are satisfied that you have identified the entire set of behavioral variables exhibited by your interview subjects, the next step is to map each interviewee against each variable range that applies. The precision of this mapping isn't as critical as identifying the placement of interviewees in relationship to each other. It is the way multiple subjects cluster on each variable axis that is significant as shown in figure.



Identify significant behavior patterns

After you have mapped your interview subjects, you see clusters of particular subjects that occur across multiple ranges or variables. A set of subjects who cluster in six to eight different variables will likely represent a significant behavior patterns that will form the basis of a persona. Some specialized role may exhibit only one significant pattern, but typically you will find two or even three such patterns. For a pattern to be valid, there must be a logical or causative connection between the clustered behaviors, not just a spurious correlation.

Synthesize characteristic and relevant goals

For each significant behavior pattern you identify, you must synthesize details from your data. Describe the potential use environment, typical workday, current solutions and frustrations, and relevant relationships with other.

Brief bullet points describing characteristics of the behavior are sufficient. Stick to observed behaviors as much as possible; a description or two that sharpen the personalities of your personas and help bring them to life.

One fictional detail at this stage is important: the persona's first name and last names. The name should be evocative of the type of person the persona is, without tending toward caricature or stereotype.

Goals are the most critical detail to synthesize from your interviews and observations of behaviors. Goals are best derived from an analysis of the group of behaviors comprising each persona. By identifying the logical connections between each persona's behaviors, you can begin to infer the goals that lead to those behaviors. You can infer goals both by observing actions and by analyzing subject responses to goal-oriented interview questions.

Develop narratives

Your list of bullet point characteristics and goals point to the essence of complex behaviors, but leaves much implied. Third-person narrative is far more powerful in conveying the persona's attitudes, needs, and problems to other team members. It also deepens the designer's connection to the personas and their motivations.

A typical narrative should not be longer than one or two pages of prose. The narrative must be nature, contain some fictional events and reactions, but as previously discussed, it is not a short story. The best narrative quickly introduces the persona in terms of his job or lifestyle, and briefly sketches a day in his life, including peeves, concerns, and interests that have direct bearing on the product.

Be careful about precision of detail in your descriptions. The detail should not exceed the depth of your research. When you start developing your narrative, choose photographs of your personas. Photographs make them feel more real as you create the narrative and engage others on the team when you are finished.

Designate persona types

By now your personas should feel very much like a set of real people that you feel you know. The final step in persona construction finishes the process of turning your qualitative research into a powerful set of design tools.

There are six types of persona, and they are typically designated in roughly the ordered listed here:

- Primary
- Secondary
- Supplemental
- Customer
- Served
- Negative

Primary personas

Primary personas represent the primary target for the design of an interface. There can be only one primary persona per interface for a product, but it is possible for some products to have multiple distinct interfaces, each targeted at a distinct primary persona.

Secondary personas

Sometimes a situation arises in which a persona would be entirely satisfied by a primary persona's interface if one or two specific additional needs were addressed by the interface. This indicates that the persona in question is a secondary persona for that interface, and the design of that interface must address those needs without getting in the way of the primary persona. Typically, an interface will have zero to two secondary personas.

Supplemental personas

User personas that are not primary or secondary are supplemental personas: they are completely satisfied by one of the primary interface. There can be any number of

supplemental personas associated with an interface. Often political personas—the one added to the cast to address stakeholder assumptions—become supplemental personas.

Customer persona

Customer personas address the needs of customers, not end users. Typically, customer personas are treated like secondary personas. However, in some enterprise environment, some customer personas may be primary personas for their own administrative interface.

Served personas

Served personas are somewhat different from the persona types already discussed. They are not users of the product at all; however, they are directly affected by the use of the product. Served personas provide a way to track second-order social and physical ramifications of products. These are treated like secondary personas.

Negative personas

Like served personas, negative personas aren't users of the product. Unlike served personas, their use is purely rhetorical, to help communicate to other members of the team who should definitely not be the design target for the product. Good candidates for negative personas are often technology-savvy early-adopter personas for consumer products and IT specialists for end-user enterprise products.

Lecture 23.

Requirements

Learning Goals

As the aim of this lecture is to introduce you the study of Human Computer Interaction, so that after studying this you will be able to:

- Understand the narratives and scenarios
- Define requirements using persona-based design

It has already been discussed how to capture qualitative information about users. Through careful analysis of this information and synthesis of user models, we can get a clear picture of our users and their respective goals. It has also been explained how to prioritize which users are the most appropriate design targets. The missing piece to the puzzle, then, is the process of translating this knowledge into coherent design solutions that meet the needs of users while simultaneously addressing business needs and technical constraints. Now we shall describe a process for bridging the research-design gap. It employs personas as the main characters in set of techniques that rapidly arrive at design solutions in an iterative repeatable and testable fashion. This process has three major milestones: defining user requirements; using these requirements to in turn define the fundamental interaction framework for the product; and filling in the framework with ever-increasing amounts of design detail. The glue that holds the process together is narrative: use of personas to tell stories that point to design.

