

Building E-education Platform for Design-Oriented Learning

by

Hai Ning

Master of Science in Architecture Studies, Massachusetts Institute of Technology, 1999
Bachelor of Architecture, Tsinghua University, 1997
Bachelor of Engineering in Computer Science, Tsinghua University, 1997

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN INFORMATION TECHNOLOGY

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2004

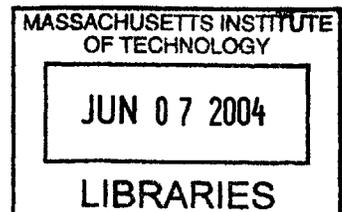
© 2004 Massachusetts Institute of Technology. All rights reserved.

Signature of Author
Department of Civil and Environmental Engineering
May 7, 2004

Certified by.....
John R. Williams
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by.....
Heidi M. Nepf
Professor of Civil and Environmental Engineering
Chairman, Committee for Graduate Students

BARKER



BUILDING E-EDUCATION PLATFORM FOR DESIGN-ORIENTED LEARNING

by

Hai Ning

Submitted to the Department of Civil and Environmental Engineering
in June 2004 in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Information Technology

Abstract

Design-oriented learning requires tools that support creative processes and student-to-student and student-to-faculty interactions. While most present E-Education systems perform as the asynchronous distribution channel for teaching material, they usually offer little support for project based design processes.

This research maps out the key learning events in design classes at MIT's Department of Mechanical Engineering, and proposes guidelines for building E-Education systems to support the unique characteristics of design-oriented learning. Two creative learning processes are identified and two independent, yet tightly related, software systems are implemented and evaluated. The first application, the Peer Review and Engineering Process (PREP), is a web system that helps instructors and students conduct and manage peer review evaluation of design concepts. The second is a real time application called InkBoard that leverages the Tablet PC and Ink medium to provide real-time collaborative sketching over TCP/IP networks. A new streaming network protocol for transferring Ink objects is proposed and implemented. A comparative study against other ink-enabled protocols is also presented.

Thesis Supervisor: John R. Williams

Title: Professor of Civil and Environmental Engineering, MIT.

Thesis Committee Memers: Professor Alexander H. Solcum, Professor Jerome J. Connor, Professor Keven Amaratunga.

Table of Contents

TABLE OF CONTENTS	1
CHAPTER 1 INTRODUCTION.....	7
PEDAGOGICAL OBJECTIVES:.....	7
<i>Exploring the uniqueness of design oriented-learning</i>	<i>7</i>
<i>Propose the requirements of e-education platforms supporting design-oriented learning</i>	<i>8</i>
TECHNOLOGICAL OBJECTIVES:	9
<i>InkBoard Messaging Protocol for Ink data network communication</i>	<i>9</i>
<i>Service oriented application architecture.....</i>	<i>9</i>
OUTCOMES AND CONTRIBUTIONS	9
THE STRUCTURE OF THE THESIS	10
CHAPTER 2 PROBLEM DEFINITION.....	13
EXISTING E-EDUCATION PLATFORM SURVEY	13
<i>COMMAND</i>	<i>14</i>
<i>Stellar.....</i>	<i>17</i>
<i>Blackboard Learning Systems.....</i>	<i>19</i>
OBSERVATIONS	20
MOTIVATION OF THE THESIS	22
METHODOLOGY	23
CHAPTER 3 DESIGN-ORIENTED LEARNING.....	25
LEARNING STRATEGY IN GENERAL	25
<i>Learning Styles</i>	<i>25</i>
<i>Felder-Silverman Model.....</i>	<i>25</i>
<i>Kolb Model</i>	<i>27</i>
<i>Teaching to All Types</i>	<i>28</i>
DESIGN AND LEARNING DESIGN.....	30
<i>What Is Design.....</i>	<i>30</i>
<i>Design Disciplines</i>	<i>31</i>

<i>Learning Theory and Teaching Design</i>	32
<i>Teaching styles in design-oriented learning environment</i>	35
THE DESIGN-ORIENTED LEARNING APPROACH	36
<i>Creative problem solving</i>	36
<i>Team collaboration</i>	38
<i>Project management</i>	38
ELECTRONIC PLATFORM SUPPORT FOR DESIGN-ORIENTED LEARNING	39
<i>Encourage independent design thinking</i>	39
<i>Facilitate communication</i>	40
<i>Build the sense of community</i>	41
<i>Provide project management functions</i>	41
CHAPTER 4 PEER REVIEW EVALUATION PROCESS	43
INTRODUCTION	43
PROJECT BACKGROUND INFORMATION	43
<i>2.007 Introduction to Design</i>	44
<i>International Design Contest (IDC) 2002</i>	45
<i>2.993. Designing Paths to Peace</i>	45
CHALLENGE: TAKING ROHRBACH ONLINE	45
<i>Engineering Design Process</i>	45
<i>Idea-generation Methods</i>	46
<i>Rohrbach 635</i>	47
<i>Technology Background</i>	48
Web Services	48
Service-oriented Architecture	49
Tablet PC	50
SYSTEM ARCHITECTURE	51
<i>Stakeholders and roles</i>	52
Developers and development platform	52
Users and client environment	52
<i>UML diagrams</i>	53
Use case diagram for site group and project team	53
Sequence diagram for Rohrbach process	54
<i>PREP architecture</i>	55

