INVESTIGATION OF DESIGN AS THE NEXT STEP IN SOFTWARE PRODUCT EVOLUTION: AN ANALYSIS OF ADDED VALUES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EKİN DİNO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL DESIGN

JANUARY 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Fatma Korkut
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Çiğdem Erbuğ
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Mehmet Asatekin	(METU – ID)	_____
Assoc. Prof. Dr. Çiğdem Erbuğ	(METU – ID)	_____
Asst. Prof. Dr. Bahar Şener	(METU – ID)	_____
Dr. Canan E. Ünlü	(METU – ID)	_____
Assoc. Prof. Dr. Tayyar Şen	(METU – IE)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Ekin, Dino

Signature :

ABSTRACT

INVESTIGATION OF DESIGN AS THE NEXT
STEP IN SOFTWARE PRODUCT EVOLUTION:
AN ANALYSIS OF ADDED VALUES

Dino, Ekin
M. Sc., Department of Industrial Design
Supervisor: Assoc. Prof. Dr. Çiğdem Erbuğ

January 2006, 98 pages

Software products are tools that find more uses in the society every day, both professionally and in daily lives of members of the society. This thesis focuses on the problems and possibilities related to current software products. By analysis of the problems and current issues in the software field, possible contribution of a design-oriented approach to software products is explored. The thesis is supported by a study in the form of a semi-structured observation.

Keywords: Human-Computer Interaction, Software Design, Interaction Design

ÖZ

TASARIMIN YAZILIM ÜRÜNLERİNİN EVRİMİNDE
BİR SONRAKİ ADIM OLARAK İNCELENMESİ:
KATILAN DEĞERLERİN ANALİZİ

Dino, Ekin
Yüksek Lisans, Endüstri Ürünleri Tasarımı Bölümü
Tez Yöneticisi: Doç. Dr. Çiğdem Erbuğ

Ocak 2006, 98 sayfa

Yazılım ürünleri toplumda üyelerinin hem profesyonel hem de günlük hayatlarında her gün daha fazla kullanım alanı bulmaktadır. Bu tez yazılım ürünlerinde karşılaşılan sorunlar ve bu ürünler ile ilgili olasılıklar üzerine yoğunlaşmıştır. Yazılım alanında karşılaşılmakta olan problemler ve ilgili konuların incelenmesi yolu ile tasarım kökenli bir bakış açısının olası katkıları araştırılmıştır. Tez yarı-yapısal bir gözlem çalışması ile desteklenmiştir.

Anahtar Kelimeler: İnsan Bilgisayar Etkileşimi, Yazılım Tasarımı, Etkileşim Tasarımı

And so it goes.

Kurt Vonnegut Jr.

ACKNOWLEDGEMENTS

My first and foremost thanks go to Assoc. Prof. Dr. Çiğdem Erbuğ, who provided insight in times of mental block, encouragement in times of despair, and a mild fear and uneasiness in times of laziness. Working, as well as conversing with her is a privilege, as I believe is the feeling of many who did.

Kerem Kuleli, with whom I've had the pleasure of becoming close friends because of this degree, also deserves a big thank you. Not only was he a fun friend but also a responsible one, reminding me of important things like registering for a semester (what can I say; I have a wandering mind).

Members of U-Test were also very helpful with their comments, ideas, effort, and friendship. I look forward to a chance to work together with them again.

I also want to thank some people in my life, who probably do not have much to do with this thesis but deserve recognition for their persistence in bearing with me: Başak Türküler Aka, Özgür Deniz Önür, Yağız Yaşaroğlu, Çağlar Karasu, Medeni Soysal, Muzaffer Büyükkaragöz, Fırat Ant, Emre Yöntem, Refik Burak Atatür, Başar Erdener, the sisters Eren and everyone at Ekodenge (among others), have been with me more or less throughout the process and some were there for the most of my life (and some mostly on wednesdays). I wouldn't want to be without them.

It may not fit in the academic prose but I have a tendency to thank musicians, writers, imaginary characters and inanimate objects who may, or may not be aware that I exist. I indulge myself with their existence. The list of specifics would be too long and somewhat out of place but nonetheless, I tip my hat to all those people, things and ideas that make me who I am.

Last, but not least, I thank my family who believe I can achieve almost anything, even when I know I can achieve almost nothing.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

With their enormous processing power and storage capacity, computers are perhaps the most important technological foundation of our modern lifestyle. Computers have their uses everywhere, secretarial work, arts, economy, astronomy, mathematics, entertainment to name some.

Software products like regular products aim to augment human intellect and capabilities to serve human needs. With increasing usage domains and wider acceptance from the public these products are not only tools for professionals but also a part of our everyday lives.

Even though this great functionality is making computers indispensable, there are major technical and usage problems related to software products. These problems continue to emerge as user demographics using computers widen. The industry responded to these problems by introducing the field of Human-Computer Interaction (HCI) which aims to improve the way humans and computers interact. This field, to date, is mostly dominated by an engineering perspective, trying to come up with generalizable and re-applicable solutions to interaction problems.

This approach not only limits creativity in the software design processes, but also fails to address other hedonic needs of users, taking efficiency as the only viable parameter of human computer interaction.

In this thesis, contribution of design to software products is presented. It is proposed interaction designer should have a responsibility to, parallel to the role of industrial designer for physical products, conceive the products and design them.

1.2 Scope of the Study

This study analyses the parameters and difficulties effecting software development process and the current practices, and elaborate on how a design oriented approach to software product domain (parallel to industrial design for conventional products) can improve software products.

1.3 Structure of the Thesis

The main research question for this thesis is:

- What can be the contribution of a design oriented view to software products?

The related sub-questions are:

- What is the current state, problems and difficulties related to the field?
- What is the current methodology for handling the problems and designing software products?
- What are the competencies of design and how they match software products?
- What should be the competencies of an interaction designer?

The thesis starts with a chapter describing what a software product is, and what the current market conditions, paradigms and problems are.

The next chapter investigates and categorizes the inherent difficulties for software design into three groups as the ones related to technology, market conditions and users.

This is followed by a chapter which describes the field of human-computer interaction, its current areas of interest and practitioners.

The next chapter elaborates on how a design oriented approach (as opposed to an engineering oriented approach) to software products and human computer interaction can contribute to this field, demonstrating the possibilities with an observational study. This chapter also notes the required competencies for an interaction designer.

The thesis concludes with a final chapter summarizing and evaluating the outcome of the study.

CHAPTER 2

SOFTWARE AS A PRODUCT

With the exponential advance in technology and the introduction of digital media, many concepts in our daily life are rapidly changing. The roles of interactive products and personal computers are ever-increasing in our day-to-day chores as well as our social and private lives. The whole society is evolving to incorporate 'information appliances' into its very fibres.

Mayhew (1992) states that the ultimate purpose of a computer-based information system is to improve human performance. Hence the reasons for existence of computer based systems are like those for regular tools mankind has been using for millennia. Brad Weed (1996) describes software tools as: "Productivity software products are tools that enable people to work and to create and are not unlike traditional tools of productivity, like screwdrivers, copying machines, fork lifts or farm machinery" (p10).

First software was designed by engineers to be used by engineers and scientists. The main aim was to harness the processing power for scientific purposes. Many technological constraints were present and their operation required expert knowledge, much like the first cars, or electric motors. Software applications, which emerged as a technological advance available to those in the computer industry and a handful of experts, in the 90s became dominant in the workplace. As the computers started to prove their value as tools of trade for an increasing number of tasks, people without extensive training on computers were required to use them.

With the emergence of the World Wide Web and its applications, and the improvements in multimedia technologies, personal computers have become as common as most home appliances, and user profiles started becoming ever-more

varied. Computers are becoming not only tools for a workplace but also primary means of communication and a very popular pastime. Computer games, online messaging programs, and even just browsing the internet are more popular every day.

According to a research by Stone (2005) 75% of Americans now use the Internet and spend an average three hours a day online. Steven Levy also reports:

> Three quarters of all Americans have access to the Internet, spending an average of twelve-and-a-half hours a week online...for those between 12 and 18, usage approaches 100 percent. Though e-mail is still the No. 1 activity, the study concludes that the Net has profoundly changed the way we spend money, keep in touch with our friends and get information (Internet users use the medium as their No. 1 source of news, despite worries about credibility) (Levy, 2005).

An extract from a study done in 2002 supports this outlook on the future:

> Given a choice of six media, one-third (33%) of children aged 8 to 17 told KN/SRI that the Web would be the medium they would want to have if they couldn't have any others. Television was picked by 26% of kids; telephone by 21%; and radio by 15% ("More Kids", 2002).

.

2.1 Software Products

Software can generally be divided into two parts, what can generally be referred to as the 'business logic', the part of the software program that defines how the work is done, and the 'user interface'.

The user interface is a computer-mediated means to facilitate communication between human beings or between a human being and an artefact. The user interface embodies both physical and communicative aspects of input and output, or interactive activity. The user interface includes both physical objects and computer systems (hardware and software, which includes applications, operating systems, and networks). Metaphors, which are sometimes utilized in computer software interfaces are defined as: **"**A method of overcoming interface complexity by exploiting users' prior knowledge by making interface objects seem like objects the user is familiar with" (Myers, 1994, p76).

A computer in itself is not a device for a specific purpose but a supplier of resources and interfaces, or a 'framework' for software to operate in. A computer with no software operating in would be like a factory with no machines. Donald Norman describes computers as "enablers" and "infrastructure" (Norman, 1998).

Today the market of utilizing that infrastructure with software tools that empower the users for various tasks is a very large one. In 2004 alone 173 million personal computers were sold (Reimer, 2005), and each require operating systems and specific software for various tasks. The richest man alive as of 2004 is in the software business (Forbes, 2005) and some of the largest companies in existence are software companies. Even those companies that do not directly sell software utilize software in various ways. Ericsson, a company renowned for its cell phones employed around 130,000 people, one fifth of which were involved with software development (embedded or otherwise), and 1000 software development projects per year were started within Ericsson Group according to data from 2001, before merging with Sony Corp. (*Carlshamre* & Rantzer, 2001). Microsoft's software sales were worth US $175 billion with an 11% market share in 2001 (McGuire, Ernsberger & Emerson, 2001).

There was even an estimated US $30 million software market for children at the age of 3 and younger in 2000 (Sandberg, 2000).

At this point it is worthy to mention that these studies are widely available for United States of America but not so for other parts of the world. However it is believed that the situation in America may help us extrapolate what the situation is, or will be for the rest of the world.

2.1.1 Technology Acceptance

The computer industry while still driven by technology (See chapter 3.2) is certainly starting to become a norm everywhere. People are sometimes required to use computers, and even when they are not, computers help ease their lives. This is largely due to the functionality that computers offer. This new functionality and its

implications are very impressive to the majority of the population, but this doesn't mean the technology is easy to use.

"During early days of a technology, it doesn't matter if it's hard to use" (Norman, 1998, p24). Similarly, the first cars and phones were hard to use as well. Donald Norman summarizes the acceptance of a product as follows in Figure 2.1:

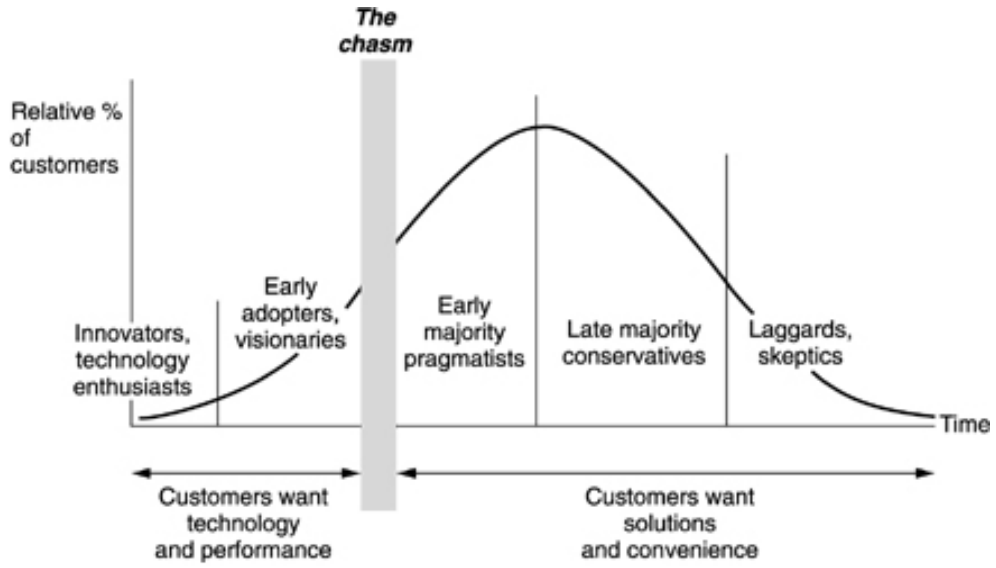

Figure 2.1: The change in customers as technology matures. (from Norman, 1998, p33).

As can be seen in Figure 2.1., several different types of consumers are interested in a product in different phases of its evolution. The first users of every technology are its innovators and a handful of people that follow developments. With introduction to the market some people called 'early adapters', people who are quick to start using a new technology are interested as well. After enthusiasts and early adapters are involved with a technology, market has to make a break through (cross the chasm) to become accepted by the majority of the population. This is the realm where users want solutions and convenience, and these are not always provided by technology alone. Figure 2.2 explains the reasons:

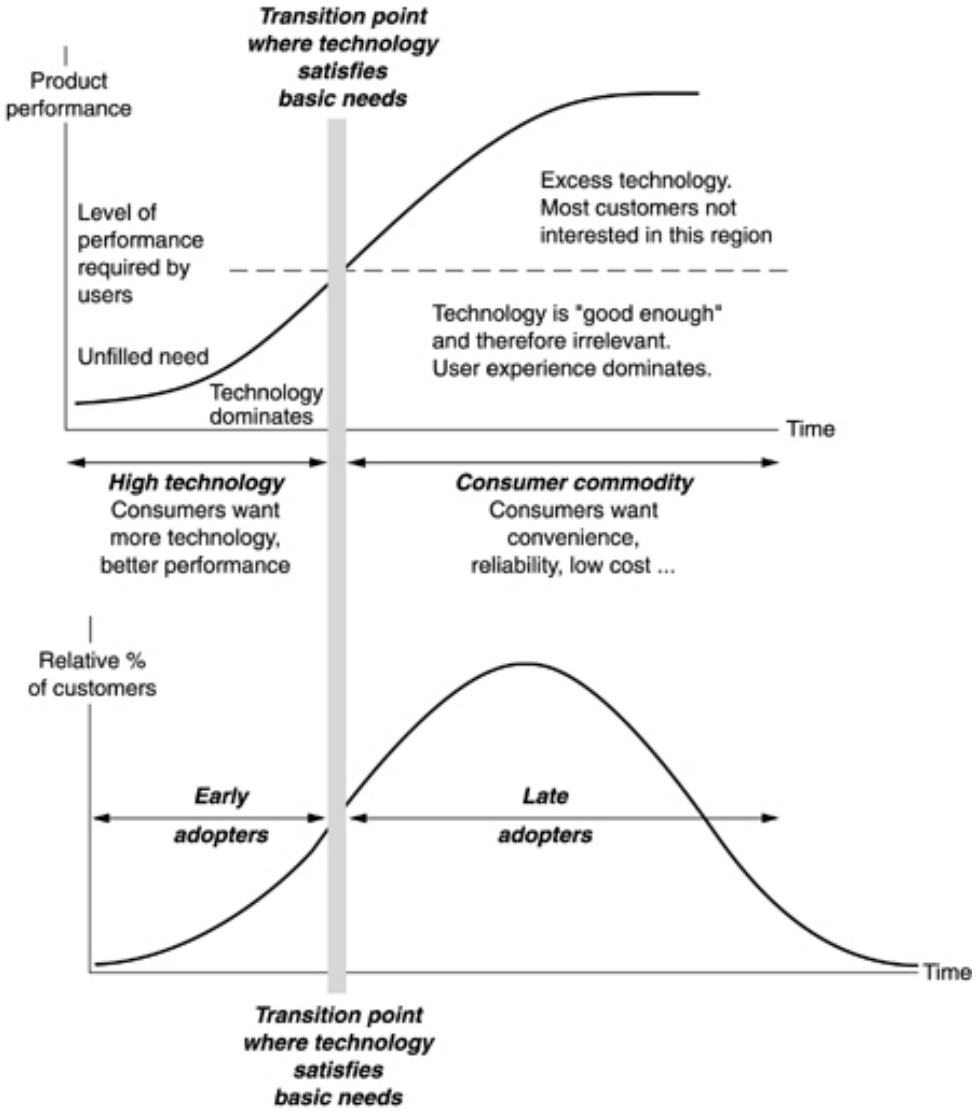

Figure 2.2: The change from technology driven products to user driven, human centred ones (from Norman, 1998, p35).

For the late adapters the determining factor is not technology but their own needs. Users that adapt to a technology in the later phase do so, not because they are interested in technology itself, but because the technology is now a norm, or a necessity. At this point users want to get work done or get satisfaction from things other than the technology itself; hence human factors become more important than technology development. Computers, in feeling the tension between technological power and human needs, can be said to be in the process of 'crossing the chasm'.

Alan Cooper (1999) explains the current situation with computers using an analogy of a dancing bear. He claims that the charm of a dancing bear in a show is not that it

dances well, but that it dances at all. He also claims that with wider acceptance and time, people will recognize that the “bear is actually a terrible dancer” (p26). Once the sheer impressiveness and feeling of supremacy hanging over computers wears off (perhaps when people that have grown with capable computers are the majority), people will concentrate on human capabilities related to computers.

Ben Shneiderman (2002) states that:

> The old computing is about what computers can do; the new computing is about what people can do (p2).

He goes on to describe what he calls ‘new Computing’, a framework for computing technology based on human needs and goals in his book Leonardo's Laptop: Human Needs and the New Computing Technologies.

Jim McCarthy, founder of a software quality training company is quoted in an article by Charles C.Mann (2002):

> Most software products have the necessary features to be worth buying and using and adopting.[...] only the extreme usefulness of software lets us tolerate its huge deficiencies. (p 34)

It can be said that, with increasing acceptance and usage from the society, pure functionality is starting to fall short of providing satisfactory products.

2.2 Problems Related to Software Products

It is evident that not everyone is happy with the way they use the computers. Like the people trying to use complicated machinery that they did not completely fathom after the industrial revolution, untrained computer users find themselves baffled and bewildered. Experience with computers improves the situation, but in the end it is still far from ideal. A popular quotation from Erasmus Smums quoted by Jef Raskin (2000), summarizes the current situation with satire:

> I don't know what percentage of our time on any computer-based project is spent getting the equipment to work right, but if I had a gardener who spent as much of her time fixing her shovel as we spend fooling with our

> computers, I'd buy her a good shovel. At least you can buy a good shovel (p.xi).