23.1 Narrative as a design tool

Narrative, or storytelling, is one of the oldest human activities. Much has been written about the power of narrative to communicate ideas. However, narrative can also, through its efficacy at engaging and stimulating creative visualization skills, serve as a powerful tool in generating and validating design ideas. Because interaction design is first and foremost the design of behavior that occurs over time, a narrative structure, combined with the support of minimal visualization tools such as the whiteboard, is perfectly suited for envisioning and representing interaction concept. Detailed refinement calls for more sophisticated visual and interactive tools, but the initial work of defining requirements and frameworks is best done fluidly and flexibly, with minimal reliance on technologies that will inevitably impede ideation.

Scenarios in design

Scenario is a term familiar to usability professional, commonly used to describe a method of design problem solving by concretization: making use of a specific story to both construct and illustrate design solutions. Scenarios are anchored in the concrete, but permit fluidity; any member of the design team can modify them at will. As Carroll states in his book, *Making Use*:

Scenarios are paradoxically concrete but rough, tangible but flexible ... they implicitly encourage 'what-if'? Thinking among all parties. They permit the articulation of design possibilities without understanding innovation Scenarios

compel attention to the use that will be made of the design product. They can describe situations at many levels of detail, for many different purposes, helping to coordinate various aspects of the design project.

Carroll's use of scenario-based design focuses on describing how users accomplish tasks. It consists of an environment setting and includes agents or actors that are abstracted stand-ins for users, with role-based names such as Accountant or Programmer.

Although Carroll certainly understands the power and importance of scenarios in the design process, there can be two problems with scenarios as Carroll approaches them:

- Carroll's scenarios are not concrete enough in their representation of the human actor. It is impossible to design appropriate behaviors for a system without understanding in specific detail the users of the system. Abstracted, role-oriented models are not sufficient concrete to provide understanding or empathy with users.
- Carroll's scenarios jump too quickly to the elaboration of tasks without considering the user's goals and motivations that drive and filter these tasks. Although Carroll does briefly discuss goals, he refers only to goals of the scenario. These goals are somewhat circularly defined as the completion of specific tasks. Carroll's scenarios begin at the wrong level of detail: User goals need to be considered before user tasks can be identified and prioritized. Without addressing human goals, high-level product definition becomes difficult.

The missing ingredient in scenario-based methods is the use of personas. A persona provides a sufficiently tangible representation of the user to act as a believable agent in the setting of a scenario. This enhances the designer's ability to empathize with user mental models and perspectives. At the same time, it permits an exploration of how user motivations inflect and prioritize tasks. Because personas model goals and not simply tasks, the scope of the problem that scenarios address can also be broadened to include product definition. They help answer the questions, "what should this product be?" and "how this product should look and behave?"

Using personas in scenarios

Persona-based scenarios are concise narrative descriptions of one or more personas using a product to achieve specific goals. Scenarios capture the non-verbal dialogue between artifact and user over time, as well as the structure and behavior of interactive functions. Goals serve as filter for tasks and as guides for structuring the display of information and controls during the interactive process of constructing the scenarios.

Scenario content and context are derived from information gathered during the Research phase and analyzed during the modeling phase. Designers role-play personas as the characters in these scenarios, similar to actors performing improvisation. This process leads to real-time synthesis of structure and behavior—typically, at a whiteboard—and later informs the detailed look and feel. Finally, personas and scenarios are used to test the validity of design ideas and assumptions throughout the process. Three types of persona-based scenarios are employed at different points in the process, each time with a successively narrower focus.

Persona-based scenarios versus use cases

Scenarios and use cases are both methods of describing a digital system. However, they serve very different functions. Goal-directed scenarios are an iterative means of defining the behavior of a product from the standpoint of specific users. This includes not only the functionality of the system, but the priority of functions and the way those functions are expressed in terms of what the user sees and how he interacts with the system.

Use cases, on the other hand, are a technique that has been adopted from software engineering by some usability professionals. They are usually exhaustive description of functional requirements of the system, often of a transactional nature, focusing on low-level user action and system responses—is not, typically, part of conventional or concrete use case, many assumptions about the form and behavior of the system to be designed remain implicit. Use cases permit a complete cataloguing of user tasks for different classes of users, but say little or nothing about how these tasks are presented to the user or how they should be prioritized in the interface. Use cases may be useful in identifying edge cases and for determining that a product is functionally complete, but they should be deployed only in the later stages of design validation.

23.2 Envisioning solutions with persona-based design

It has already been discussed that the translation from robust models to design solutions really consists of two major phases. Requirements Definition answers the broad questions about what a product is and what it should do, and Framework Definition answers questions about how a product behaves and how it is structured to meet user goals. Now we look Requirement Definition phase in detail.