<i>Rendering engine</i>	56
User management WFUC.....	57
PREP Matrix WFUC.....	57
Discussion WFUC.....	59
<i>Web/Windows Services</i>	59
Registration web service.....	59
Document web service	61
Discussion web service	61
Email alert Windows service.....	61
Email web service	62
CHAPTER 5 IMPACT OF PREP	63
IDC 2002	63
2,993 PATHS TO PEACE	65
DESIGN OF A TOTALLY BURIED-DISTRACTION DEVICE.....	67
OBSERVATIONS	69
CHAPTER 6 SKETCH AND DESIGN	71
THE RELATIONS OF DRAWING TO PROBLEM SOLVING.....	71
<i>To archive the geometric form of the design</i>	71
<i>To communicate ideas</i>	72
<i>To act as an analysis tool</i>	72
<i>To simulate the design</i>	72
<i>To serve as a completeness checker</i>	73
<i>To act as an extension of designer's short-term memory</i>	73
ROLE OF SKETCHING	73
<i>Cognitive analysis</i>	74
<i>Visual Thinking</i>	75
BACK TO THE DRAWING BOARD	76
<i>Loss of Public Forum</i>	77
<i>Lack of abstraction, ambiguity, vagueness and imprecision</i>	77
Abstraction	77
Ambiguity and vagueness.....	78
Imprecision.....	78
<i>No incremental formation process support</i>	78

<i>No easy way for rapid exploration of alternatives</i>	78
THE PROMISE OF THE TABLET PC.....	79
<i>Pen-based natural user interface</i>	79
<i>Advanced operating system</i>	80
<i>Enhanced mobility and feature-rich applications</i>	80
<i>Integration with Office suite</i>	81
<i>Personalization</i>	81
<i>Extend Existing Applications with Digital Ink Handwriting</i>	81
SUMMARY.....	82
CHAPTER 7 INKBOARD SOFTWARE DESIGN	83
MOTIVATION AND GOALS.....	83
<i>Motivation</i>	83
<i>Goals</i>	84
Design-oriented education needs.....	84
Scalability.....	84
Application usability	85
RELATED WORKS.....	85
<i>Conference XP</i>	85
<i>ReMarkable Texts</i>	86
TABLET PC SDK.....	87
<i>Overview</i>	87
<i>Managed APIs</i>	88
<i>Ink controls</i>	89
<i>COM Automation APIs</i>	89
INK – THE NEW NATIVE DATA TYPE	90
CHALLENGE: BUILDING INK-ENABLED COMMUNICATION TOOLS	91
SYSTEM ARCHITECTURE.....	92
<i>P2P or client/server?</i>	92
<i>InkBoard Client Architecture</i>	94
InkBoard drawing area.....	94
Other UI tools and thread safety.....	97
<i>Network Manager</i>	98
<i>InkBoard Server Architecture</i>	99

Network Manager.....	100
Ink Studio database.....	101
INKBOARD USABILITY	103
<i>Sign-in dialog box</i>	104
<i>InkBoard toolbar</i>	105
<i>Color picker</i>	106
<i>Timeline</i>	107
<i>Presence and ink layers</i>	107
CHAPTER 8 INKBOARD MESSAGING PROTOCOL	109
THE ANATOMY OF IMP	109
<i>Base class InkBoardMessage</i>	109
<i>Derived class IBM_Stroke</i>	112
<i>Derived class IBM_StudioInfo</i>	113
<i>InkBoard message helper class</i>	113
<i>Transferring InkBoardMessage</i>	114
Asynchronous client socket.....	115
<i>Preserve message boundary</i>	116
A COMPARATIVE STUDY OF IMP VS. REMARKABLE TEXTS	117
<i>ReMarkable Texts</i>	118
RTP	118
RTDocs and RTInk	118
Multicast network.....	119
<i>IMP Assessment Tool</i>	121
Software architecture.....	121
The synchronization of time.....	122
Collected data.....	124
<i>Analysis of the IMP in comparison with RTDocument + RTP</i>	127
Advantages.....	127
Disadvantages.....	128
CHAPTER 9 INTEGRATING CONFERENCE XP WITH INKBOARD	131
CONFERENCE XP & INKBOARD	131
TECHNOLOGY OVERVIEW OF CONFERENCE XP	131
<i>Conference layer</i>	132