Much effort is being put into software products but still there are problems with the way people use them. Inefficiencies and frustrations related to computers are a constant topic in the literature. Three pioneers and leading names of human computer interaction (Norman, 1998; Cooper, 1999; Raskin, 2000) as well as other researchers all claim computers are hard to use as they are today, and there are various books on improving computers and computerized products. Brad Myers (1994) states that “even though American industry has invested heavily in information technology, the expected improvements have not been realised” (p 74).

Studies are conducted on problems with software products that are everyday products for an ever-increasing number of users (Anderson, 1999). Although it is not possible to list all the studies done on software problems, some examples , are presented here to draw a broader picture of the situation.

Operating systems utilizing desktop metaphor are a point of interest and found to be an area quite problematic, even though this metaphor is the de facto standard for most operating systems. One of the studies suggests that the metaphor is used the same way by the users as when it was first introduced (Ravasio, Schär & Krueger, 2004). The introduction of the software metaphor took place together with the introduction of graphical user interfaces 30 years ago and this could imply that several generations of products saw no effective improvements.

A study done on search engine usage concludes that “many users clearly have multiple misconceptions about how search engines process their queries” (Muramatsu & Pratt, 2001).

A study done by observing Microsoft Word usage patterns of participants shows that out of 265 first level functions, 42 were not used by any participant, 118 were used by less than 25% of the participants, and only twelve were used regularly by more than 75% (Baecker, Booth, Jovicic, McGrenere, & Moore, 2000). A second study, tracking Microsoft Word use of 16 participants show that of the 642 commands available, just 20 commands accounted for 90% of use. The average person used

only 57 commands in six months, and all of the users together used only 152 commands in 18 months (Linton, Joy, & Charron,1999).

Another study (Dino & Erbuğ, 2005) conducted on word processors suggests that people propagate their computer knowledge by habits and memorization rather than using software with common sense and ease.

Ben Shneiderman (2000) reports that “A survey of 6000 participants has shown an average of 5.1 hours per week wasted in front of computers, more time is wasted in front of computers than on highways” (p84).

An observation from the Logical User Centered Interaction Design Group (L.U.C.I.D.) homepage:

> The lack of attention to good software design is costing corporations 80 billion dollars a year, according to the *Standish Group*. Research by Professor *Thomas K. Landauer* suggests that the quality of software may be depressing national growth by 3-4%. And, if we care, it is making millions of people miserable.
>
> The Gartner Group has characterized the state of software development as chaos. 25% of software developments fail outright. Another 60% produce a sub-standard product. In what other industry would we tolerate such inefficiency? Imagine if 25% of all bridges fell down or 25% of airplanes crashed (Kreitzberg, n.d.).

This can be seen as the outcome of the immaturity of an industry, which is still in the process of development.

As can be seen from these examples and numerous others, there clearly is something wrong with the way we use computers today. After years of use this inefficiency cannot be blamed on the immaturity of the technology or incapability of the users. Clearly there is something wrong with software products, and this suggests an evolution perhaps in the path of conventional products (see Chapter 5.1).

The problems with software products are more or less the same with most electronic products with digital interfaces. A study done in America shows 43% of the time

people spend with electronic products is spent on figuring out how they work and most people will learn how to use 35% capacity of a product and stop (Lardner, LaGesse, Rae-Dupree, & Roane, 2001). This could point either to an abundance of functionality on product side or a lack of capacity or understanding on user side.

This excess functionality is much more frequent in software because the cost of implementing a function in a specific product, once it is developed, is none. In a physical product interfaces like buttons or LED's all cost money, but a button in a software product costs nothing other than the development costs. In addition to this, to justify the development costs one would need to use the function in as many products as possible. This leads to overcrowding in the products.

A concept defined by Donald Norman (1988) is "Featuritis", or "Creeping Featurism", which means giving priority to the number of features, or capabilities of a product over factors like user needs or ease of use. In Norman's words:

> Complexity probably increases as the square of the features: double the number of features, quadruple the complexity. Provide ten times as many features; multiply the complexity by one hundred (p174).

In essence "Featuritis" or "Bloat" causes human or system performance to diminish to such extent that the addition of the features is not justifiable. By the definitions and data above, it can be said that some of the most widely used software products today are bad cases of "Bloat" or "Featuritis". To quote Jef Raskin (2000);

> "We are justifiably annoyed when a product, a piece of software, or a computer system imposes complexity beyond our needs and presents difficulties we do not understand. We want to do some simple word processing but we are forced to take in hand the hundreds, or in case of Microsoft Office, thousands of commands and methods we do not need." (p143).

These observations, though valid, may not be completely avoidable. Software is a unique area governed by unique conditions as well as conditions valid for other markets. Many variables cause difficulties for the design of software products, and these will be examined in chapter 3.

CHAPTER 3

DIFFICULTIES RELATED TO SOFTWARE PRODUCTS

Producing software is, like producing conventional products, a multidisciplinary and complicated process which requires expert knowledge. Additional to inherent complexities of conventional products, or complex system development, some different considerations are present as well. The time where most of these difficulties are felt is when users try to interact with computers to achieve a goal.

Mick and Fournier (1998) observe eight paradoxes of technological products as:

> Control/chaos: Technology can facilitate regulation or order, and technology can lead to upheaval or disorder.
> Freedom/enslavement: Technology can facilitate independence or fewer restrictions, and technology can lead to dependence or more restrictions.
> New/obsolete: New technologies provide the user with the most recently developed benefits of scientific knowledge, and new technologies are already or soon to be outmoded as they reach the marketplace.
> Competence/incompetence: Technology can facilitate feelings of intelligence or efficacy, and technology can lead to feelings of ignorance or ineptitude.
> Efficiency/inefficiency: Technology can facilitate less effort or time spent in certain activities, and technology can lead to more effort or time in certain activities.
> Fulfils/creates needs: Technology can facilitate the fulfilment of needs or desires, and technology can lead to the development or awareness of needs or desires previously unrealized.
> Assimilation/isolation: Technology can facilitate human togetherness, and technology can lead to human separation.
> Engaging/disengaging: Technology can facilitate involvement, flow, or activity, and technology can lead to disconnection, disruption, or passivity (p126).

Due to its firm rooting in technology, all these paradoxes are inherent in any software development effort as well hence; software design is very difficult and complicated.

> While traditional industrial design concentrates on the product's functionality and its appearance as an object, interaction design requires a different emphasis because a computer-based device must not only work and look well in itself: it must also be designed so that our interaction with it, the way we exchange information with it and tell it our wishes, is clear and efficient. Only then can it be an experience that improves the quality of our everyday life (Interaction Design Institute IVREA, n.d.).

Brad Myers (1994), after methodologically listing and discussing requirements and difficulties associated with design of user related parts of a software program, reaches the conclusion that "user interface design and implementation are inherently difficult tasks and will remain so for the foreseeable future" (p73).

The three considerations when putting forth a product, according to Alan Cooper (1999) are: What's Capable (Engineering), what's Viable (Business), what's Desirable (Design). A study by Rauch, Kahler, & Flanagan (1996) reports that one third of all software projects fail and top five reasons for this are Lack of user input, lack of executive support, incomplete requirements and specifications, changing requirements, and technological ineptness. These five reasons fit into the classification that is suggested as well. The difficulties related to designing software products will be grouped accordingly as: 'difficulties related to technology', 'difficulties related to the market', and 'difficulties related to the users' for ease of discussion.

3.1 Difficulties Related to Technology

Computers are inherently complicated machines. Computers consist of various types of memory spaces only capable of storing binary data and various elements capable of processing this data. Binary data means data consisting of "on" or "off" states, or in their more popular notation 'zeroes' and 'ones'. The power of a computer system comes from the fact that it can process this data very fast and

accurate. In other words, the power of a computer system is not its versatility, or efficiency but its brute speed and determinism.

Computer languages turn a device whose main capability is doing basic arithmetic and logic operations on binary data (unintelligible to most of the society) into an immensely useful tool. Passing through several layers, these languages convert the raw data into formats users can interpret in various styles and vice versa.
The computer can interpret only what is known as “machine language” which is basically comprised of a specific order of zeroes and ones. One step higher in the hierarchy of languages is assembly languages which are more human readable versions of machine languages. The next step is the high level languages (c++, basic etc.). These high level languages are where most of the programming is done for current software design. These principles are valid for most embedded software as well; the language used to program the menu of a television set, under most circumstances is a high level language.

Programs (Code) written in a high level language is first translated to assembly (using a programs called compilers) and then to machine language (using programs called assemblers) thus are turned into a language “understandable” therefore executable by a computer. For example when a programmer writes a line of code telling the computer to add two and two, then write the result on the screen in green letters, compiler and assemblers convert these instructions to machine language in the most efficient manner.

Due to this hierarchical nature of computer architecture, a computer programmer, if not designing a compiler, programming language or assembler, does not need to know neither machine language, nor assembly. A programmer’s main concern is formulating and implementing a programs look and behaviour in these high level languages. This is a very difficult and demanding job.

Michele Tepper (2002) notes that:

> Windows XP has 45 million lines of code. At that scale, it’s nearly impossible to conceptualize all the ways in which one part of the program might interact with another part, or with another piece of software, and if you can’t conceptualize it, you can’t debug it (p40).

Debugging is clearing a program of bugs by means of repeatedly operating written code to observe results. A bug in the code is:

> …the term [bug] refers to programming flaws--commands that don't accomplish the desired result because computers have a habit of following the letter rather than the spirit of the instructions handed to them (Murphy, 2003, p147).

Most of the bugs in a code are typing mistakes, but one of the first bugs (also known as bug zero) actually was a moth stuck in the relay of a 1945 Harvard Mark II Aiken Relay Calculator, one of the earlier computers, inspiring the name for the concept.

A report prepared by Research Triangle Institute (2002) while summarizing the results of the research states that “over half of software bugs are currently not found until downstream in the development process”, and reports the findings of a case study carried out on two major industries. The findings show that the annual cost of inadequate software infrastructure in these two sectors is $5.85 billion. The same study, with some extrapolation, and taking out software testing costs, claims that the impact of inadequate software testing on U.S. economy is $59.5 billion annually.

As a software product is formed of these components, applicable, and perhaps understandable only by professionals of the field, applying user-centred design, a field in which extensive knowledge of human behaviour is required is not easy. Only exceptional people can comprehend both technological and human requirements and limitations at the same time.

This calls for many layers of communication between the people implementing a product at the technological level, and people designing the product, if they are not the same people. This is a deadlock situation, and certainly is one of the bottlenecks of software design. The differences that are liable to occur between the viewpoints of the people that design and implement a product are discussed in chapter 5.

3.2 Difficulties Related to the Market

Donald Norman (1998), when discussing the reasons for the complexity of the personal computer gives three reasons. These reasons are trying to devise a single

machine for every purpose, a single machine to be used by everyone, and the business model of PC and software market (p77). The first two reasons can be viewed as a part of the third, the business model.

Software market with its billion dollar lawsuits (CNNMoney, 2002; LaMonica, 2003) is a very unique and violent market. A view on Microsoft Corporation's inner environment (Cooper, 1999, p110) gives an idea how stressed this environment can be and how difficult is to produce software products in such a market.

In essence, for the purposes of this study, the difficulties related with the market can be grouped under three items:

- Market's tendency to utilize technological advances as purchase justification and versioning cycle.
- Standardization.
- Market's domination by a few major players: market monopoly.

3.2.1 Technology Driven Market

One of the problems with the way software market works is that it is driven almost solely by technology. The market when putting forth a new product, takes technology not as the tool but also as the starting point of concept generation. Veryzer and deMozota (2005) observe:

> Although concept generation would seem to require an appreciation of both technical capabilities and a clear sense of user needs, in practice this often is not the case—particularly for technology-driven products (p 135).

Problems related with Microsoft's commonly used operating system help visualize the depth and breadth of the difficulties:

> Microsoft released Windows XP on Oct. 25, 2001. That same day, in what may be a record, the company posted 18 megabytes of patches on its Web site: bug fixes, compatibility updates, and enhancements. Two patches fixed important security holes. Or rather, one of them did; the other patch didn't work. Microsoft advised (and still advises) users to back up critical files before installing the patches. Buyers of the home

> version of Windows XP, however, discovered that the system provided no way to restore these backup files if things went awry. As Microsoft's online Knowledge Base blandly explained, the special backup floppy disks created by Windows XP Home "do not work with Windows XP Home (Mann, 2002, p34).

The problems mentioned above are linked to the fact that the programmers were either too rushed or too careless to correct obvious mistakes.

Veryzer and deMozota (2005) discuss in length the effects of technology on user oriented design. Even though this technology centred view is plausible for early adapters of a technology, it is not the main factor for the majority of the population (Norman, 1998; Cooper, 1999). Even so, companies still produce versions faster than the market can absorb them (Sawyer, 2001). The reason for this is the market's tendency to take advantage of the rapid development in personal computer technology, selling as many products as they can. The now legendary Moore's law states that the number of components in an integrated circuit (therefore effectively the speed of a computer) doubles every 18 months (Moore, 1965). This trend was observed by Gordon Moore in 1965 and is still holding, throughout several generations of technology. This leads to proportional increase in computers' technological capabilities, for example the amount of data they can process in a given time.

Technology driven market, in order to justify a new version, uses the advance of technology and new functions as the reasons of purchase:

> As the firms are urged to introduce novel products repetitively to stay in the market, they aim to direct consumer demands for their continuous innovativeness. Therefore, a product's obsolescence is mostly reliant on the introduction of goods with novel features, rather than its adequacy to to achieve a certain task (Gültekin, 2004, p6).

As the users needs do not change as fast as new versions are introduced (sometimes annually), this mechanism cannot be said to be very user-centred. One might even argue that complete satisfaction of a user by an existing version would hinder the sales of the next version of a product:

> If you've developed a great product, your goal is to develop a better one that will make the first one obsolete. If you don't make it obsolete, someone else will (Tapscott, 1995, p.59).

The arguments that claim each new version is as usable (or user-centred) as the technology at the shipment time allows are not valid for the exact reasons discussed in these chapters. The relation between a version's contents and technology is closer to Nathan Myhrvold's (1997) statement (which he named "Nathan's first Law"): "Software is gas – it expands to fit the container it is in", meaning that current software is designed to fill current technological capacities, not satisfy current user needs. Barbara Mirel (2000) reports the story of a software product company which started out as very successful "user first" company but later on was taken over by people who thought "the dazzling power of technology would win the market" and went on to a complete failure. The story of the company (of which the narrator was an employee) clearly exemplifies the differences between user centred and technology oriented viewpoints, and how the market is governed by the latter.

This continuous production cycle that always assumes a "next version" tends to put forth products that include whatever is ready by the shipment date, rather than a product that encompasses a user-centred design.

3.2.2 Standardization

The continuous version cycle mentioned in the previous section brings some considerations with it. One of these considerations is standardization. When products are introduced to the market so frequently, almost no new products can break free of previous products in terms of their interfaces and usage. That would mean additional learning costs on the users' part which is pretty unacceptable, so interface language and standards from a time when human considerations were not so heavily studied tend to persist through new versions. Major software developers force style guides to standardize all software products in an attempt to minimize technological and cognitive overheads (Cooper, 1999).

Another place where standards are discussed is software components used in programming. Steve Sawyer (2001) gives an example:

> The engineering roles of standard software components, including application protocol interfaces and protocols, are another potential force. Standards help reduce the variability of base components and architectures, allowing increased component use and reuse. These standards are often market base in that they are shaped by the practices of several influential vendors and consumers. Such control gives the vendor (or consumer) the ability to influence its own role in the software product market (p 102).

Also, Fagin (1999) comments about standardization of the industry:

> But if we give up adherence to a standard, or decide to replace an older product with a newer one, we must reallocate resources. We are faced with a thorny problem: Under what circumstances should a standard be abandoned? At what point does it make sense to discard the resources invested in methods of older knowledge, and adopt those discovered more recently? We are faced with a standardization / innovation trade-off, an important economic problem. (p16).

Standardization, while a very useful tool, if taken as a starting point for software development, can hinder all innovation efforts and creativity. This is discussed in Chapter 5.

Jef Raskin (2000) summarizes the problem related to standardization and interface development tools that enable users to create interfaces quickly utilizing pre-programmed elements:

> Creating fine interfaces can require undertaking intensive and expensive work. Interface-building tools, such as Visual Basic or Visual C++, are marketed as lowering development costs and speeding implementation. [These tools] enshrine current paradigms and thus unduly limit the scope of what you can do. Similarly, the Macintosh or Windows interface guidelines and a portion of the heuristics presented by books on interface design occasionally give advice that is demonstrably incorrect—often due to the company's need to maintain compatibility with earlier versions of the interfaces and to the misperception that users will inevitably revolt if old, familiar interface methods are abandoned (p4).

The standards of the software industry are also forced by the monopolistic structure of the market, due to the resources invested in developing them technology wise, and to utilize the consumers' previous experiences usability wise.

3.2.3 Market Monopoly

A few companies (e.g. Microsoft Corp., Intel Corp.) hold monopolistic or near-monopolistic positions in the software product market (Spinello, 2003). Especially Microsoft is a unique company in its standing in the global market. Microsoft Windows operating system runs on 94% of the desktop and laptop computers in the world (Metz, 2005). The monopoly and antitrust case against Microsoft can said to be the poster child of monopolization in global economy. The Department of Justice of United States of America's arguments against Microsoft were:

> • Microsoft has a monopoly in the PC operating system market. This monopoly is protected by "network effects".
> • The emergence of the Internet, WWW standards, and Java are seen by Microsoft as a threat to that monopoly.
> • Microsoft's actions in response to this threat (such as its promotion of Internet Explorer through restrictive tie-ins with the Windows operating system) are illegal violations of sections 1 and 2 of the Sherman Act of 1890. They are also anticompetitive and detrimental to consumer welfare.
> • Preventing Microsoft from engaging in these practices will promote innovation and competition in the browser market. This will enhance consumer welfare (Fagin, 1999, p17).

An important point here is the innovation that is prevented by the monopolistic conduct. For example, an article in Economist ("Firefox Swings",2005) claims that the emergence of Firefox, an open source, free web browser in the market, forced Microsoft to invest in web browser development efforts. It is claimed in the article that Microsoft, after fall of Netscape (aforementioned lawsuit), without any competition slowed down development efforts.