Defining the requirements

The Requirement Definition phase determines the what of the design: what functions our personas need to use and what kind of information they must access to accomplish their goals. The following five steps comprise this process:

1. Creating problem and vision statement
2. Brainstorming
3. Identifying persona expectations
4. Constructing the context scenario
5. Identifying needs

Although these steps proceed in roughly chronological order, they represent an iterative process. Designers can expect to cycle through step 3 through 5 several times until the requirements are stable. This is a necessary part of the process and shouldn't be short-circuited. A detailed description of each of these steps follows.

Step1: Creating problem and vision statement

Before beginning any process of ideation, it's important for designers to have a clear mandate for moving forward, even if it is a rather high-level mandate. Problem and vision statements provide just such a mandate and are extremely helpful in building consensus among stakeholders before the design process moves forward.

At a high level, the problem statement defines the objective of the design. A design problem statement should concisely reflect a situation that needs changing, for both the

personas and for the business providing the product to the personas. Often a cause-and-effect relationship exists between business concerns and persona concerns.

For example:

Company X's customer satisfaction ratings are low and market share has diminished by 10% over the past year because users don't have adequate tools to perform X, Y and Z tasks that would help them meet their goal of G.

The connection of business issues to usability issues is critical to drive stakeholders' buy-in to design efforts and to frame the design effort in term of both user and business goals. The vision statement is an inversion of the problem statement that serves as a high-level design vision or mandate. In the vision statement, you lead with the user's needs, and you transition from those to how business goals are met by the design vision:

The new design of Product X will help users achieve G by giving them the ability to perform X, Y and Z with greater [accuracy, efficiency, and so on], and without problems A, B, C that they currently experience. This will dramatically improve Company X's customer satisfaction ratings and leads to increased market share.

The content of both the problem and vision statement should come directly from research and user models. User goals and needs should derive from the primary and secondary personas, and business goals should be extracted from stakeholder interviews.

Step 2: Brainstorming

Brainstorming performed at this earlier stage of Requirements Definition assumes a somewhat ironic purpose. As designers, you may have been researching and modeling users and the domain for days or even weeks. It is almost impossible that you have not had design ideas percolating in your head. Thus, the reason we brainstorming at this point in the process is to get these ideas out our heads so we can "let them go" at least for the time being. This serves a primary purpose of eliminating as much designer bias as possible before launching into scenarios, preparing the designers to take on the roles of the primary personas during the scenario process.

Brainstorming should be unconstrained and critical—put all the wacky ideas you've been considering (plus some you haven't) out on the table and be the prepared to record them and file them away for safekeeping until much later in the process. It's not likely any of them will be useful in the end, but there might be the germ of something wonderful that will fit into the design framework you later create. Holtzblatt & Beyer describe a facilitated method for brainstorming that can be useful for getting a brainstorming session started, especially if your team includes non-designers.

Step 3: Identifying persona expectations

The expectations that your persona has for a product and its context of use is, collectively, that persona's mental model of the product. It is important that the representation model of the interface—how the design behaves and presents itself—should match the user's mental model as closely as possible, rather than reflecting the implementation model of how the product is actually constructed internally.

For each primary persona you must identify:

- General expectations and desires each may have about the experience of using the product
- Behaviors each will expect or desire from the product
- Attitude, past experience, aspirations and other social, cultural, environmental and cognitive factors that influence these desires

Your persona descriptions may contain enough information to answer some of these questions directly; however, you should return to your research data to analyze the language and grammar of how user subjects and describe objects and actions that are part of their usage patterns. Some things to look for include:

- What do the subjects mention first?
- Which action words (verbs) do they use?
- Which intermediate steps, tasks, or objects in a process don't they mention?

After you have compiled a good list of expectations and influences, do the same for secondary and customer personas and crosscheck similarities and differences.

Step 4: constructing context scenarios

Scenarios are stories about people and their activities. Context scenarios are, in fact, the most story-like of the three types of scenario we employ in that the focus is very much on the persona, her mental models, goals, and activities. Context scenarios describe the broad context in which usage patterns are exhibited and include environmental and organizational considerations. Context scenarios establish the primary touch-points that each primary and secondary persona has with the system over the course of a day, or some other meaningful length of time that illuminates modes of frequent and regular use. Context scenarios are sometimes, for this reason, called day-in-the-life scenarios. Context scenarios address questions such as the following

- What is the setting in which the product will be used?
- Will it be used for extended amounts or time?
- Is the persona frequently interrupted?
- Are there multiple users on a single workstation/device?
- What other products is it used with?
- How much complexity is permissible, based on persona skill and frequency of use?
- What primary activities does the persona need to accomplish to meet her goals?
- What is the expected end result of using the product?