<i>Managed DirectShow layer</i>	132
<i>RTP layer</i>	132
INTEGRATION WITH INKBOARD	133
<i>ActiveX controls</i>	133
<i>UI consideration</i>	134
FUTURE IMPLEMENTATIONS	135
<i>Provide support for unicasting</i>	135
<i>Record conference sessions for future playback</i>	135
CHAPTER 10 CONCLUSION	137
SUMMARY	137
CONTRIBUTIONS	137
<i>Pedagogical contribution</i>	138
PREP	138
InkBoard.....	139
<i>Technological contribution</i>	140
Service-oriented software architecture	140
InkBoard messaging protocol.....	141
FUTURE RESEARCH	142
<i>Design advisors</i>	142
<i>Shape recognition</i>	142
<i>Unicast conferencing</i>	143
<i>Case studies</i>	144
<i>Deployment and integration</i>	144
APPENDICES	147
ACKNOWLEDGEMENT	147
SELECTED BIBLIOGRAPHY	149
FOOT NOTE	155

Chapter 1 Introduction

The Internet has largely changed the higher-education landscape since the late 1990s. With the advance of the technology, the barrier-to-entry of building web sites for education has come down quite considerably over the past few years. Universities as well as corporate training programs are aggressively exploring the opportunities of leveraging the power of the Internet to facilitate higher-education.

Learning style theory proposes that different people learn different types of subject in different ways. It is to the learners' advantage that they chose the most comfortable way to learn, that is efficient and productive. It is also known that particular learning objectives sometimes dictate specific learning activities. For example, the only way to learn to ride a bicycle is by practicing riding it, not by watching other people riding it, nor by reading instruction of how to ride it, nor by listening to instructions from experienced riders. The same is true for several other fields of study, and learning how to design artifacts is one of them.

This thesis focuses on design-oriented learning, and aims to achieve the following pedagogical and technological objectives.

Pedagogical Objectives:

One of the goals of the thesis is to study the unique characteristics of design-oriented learning, and in turn, propose requirements for building e-education platforms that support these unique aspects.

Exploring the uniqueness of design oriented-learning

Archer notes "Design is that area of human experience, skill and knowledge which is concerned with man's ability to mould his environment to suit his material and spiritual needs".¹ Design, and particularly artifacts design, is a unique activity. It is an open-ended problem-solving process that often forces the designer to choose the solution that best satisfies the requirements among all the possible options. The process of generating novel design concepts and selecting

the attributes of the final product is the focus of design-teaching in fields such as mechanical engineering, architecture, urban planning, etc.

Study of design courses offered by MIT's Department of Mechanical Engineering, reveals that there are three unique characteristics associated with design-oriented learning that are not found in other courses, namely,

1. Creative problem solving
2. Team collaboration
3. Project management

Propose the requirements of e-education platforms supporting design-oriented learning

Based on these studies, we propose that an e-education platform for design-oriented learning, should address the following issues:

1. Encourage creative, independent design thinking.
2. Provide tools for publishing an individual's ideas.
3. Implement tools for synchronous and asynchronous communication among students.
4. Provide private space for small design teams to exchanging information that builds a sense of community.
5. Provide tools for project management.

Two proof-of-concept software systems were built based on the above requirements. Peer Review Evaluation (PREP) system is a web tool enabling design teams to conduct peer review of the design concepts. InkBoard, a desktop application running on Tablet PCs, is a sketch-sharing tool for teams to communicate their design concepts using ink strokes on Tablet PC screens. It also leverages video and audio conferencing capabilities. Both systems are targeted to provide a set of much more effective tools to augment present web-based e-education platforms in the context of design-oriented teaching and learning.

Technological Objectives:

InkBoard Messaging Protocol for Ink data network communication

In the process of designing and building the PREP system and InkBoard application, a number of challenging technical issues arise. In particular, “ink”, is a new data type for the Tablet PCs to record digital pen strokes. Protocols for exchanging Ink in a network environment have not been extensively studied, particularly, serializing Ink and streaming it across the network. Here the Ink Messaging Protocol (IMP) is proposed and tested. It is compared with related technologies and an analysis of advantage and disadvantages is presented.

Service oriented application architecture

Component-based and service-oriented design for web development is proposed as a solution to the issues of the current typical web systems, namely flexibility, maintainability, and scalability. PREP system follows the philosophy of component-based and service-oriented. This thesis develops a web programming framework that constructs web portals from reusable web components. These web components or building blocks are in turn supported by a range of web services that can be updated or even exchanged without affecting the client layers.