An example of innovation in the absence of monopolies would be the Internet itself:

> The best example is the Web itself. Built by Tim Berners-Lee, drawing on years of computer science innovations in the non Windows world, it became a global force far more quickly than Windows did under the rigid control of William H. Gates III. Today, forums of engineers, such as the World Wide Web Consortium, keep the Web shipshape, and no company dictates what features should be added, or when.
>
> As a result, innovations and standards are being spawned at an astonishing pace. You can see this in explosion of MP3, a standard for compressing music files; in XML, which is the next lingua franca of the

> Web; and in the Java programming language, which Sun Microsystems Inc. is handing, toll-free, to the e-commerce community (Gross, 2000, p44).

As can be seen the software market, like other high-tech markets, is under stress from various complicated factors.

3.3 Difficulties Related with the users

There are discussions related what literacy means in today's world (Burniske, 2000). Do basic requirements of living in the society end at knowing how to read and write? Using an automated teller machine or telephone? How much do we have left until using a computer will be as basic a requirement as having a signature? Some companies, even now, require an e-mail address for job applications. Does everyone have to learn how to use computers as they are today or do computers have to change to become usable by everyone?

Goodman as reported by Mick and Fournier (1998) states that:

> ''For the first time, many of us are living in a domestic partnership with machines whose primary feature is to make us feel dumb" (p 130).

Also, Beyer and Holtzblatt (1997), write that some estimates report that only slightly more than 30% of the code developed in application software development ever gets used as intended by end-users and suggest that this is due to developers' lack of understanding of the users.

Brad Myers (1994) observes:

> First command to user interface designers is "know thy user." This has been formalized to some extent by the HCI sub-field of "task analysis." Unfortunately, this is extremely difficult in practice (p 75).

As Myers states, the aspects of design related to the users is very difficult indeed, as here are quite a few important factors that are related to users of software products. For the purposes of this thesis, these issues can be listed as: user variations, cognitive friction, learning, expectations, and context considerations. At another level from these factors, but still very important, is the factor of time.

First of the difficulties arises in tandem with the market model of software products that tries to devise a single product for all users. A product's capacity to be used by everyone is called "Universal usability" and Baecker et.al. (2000) comment on the factors affecting universal usability are:

> A significant impediment to universal usability is the complexity inherent in many of today's software systems. Complex functionality, data complexity, and the complexity of learning about these systems all affect usability (p23).

Many software products today aim for universal usability. Most of the software products are defined by the tasks they accomplish, but not for whom they accomplish it. There is no differentiation between a secretarial word processor and a primary school word processor, even though their needs and user profiles are probably different.

It is also discussed if universal usability should be a goal for every product.

> Developers of horizontal products like operating systems and communication software quickly discover that truly homogenous sets of users are rare. They often end up adding lots of features to their products to attract the broadest number of users. But the products can turn out to be expensive to produce and update (Cusumano, 2003, p16).

As noted above, homogenous user groups are rare. Users have different attributes like learning capabilites, cognitive capabilities or social standing. A product that aims to satisfy all of the requirements could be too general, and a product that aims to satisfy the mean could be too narrow for use (Norman, 1998).

Another issue related to users is cognitive friction. "Cognitive Friction" is defined as "the resistance encountered by a human intellect when it engages with a complex system of rules that change as the problem permutes" (Cooper, 1999, p19). Personal computer, by this definition is certainly a device of very high cognitive friction. A personal computer can be in an unlimited number of states depending on the software installed and the current state of the software. This is demonstrated by an example: typing "erase all" in a word processor just types the words on the screen. Typing "erase all" in another state might actually erase the contents of your

hard drive. Each tool in an application puts the personal computer in a different state, and the use of your mouse changes accordingly. A selected are may be painted green or deleted depending on the current state.

It should be noted that this is not the same thing with task complexity. While playing a musical instrument is difficult and complex (more complex than most software programs) an instrument does not have many states. An input, for example fingering a C sharp note on the guitar results in the exact same result every time.

One of the causes of this high cognitive friction is the number of features included in software programs. These features, most of the time cannot be worked out by common sense but require learning and memorization (Dino & Erbuğ, 2005).

This leads to the third issue related to users. People do not want to re-learn these with every new release so old ways of interaction propagate. A study done in a law enforcement agency by introducing a new more usable software product to replace an old more cumbersome product shows people show very high resistance to new products just because they do not want to re-learn how to do everything, even if it means inefficiency. The cost of learning new technology is too high (Hauck & Weisband, 2002).

Jonathan Grudin (1989) summarizes this problem by the statement: “Ease of learning can conflict with ease of use” (p1166)

And Brad Myers(1994) notes:

> Time is valuable, people do not want to read manuals, and they want to spend their time accomplishing their goals, not learning how to operate a computer-based system (p 74).

The next issue is user expectations. As the market is technology driven, the users demand the functionality as well (Lardner et.al, 2001). People want the functions they use and the functions they don’t in case they need them. Like people that drive 150hp cars that use twice as much fuel than possible at 50km/h, computer users want all the capabilities possible. The research done on this subject shows that even though bloating is considered to be a big problem most of the users do not want stripped down “lite” versions of the software (McGrenere & Moore, 2000).

People want the technology that impresses them. Majority of the population want the latest technology, and functionality (Lardner et.al, 2001). People do know what they want rather than what they need.

Alan Cooper (1999) calls this “the consumer driven death spiral” (p220). When a designer starts to design products according to what users want it is difficult to come up with a product with integrity.

So, in essence, users ask for what turns out to be complexity, as this excerpt from Robert Veryzer (1998) summarizes:

> Changes in a product capability may offer high relative advantage, but such advantages may be offset in part by increased product complexity. Products involving consumption pattern changes can require customers to alter thinking and this can result in resistance to the product (p138).

Another one of user-related issues is the considerations that arise when the computers are used in a context. When a product becomes as ubiquous as computers, the context in which it is used (socially or otherwise) gains importance. Things like file sharing software or instant messaging software cannot be examined outside social context and this cannot be explained in terms familiar to engineers.

As Elaine Ann (2003) reports:

> As one of the prominent industrial design events U.S. IDSA Design About: Interactive Edges summarizes: “We can no longer think about products as isolated objects that are designed, produces, and inserted into people’s lives, nor can we think about products consisting of hardware design and software design. Hardware and Software need to become one, and products need to be thought of as part of a bigger system of objects and spaces (p 2).

Time, as an overall determinant, is another issue that should be considered when designing products. Time is a crucial factor in many areas like learning, expertise, habit formation or even bonding, and should be taken into account when designing.

Although these issues related to users are addressed by many academicians, the field of HCI, in reality, operates in a manner that has a tendency to disregard them.

CHAPTER 4

HUMAN-COMPUTER INTERACTION TODAY

The discipline of Human-Computer interaction is a relatively new one. Not only are computers a relatively new field but also the many facets their interactions with humans came to interest after computers were accepted as products for human use rather than a solely scientific topic or tool.

Bonnie E. John and Len Bass (2001) summarize the state of HCI before 1980 as follows:

> Prior to 1980, neither usability nor architecture were of much concern to the builders of software systems. Most systems were monolithic and both the user interface and the architecture were more emergent than designed. Getting the computer to do useful work at all within the confines of memory and processor speed was still the main challenge. Consequently, it was very hard to change a system in any meaningful way after it was built; specifications were top-down using a waterfall method of development, and user-testing, a technique from human factors and cognitive psychology, was used by the larger and more enlightened organizations (p330)

After 1980, parallel with the emergence of graphical user interfaces and increased public use of computers (Myers, 1998) user interface started to become a separate part of a computer program. John Caroll marks March 1982 National Bureau of Standards Conference called “Human Factors in Computer Systems” as a key event while groups such as Special Interest Group in Human Computer Interaction (SIGCHI) also were formed during this period (Rozanski & Haake, 2003). In the CHI 1985 conference, Professor Allen Newell of Carnegie-Mellon University introduced the then-breakthrough suggestion that psychological science of the user can have an impact on the user interface. The scientific model he suggested had the following properties:

- It focuses on design, not evaluation.
- It has the form of an engineering-style theory based on task analysis, calculation, and approximation.
- It is used by interface designers at design time .(Moran, 1985)

This was the first time a user centred approach was introduced to design-time as opposed to evaluation.

While John Caroll, as quoted by Evelyn P. Rozanski and Anne R. Haake (2003), with a much more recent viewpoint, define HCI as:

> HCI is the study and practice of usability. It is about understanding and creating software and other technology that people will want to use, will be able to use, and will find effective when used. The concept of usability and the methods and tools to encourage it, achieve it, and measure it are now touchstones in the culture of computing (p181)

Additionally, Special Interest Group in Computer Human Interaction (SIGCHI) defines HCI as 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" (SIGCHI, 2004)

It can be seen that today, the relation between humans and computers is starting to be the decisive factor for the software industry. People that can, and will use a software program however difficult or unpleasant it is to use are not the majority of users anymore (they used to be – see Chapter 1). Rauch et.al. (1996) report that "usability is the characteristic most often identified with quality in a survey of 500 business computer users", and go on to state the emphasis of usability of software products with a variety of examples.

It is proposed by McCrickard, Chewar & Somervell (2004) that:

> Understanding HCI will allow the interface designer to produce products that are usable by everyone, extending the impact of computing and communication to a diverse set of users within many domains, academic disciplines, and outside demographics. (p 31)

These high hopes for HCI are only natural, and describe the ultimate aim of the field. When it comes to how to achieve these results there is much discussion in the field

(Ehrlich & Henderson, 2000; Marcus, 2002; Armitage, 2003). It is worth noting that usability testing alone uncover but do not fix design problems. (Ferre, Juristo, Windl & Constantine, 2001).

The field of human computer interaction aims to better the interaction between users and their computerized tools. To achieve this, majority of the practitioners concentrate on usability, and most of the times the main method is usability testing.

> The role of many HCI professionals seems to be squarely in the arena of testing or evaluation. For example, the Usability Professionals Association is 1600 members strong and their conference contains many presentations about testing and evaluation techniques and how to communicate test results to effect change. The panel at CHI2000 entitled `Scaling for the Masses: Usability Practices of the Web's Most Popular Sites' revealed user testing as the main, and sometimes only, technique used by the organizations hosting these sites.(John & Bass, 2001)

4.1 The Backgrounds of HCI Professionals

Traditionally, people whose main profession are computers are mainly engineers or computer scientists. Human-Computer interaction is perceived as just another bit of knowledge or consideration these people need to take into account. Almost all HCI courses or departments aim computer engineers or computer scientists. Brad Weed (1996) states:

> Visit just about any university design department and ask where you can study computer interaction design and you will be led either to the computer lab where student are doing traditional design projects online or to the computer science department. (p10)

Even approaches that try to humanize human-computer interaction are aimed at computer scientists. For example a study that tries to teach interface design with a studio approach is opened for "computer science majors & computer information technology or multimedia minors". One of the conclusions of the study is as follows:

> For HCI educators the difficulty is that we come ill prepared to teach design itself, much less the real world application of it. Traditionally HCI design has been taught as an abstract process of iterative user centred design with a recommended set of design aids such as task analysis,

> GOMS (goals, operators, methods, and selection rules), guidelines, heuristic evaluation and usability testing. (Reimer & Douglas, 2003, p192).

An observation is that “HCI courses within the computer science curriculum tend to be survey courses on various elements of interface design, or a course that has students building cool interfaces” and “As the focus of design has shifted from the command line to graphical interfaces to off-the desktop ubiquous computing paradigms, the computer science undergraduate major must have a solid appreciation of HCI topics to succeed.” (McCrickard, Chewar &Somervell, 2004, p31)

Jerry B. Weinberg suggests teaching ethnography to computer scientists to help with HCI and usability testing. (Weinberg & Stephen, 2002). There are numerous papers that explain usability concepts and techniques to software developers (Ferre et.al., 2001). A study shows that engineers in a usability course “solely focus on overt problems negatively expressed by their users, proposed simplistic and unimaginative solutions, and could not specifically describe their users’ characteristics and intentions” (Sugar, 2001, p247). This is not a problem of engineers’ ability to grasp these concepts but their ability to master these abilities in addition to their current ones. It cannot honestly be expected from people who are already working in one of the most difficult fields to master another with completely different requirements, parameters, and perhaps points of view. A survey reported by Raymond McLeod, Jr (1996) illustrates this point. The survey, spanning 647 college instructors teaching analysis and design, showed that ‘human factors considerations’ and ‘prototyping’ were ranked fifth and sixth in course emphasis. This means the part of a product most relevant to the end user was the fifth most important for the designer of the same product, which is unacceptable for consumer products.

There are human computer interaction courses offered in design departments as well, but these courses tend to focus on multimedia design and adding interactive functionality to conventional products rather than interaction design of software tools.

These complicated machines need expert knowledge to grasp, program and design. But most of these considerations are related to implementation, much like the interior operations of a car, or a television. The industry learned that letting the engineers design parts of a product that will directly affect the end user is not the best approach. Knowledge of humans and how they interact with their environment is a very different area of expertise and interest.

4.2 The Problems Encountered in HCI

When the majority of the people working in a field adopt an engineering perspective, topics of discussion tend to converge on quantifiable and reproducible results. Usability metrics probably are the only quantifiable aspects of the interaction between humans and computers (this is the case in industrial design as well). Probably because of this, current literature falls short of the expectations of the practitioner. Wixon (2003) discusses why current HCI literature fails the practitioner by focusing on criticism rather than coming up with new methods for design, and how the practitioner expects something resembling guidelines. Scott McCrickard, Chewar and Somervell (2004) state that:

> HCI can be viewed as a science. Some argue that interface designs are perfected over time, starting with observation that leads to hypotheses and testing, accumulating knowledge that eventually forms theory. Reducing interfaces to basic units that can be observed and tested in a variety of conditions provides laws that describe how these units interact, leading to new hypotheses and more constraints, rules, and exceptions to rules. (p31)

And also:

> HCI is emerging as an engineering discipline. In reflecting on key objectives of engineering, primary concerns are with efficiency and reliability. […]Training HCI students to solve problems by using procedures and analytical methods supports and extends usability engineering practices. (p31)

This view suggests any kind of design can be broken down into smaller conceptual pieces that can, for future problems be brought together for a new solution, and this is all there is about design. Engineering perspective demands guidelines and reproducible results.

There are studies that turn to software architecture (or the way software is written and implemented) to solve the usability problems that arise (Folmer, van Gurp & Bosch, 2003; John & Bass, 2001) and putting usability among other key attributes of software products (e.g. security, efficiency) during design phase. Equating usability (or any other user factor) of a product to engineering factors doesn't answer the question of who will be making related decisions.

The difficulties related with users are already discussed in Chapter 3.3. Current industry mechanics have a tendency to fail to incorporate these user considerations into HCI. Tepper (2002) claims that the whole model of the software industry causes these problems and gives an example of a programmer friend that cannot understand calendar/email combination programs (which are very widely used - like Microsoft Outlook) because, in his mind they are separate programs with separate functionality. He also writes about the view of developers about users:

> Nathan Myhrvold, Microsoft's former CTO, was quoted in July's MIT Technology Review as saying: "Users are tremendously non-self-aware… Software sucks because users demand it to. (p38)

Cooper (1999, p127) observes that the industry sees the users of software products as "elastic users". He claims that developers define users as they see fit. On one scenario they expect the user to navigate the windows file system, defining the elastic user as "accommodating, computer-literate power user". On another they "find it convenient to step the user through a difficult process with a mindless wizard, defining the elastic user as "obliging, naïve, first-time user".

A letter written to, and printed by Interactions Magazine of ACM, one of the most important magazines dealing with Human-Computer Interaction by a software designer and programmer Thomas McCoy (2002) takes a snapshot of the situation very realistically.

> A major cultural change would be needed, not only with computers but with all technology, to make people realize that bad design can make devices hard to use. (p14)
> [...]
> Most of us don't know squat about usability techniques. We often come from science or engineering backgrounds and have no worries using

> complex systems. When we build applications, we construct them for people like ourselves to use and have difficulty even understanding what their problem might be. (p14)

He goes on to write about how usability experts hand them recommendations and come at the end of the project to test the product, taking no part in design or implementation, only commenting on the final product.

The current state of designers in software companies are summarized by Norman (1998):

> Most companies have a few people scattered here and there who work on the user interface (variously called or). There are technical writers and industrial designers, perhaps a few graphical designers. But these people are seldom in one organization, seldom given much power. They are usually relegated to the junior, minor ranks, called upon at the tail end of product development to "make it easy to use," "make it pretty," "explain how to use it." This is not the way to deliver quality user experience (p.49).

Another point that Norman, among others, makes is that many of the problems related to software (and many interactive products) arise from the fact that the initial problems related to these problems are ill-defined. HCI deals with the question of solving usability problems related to software problems but it is not clear who is putting forth the initial specifications and requirements (the problem) related with a product. The process of putting forth the interaction problem to be solved is a multi-disciplinary and complicated task in itself.

In the light of the above discussion, another viewpoint, or role focusing on user needs and capabilities can both ease the process and help create results that are both original and usable. This viewpoint need not only focus on concrete factors like efficiency or usability, but hedonic needs and opportunities as well, synchronizing the efforts with the computerization of our societies.

CHAPTER 5

A DESIGN ORIENTED VIEW FOR SOFTWARE PRODUCTS

To be able to discuss a “design oriented” view, a definition of design is should be made first. There are things that industrial designers or architects can do that mechanical or civil engineers cannot, hence these professions exist.

Design is a word with many meanings. For someone interested in fashion, design is what separates one piece of clothing from another. For an electrical engineer, design is what needs to be done to make a circuit board operate in par with specifications. These two designs surely have little in common. The American Heritage Dictionary of the English Language (The American Heritage Dictionary of the English Language: Fourth Edition, 2005) has 17 meanings under “design” both nouns, and verbs.

Fallman (2003) writes about the three accounts of design, namely, conservative, romantic, and pragmatic accounts. Conservative account is a view on design based on scientific and engineering disciplines. The design process roots from requirements, or specifications and step by step advances towards a result (solution). This is achieved by a methodology in conservative account and every method is well documented and replicable. This shows that conservative account assumes a problem is to be solved and rational steps lead to a solution.

The second account is the romantic account which holds the designer above all else. It holds art as a role model, and accepts the presence of a “mystical element” in design. Replication is not only improbable but also discouraged, creativity is promoted over methodology, and individual judgment over logical reasoning.

Because of this the design process is a black box as opposed to the transparency of the conservative account.