To ensure effective context scenarios, keep them broad and relatively shallow in scope. Resist the urge of dive immediately into interaction detail. It is important to map out the big picture first and systematically identify needs. Doing this and using the steps that follow prevent you from getting lost in design details that may not fit together coherently later.

Context scenarios should not represent system behaviors as they currently are. These scenarios represent the brave new world of goal-directed products, so, especially in the initial phases, focus on the goals. Don't yet worry about exactly how things will get accomplished—you can initially treat the design as a bit of a magic black box. Sometimes more than one context scenario is necessary. This is true especially when there are multiple primary personas, but sometimes even a single primary persona may have two or more distinct contexts of use.

An example text scenario

The following is an example of a first iteration of a context scenario for a primary persona for a PDA/phone convergence device and service; Salman, a real-estate agent in Lahore. Salman's goals are to balance work and home life, cinch the deal, and make each client feel like he is his only client. Salman's context scenario might be as follow:

1. Getting ready in the morning, Salman uses his phone to check e-mail. It has a large enough screen and quick connection time so that it's more convenient than booting up a computer as he rushes to make his daughter, Alia, a sandwich for school.
2. Salman sees e-mail from his newest client who wants to see a house this afternoon. Salman entered his contact info a few days ago, so now he can call him with a simple action right from the e-mail.
3. While on the phone with his client, Salman switches to speakerphone so he can look at the screen while talking. He looks at his appointments to see when he's free. When he creates a new appointment, the phone automatically makes it an appointment with client, because it knows with whom he is talking. He
4. Quickly keys the address of the property into the appointment as he finishes his conversation.
5. After sending Alia off to school, Salman heads into the real-estate office to gather the papers he needs for the plumber working on another property. His phone has already updated his Outlook appointments so the rest of the office knows where he'll be in the afternoon.
6. The day goes by quickly, and he's running a bit late. As he heads towards the property he'll be showing client, the phone alerts him that his appointment is in 15 minutes. When he flips open the phone, it shows not only the appointment, but a list of all documents related to client, including e-mail, memos, phone messages, call logs to client's number, and even thumbnail pictures of the property that Salman sent as e-mail attachments. Salman presses the call button, and the phone automatically connects to client because it knows his appointment with him is soon. He lets him know he'll be there in 20 minutes.
7. Salman knows the address of the property, but is a bit unsure exactly where it is. He pulls over and taps the address he put into the appointment. The phone downloads directions along with a thumbnail map showing his location relative to the destination.
8. Salman gets to the property on time and starts showing it to client. He hears the phone ring from his pocket. Normally while he is in an appointment, the phone will automatically transfer directly to voicemail, but Alia has a code she can press to get through. The phone knows it's Alia calling, and uses a distinctive ring tone.
9. Salman takes the call—Alia missed the bus and needs a pickup. Salman calls her daughter and ask her that he will pick her up after 30 minutes.

Now how the scenario remains at a fairly high level, not getting into too specific about interface or technologies. It's important to create scenarios that are within the realm of technical possibility, but at this stage the details of reality aren't yet important.

Step 5: identifying needs

After you are satisfied with an initial draft of your context scenario, you can begin to analyze it to extract the persona's needs. These needs consist of objects and actions as well as contexts. It is preferred not to think of needs as identical to tasks. The implication is that tasks must be manually performed by the user, whereas the term needs implies simply that certain objects need to exist and that certain actions on them need to happen in certain contexts. Thus, a need from the scenario above might be: Call (action) a person (object) directly from an appointment (context) If you are comfortable extracting needs in this format, it works quite well; you can separate them as described in the following sections.

Data needs

Persons' data needs are the objects and information that must be represented in the system. Charts, graphs, status markers, document types, attributes to be sorted, filtered, or manipulated, and graphical object types to be directly manipulated are examples of data needs.

Functional needs

Functional needs are the operations that need to be performed on the objects of the system and which are eventually translated into interface controls. Functional needs also define places or containers where objects or information in the interface must be displayed.

Contextual needs and requirements

Contextual needs describe relationships between sets of objects or sets of controls, as well as possible relationship between objects and controls. This can include which types of objects to display together to make sense for workflow or to meet specific persona goals, as well as how certain objects must interact with other objects and the skills and capabilities of the personas using the product.

Other requirements

It's important to get a firm idea of the realistic requirements of the business and technology you are designing.

- Business requirements can include development timelines, regulations, pricing structures, and business models.
- Technical requirements can include weight, size, form-factor, display, power constraints, and software platform choices.
- Customer and partner requirements can include ease of installation, maintenance, configuration, support costs, and licensing agreements.

Now design team should have a mandate in the form of the problem and vision statements, a rough, creative overview of how the product is going to address user goals in the form of context scenarios, and a reductive list of needs and requirements extracted from your research user models, and scenarios.