Outcomes and contributions

The PREP system was built and used in a variety of environment including design contest, design classes, and real collaborative design tasks. It improved the teaching and learning experience in the design courses conducted by Department of Mechanical Engineering at MIT.

InkBoard generated attentions from education communities outside MIT, as well as from Microsoft Research teams. It is publicly available for download, and is being test-driven by a number of institutions as of the writing.

Using an assessment tool we built, we observed that the performance level of Ink Board Messaging protocol is within the same range of Real Time Protocol running in Multicasting

networks, and it has the advantage of being able to operate in less-restrictive network environment supported by most educational institute and ISPs. The proposed and implemented service-oriented software architecture also proves to enhance the flexibility, scalability and customizability of web systems in general.

The Structure of the Thesis

This thesis has ten main chapters that cover the major topics of building e-education platform to support design-oriented learning.

- Chapter 1, *Introduction*, outlines the objectives and summarizes the contributions of the thesis.
- Chapter 2, *Problem Definition*, surveys the existing e-education platforms, analyzes their merits, and provides motivations for the thesis.
- Chapter 3, *Design-oriented Learning*, discusses in detail the three unique features of design-oriented learning, and proposes guidelines for building e-education platforms to support such learning activities.
- Chapter 4, *Peer Review Evaluation Process*, introduces PREP, the first web application that addresses the peer review process, a widely-practiced design methodology.
- Chapter 5, *The Impact of PREP*, presents the user data collected with the PREP system, analyzes the results.
- Chapter 6, *Sketching in Design*, analyzes the importance of drawing, preliminary sketches in particular, in the early phases of design, and provides the motivation of building the InkBoard application.
- Chapter 7, *InkBoard*, presents the architecture of first Tablet PC application designed to enable ink-sharing and audio/video conferencing capabilities for design teams.

- Chapter 8, *Ink Messaging Protocol*, analyze in detail the protocol designed for transporting Ink data across a TCP/IP network, provides a comparative study with other similar technologies, and describes the advantages and disadvantage of the Ink Messaging Protocol.
- Chapter 9, *Integrating Conference XP with InkBoard*, describes the integration of Conference XP audio/video conferencing components with the InkBoard application.
- Chapter 10, *Contributions*, summarizes the pedagogical and technological contributions of this thesis, and proposes the next steps for design-oriented learning support.

Chapter 2 Problem Definition

With the advance of computer technologies, the quest for e-learning, or computer-mediated education, began in the 1980s. Initially, it was limited to educational software being stored on magnetic tapes and run on stand-alone PCs. The media has since moved on to magnetic disks, CDs and DVDs. In the meantime, the communication mechanism between computers has gone through much more fundamental changes. From local area networking (LAN) leading up to the Internet that connects computers all over the world, from archaic command-line based telnet sessions to rich multimedia-packed HTML web pages filled with images, video clips and colorful animations, it gives rise to e-learning a huge up-lift at the global scale, and creates an unprecedented learning experience. Online Web-based Learning (OWL) has emerged as one of the fastest moving trends in educational technology today². Chris Dede states that “In developed countries, sophisticated computers and telecommunications are on the verge of reshaping the mission, objectives, content, and processes of schooling.”³ In this chapter, we will discuss the merits and shortfalls of popular e-learning tools and explains the motivation of the thesis.

Existing E-education Platform Survey

Today, setting up a web server and authoring HTML documents online has become as easy as developing and distributing paper-copy class notes. Finding a class at MIT that does not have a web site where students can go and look for information is increasingly difficult. In fact, the Open Course Ware project (OCW) at MIT has announced that they have put 500 MIT courses online, including all syllabi, reading lists, lecture notes and assignments, free and open to the whole world.

For something as simple as putting up a web page online, to do it on such a scale – considering MIT alone offers more than 600 courses per semester – would require a complex yet powerful information system in place. OCW admits that “brute-force HTML” was not scalable after the initial pilot that published 32 subjects by just doing that. Electronic platforms, or web-based course content management tools to be more precise, are the inevitable choice for such tasks. To

get a closer view at the existing e-education platforms, their strength and weakness, we analyze three typical course content management systems that have been popular at MIT: *COMMAND*, a successful home-grown solution coming out of a small research lab; *Stellar*, an institute-backed operation with a fair amount of resources and ambitious goals in a research-oriented academic environment; and finally *Blackboard Learning Systems*, the leading commercial software vendor for course management systems targeting at global education market.

COMMAND

Course Management and Delivery System, or COMMAND, was built by the Intelligent Engineering Systems Lab in the MIT Department of Civil and Environmental Engineering in 1997. It started with one software engineering course in the Department of Civil and Environmental Engineering, and spread out to the whole MIT, including the Schools of Architecture, Engineering, Management (Sloan School) as well as Humanity. In the early days of web facilitated education, it was a wild success within MIT, considering it was created by a group of four students and one professor in three months with virtually no marketing campaign other than the word of mouth, emails and was supported by one, sometimes two, graduate students.