The third and final account is the pragmatic account. This account deals with specific design situations; hence a design is never disconnected from its surroundings and their effects. This means each design "situation" is unique. The designer is a "bricoleur", a tinkerer, who designs by reflection-in-action (Schön, 1983) rather than applying methodologies and theories.

Fallman summarizes these three accounts by the following table:

Table 5.1: Three Accounts of Design. (from Fallman, 2003, p227)

	Conservative Account	**Pragmatic Account**	**Romantic Account**
Designer	An information processor; a *'glass box'*	A reflective, know-how bricoleur; a *'self-organizing system'*	A creative, imaginative genius; an artist; a *'black box'*
Problem	Ill defined and unstructured; to be defined	Unique to the situation; to be set by the designer	Subordinate to the final product
Product	A result of the process	An outcome of the dialogue; integrated in the world	A functional piece of art
Process	A rational search process; fully transparent	A reflective conversation; a dialogue	Largely opaque; mystical
Knowledge	Guidelines; design methods; scientific laws	How each problem should be tackled; compound seeing; experience	Creativity; imagination; craft; drawing
Role model	Natural sciences; engineering; optimization theory	Bricolage; human sciences; sociology	Art; music; poetry; drama

This view of design is helpful in determining the relevant factors and specific situations may require a mixture of the three accounts. The pragmatic account, in evaluating each unique design situation and evaluating, seems to encompass the

tools of the other two accounts, and is helpful to understand the proposed view for HCI.

Another division in design is to two categories: engineering design and creative design (Löwgren, 1995). The main difference between the two is that engineering design assumes the problem definition is precise and final and seeks to find a solution-a singular solution. While doing so engineering tries to optimize many factors. Hence engineering design is a transition from requirements (abstract) to artefact (concrete). In engineering design, the identity of the designer is immaterial due to its basis in methodologies.

In contrast, creative design starts with questioning and trying to understand the problem itself:

> Creative design is about understanding the problem as much as the resulting artefact. Creative design work is seen as a tight interplay between problem setting and problem solving. In this interplay, the design space is explored through the creation of many parallel ideas and concepts. The given assumptions regarding the problem are questioned on all levels. Creative design work is inherently unpredictable. Hence, the designer plays a personal role in the process (Löwgren,1995, p88).

Table 5.2: A summary of generalized differences between the two perspectives. (from Löwgren,1995,p88).

	Engineering design	**Creative design**
process as a whole	entirely convergent	has divergent aspects
key question	how: determining how to solve problem	why: is this the right problem to solve?
stages in the solution	sequential: one candidate solution is refined	parallel: many alternatives are explored
nature of process	analytical: can be described as structured	creative: inherently unpredictable
purpose	one satisfying solution	explore possibilities before committing
ownership	impersonal: designer is an objective instrument	personal: the designer is "present" in the design; assumes responsibility for social action

Both these classifications show a contrast between what design is for engineers, and for other "designers" like architects and industrial designers. For the sake of fluency, the terms "engineering design" will be used for the design that is done by

engineering and “creative design” will be used for the design that is the topic of the chapter.

Another definition is done within the elements of design as; logos, ethos and pathos:

> In his analysis of design as rhetoric, and the designed object as argument, Buchanan (1989) discusses three aspects of the design argument: logos, ethos, and pathos. Logos is technological reasoning, “the way the designer manipulates materials and processes to solve practical problems of human activity.” .For instance, most hammers are based on the basic premise of a lever supporting a head, made in a size fitting comfortably in a person’s hand. Ethos is the character of the designed product, as it reflects its maker: “part of the art of design is the control of such character in order to persuade potential users that a product has credibility in their lives”. Pathos, finally, concerns emotional persuasion and can be considered the connection between design and fine arts. Emotional persuasion is often in focus when we think about “form”, e.g., persuasion in terms of “beautiful” or “artful” design”(quoted in Redström, 2001, p11).

Taking these as the parameters of design, we can conclude that engineering design is neither interested in nor capable of ethos and pathos, due to personal, and human centred aspects of both. Logos can, but ethos and pathos cannot be generalized, impersonalized or formulized, so cannot be brought forth by an engineering perspective. These aspects are also related with the Meta product:

> According to Monö, a meta-product is the result of “all the interpretations and ideas ‘behind’ the physical product, such as prejudices, status, nostalgia, group affiliation and so on” (Redström, 2001, p 10).

Also, Bryan Lawson (1990) in his book “How Designers Think” discusses in length the nature of design problems, design solutions and designers. His views lead to a less structured solution, and perhaps show that design cannot be formulized, but is an act of direct involvement in a specific design situation with numerous variables. One point mentioned in the book is that the aim of design is not optimization, or standardization (p 22-23).

A designer needs to understand people, the technology he/she is designing for, and do the design itself (Lawson, 1990, p6). This requires the designer to be involved directly with various aspects of the whole process:

> This process [design process –in which something is created] of giving form to something calls for a certain level of participation and commitment on behalf of the people that are involved in the design process. This metaphorically resembles the way carpenters in a direct way be must be involved with the materials of carpentry; its physical tools, techniques, and materials. Without this direct involvement, something new cannot be brought into being, whether a baker, a software engineer, or an industrial designer. To design is hence about getting oneself involved in a conscious aim to create and give form to previously nonexistent artefacts, i.e. to make things work in the real world (Fallman, 2005, p2).

The discussions above have their reflection on the practical world of industrial design as well. The creative designer's ability to understand and evaluate requirements, and develop personal outputs that not only address specific goals but added values is the driving factor in several product fields. What makes one product better than another in the eye of a consumer and what gives a customer pleasure is not only "Logos" but also "Ethos" and "Pathos" (the Meta-product); not engineering but aspects of creative design. This would not be so when the same product was first invented, a time when technology was the driving factor. But it can be argued that hedonic pleasures arise from the usage, or owning of a new technology per se.

Chris Conley (2004), an academic of industrial design summarizes the core competencies of design that are relevant to industry as:

> 1. The ability to understand the context or circumstances of a design problem and frame them in an insightful way
> 2. The ability to work at a level of abstraction appropriate to the situation at hand
> 3. The ability to model and visualize solutions even with imperfect information
> 4. An approach to problem solving that involves the simultaneous creation and evaluation of multiple alternatives
> 5. The ability to add or maintain value as pieces are integrated into a whole
> 6. The ability to establish purposeful relationships among elements of a solution and between the solution and its context
> 7. The ability to use form to embody ideas and to communicate their value (p46).

These competencies are bound to fit the rising requirements in the software field, and the role suggested to provide added values based on these competencies is "Interaction Designer".

5.1 The Role of Interaction Designer

Interaction design is considered a sub-discipline of design which examines the role of embedded behaviours and intelligence in physical and virtual spaces as well as the convergence of physical and digital products. A formal definition is: “That part of the design that will directly affect the ultimate end user of the product. Interaction design may even include the selection of a programming language” (Cooper, 1999, p22).

The role incorporating the design oriented attitude proposed by this thesis is “interaction designer”, and the work done is “interaction design”. This term “interaction” was first coined by Morridge as:

> It describes the design of the behaviour of products, its task flow and structure of information, making technology usable, understandable and pleasant for people to use (quoted in Ann, 2003, p2).

Thackara (2001) reports ten articles that Interaction Design Institute Ivrea of Italy lists about interaction design which can be summarized as:

- People are innately curious, playful, and creative so technology will persist.
- People will not be treated as factors in a system
- Experiences will not be designed for individuals, unless asked.
- We do not believe in idiot-proof technology, because we are not idiots and neither are you.
- We will focus on services, not on things. We will not flood the world with pointless devices.
- “Content” is something you do, not something you are given.
- We will think about the consequences of technology before we act, not after.
- We will not pretend things are simple when they are complex.
- We believe that place matters, and we will look after it.
- We will not fill up all time with content.

These articles, even though loosely stated, show the sensibilities related to design of software products. It ahould be noted that all these sensibilities take people as the centre of argument, not technology. In a sense, these articles are very design

oriented. In fact there are several institutions like Ivrea that root from design, to create interactive products but they concentrate mostly on physical products with interactive capabilities rather than software products.

The suggested role of interaction designer for software products is analogous with the role of industrial designer's physical products.

5.1.1 An Analogy With Industrial Design

The points of analogy of interaction design and industrial design are twofold; the first is the emergence and history of relevant fields, the second is the role they play in the development process.

5.1.1.1 Similarities in the Histories of the Fields

Emergence of industrial design as a renowned discipline follows the events of the industrial revolution. One of the pioneers of the idea of an analogy between industrial design and interaction design, Brad Weed (1996) states:

> Just as the industrial revolution helped form the discipline of industrial design, the technological revolution is helping form interaction (p11).

Another analogy made with industrial revolution is that the businesses fitted with steam power and the businesses designed for steam power were very different and same will go for the computer age. The meaning of this is that the businesses designed assuming the presence of a computerized world will make much more difference than those fitted with computers as an add-on (Ehrlich & Henderson 2000).

The examination of the events after the industrial revolution gives us clues about what to expect. After the technologies brought forth by the industrial revolution "crossed the chasm" (See chapter 2.2.1) the discipline of industrial design was introduced to make them usable, likeable and even buyable by the majority of the public. The cutting edge technology (e.g. electricity at the beginning of the century) that was not available to the majority of the public when it was first introduced, step

by step becomes the norm of the society. For example a food processor, while the technology it utilizes is relatively new (hundred years), cannot be said to be a cutting edge technological product anymore. If anything, the design of the food processor is centred around human needs, desires, and abilities; the design attributes that are in the field of industrial designers. The human centeredness of computing technologies can expected to be the logical next step for the society.

The first technological designs were dominated by technology at the beginning of the century. Ergonomics (usability), a branch sometimes attached to design, and sometimes engineering became popular with military implications during the world wars, and is still a well respected field sometimes overrode design. The then fledgling field of industrial design mainly had exterior contributions mostly aesthetic. After saturation of the market, and development of industrial design as a discipline, design is the coordinating and decisive element in any product development effort (Lorenz,1990, p3-15; Veryzer & de Mozota,2005). The same pattern can be followed in the development of computer technologies, technology domination, followed by engineering centred usability considerations and exterior contributions of design. Even several glimpses of the power of a design oriented approach especially for Macintosh computers (Alben, Faris & Saddler, 1994) was seen in early 90's . The logical next step would be the human centred design and development process in the footprints of industrial design.

The emergence of streamlining in industrial design is an interesting example in the histories of both fields. Streamlining was a famous style of external contribution of industrial designers (most famous of them Raymond Loewy – see Figure 5.1) to some mechanical designs in mid-century. Most striking examples are the trains and automobiles of that period, which utilized curvy, stylish exteriors that had little to no meaning other than their aesthetic contributions (Lorenz, 1990, p15). The point of interest here is that the exact phrase "streamlining" has been used by Yahoo! (see Figure 5.2) among others after redesigning the interfaces of their applications, utilising softer edges for their interaction elements. (Olsen, 2002).

Figure 5.1: Streamlined pencil sharpener of Howard Loewy (from Official Website of Raymond Loewy,n.d.).

Figure 5.2: Yahoo! Interface that is classified as streamlined by the company (from Yahoo! Mail website, n.d.).

Another similarity between the course of industrial design products and software products can be observed in the area of ergonomics. Ergonomics used to be the dominant factor in design especially for work related tasks, but is now a contributing factor among others (e.g. style, pleasurability, hedonic factors etc.) (Berkman, 2002). Similar to this, the field of human computer interaction firstly based usability testing on the factors of effectiveness and efficiency, rooting from considerations related to work-related tasks. The phase of industrial design when ergonomics (or usability) was emphasized over every other factor (like user hedonics) seems to be replicated by the field of human computer interaction with a lag of several decades.

In an overview, Weed (1996) notes three pioneers of industrial design, namely Raymond Loewy, Walter Dorwin Teague, and Henry Dreyfuss, and how their approaches are to be mirrored in interaction design:

> What is most interesting about these three pioneers of industrial design is their unique contributions to the discipline of design. Loewy was skilled in incorporating marketing constraints and making products desirable. Teague was skilled at incorporating engineering constraints and making products useful. And Dreyfuss was skilled at incorporating human constraints and making products usable. These three elements: desirability, usefulness, and usability, lead not only to the creation of great physical consumer products, but also to the creation of great software products. (p9)

As computing technology and specifically computers become tools of everyday use, people not interested in computing technology itself, but the uses of it start to use it with increasing frequency. The three properties mentioned above (desirability, usability, usefulness) are bound to be the driving factors of this emerging market involving computers. A role similar to industrial design seems unavoidable for computing technologies.

5.1.1.2 The Roles in the Process of Product Development

The development process of software products is going through an ongoing evolution, and the roles involved are frequently redefined as well. Figure 5.3 summarizes the evolution this process and related roles has been going through so far.

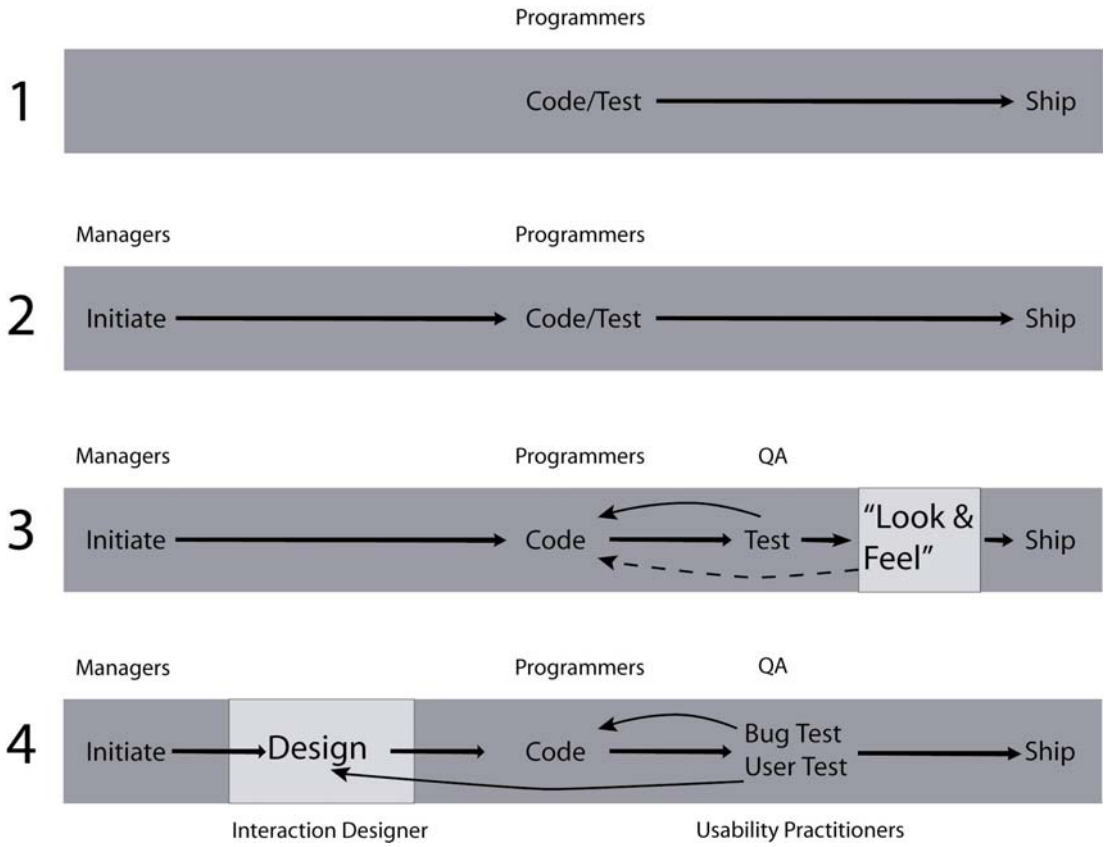

Figure 5.3: Evolution of software development process (adapted from Cooper 1999, p6)

The numbers represent the 4 periods software development has gone through at various times. The related flows will be briefly explained below:

Flow 1: The evolution started with the invention of computers. The programmers, who were generally the users as well, did all the work. These people programmed computers for their own uses and scientific purposes.

Flow 2:.Computer programs started to be in demand from other areas and purposes. Requirements were introduced for specific software and managers were brought in to analyze the market and specify the requirements. The work of realizing software itself was completely done by programmers.

Flow 3: With the maturation of the industry, evaluation became a separate profession and a step in the process. Design and usability became involved, but later on in the process focusing mostly on visual presentation. “Today, common

practice includes simultaneous coding and design followed by bug and user testing and then revision."(Cooper, 1999, p6)

Flow 4 (Suggested Flow): User needs and design is the focus of the process. Design precedes all coding and all decisions are made before programming. Programmer's role is to realize the design decisions, in other words implement them. This is much like what other engineering disciplines that work in collaboration with design oriented professions do.

The person responsible for the design phase of, for the final flow, is the "interaction designer". This role shoulders design responsibilities; has knowledge of the market to recognize what is desirable, has knowledge of production methods to know what is realizable, and most importantly has knowledge of the users to know what is useful(and usable). This role in software production is the same with industrial design for other product areas.

5.1.2 Engineering and Design Oriented Perspectives

Another important similarity between industrial design and interaction design would be the clash of engineering and design perspectives.

Engineering design involves separation of design and analysis. Design for engineering is a way of fulfilling the design specifications and design can be hierarchically decomposed (Löwgren, 1995). No mention of the mechanism of forming or evaluating the requirements is made, so it is unclear who prepares the requirements for an engineering problem, they are just assumed to be given.

The clash of engineering perspective and design perspective has been an echoing topic in literature (Veryzer, 2005). This is also an escalating issue in HCI:

> Just as the early pioneers of industrial design struggled amidst the armies of engineers to make the products usable, useful, and desirable, so too must the designers of the software industry strive to make software products usable, useful, and desirable (Weed, 1996, p11).

Cooper (1999) illustrates the point by telling of a case where a writer named Fred Moody worked with Microsoft through the life-cycle of a project. Other than the turbulent tale of the project, a passage worth noting is this:

> Designers come to Microsoft from the arts; developers from the world of math and science. Developers looked down on designers because their thinking seemed fuzzy and unstructured, their tastes arbitrary. Designers felt that developers were unimaginative, conservative, and given to rejecting their designs out of hand without trying to find a way to make them work. Because programming was inexplicable to designers, they had no way of assessing a developer's insistence that their designs were unprogrammable. "Designers ", Tom Corrdry liked to say, "are invariably female, are talkative, live in lofts, have vegetarian diets, and wear found objects in their ears. Developers are invariably male, eat fast food, and don't talk except to say 'not true'" (p113).