The design guideline behind the COMMAND system was to create a universal template, or container, for lecture-based courses with basic functionalities such as content storage and retrieval so that professors can easily create a course web site driven by database out of the template and put his/her content online for students to review. This simplistic approach is probably the main reason that Command was so successful. Back in 1997, not many easy-to-learn yet powerful tools were out there to enable teachers to build content-rich course web sites. Electronic versions of the teaching materials were already available for most of the courses; but it was the difficulty of putting them online in a well-organized fashion and the burden of having to learn coding up HTML or sometimes writing database-driven web applications by themselves, that drove the teaching staff to Command.

COMMAND was built on Lotus Domino, a rapid database/web site development environment. A template database that holds all the business logic functionalities sits on the server waiting for

request. Professor/TA can send a request through the web interface, and a course database would be created out of the template after the verification of the authenticity of the request. The course database will hold all the content later being created by its users.

The basic functions Command provides are:

- Course Documents
 - About Course
 - Readings
 - Lecture Notes
 - Assignments
 - Solutions
 - Course Help Files
- Homework Administration
 - Homework Submission
 - Grading
 - Release of Grades
 - Grade History
- Collaboration
 - Class Discussion
 - Private Group Discussion
 - Chat
 - Contact Info (Directory)
- Administration Tools
 - Sections
 - Announcements
 - Calendar
 - Team Formation
 - Presentation Scheduler

- Video Streaming
 - Lectures
 - How-To

As of Spring 2003, COMMAND had 16,199 users, 476 course web sites, 84 team spaces, 2794 administrative email messages, and over 200,000 pages of content.

Table 1 Sample COMMAND course site sizes in megabytes

Course Number	Size	Course Number	Size
1.00	5.44 MB	1.041	30.91 MB
1.00Fall01	7.43 MB	1.070	0.18 MB
1.00Spr02	14.6 MB	1.103	4.05 MB
1.010	0.17 MB	1.118	59.33 MB
1.012	127.8 MB	1.118_F01	112.08 MB
1.013	118.41 MB	1.120F99	39.1 MB
1.033	104.29 MB	1.124	20.11 MB
1.040	43.39 MB	1.124_F00	6.47 MB
1.040	43.39 MB	1.125	11.62 MB

Table 2 Sample COMMAND course site file types

File Types Counts – Course 1.118	
<ul style="list-style-type: none"> • Document folder: Assignments <ul style="list-style-type: none"> • HTML, 1 • Document folder: By Category <ul style="list-style-type: none"> • HTML, 2 • Document folder: Course Material <ul style="list-style-type: none"> • HTML, 1 • Document folder: HomeworkByNumber <ul style="list-style-type: none"> • XLS, 3 • TXT, 6 	<ul style="list-style-type: none"> • Document folder: Lecture Notes <ul style="list-style-type: none"> • PPT, 24 • PDF, 3 • ZIP, 1 • HTML, 1 • Total: 29 • Document folder: Readings <ul style="list-style-type: none"> • XLS, 2 • TXT, 1

<ul style="list-style-type: none"> • PDF, 1 • ZIP, 14 • HTML, 50 • GZ, 2, • VSD, 3 • Total: 80 	<ul style="list-style-type: none"> • PDF, 14 • PS, 2 • ZIP, 7 • EXE, 2 • HTML, 2 • HTM, 14 • Total: 44 • Document folder: Solutions <ul style="list-style-type: none"> • HTML, 1 • HTM, 1 • Total: 2
--	--

COMMAND was switched off on August 2003, and replaced by a new system called Caddie.NET. One of the exemplary tools presented in this thesis, Peer Review Evaluation Process System, was built on the platform of Caddie.NET. The details are discussed in Chapter 4.

Stellar

Stellar is another course management system supporting teaching and learning across MIT which began in Fall 2001. It is currently on version 1.5. Unlike COMMAND which was a spontaneous research project with very limited resource that had grown big, Stellar is formally endorsed by MIT, developed by Academic Media Production Services (AMPS), and deployed institute-wide to support over 650 courses as of the writing.

The main functions offered by Stellar include:

- Integrated authentication with MIT web certificates.
- Teaching Materials:
 - Lecture Notes
 - Lab
 - Assignments
 - Readings

- Exams
- Resouces
- Discussion
- Membership (Staff, students directory)
- Schedules
- Homework Submission
- Search

Stellar adopts the same model as COMMAND did, taking requests from professors/TAs over the web, creating course lockers upon verification of the requests. It offers an easy way for faculty, instructors and teaching assistants to create secure class web sites, organize class materials for students, handle homework assignments, and engage students in discussion using the online tools. It also boasts production quality software, with usability reviews and full-scale software quality assurance procedures. The features such as customizable look and feel with distinctive logos and color schemes also appeal to many MIT faculties and students. Stellar also has actively-updated user help files, documentations, FAQs that help users to fully leverage its functionalities. It even runs a few workshops during the summer, training interested faculties and teaching staffs to prepare for the subsequent fall semester courses.

The system behind Stellar consists of two Linux servers running Apache web servers, one 4-processor Linux Server running Tomcat, a Java Servlet application server, and two Oracle database servers for data redundancy. The actual content is stored in the Oracle database, upon requests from the client browser which are relayed by the Apache servers to the Tomcat server, the Java servlets perform the necessary calculation and retrieve data from Oracle database, assemble the result into XML strings back to the Apache server. The XML strings are then translated into HTML pages by Java XML API functions with pre-defined XSLT files before being passed back to the client browser.

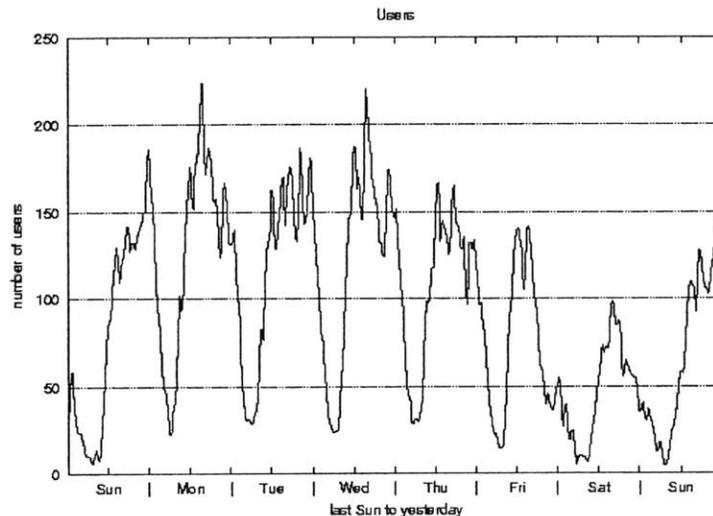


Figure 1 Stellar usage over the week of 9/28/2003 ~ 10/6/2003

Stellar apparently has been successful so far, attracting increasing number of participating courses around MIT. However, considering the amount of resources, and the high profile it maintains being the official MIT course management system, it seems fair for one to expect Stellar to offer much more functions and much better services than COMMAND did. However, this remains to be seen.

Blackboard Learning Systems

Blackboard is a leading enterprise software company for e-education, offering course content management systems for high education, K-12 education, as well as corporate and government training. According to their website, currently there are more than 2,000 colleges and universities in over 100 countries that use its Blackboard course management system⁴. Its flagship product – Blackboard Learning System – just released in version 6, boasts “feature-rich learning environment, pedagogical flexibility, complete control of the course design and unmatched ease-of-use”⁵.

For a leading commercial software vendor of course management system, Blackboard Learning Systems offer a wide range of features with rich user experiences including the following:

- Content Management
 - User-controlled, text-based naming and navigation for course content areas
 - Move and copy content items, folders and learning units within course content area and between distinct courses
 - Ability to include SCORM-compliant Microsoft LRN content in any content area.
 - Integrated math and science notation.
 - Grade book and homework capabilities.
- Assessment Management
 - Assessment engine offers dynamic, rule-based assessment authoring tool.
 - Expanded feedback and delivery options.
- Collaboration
 - Free-form chat, chat lectures, questions and answer chats.
 - Class tours and group web browsing.

The success of Blackboard, a company only started around 1997, gained momentum during the dot com frenzy, survived the economic downturn and started to generate profit. Technology superiority seems to be the first thing it has to be able to achieve. And yet in today's web platform backend turf war between Microsoft enterprise systems and J2EE technology backed up by Sun Microsystems, joined occasionally by other big names like IBM and Oracle from time to time, it is very surprising to see Blackboard's platform rest its complicated business logic on PERL, a widely acknowledged development tool suitable for quick and dirty solutions as opposed to enterprise-level scalability and maintainability.

Observations

The majority of our classroom learning experience today in most education institutes is an authority-based, lecture-centered environment. And this fact has clearly shown its influence in

designing and implementing the e-education platforms that serve as auxiliary tools for classroom learning. All three different course management systems show very similar focus on functionalities of document storage and retrieval made easy for instructors and students. Granted, lecture-centered classroom learning is useful for introducing students to concepts, or for providing information about specific topics. With the supplement of actual hands-on exercises the lectures can be a very effective teaching approach.