These events take place before interaction design was an issue, and the "designers" mentioned are the graphic designers employed by Microsoft, but the clash and even the absence of designers responsible for the operation of the product itself illustrate the point.

Charles Simonyi who led a team of programmers in the development of 'Bravo', the first "what you see is what you get" (better known as WYSIWYG) word-processing editor comments about the two perspectives:

> [There are two meanings to software design]. One is, designing the artefact we're trying to implement. The other is the sheer software engineering to make that artefact come into being. I believe these are two separate roles -- the subject matter expert and the software engineer." (quoted in Rosenberg, 2004, p1)

This coincides with the clash of engineering and creative design as explained by Löwgren (1995):

> In short, professional software development models in general are based on an engineering design perspective. In one sense this is perfectly appropriate given that they were intended to govern the internal design, i.e. the construction of the software. There is no doubt they contribute to the verifiability, maintainability, and other properties that are crucial to the quality of the product from a constructional point of view. The problem arises when they are interpreted as models for external design i.e., design of the external behaviour and appearance of the product, the services it offers its users and its place in the organization (p88).

Fallman (2004) introduces two views of design that he claims is also valid for human computer interaction. “Design-oriented research” and “research-oriented design” (see Figure 5.4). The former tries to generate knowledge, utilizing design as a field of experimentation, and the latter tries to generate artefacts, using research as a source of tools.

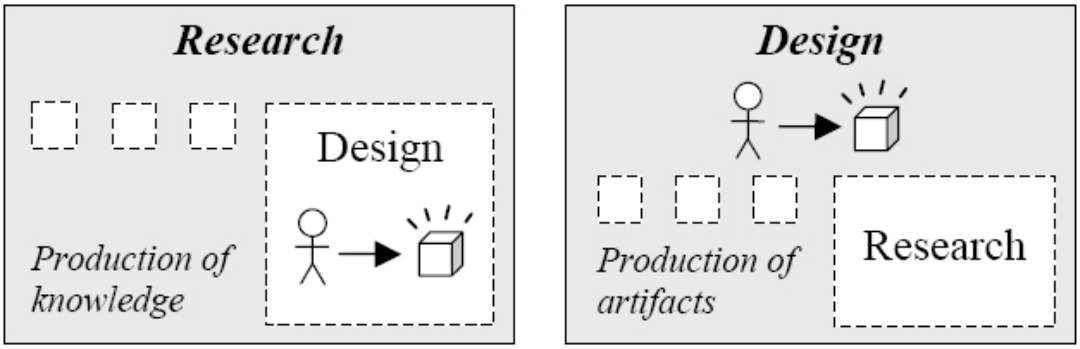

Figure 5.4: Design Oriented Research and Research Oriented Design (from Fallman, 2003, p231).

This distinction matches the distinction between engineering and creative design perspectives. In fact, most of current usability work fits the definition of “design-oriented research”, trying to come up with replicable knowledge. Usability utilizes design ideas, their implementations and tentative designs (like paper prototypes) to reach generalizable conclusions about usability (knowledge).

5.2 Relation Between Usability and Design

Current literature of HCI deals mainly with usability problems. Efficiencies with standard interface elements, categorizations and so forth. These studies root mainly from engineering, traditional usability and cognitive psychology. (See chapter 4). This approach leads to standardization, produces guidelines that are aimed to be used by software developers (mainly engineers) for their software to be usable. Human centred iterative cycles are suggested to developers, in order to test their software with users and better them with each cycle.

This is like letting engineers design a complete car, and testing it with users in order to better its design. No innovative product can be expected to come out of this cycle.

Brad Weed states that iterative design cannot replace having good designers (Weed, 1996). Furthermore this view of design keeps the whole idea of a specific design situation, and context which are central to creative design out of the equation. Djajadiningrat et al. (2000) argues that:

> "While usability is often a laudable goal, it isn't enough. Focusing on ease of use tends to encourage a narrow view of what 'use' is with respect to technology, emphasising efficiency and productivity over exploration or curiosity. With a correspondingly narrow range of models for usability, interaction tends to be self-similar, mundane, and ultimately boring" (p66).

An example for this is a study (Hauck & Weisband, 2002) where a new software with much higher usability measurement results was introduced to a law enforcement agency in which an older software was already being used. Much resistance was shown to the new software because of the context in which it was being used. While the new software was much better in terms of functionality, it was not developed for the specific design situation, and there was a high resistance to change.

Another issue about usability testing is that usability tests that are easier to implement, take shorter time, and use up less funding are tests that require people to use the software for a first time for a short period. This kind of testing that is the method most used in the industry is more like "learnability" tests, omitting the factor of usage time (Mendoza & Novick, 2005). This might be valid for everyday products that do not require any form of training or getting-used-to, but not for professional tools or products for which learning is a determining factor. Some devices require learning, but once learned, usage is a rich and relatively simple process. Same could be possible for some software products. Utilizations of humane factors like this is definitely in the area of design.

Studies show that while consistency is a valuable tool, both with metaphors and with previous products, it is merely a tool but not a goal for each product (Grudin, 1989). Consistency is a by-product of both the marketing strategy which puts out new versions frequently and tries to avoid learning costs, and engineering viewpoint which seeks replicable results and guidelines.

Designing with just usability criteria, and user feedback means letting the users make design decisions. One rule for design is to listen to, and take input from the user but never let the user directly effect a design (Cooper, 1999, p123). This is why professional designers are required. Filtering and analyzing these all inputs into a coherent, useful, usable and pleasurable product is the profession of designers.

Another point of view for the discussion of usability is brought forth by Johan Redström (2001):

> Traditionally, human-computer interaction research tended to focus on how to effectively support people in accomplishing, primarily work related, tasks, and therefore the notion of usability was central. As computational things become everyday things, what we design for can not be restricted to how to enable people to become more productive. Thus, there is a need for complementary design philosophies" (p2).

In conclusion, while usability is a very valuable tool of measurement and may provide insight on design decisions, it cannot replace design or provide ever-valid solutions.

5.3 Possible Contributions of a Design Oriented View

The contributions of a design oriented view to software development process can be in several fundamental areas.

- Design can specify the overall characteristics and properties of a product, shifting the focus from technology to people.
- Design can fuse innovation, breaking free of the standardization tendency of the industry.
- Designers can add meaning to software design in order to produce products that not only function and are usable, but also satisfy hedonic needs.

5.3.1 Shift of Focus from Technology to People

The industry's tendency to focus on technology is already discussed (Chapter 3.2). It is also stated that in the maturity of a product, user needs surpass technological constraints or requirements. As a result of this, cognitive capabilities became important, but in a scientific manner (See chapter 4).

Even without consideration of the emotional aspects of design, there tends to be a tension between technological pressures and user considerations in the market. The constant technological input causes important effects in the market: "(1) product applications often are formulated with only a partial sense of a market, and (2) products may become obsolete before there is an opportunity for refining or evolving them" (Veryzer & deMozota, 2005, p129). In turn, the industry at best tries to keep user considerations in par with technological advances. This usually is at the level of making the users be able to use the technological advances.

This tension is highest in the software industry where there are little to no material production costs, and the cost of including a function, once it is developed is close to nothing. This leads to utilizing every possible technological advance in a product, and frequent versioning of a product. Thackara (2001) defines the current state of the industry as "industrial autism" to point out the detachment of the companies from human beings to focus on technology.

This focus on technology and the ability of users to utilize the technology is actually insufficient to design good products. "The design challenge is to integrate the various technological components into a system that is consistent with existing or evolving consumer usage patterns and needs" (Veyzer & deMozota, 2005, p132). One of the competencies of design is the capability to analyze the users to formulate their needs. An important point here is that formulating user's needs does not mean asking them. As discussed in Chapter 3.3 users may not know what they need or even what they want. Their inputs to the design process, while valuable are not the decisive factor. It is up to the designer to put forth a product.

With a design oriented view, no product is valid for everyone, for the same reasons discussed above. A word processor for high school students can be different from

that for secretarial work, or for retired people. Each product aims a different thing, for different people, adding different values to the design. The first step of such products, the formation of “conceptual” designs are the domain of designers. Each design situation is unique and should be evaluated so.

As Thackara (2001) states: “Interaction design can help shift the focus of innovation from pure technology to the contexts of daily life”(p 51). And:

> As products continue to embody progressively complex technologies and to offer a myriad of capabilities, it is often the user’s ability to understand and to appreciate the product that stands as a principal design constraint as well as being a key element in marketplace success (Veryzer & deMozota, 2005, p131).

When the focus of designs is shifted from technology to people, other factors related to people emerge.

5.3.2 Adding Meaning to Software

There are other factors than functionality and usability for everyday products. There is no generic scissor, or blender that fits every user’s tastes, personality, and needs, even when the functionality is the same. With the increasing usage interactive products and software can said to be everyday things (Redström, 2001). This viewpoint, in parallelism to the events in industrial design history, is forming in HCI as well. Overbeeke et. al. (2003) state:

> The resulting products reflect their maker’s training. Psychologists make products that are very ‘cognitive’ (or instruct designers to do so). Software engineers design interfaces that resemble the logic of programming. […] For too long psychologists have led designers to make overly cognitive designs. We repeat: Design should be left to designers (p.7-8).

This view argues that “user centred design” is evaluated as a scientific discipline that tries to quantify human attributes, which is wrong. The proposition is to put emphasis back on design. It is suggested that the three levels of skills: cognitive (knowing), perceptual-motor skills (doing), and emotional skills (feeling), should be considered in all interaction problems.

As software products excel in functionality, or with the services they offer, user goals related to these services are of increasing importance. Computers are not perceived as productivity tools anymore; they are quickly becoming the prime means for entertainment and communication (Ehrlich & Henderson, 2000). When a product is not solely work related, the dominating factors in its use cannot be accepted as factors like efficiency and effectiveness.

Taking this into account interaction designers, in the last decade tried to find new values in their designs. ACM/Interactions Design Awards Committee's definition of 'Quality of Experience' (see Figure 5.5) shows a shift in the viewpoint of some software (and interactive product) producers:

> By "experience" we mean all the aspects of how people use an interactive product: the way it feels in their hands, how well they understand how it works, how they feel about it while they're using it, how well it serves their purposes, and how well it fits into the entire context in which they are using it. If these experiences are successful and engaging, then they are valuable to users and noteworthy to the interaction design awards jury. We call this "quality of experience" (Alben, 1999, p12).

Figure 5.5: Criteria for ACM/Interactions Design Awards 1996 (from Alben, 1999, p14).

The whole experience related with usage of software products is changing. When predicting the future of software products Jonathan Grudin comments: “Experience is specialized and personal, so design will be much more specialized”(Ehrlich, 2000). The hedonic values that are meaningful to the society and individual therein can be understood and designed for, only by a design oriented view, which seeks not replicability but meaning in its designs.

Raskin (2000) names the new breed of interfaces as “humane interfaces”. He claims: “An interface is humane if it is responsive to human needs and considerate of human frailties” (p.6) and also: “No matter how complex a task a product is trying to accomplish, simple parts should remain simple“ (p1). This humaneness of an interface is what gives it properties other than those required by productivity tools. It is emphasized that these humane bonds are formed with productivity tools as well, even though they are not among the requirements.

Fallman (2005) states that the users have a tendency to use artefacts in ways which the designer did not intend: “the volitions, structures of power, structures of gender, meanings, assumptions, presumptions, beliefs, and worldviews, with which a natural scientist usually does not deal” (p3). Uses of the tools by each unique user vary, and it is the designer’s grasp of these humane factors is what should be essential in artefact design, if our products are to have meaning. This whole idea of a context also falls in the area of design.

5.3.3 Innovation

The pool of available technology is not the guidance factor but a tool for the designer to utilize in realizing the designs. This is also one of the ways in which design can fuse innovation. When design precedes technology, it can force the direction of innovation. Design can be a way of leading technology as well as serving it to the society.

Designers can produce innovative products, in a way that would directly affect the user. While innovation in engineering is the thing that brings technology to a position to realize such products as computers, innovation in serving this technological power to humans requires a different expertise. To be able to design products that

encompass such a broad range of human factors the designer must grasp, and be able to design for human needs. For the same reason, while solving specific design situations, a designer would consult his/her own intuition rather than a guideline. This is not to say a designer would not utilize previous experiences of a user, just that those experiences would be another input to the design, as opposed to starting points. As Raskin (2000) states:

> Where real improvement can be achieved by making major changes, the interface designer must balance the legitimate use of familiar paradigms, which ease the learning process, against the enhanced usability that can be attained by abandoning them. (p4)

As human factors in software industry start to dominate it, the human factors within the development team, as well as those of the end users gain importance (Anderson et. al., 2001). Also, with a design oriented approach, the identity of the designer gains importance, which takes core of design process further away from engineering design. “Interaction designers have to understand people, how they experience things, how they themselves interact, and how they learn” (Ann, 2003). With constant introduction of new contexts for computers this tendency towards human factors is bound to increase, and interaction design will be less about technology than its applications to human contexts, much like industrial design. A good example of these new interfaces is computer games.

5.3.4 Computer Games as an Example for Design oriented View

Computer games are software products that are designed and marketed differently from any other software that are currently on the market. The design process of computer games starts from conceptual phase and is not taken to programming until at a much later stage (Rouse, 2001). The designing of the whole experience and goals are central to the design of games, so neither technology nor cognitive considerations (e.g. usability) are defining factors for the game designs.

The perspective of user research practitioners related to games design is more like a complementary tool:

> At some level, we as user researchers can provide input at nearly every stage of the development process that is useful for design, but in the end, it is not up to us to design a game. (Pagulayan & Steury, 2004, p70)

This approach that focuses on the experience, and defines a process that is entirely design oriented. This approach also has roots in the fact that the goals of a computer game are set by the designer as well. There is no real world goal to achieve.

> Computer users have tasks they need to perform, and are therefore motivated to overcome poorly designed interfaces. With video games, there is no external motivation for the task – if the game's interface is not compelling and entertaining, the product fails in the marketplace. (Gold, Skelly & Thiel, 1994, p177)

The point of importance here is that the entertaining factor, or any factor that is stimulant for the emotional skills of the user are addressed by the designer, as opposed to cognitive or motor skills which, to an extent can be addressed by a thoroughly scientific approach. Interaction design is suggested to be a design oriented field that is considerate of the emotional skills as well as others.

5.3.5 Competencies of an Interaction Designer

Meeting most of the aforementioned challenges is what industrial designers have been doing for conventional products in the last century. Factors like producability, usability, or satisfaction occur in industrial design as well. The role of interaction design should root from the design oriented viewpoint that is used by industrial design, and by utilizing software technologies, help produce the tools of the information age.

> The role of design in HCI must not simply be seen either as a question of problem solving, as an art-form, or a bustle with reality: it is on the contrary an unfolding activity which demands deep involvement from the designer.(Fallman, 2003, p131).

This whole involvement from the designer in the aspects of design requires background knowledge from various fields. As design can be seen as a linking role

for product development, at least communicative knowledge in several fields are required.

The competencies of an interaction designer, taking the list of Conley (2004) (also noted in the start of Chapter 5) as the basis, would be:

1. The ability to understand the context or circumstances of a design problem and frame them in an insightful way
2. The ability to work at a level of abstraction appropriate to the situation at hand
3. The ability to model and visualize solutions even with imperfect information
4. An approach to problem solving that involves the simultaneous creation and evaluation of multiple alternatives
5. The ability to add or maintain value as pieces are integrated into a whole
6. The ability to establish purposeful relationships among elements of a solution and between the solution and its context
7. The ability to use form to embody ideas and to communicate their value

These general competencies or abilities match those proposed for an interaction designer with accuracy. Conceptualization and problem solving abilities are central to providing a product that satisfies user needs without relying excessively on technology and presenting the solution in a relatively limited physical interface (e.g. computer screen, mouse, keyboard). Design is as much about eliminating parts, as it is about adding them.

Those related to the way of thinking (similar to reflection-in-action) help the user to be innovative as well as practical. This way of thinking cannot be represented by guidelines.

The aesthetic and form-creating capabilities are also an important point of discussion. While form in a physical sense is not talked about in software products, graphically communicating ideas is very important in computers. Also, with the advance of computer graphics technology, three dimensional imagery, as well as traditional graphics would be at the designer's disposal. The usage of all these visual media requires a firm grasp of aesthetics and form, which, up to now were added to projects by graphical designers. This separation of roles between

functioning and visualization can also said to be hindering the wholeness of the products, causing eclecticism.

Two other important areas of knowledge that would be required from interaction designers are usability and computer technology knowledge. While the designer would be required to neither conduct usability tests, neither write the code for a software program, knowledge about the terminology and conduct of both fields are essential. Knowledge about usability is required to know the essentials of human capabilities and to be able to communicate with the people who conduct usability tests. Knowledge of computer technologies is required to recognize what possible and producible.

In the light of the above discussion, and taking into account the similarities of industrial design, and the proposed role of interaction design, a study was conducted. This study aimed to demonstrate the different approaches shown by people with engineering and industrial design backgrounds when designing software products.

5.4 A Semi-Structured Observation on Possibilities of a Design Oriented Perspective

The following study was conducted to demonstrate the possible contributions of a design oriented perspective on software design phases. While not aiming numeric or conclusive results, the study shows the differences between the two approaches and gives an idea about the current dilemmas in software development processes.

5.4.1 Study Method

The study is carried out as a “Semi-Structured Observation” (Cohen, Manion & Morrison, 2000, p305). A semi structured observation has an agenda of issues and a predefined setting, but the data is gathered in a less systematic or pre-determined manner. This study in particular does not seek numerical results but insights on the current paradigms.

5.4.1.1 Participants

As the aim of this study was to point out the differences between design approaches and final designs of engineers and industrial designers and examine them in a mainly qualitative manner, a participant set of five industrial designers and five electrical engineers were selected. The engineers were working in the field of computers. Throughout this thesis the engineers will be addressed as E1-E5 and designers as D1-D5. Individual data about them can be found in Appendix A.

The metrics used while selecting the participants were as follows:

- None of the participants had formal HCI education.
- All of the participants are well acquainted with computers and software.
- All of the participants have completed their undergraduate education in their respective fields.

5.4.1.2 The Task and Context

The required task was to tentatively design an e-mail client software to work in an undefined software environment (operating system). The only constraint in the interface requirements was the usage of keyboard and mouse as input, and monitor and audio devices as output devices. The users were asked to sketch the details of an e-mail client program with the most emphasis on interaction. They were told they could add functionality if they wanted but the given functionality was sufficient for completion. They were given the requirements to be fulfilled for the client program on small cards, used paper and transparency mediums for their presentations. The task cards were shuffled before being given to a participant so as not to offer any way of grouping. The task description, and task cards explaining the functionality can be found in Appendix B.