However, there are problems with the lecture-centered classroom approach with learning materials that deals with experience-oriented, “knowing-how” type of tacit knowledge⁶ as opposed to concept-oriented, “knowing-what” type of explicit knowledge. The traditional classroom is designed as a broadcasting medium where instructors present their knowledge on a certain subject. There is no guarantee that the broadcasting has achieved its purpose of transferring information to the students sitting in the classroom, let alone making sure that information be internalized by the students at the same time. In-class discussion sessions are not feasible particularly in larger classes with many students. The commonly used assessment tools such as after-class assignments and quizzes can only be practiced after the lecture, which leaves a time gap in-between when immediate feedback could be more favorable.

With limited teaching resources it is difficult to generate direct interaction between instructors and students. With no effective communication channels that are established off lecture time among students themselves before the popularity of the Internet in educational environment, there was very little the instructor could do. The advancement in technology and rapid adoption of the learning technology provide instructors the means they need to address the inadequacy of the broadcast teaching method. Modern effective learning environments should integrate four dimensions⁷: learner-centered, knowledge-centered, assessment-centered and community-centered. The old teacher-centered approach, though it has been practiced for the most part in the history of contemporary education, no longer has the capability to create and deliver the integrated learning environment necessary in the design for innovative teaching today. E-education platforms, built on the nowadays ubiquitous Internet, should be able to address the aforementioned four aspects instead of merely replicating the old lecture-based teacher-centered approach electronically.

Granted, the COMMAND system, Stellar as well as Blackboard Learning System all provide some mechanism for encouraging interactions between students and instructors. However, most of them are limited in the form of online discussion forums. While general discussion board does fulfill some requirements of indirect communication and in some case generate fair amount of traffics among interested parties, without an accountable moderator, its generality often leads the conversation out of focus. Also, all three systems use the similar model of generating course sites based on template sites, which contributes the ease and efficiency of getting new course sites up quickly, but at the same time suffers the consequence of forcing the organizational structure of the teaching materials belonging to different courses conforming to the same hierarchy.

It is clear that a web-based course management system mirroring the lecture-centered classroom teaching approach has the above limitations even when the subject is concept-oriented or fact-based. In other types of learning subjects such as design-oriented courses where interactions between students and teachers as well as among students are vitally important and teaching materials are structured differently from course to course, this kind of cookie-cutter platform will not be able to help to generate the desired learning experience.

Motivation of the Thesis

The Motivation of this research is to explore how to build e-education platforms to support the needs of design-oriented learning. There are many design-oriented courses taught in different disciplines in MIT, such as architecture studios, building structure designs, and urban planning studies. One of the very interesting design fields is the robot design courses offered by the Department of Mechanical Engineering. This type of courses usually requires students create a solution to a problem within a well-defined boundary. Most of the time the solution itself is not the ultimate goal; rather, the instructors try to help students establish a systematic process of reaching a solution to the problem.

Admitted, the popular course management systems provide some level of support in this particular type of learning. However, the main elements in design-oriented learning can not be addressed simply by document storage and retrieval system. Before we dive into the distinct

features of an e-education platform that supports the requirements of design-oriented learning, we need to study the uniqueness of design-oriented learning.

Methodology

The principle methodology of this research can be summarized in 6 steps.

- Step 1: Study the learning theories in existing literatures regarding design-oriented learning.
- Step 2: Gather teacher and student experiences on a few typical design-oriented courses taught at MIT Mechanical Engineering Department. Identify some of the key learning events.
- Step 3: Find out the unique characteristics of design-oriented learning courses in comparison with non design-oriented courses.
- Step 4: Propose the requirements of an E-Education platform that targets supporting design-oriented learning.
- Step 5: Build experimental systems based on the proposed guidelines.
- Step 6: Run user tests on the experimental systems and gather feedbacks.

Chapter 3 Design-oriented Learning

It is crucial to understand the activity of design before the unique characteristics of design-oriented learning can be explored. This chapter tries to define what is design-oriented learning, what is different from other types of learning and how to deal with these unique characteristics when design an e-education platform to support design-oriented learning.

Learning Strategy in General

Learning Styles

Most people approach something new in a similar fashion each time; they usually develop a pattern of behavior they use for learning new things. They learn in many ways – by seeing and hearing; reflecting and acting; reasoning logically and intuitively; memorizing and visualizing and drawing analogies and building mathematical models. This pattern is called *learning style*. Although with different learning tasks, typical learners don't approach them in exactly the same way, they usually develop a set of behavior patterns overtime that is relatively consistent and they feel most comfortable with. Information about learning styles can serve as a guide to the design of learning experiences that match learner's learning style.