Participants were given no time limit for their design process, though they were informed that it was expected to take between 1.5 and 2 hours. It was assumed that this period is not the kind of time a real design process takes, but was enough for determining the participants' approach to design and software concepts.

There were no metrics of success taken into account as the task was not a usability test but a way of pointing out differences between approaches of the two professions.

Also, at the end of the study a semi-structured interview was conducted with the participants.

5.4.1.3 Test Environment

The study took place in U-Test laboratories (METU/BİLTİR Usability Testing Unit) observation room. No external interference was present other than the periodic visits by the tester.

All the design processes were recorded via two cameras (one recording the participant, one recording the work done).

5.4.2 Experiment Results

While this study was designed to point out the differences presented by engineers and industrial designers, in terms of interaction and functionality (as an evaluation of graphical design would be lopsided for sure) some other points worth mentioning will be discussed here as well.

Before beginning to investigate the test results, it is probably worthy to mention that nine of the participants came up with all around final designs while one participant (D4) came up with no design, though even with no final design some data can be found in the test.

With a participant space of five engineers and five designers, generalizable outcomes were not expected from the study, but many clues hinting a further study could be found about the benefits of a formal industrial design education supply to the field of software design.

No quantitative metrics were introduced during test design period due to the qualitative nature of the experiment, the test results will be discussed under two headings, namely:

- Interface elements used and interaction approach
- Functionality

5.4.2.1 Interface Elements Used and Interaction Approach

Table 5.3 shows the interface elements adopted by the participants directly or with a little manipulation. The participants confirm most of these usages; others are obvious usages of classic windows elements (like the taskbar).

Table 5.3: Interface elements used by the participants. Grey areas represent usage of the elements. D4 presented no final design.

	E1	E2	E3	E4	E5	D1	D2	D3	D4	D5
Tree view										
Tabs										
List view										
Context menu										
Task bar										
Windows style menu										
Drop down combo box										
Link style buttons										
Expanding menus (accordions)										
Check boxes										
Scroll Boxes										
Window structure										
Windows style buttons										
Drag and Drop										

When these usages are examined by themselves, they give no fundamental information on conclusive differences of the approaches of industrial designers and engineers but still form a foundation for later co-relational analysis. Note that these elements are not all possible interface elements and are not of the same weigh or

reoccurrence. These are characteristic elements encountered while examining the final designs submitted by the participants. But even through these tentative figures one would be inclined to note the designers' tendency to avoid them with a 4.25 average element usage to the engineers' 6.6 (excluding D4).

What one cannot deduct from the above table is the new interface elements or styles proposed by the participants. When examining the 4 designs with least number of recognizable elements used, one notices that designs of D3 and D4 propose new interaction styles and designs that can be considered "creative". The designs of E2 and E3 are just "simple" with no new interface element or style introduced.

At this point it is probably worthy to mention that the engineers' designs give the feeling of caring about linking all functionality, but not how those links can be put through to an ordinary user. This is probably due to lack of interaction and perception education, and it is open to discussion if this was expected of them.

When each design is taken into account subjectively (see Figure 5.6), in terms of the interface elements used, the designs of D3 and D5 really stand out due to their unusual interaction styles, Designs of D1 and E2 have somewhat common interaction elements with little twists or differences and D2 uses interaction elements that are readily available, but not from Microsoft domain but Macromedia domain, whose programs he frequently uses. The other designs have no interaction elements other than windows standards. Hence on a subjective scale of independence in terms of interface elements (D4 is included regarding his design approach and comments):

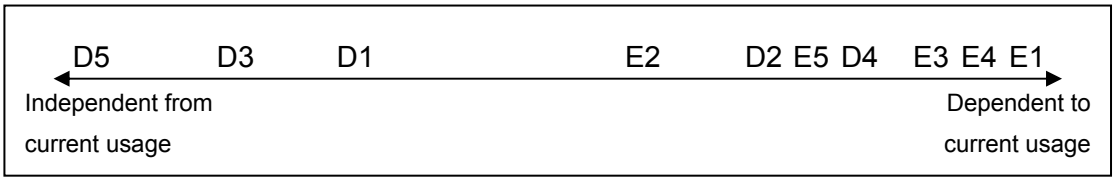

Figure 5.6: Subjective figure showing dependence to current interface elements.

All participants to some degree admitted to being influenced by the software they use when designing the interface elements. In D5 this influence was limited to his

letters and other objects resembling windows while for all engineers except E3 (who changed the way context menus are reached with a little twist) used standard interface elements without questioning. One engineer even saw these details out of the scope of work that was asked of him.

With a broad analysis, the engineers' idea of redesigning the interface seems to be shuffling, and re-organizing standard menus and functions. Designers on the other hand seem to take a more bottom-up approach, reconstructing the interface from the beginning.

5.4.2.2 Functionality

When the task cards and instructions were given to the participants, the points emphasized were interaction style and creativity. When and if participants asked is additional functionality was necessary the answer was "No, but you can add functionality if you see fit". In retrospective, this may have been a bad idea as we now have implementations of everything from virus protection to calendars, which makes quantitative comparisons harder.

Table 5.4: Extra functions used by the participants. Grey areas represent implemented functions. D4 presented no final design.

	E1	E2	E3	E4	E5	D1	D2	D3	D4	D5
Attachments										
Multiple accounts										
Import/export mails										
Send and receive mail buttons										
Folders for storing mails										
Extra data fields in address book										
Signature										
Avatars										
Word tools										
Log keeping										
Virus protection and blocking										
Address book grouping										

Again, like the interface elements discussed in the previous section, these extra functions are the functions included in their designs with some emphasis by the participants.

One of the most important points that arose during the research was the difference between individual approaches to the design process. Some of the participants tried to take the e-mail client program they use or are comfortable with, and re-design it, changing the parts they feel are wrong, adding functions they feel are necessary. Other participants examined the given requirements and designed a program from scratch. This difference does not exactly overlap with the engineer/designer distinction, but it is a point worth mentioning that the only designer that tried to directly redesign a program (Microsoft Outlook) failed to present a final design.

During the interviews all the engineers and D4 (who submitted no design) frequently referred to the problems they were having with their current e-mail program, which supports the redesigning approach. This , combined with the participant's strong computer background, may show the strong influence of experience on designs.

Figure 5.7 represents each participant's relative emphasis on extra functionality (D4 is included regarding his design approach and comments).

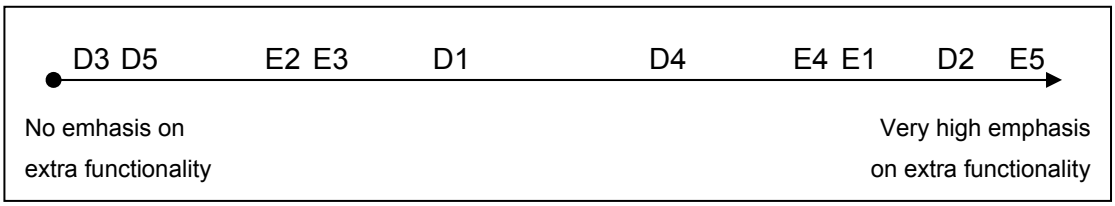

Figure 5.7: Subjective figure showing emphasis on extra functionality.

The relation between the two subjective figures (Figure 5.6 and Figure 5.7) suggests a correlation between lack of extra functionality and independence of interaction language. This, when investigated, puts forward two reasons:

- Less functionality requires less interface elements (as in the designs of E2 and E3).
- Less emphasis on functionality allows more effort on interaction (As in designs of D3 and D5).

5.4.3 Conclusion of the Observation Study

On retrospective the conclusions that can be achieved are these:

- The engineers' idea of designing a software program is based mainly on improving the functionality (or taking out functionality they see unnecessary) of current software while designers tend to design from scratch for the requirements at hand.
- Engineers had almost no tendency to question basic interface elements while designers started their designs by questioning them (not always falsifying though).
- Designers had the tendency to use drag-and drop interfaces; a graphical and intuitive form of interaction.
- The designers that tried to implement baseline functionality came up with creative results while engineers doing so came up with plain and simple ones. In other words while trying to implement improved functionality (probably especially in a cramped time period) the differences between designers and engineers start to diminish, as the designers too start needing to fall back on accepted interface languages to be able to solve interaction in cramped interfaces. This is probably one of the most important points that arose in this research.

There are fundamental differences in the two professions' approach to software designs even though these differences cannot always be seen in the outcomes of the designs.

While not conclusive, the points discussed above show the capability of a design oriented view to bring a new perspective to software products.

5.4.3.1 Shortcomings of the Study

Being too broad a study and a study aiming just to guide further studies rather than putting forward new data, it cannot be said that it failed expectations. Some shortcomings and loose ends of the study are as follows:

- The participants are not really homogenous, in study like this where individuality has a lot of effect, a much larger space is required for conclusive results.
- Due to loose design requirements in a product, which everyone uses, personal grasp of the problem at hand differed a lot, hindering the comparison process. (Added functionality needs added interfacing. How can a product with the basic requirements be compared to a product that even has virus protection, other than the data that is given by the assumption itself.)
- In the name of easing the pressures of the design process presentations were deemed unimportant, being recorded as they were explained, but this makes them really hard to decode.

A larger scale test with a better distinction of professions and a well defined problem structured for a specific target data (for example “how do they handle selections?”, “How do they handle mouse interaction?”) could even have given scientifically quantifiable results.

CHAPTER 6

CONCLUSION

Considering the difficulties and current paradigms related to software products, it cannot be claimed that software design is an easy process. Looking back on the difficulties related to software products, which were classified under “technology”, “market”, and “user” considerations, it is certain that producing good software products is a hard, multidisciplinary work.

Computers are still considered as a “new” technology. What computers can achieve functionally are persistently exciting for the majority of the population. The ability to edit a document with a word processor or look up a topic in the world wide web are so much easier than the old ways of achieving the tasks that the functionality in itself is a reason for purchase. This is why current industry focuses on technology as the driving factor.

As time passes, computers settle in the ways of the society, transforming from a supplementary technology to a fundamental one. This is so not only for work related tasks, but also in social context. With the depth and breadth of the areas of usage and user profiles, problems with usage of software products arise. Even when the functionality is sufficient issues related to usage and satisfaction are topics that are frequent in literature. As presented in chapter 4, human-computer interaction was introduced to answer the usability problem, mainly as the scientific means to make software products usable.

The usage setting of software products presents a mirror-view of conventional products like utility tools, or home appliances. Software products are tools we use to produce, communicate or entertain ourselves. As discussed in the final chapter, a

design-oriented view for software products may be better suited to match the situation presented by such products.

As the field advances the problems related to users are addressed with frequency, but current trends in human-computer interaction cause this interest to be overtly engineering oriented. Human factors in design of software products are seen as reusable and quantifiable data that is to be used so standardize and better all computer interaction.

As opposed to this search for standardization and quantification, a design oriented view, embracing the uniqueness of each design situation does not seek standardization. On the contrary, a design-oriented view seeks to create unique artefacts that satisfy hedonic needs as well as utilitarian ones. Also in satisfying utilitarian needs, a design oriented view can analyze user needs, and available technologies and offer a set of specific functionality for a specific situation. This helps in shifting the focus from technology to design. An important point here is that both technology and usability are very important scientific disciplines, and the required knowledge about both of them are very important for a designer. Usability knowledge is important because due to the immerse functionality and limited interface of computers human cognitive capabilities are of great importance. Technological knowledge is important to recognize design opportunities and know what is producible. An interaction designer, utilizing a design oriented approach would need to have attributes similar to those of an industrial designer (discussed in chapter 5.3.5).

An interaction designer, utilizing a design oriented approach would need to have attributes similar to those of an industrial designer (discussed in chapter 5.3.5). Other than the knowledge of design and considerations for its sensitivities, an interaction designer would need to have specific knowledge about usability, and computer technologies. Usability knowledge is important because due to the immerse functionality and limited interface of computers human cognitive capabilities are of great importance. Technological knowledge is important to recognize design opportunities and know what is producible.

As a result, to answer the main research question:

A design oriented view is capable of shifting the focus from technology to users, to produce products that can satisfy hedonic needs, as well as functional requirements. This viewpoint could also fuse the creation of more creative software products, as the designer would be inclined to break free of current paradigms.

The interaction designer's role in the production process is to determine the requirements in tandem with marketing, putting forth a design, taking technological limitations into consideration. In this process, usability should be a constant concern, but not a driving factor as excessive focus on usability, as discussed previously would lead to standardization and undermine innovation.

As a conclusion, such a design process, led by the interaction designer, taking humane and social factors, as well as functional needs into consideration, using technology and usability as tools (as opposed to goals) in the process will lead to software products that will better suit human needs.

REFERENCES

Alben, L.,Faris, J., Saddler, H. (1994) Making it Macintosh: designing the message when the message is design, *Interactions*, 1(1), 11-20.

Alben, L. (1996), Quality of experience: defining the criteria for effective interaction design, *Interactions*, 3(3), 11-15.

Anderson, R.(1999), Organizational limits to HCI: conversations with Don Norman and Janice Rohn, *Interactions*, 7(3), 36-60.

Anderson, J., Fleek, F., Garrity, K. & Drake, F. (2001), Integrating Usability Techniques into Software Development, *IEEE Software*, 18(1), 46-53.

Ann, E., (2003), Interaction Design:Industrial Design in the Information Age, Hong Kong: Kaizor Innovation.

Armitage, J. (2003), From user interface to über-interface: a design discipline model for digital products, *Interactions*, 10(3), 18-29.

Baecker, R., Booth, K., Jovicic, S., McGrenere, J. & Moore, G.(2000), Reducing the gap between what users know and what they need to know, *Proceedings on the 2000 conference on Universal Usability*, 17-23, New York: ACM Press

Berkman, A. E.(2002), The Influence of Ergonomics on Marketing and product styling. Ankara: Publications Committee of METU Faculty of Architecture

Beyer, H & Holtzblatt, K.,(1997) Contextual Design, *Interactions*, 6(1),32 -42

Buchanan, R. (1989). Declaration by Design: Rhetoric, Argument, and Demonstration in Design Practice. In: Margolin, V. (Ed.): Design Discourse; History, Theory, Criticism, 91-109. Chicago: The University of Chicago Press.

Bug Hall of Fame (n.d.), http://www.asktog.com/Bughouse/BugHallOfFame.html, Last accessed on January 2, 2006.

Burniske, R.W., (2000), Literacy in the Cyber Age, *Ubiquity*, 1(12), 3

Carlshamre,P. & Rantzer, M. (2001), A Narrative Approach to User Requirements for Web Design, *Interactions*, 7(6), 31-35.

CNNMoney (2002), “Netscape sues Microsoft”, http://money.cnn.com/2002/01/22/technology/netscape/, Last accessed on January 2, 2006.

Cohen, L., Manion, L. & Morrison, K. (2000), Research Methods in Education, 5th Edition, NY: RoutledgeFalmer.

Conley, C. (2004), Leveraging Design's Core Competencies, *DMI Review*, 15(3), 45-51.
Copper, A., (1999), The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity, Indiana: Sams.

Cusumano, M.(2003), Beware the Lure of the Horizontal, *Communications of the ACM*, 46(7),15 -17

Dino, E. & Erbuğ, Ç. (2005), Effects of Mass Marketed Software Products on the Evolution of Software Interface Language, Proceedings of HCI2005.

Djajadiningrat, J. P., Gaver, W. & Fres, J. W. (2000): Interaction Relabelling and Extreme Characters: Methods for Exploring Aesthetic Interactions. In: Conference proceedings on Designing Interactive Systems (DIS) 2000, 66-71. ACM Press.

Ehrlich, K. & Henderson, A. (2000), (Inter)facing the millennium: where are we (going)?, *Interactions*, 7(1), 19-30.

Faggin, B. (1999), Computers, Science, and the Microsoft case, *ACM SIGCAS Computers and Society*, 29(2), 15-22.

Fallman, D. (2003) Design-oriented human-computer interaction, *Proceedings of the SIGCHI conference on Human factors in computing systems*, 225-232.

Fallman, D. (2004) Design oriented-research versus Research-oriented Design, Workshop Paper, *CHI 2004 Workshop on Design and HCI, Conference on Human Factors in Computing Systems*

Fallman, D. (2005) Why Research-oriented Design Isn't Design-oriented Research, *Proceedings of Nordes: Nordic Design Research Conference*.

Ferre, X., Juristo, N., Windl, H. & Constantine, L. (2001), usability basics for Software Developers, *IEEE Software*,18(1), 22-29

"Firefox Swings to the Rescue",(2005), *Economist*, Vol. 377(8457), 64-64.

Folmer, E., van Gurp, J. & Bosch, J.(2003) A framework for capturing the relationship between usability and software architecture. *Software Process: Improvement and Practice,* 8(2), 67-87.

Forbes, (2005),The World's Top Ten Billionaires, Forbes, 175(6), 166-167

Gold, R., Skelly, T. & Thiel, D. (1994), What HCI Designers Can Learn From Video Game Designers, Conference *companion on Human factors in computing systems*, 177-178.

Graham, L. (1999). The Principles of Interactive Design. New York: Delmar Publishers.

Grudin, J.,(1989),The case against user interface consistency, *Communications of the* ACM ,32(10), 1164 - 1173

Hauck, R.V. & Weisband, S. (2002), When a better interface and easy navigation aren't enough: examining the information architecture in a law enforcement agency, *Journal of the American Society for Information Science and Technology* 53(10), 846 - 854

Interaction Design ınstitute IVREA (n.d.), “Interaction Design”, http://www.interaction-ivrea.it/en/about/interactiondesign/index.asp Lat accessed on January 2, 2006.

John, B.E. & Bass, L. (2001), Usability and Software Architecture, *Behaviour & Information Technology*, 20(5), 329-338

Kreitzberg, C. (n.d.), “The L.U.C.I.D. Computing Movement”, http://www.cognetics.com/papers/charlie/charlie2.html, Last accessed on January 2, 2006.

LaMonica, P.R., (2003), “Microsoft to pay AOL $750M”, http://money.cnn.com/2003/05/29/technology/microsoft/, Last accessed on January 2, 2006.

Lardner, J. & LaGesse D., Rae-Dupree, J., & Roane, K. (2001, January 15). Overwhelmed by tech. U.S. News and World Report, 30-36

Lawson, B.(1990), How Designers Think: The Design Process Demystified, Cambridge: Butterworth Architecture.

Levy, S. (2004), No Net? We'd Rather Go Without Food, *Newsweek,* 144(15), 14

Linton, F., Joy, D. & Charron, A. (1999). OWL: A Recommender System for Organization-Wide Learning. *MITRE Technical Report MTR 98B0000025V00S00R00*, USA: MITRE.

Lorenz, C. (1990). The Design Dimension: New Competitive Weapon for Product Strategy and Global Marketing, Cornwall: T.J. Press Ltd.

Löwgren, J.(1995) Applying design methodology to software development, *Proceedings of the conference on Designing interactive systems: processes, practices, methods, & techniques*, 87-95.

Mann, C.C. (2002, July), Why Software is so Bad, *MIT Technology Review*, 33-38

Marcus, A. (2002, September). Dare We Define User-Interface Design? *Interactions,* 9(5), 19-24.

Mayhew, D. (1992). Principles and Guidelines in Software User Interface Design. New Jersey: Prentice Hall.

McCoy, T.(2002), Letter From the Dark side: Confessions of an Applications Developer, *Interactions*, 9(6), 11-15.

McCrickard, D.S., Chewar, C.M., Somervell, J. (2004), Design, science, and engineering topics?: teaching HCI with a unified method, *Proceedings of the 35th SIGCSE technical symposium on Computer science education,* 31-35.

McGrenere, J., & Moore, G. (2000), Are we all in the same "bloat"?, *Proceedings of Graphics Interface* 2000, 187-196

McGuire, S., Ernsberger Jr., R. & Emerson, T.,(2001), Software Pirates, Beware. *Newsweek*, 138(18), 68-72

McLeod, R. (1996), Comparing Undergraduate Courses in Systems Analysis and Design, *CACM*, 39(5), 113-121.

Mendoza, V. & Novick, D.G., (2005), Usability Over Time, Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information, 151 – 158, New York: ACM Press

Metz, C. (2005), "Living in a Windows World", http://www.pcmag.com/article2/0,1895,1861732,00.asp, Last accessed on January 2, 2006.

Mick, D.G. & Fournier, S.(1998), Paradoxes of technology: Consumer cognizance, emotions, and coping strategies, *Journal of Consumer Research*, 25(2), 123-143

Mirel, B, (2000)Product, process, and profit: the politics of usability in a software venture, ACM Journal of Computer Documentation, 24(4) ,185 - 203

Moore, G. E. (1965). Cramming More Components Into Integrated Circuits, *Electronics*, 38(8).

Moran, T.P. (1985), Summary of Allen Newell's CHI '85 Address: "The Prospects for Science in Human-Computer Interaction", *SIGCHI Bulletin*, 17(1)

More Kids Say Internet Is the Medium They Can't Live Without (2002, April), http://www.statisticalresearch.com/press/pr040402.htm, Last accessed on January 2, 2006.

Muramatsu, J. & Pratt, W., (2001), Transparent Queries: investigation users' mental models of search engines, *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval,* 217-224, New York: ACM Press.

Murphy, V. (2003), The Exterminator, *Forbes*, 171(11), 147-150

Myers, B. A., (1994, January). Challenges Of HCI Design And Implementation. *Interactions,* 1(1), 72-83.

Myers, B. A., (1998), A Brief History of Human Computer Interaction Technology, *Interactions,* 5(2), 44-54.

Myhrvold, N, (1997), "The Next Fifty Years of Software", http://www.research.microsoft.com/acm97/nm/, Last accessed on January 2, 2006.

Neil, G. (2000), Lessons From The Web-A World Without Monopolies , *Business Week*, 3687, 44-44.

Norman, D. A., (1988). *The Design of Everyday Things*. New York, NY: Doubleday.

Norman, D. A. (1998) The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution. London: The MIT Press.

Official Website of Raymond Loewy (n.d.), http://www.raymondloewy.com, Last accessed on January 2, 2006.

Olsen, S. (2002), "Yahoo Relaunches With Streamlined Look", http://news.com.com/2100-1023-941301.html, Last accessed on January 2, 2006.

Overbeeke, C.J., Djajadiningrat, J.P., Hummels, C.C.M., Wensveen, S.A.G., & Frens, J.W. (2003). Let's Make Things Engaging. In: Blythe, M.A., Overbeeke, C.J., Monk, A.F., & Wright, P.C. (Eds.) *Funology: From Usability to Enjoyment.* Dordrecht: Kluwer, 7-18.

Pagulayan, R. & Steury, K. (2004), Beyond Usability in Games, *Interactions*, 11(5), 70-71.

Raskin, J. (2000), The Humane Interface: New Directions for Designing Interactive Systems, Massachusetts: Addison-Wesley.

Rauch, T, Kahler, S., & Flanagan, G., (1996) Usability Techniques - What Can You Do?, *SIGCHI Bulletin*, 28(4), 63-64
Ravasio, P.,Schär, S.G. & Krueger, H.(2004), In Pursuit of Desktop Evolution, *Interactions*, 11(6), 9-10

Redström, J. (2001). Designing Everyday Computational Things. Dissertation, Göteborg University.

Reimer, J (2005, December), "Total Share: 30 Years of Personal Computer Market Share Figures", http://arstechnica.com/articles/culture/total-share.ars, Last accessed on January 2, 2006.

Reimer, Y.J. & Douglas, S.A. (2003), Teaching HCI Design with the Studio Approach, Computer Science Education Journal, 13(3), 191-205.

Research Triangle Institute(2002), Planning Report 02-3: The Economic Impacts of Inadequate Infrastructure for Software Testing, Noth Carolina: Author.

Rogers, M. (1999), Monopoly Power, Innovation and Economic Growth, *The Australian Economic Review*, 32(1), 96-104.

Rosenberg, S. (2004) Why Software Still Stinks, http://www.salon.com/tech/col/rose/2004/03/19/programmers_at_work/, Last accessed on January 2, 2006.

Rouse, R., III. (2001). Game Design: Theory & Practice. Plano, Texas: Wordware Publishing, Inc.

Rozanski, E.P. & Haake, A.R. (2003), The Many Facets of HCI, *Proceedings of the 4th Conference on Information Technology*, 180,185

Sandberg, J.,(2000), Multimedia Childhood, *Newsweek*, 136(17a), 78
Sawyer, S (2001), A market-based perspective on information systems development, *Communications of the ACM*, 44(11),97 - 102

Schön, D. A. (1983) *The Reflective Practitioner. How professionals think in action*, London: Temple Smith.

Shneiderman, B., (2002), Leonardo's Laptop: Human Needs and the New Computing Technologies, New York: MIT Press

SIGCHI (2004), "ACM SIGCHI Curricula for Human-Computer Interaction" , http://www.sigchi.org/cdg/cdg2.html, Last accessed on January 2, 2006.

Spinello, R.A. (2003), The case against Microsoft: An ethical perspective, *Business Ethics: A European Review*, 12(2), 116-132.

Stone, B.,(2005), Hi-Tech's New Day, *Newsweek*, 145(15), 60-64

Sugar, W. (2001) What is So Good About User-Centered Design? Documenting the Effect of Usability Sessions on Novice Software Designers, Journal of Research on Computing Education, 33(3), 235-250.

Tepper, M. (2002), Why Software Still Stinks, *NetWorker*, 6(3), 40 - ff

Thackara, J. (2001), Design challenge of pervasive computing, *Interactions*, 8(3), 46-52.

The American Heritage Dictionary of the English Language: Fourth Edition (2005), http://www.bartleby.com/61, Last accessed on January 2, 2006.

Tapscott, D.(1995), The Digital Economy: Promise and peril in the age of networked intelligence. New York: McGraw Hill.

Veryzer, R.W. & de Mozota, B.B. (2005), The Impact of User-Oriented Design on New Product Development: An Examination of Fundamental Relationships, *Journal of Product Innovation Management*, 22(2), 128-143.

Veryzer, R.W. (1998). Key Factors Affecting Customer Evaluation of Discontinuous New Products. *Journal of Product Innovation Management*, 15 (2), 136–150.

Veryzer, R.W. (2005). The Roles of Marketing and Industrial Design in Discontinuous New Product Development. *Journal of Product Innovation Management*, 22 (1), 22–41.

Yahoo! Mail Website(n.d.), http://mail.yahoo.com, Last accessed on January 2, 2006.

Weed, B (1996, July). The Industrial Design Of The Software Industry. *ACM SIGCHI Bulletin,* 28(3), 8 - 11.

Weinberg, J.B. & Stephen, M.L. (2002), Participatory design in a human-computer interaction course: teaching ethnography methods to computer scientists, P*roceedings of the 33rd SIGCSE technical symposium on Computer science education*, 237 – 241.

Wixon, D. (2003), Evaluating usability methods: why the current literature fails the practitioner, *Interactions*,10(4), 28-34.

APPENDIX A

OBSERVATION SUMMARIES

ENGINEER 1 (E1)

Occupation: Electrical and Electronics Engineer

Computer Background: Has used Commodore 64, Amiga 500, And PC. Proficient in Electronic simulation programs, Microsoft Office products, Visual C++, and MatLab. Uses Adobe Photoshop.

Design Process: Started working with cards. Used cards for grouping and didn't work with them much. Quicker than most to start drawing. Didn't change his mind much. Said: "The design will resemble Outlook, but maybe something will come to my mind." (At the very beginning of design).

Final Design:

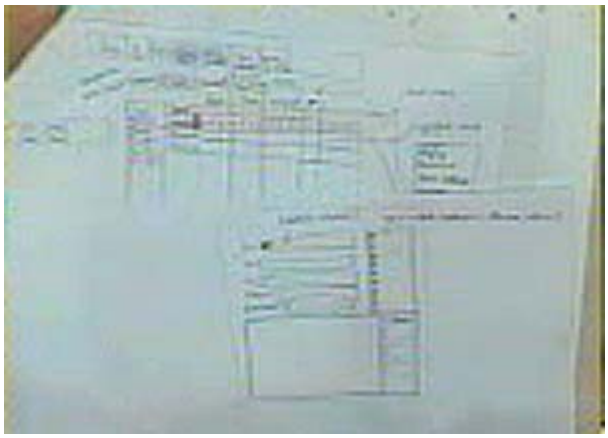

Figure A.1: Design of E1

- The design has “File, Edit, View, Tools, Preferences” menus in the exact Windows manner.
- Window structure is used.
- The design has a “taskbar” alert.
- Uses tabs, tree-view architecture, E-Mail list with headers.
- Interesting use of tabs (tabs are upside down as opposed to their real-life counterparts)

Extra functions are:

- Multiple accounts
- Import/export mails
- Send and receive mail buttons (like in outlook)
- Folder architecture for storing mails
- Extra data fields in address book (telephone, secondary mail etc.)
- Signature
- Attachments

Other Notes: Did not question the basic functionality but tried to improve it by adding little twists. Did not think much on the layout or screen usage. Not a very original design overall, but not lacking in functionality.

ENGINEER 2 (E2)

Occupation: Electrical and Electronics Engineer

Computer Background: Has used PC, and UNIX. Proficient in Matlab, Visual C++, Office Programs

Design Process: Said she wasn't expecting something much different from Outlook right at the beginning. Grouped cards and later said she just read the cards once, and seeing nothing new, she would just check at the end if they were fulfilled. The design process had no rollbacks. She stuck with most of the things she initially tried.

Final Design:

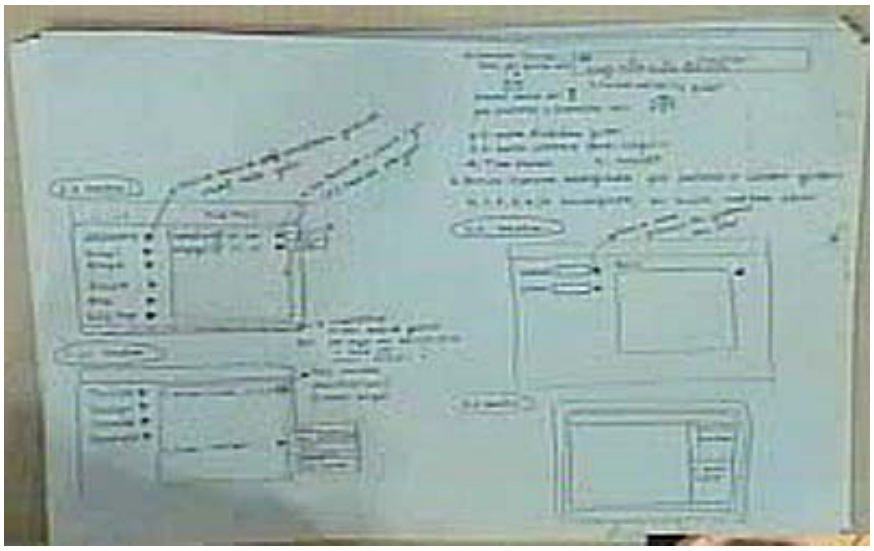

Figure A.2: Design of E2

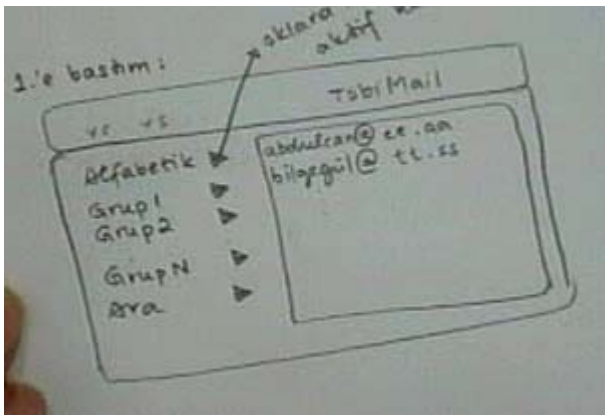

Figure A.3: Design of E2

Design uses Context menus, window structure

Extra functions are:

- Grouping of address book

The final design is very plain and can be considered elegant. Most functionality is straight forward, though almost all interaction is achieved through interaction with little triangles. These triangles function like right mouse button context menus for the item they are near.

Other Notes: Plain and simple design addressing the necessary functionality and not much more. The designer does not find her own work very satisfactory and most of the difference is in interaction wrappers.

ENGINEER 3 (E3)

Occupation: Electrical and Electronics Engineer

Computer Background: Has used PC, and UNIX. Proficient in Matlab, Visual C++, Office Programs

Design Process: Worked an enormous amount of time with the cards (until completely satisfied) and didn't touch them later. After he was satisfied with the arrangement of cards, started sketching with little or no regard to screen usage.

Final Design:

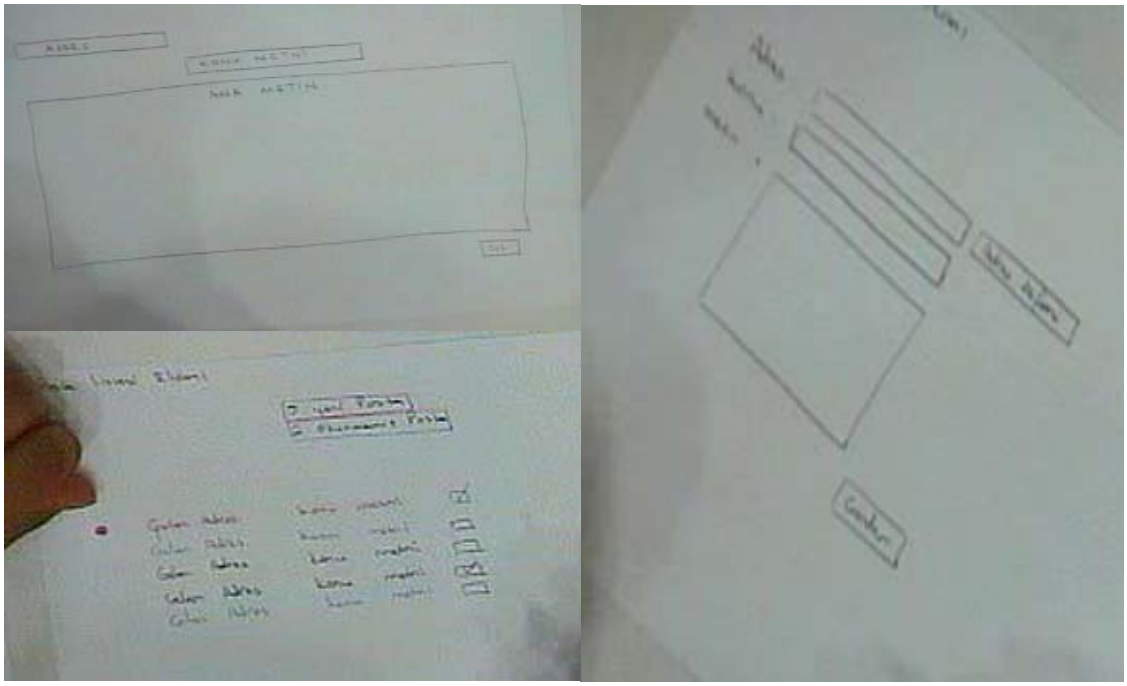

Figure A.4: Design of E3

Design uses window structure, check boxes, and almost all basic windows elements.

Extra functions are:

- Grouping of mails

The final design is very simple looking, but has no regard for screen usage. Some very awkward interfaces (like entering the number of entries to be made, then entering them). Concentrates mostly on what is where, but not “how” it is there. Most improvements were minor modifications to the way outlook or web based mailing programs behave.

Other Notes:

Plain and simple design addressing the necessary functionality and not much more. Lacking the elegance.

ENGINEER 4 (E4)

Occupation: Electrical and Electronics Engineer

Computer Background: Has used PC, and UNIX. Proficient in Matlab, Visual C++, Office Programs

Design Process: Said he would be influenced by outlook right away. The design process had no rollbacks. He stuck with most of the things he initially tried. The process was mainly focused on functionality.

Final Design:

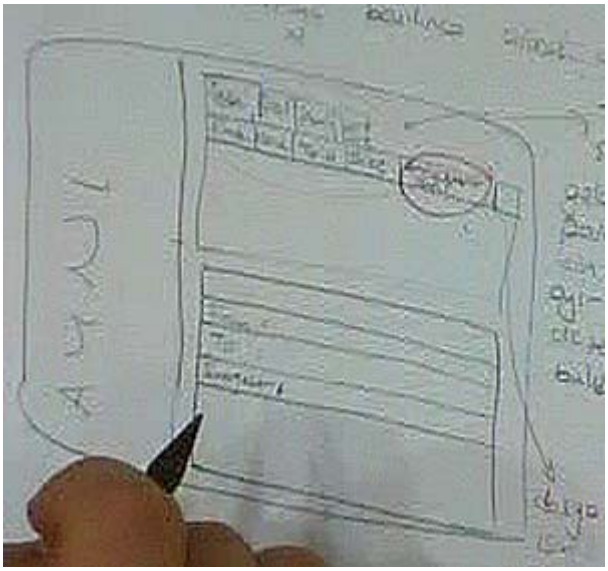

Figure A.5: Design of E4

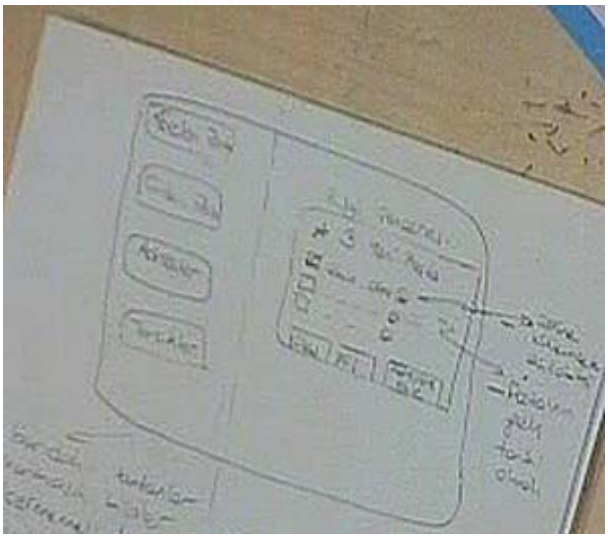

Figure A.6: Design of E4

Design uses window structure, check boxes, tabs, and most windows standards.

Extra functions are:

- Banning messages
- Extended address book.

The final design is thorough and focuses mainly on functionality. Main functions are grouped on the left much like a web page and all design and the little twists are described in detail. All in all the design is a plain cross between yahoo mail and outlook (admitted by the designer) with small nuances.

Other Notes: The design process was focused mainly on covering up messy functionality of outlook or yahoo. More of a clever redesigns process in terms of quick interaction but nothing new, really.

ENGINEER 5 (E5)

Occupation: Electrical and Electronics Engineer.

Computer Background: Has used PC. Proficient in almost everything related to computers.

Design Process: Did some grouping and started a tentative flowchart but then fell into tedious blueprinting. Was not very talkative through the three-hour process. Was conscious of the screen positioning throughout

Final Design:

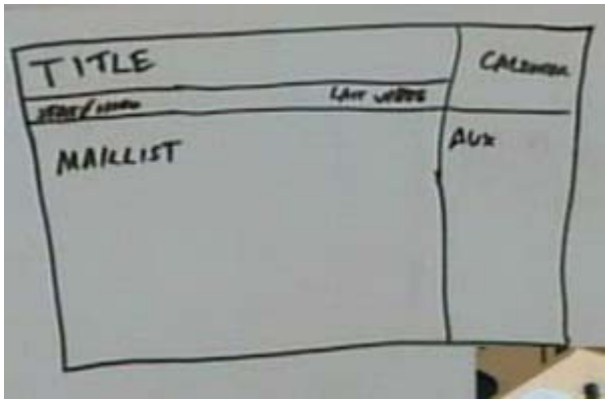

Figure A.7: Design of E5

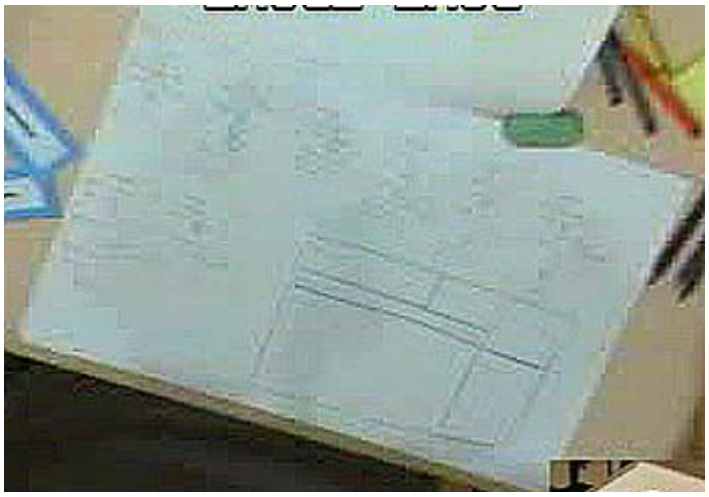

Figure A.8: Design of E5

Final design uses an interface more like a web interface largely due to the web based mail client the designer took as his inspiration. Drop down boxes and links galore.

Extra Functions are too many to discuss.

The final design is very well thought of, and very impressive for a three-hour design. But nothing groundbreaking in terms of interaction and functionality, really. All in all it's a very clever hybrid.

Other Notes: Another victim to the redesigning tendency. This design looks like a patchwork of working parts, handling many things a mail user might want but we didn't ask for.

DESIGNER 1 (D1)

Occupation: Industrial Designer

Computer Background: Has used Amiga 500 and PC. Proficient in Office programs, Animation programs, illustration and modeling programs.

Design Process: Asked several questions on the nature of the operating system (Is the product to be web-based, can it have additional functionality etc.)

Used the cards extensively and used multiple relations between them. Started the design process by sketching blocky windows interfaces and then put a thematic "jacket" on at the end.

Final Design:

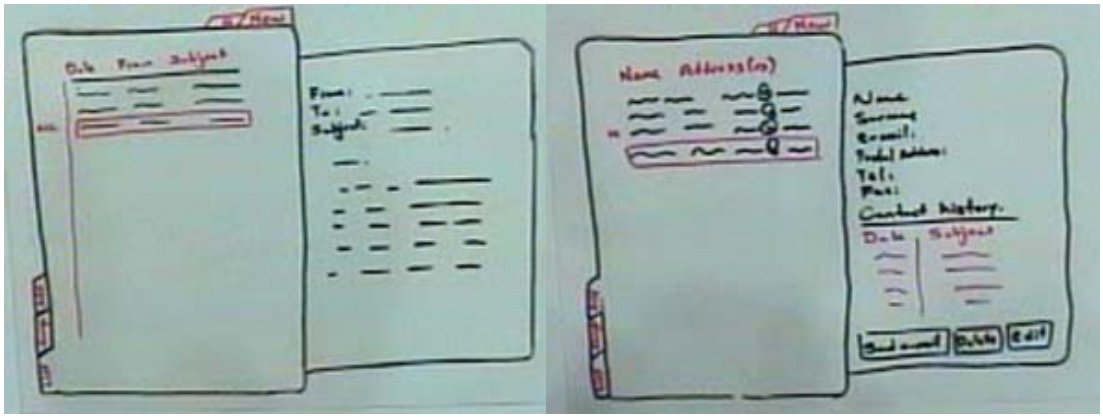

Figure A.9: Design of D1

The design uses an idea of a tabbed file, which has papers coming out of its back end (again as opposed to its real world counterpart).Design uses tabs, and lists.

Extra functions are:

- Multiple accounts
- Attachments

Other Notes: The designers' touch in this case seems to be a superficial redressing of the interface**.** Mentioned the limitedness of a computer interface. Claimed much more could be done with a PDA. Claimed cursors, context menus and other things are mandatory with this interface.

DESIGNER 2 (D2)

Occupation: Industrial Designer

Computer Background: Has used PC, Mac and Linux. Proficient in Macromedia programs, graphic design and modeling software etc.

Design Process: Asked the motive behind this design process. Formed a flow using the requirements and past experience with e-mail. Took an e-mail address (specifically incoming) as a starting point for the flow. Mentioned a functional break down he was used to (Mail-Related, System-Related, General). Talked throughout the design process explaining most of her decisions. Some of his considerations are closeness of the concept of e-mail to mails in paper medium (suggesting a more mp3 player type approach) and usage of visual space.

Claims to be influenced by the software he uses.

Final Design:

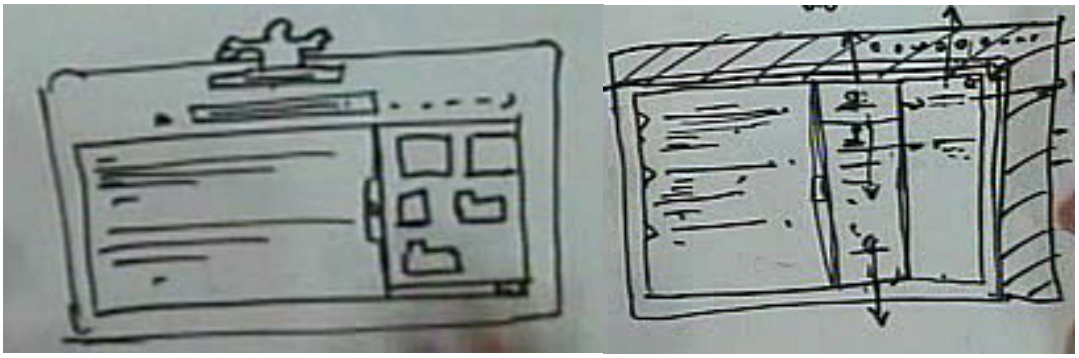

Figure A.10: Design of D2

Design uses accordions (expandable menus), avatars (representative images of a person), expanding context menus, window structure check boxes etc.

Extra functions are:

- Multiple accounts
- Avatars (representative icons for people)
- Word tools for text editing (justify, bold etc.)
- Log keeping for e-mails.
- Little mention of blocking and virus protection

The design looks more like an instant messaging program than an e-mail program. Dynamic and well thought, for the shortest time taken of all designs done.

Other Notes: Using an approach of flow analysis, the final design seems to have ended closer to an “information exchange” tool. This is largely interesting due to the ability of a different approach (instead of redesigning) of ending somewhere fundamentally different.

DESIGNER 3 (D3)

Occupation: Industrial Designer

Computer Background: Has used Commodore 64, Amiga 500 PC. Proficient in Office programs, 3D Modeling programs, animation programs. Knows a little scripting. Designed for the web using flash.

Design Process: Worked with cards and grouped the functions in a simple manner. Focused the process on interaction, added NO new functionality, and did a lot of rolling back during design.

Final Design:

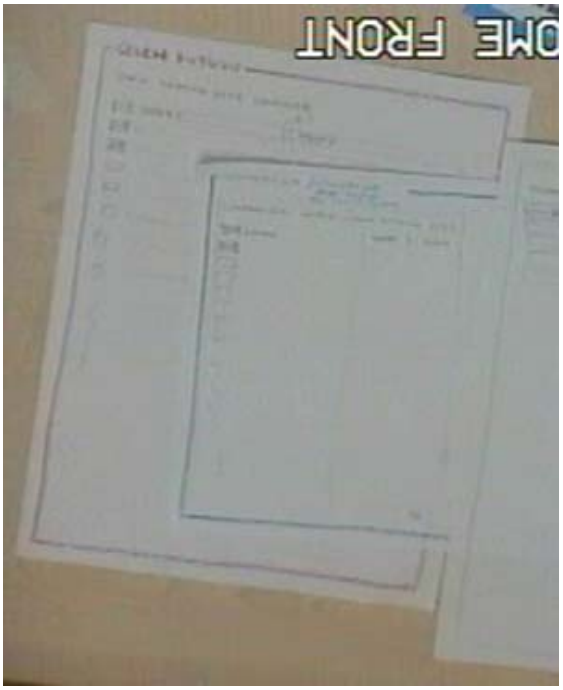

Figure A.11: Design of D3

Figure A.12: Design of D3

Design uses overlapping window-like structure no other recognizable elements are directly used other than editable text areas and mail icons.

Interaction is pretty original with three overlapping windows that are not movable and are activated when the mouse is over them. Two “thrash bin” areas are accessible from all windows and handle deleting.

All in all a pretty interesting design, if somewhat vague.

Other Notes: A plain and interesting result. Overlapping ideas are conveniently used (like one thrash for all deleting processes). Not too many questions asked which suits the spirit of the experiment.

DESIGNER 4 (D4)

Occupation: Industrial Designer
Computer Background: Has used PC. Proficient in Office programs, 3D Modeling programs, animation programs. Designed for the web.
Design Process: Started examining and grouping the cards to a vague grouping, then did a lot of sketching with tedious notes. Concentrated on multiple users/accounts and synchronization.

Final Design: NO FINAL DESIGN. Actually no design at all. Tried to re-design outlook but it seems he couldn't remember any problems with it other than synchronization.

Tried to adopt a different context menu style used in another software but didn't finish it.

Other Notes: Seems he tried to better outlook rather than sketch a simple e-mail client program. He said things like "When doing something like this one has to get the good sides of all products like hotmail and outlook".

DESIGNER 5 (D5)

Occupation: Industrial Designer
Computer Background: Has used PC. Proficient in Office programs, 3D Modeling programs, animation programs etc. Knows a little Flash.

Design Process: Examined the functions for a while but didn't do much grouping. Then started sketching things like post boxes and mail delivery equipment. A pretty marginal interface, which according to the designer is "closer to the origins of the idea of mailing".

Did a lot of rolling back and developed ideas right to the end. Most of it had to do with the real life analogies of mailing.

Final Design:

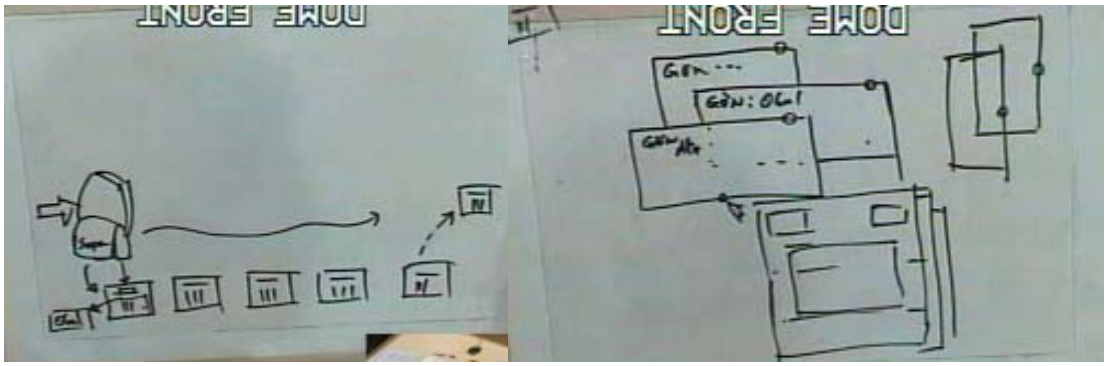

Figure A.13: Design of D5

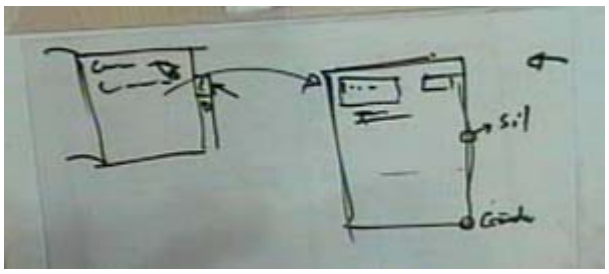

Figure A.14: Design of D5

Final design uses no instantly recognizable windows style interfaces. It is basically mailboxes, envelopes, papers, an address book (a real-looking one with hand written addresses in it etc.).

Interaction is achieved through “hot points” (red dots on the interface) whish when clicked on, accomplish a task (for example to delete a mail, click the red point on its edge and watch it being ripped in two.)

It is a design of questionable realism with today’s standards but has a certain romantic appeal. (Incoming mails have fold-marks on them).

Other Notes: Perhaps the most impressive example of an industrial designer doing software works with no restraints in this test. This is certainly applicable design (see the computer games today), with visual appeal, but no resembling software or so it seems. The designer even suggested box-type mailboxes like the ones we use to replace American looking flagged ones.

APPENDIX B

TASK CARDS AND DESCRIPTION USED IN OBSERVATION

E-mail list	New mail alert
Reading e-mail	New mail information
Read mail information	Important mail information
Deleting an e-mail	Writing new e-mail
Saving an address	Deleting an Address
Mail to a saved address	Reply to an e-mail
Forward an e-mail to a new address	Address book
Address that will receive the e-mail	Address that has sent the e-mail
E-mail subject text	E-mail main text

E-posta listesi	Yeni posta uyarısı
E-posta okuma	Yeni posta bilgisi
Okunmuş posta bilgisi	Önemli posta bilgisi
Bir postayı silme	Yeni e-posta yazma
Bir adresi saklama	Bir adresi silme
Saklanmış bir adrese posta atma	Bir postaya cevap verme
Bir postayı yeni bir adrese yollama	Adres defteri
E-posta yollanacak	E-posta yollayan adres
E-Posta konu metni	E-posta ana metni

Hello,

First of all, thank you for participating in this study and we hope this will be an interesting experience for you as well.

To briefly summarize what we want you to do:

A new operating system is about to be released. The producers of this operating system want to put forth their own software designs, completely independent from currently used software products. What we want you to do is to design an e-mail software interface to take part in this operating system. What limits you is the current physical interface of computer systems (Keyboard, monitor, and mouse…)

The tools you can use while designing are as follows:

- Cards that define the parts and the functionality expected from the software
- Acetate working medium the size of a computer screen
- Scrap paper and coloured pencils
- Marker pencils and sticky papers to work on the acetate surface.

About one and a half hours may be enough to complete a design.

Your questions and comments throughout the design process would be very helpful for us.

Thank you for your effort.

Merhabalar....

Öncelikle bu çalışmaya katıldığınız için teşekkürler, umarım sizin için de ilginç bir deneyim olur.

Yapmanızı istediğimizi kısaca özetlemek gerekirse:

Yeni bir işletim sistemi piyasaya sürülmek üzere. Bu işletim sisteminin üreticileri şimdiye kadar kullanılan yazılımlardan bağımsız olarak kendi yazılımlarını tasarlayarak ortaya koymak istiyor. Sizden beklentimiz bu işletim sistemi içinde yer aşlacak bir e-posta yazılımının arayüzünü tasarlamanız. Sizi kısıtlayan şey ise varolan fiziksel bilgisayar arayüzünü kullanmanız gerekmesi (Klavye, ekran ve fare...).

Bu tasarımı yaparken faydalanacaklarınız şunlar:

- Yazılımdan beklenen işlevselliği ve parçaları tanımlayan kartlar.
- Ekran boyutunda asetat çalışma alanı.
- Müsvette kağıtları, boya kalemleri...
- Asetat alan üzerinde çalışmak için marker kalemler, yapışkan not kağıtları...

Tasarımı tamamlamak için yaklaşık birbuçuk saat süre yeterli olabilir.

Tasarım süresince sorularınızı ya da yorumlarınızı paylaşmanız bizim için çok yararlı olacaktır.

İyi çalışmalar...