Felder-Silverman Model

Richard M. Felder proposes that a student's learning style may be defined in general by answers to five questions⁸:

- What type of information does the student preferentially perceive: sensory (external) – sights, sounds, physical sensations, or intuitive (internal) – possibilities, insights, hunches?
- Through which sensory channel is external information most effectively perceived: visual – picture, diagrams, graphs, demonstrations, or auditory – words, sounds? (Other sensory

channels – taste and smell, with the exception of touch – are relatively unimportant in most educational environments and will not be considered here.)

- With which organization of information is the student most comfortable: inductive – facts and observations are given, underlying principles are inferred, or deductive – principles are given, consequences and applications are deduced?
- How does the student prefer to process information: actively through engagement in physical activity or discussion, or reflectively – through introspection?
- How does the student progress toward understanding: sequentially – in continual steps, or globally – in large jumps, holistically?

Similarly teaching style by instructors can also be defined in terms of the answers to the above five questions:

- What type of information is emphasized by the instructor: concrete – factual, or abstract – conceptual, theoretical?
- What mode of presentation is stressed: visual – pictures, diagrams, films, demonstrations, or verbal – lectures, readings, discussions?
- How is the presentation organized: inductively – phenomena leading to principles, or deductively – principles leading to phenomena?
- What mode of student participation is facilitated by the presentation, active – students talk, move, reflect, or passive – students watch and listen?
- What type of perspective is provided on the information presented, sequential – step-by-step progression (the trees), or global – context and relevance (the forest)?

Table 3 Felder-Silverman model: dimensions of learning and teaching styles

Preferred Learning Style		Corresponding Teaching Style	
Sensory	Perception	Concrete	Content
Intuitive		Abstract	
Visual	Input	Visual	Presentation
Verbal		Verbal	
Inductive	Organization	Inductive	Organization
Deductive		Deductive	
Active	Processing	Active	Student participation
Reflective		Passive	
Sequential	Understanding	Sequential	Perspective
Global		Global	

Kolb Model

David Kolb proposed another well-known model in analyzing learner's learning styles⁹. He classifies students as having a preference for 1) concrete experience or abstract conceptualization (how they take information in), and 2) active experimentation or reflective observation (how they internalize information). The four types of learners in this classification scheme are:

- Type 1 (concrete, reflective). A characteristic question of this learning style is "Why?" Type 1 learners respond well to explanations of how course material relates to their experience, their interests, and their future careers. To be effective with Type 1 students, the instructor should function as a motivator.
- Type 2 (abstract, reflective). A characteristic question of this learning style is "What?" Type 2 learners respond to information presented in an organized, logical fashion and benefit if they have time for reflection. To be effective, the instructor should function as an expert.
- Type 3 (abstract, active). A characteristic question of this learning style is "How?" Type 3 learners respond to having opportunities to work actively on well-defined tasks and to learn by trial-and-error in an environment that allows them to fail safely. To be effective, the instructor should function as a coach, providing guided practice and feedback.

- Type 4 (concrete, active). A characteristic question of this learning style is “What if?” Type 4 learners like applying course material in new situations to solve real problems. To be effective, the instructor should stay out of the way, maximizing opportunities for the students to discover things for themselves.

Table 4 Kolb model: dimensions of learning and teaching style

Learning Style	Typical Question	Teacher’s Role
Type 1: Concrete & Reflective	“Why?”	Motivator
Type 2: Abstract & Reflective	“What?”	Expert
Type 3: Abstract & Active	“How?”	Coach
Type 4: Concrete & Active	“What if?”	Moderator

By practicing respective teaching style targeted at different categories of learning style, the learners show better understanding of the learning material and achieve the learning objective successfully. This was demonstrated in Vanderbilt University Chemical Engineering department’s Kolb Learning Style Inventory Program, Brigham Young University College of Engineering and Technology’s program based on Kolb learning styles, as well as the Felder-Silverman learning style program at Civil Engineering Department in the University of Western Ontario¹⁰.

Teaching to All Types

Limited teaching resources in typical classes in a college teaching environment reflect the low ratio of teachers versus students. Most of the time, teacher has to accommodate a class of students with different learning styles. The instructor needs to ensure his/her courses present information that appeal to a range of learning styles. Felder suggests the following approaches to achieve the goal of teaching the same materials to all learning types.¹¹

- Teach theoretical material by first presenting the phenomena and problems that relate to the theory.
- Balance conceptual information with concrete information.

- Make extensive use of sketches, plots, schematics, vector diagrams, computer graphics, and physical demonstrations in addition to oral and written explanations and derivations in lectures and readings.
- To illustrate an abstract concept or problem-solving algorithm, use at least one numerical example to supplement the usual algebraic example.
- Use physical analogies and demonstrations to illustrate the magnitudes of calculated quantities.
- Occasionally give some experimental observations before presenting the general principle, and have the students see how far they can get toward inferring the latter.
- Provide class time for students to think about the material being presented and for active student participation.
- Encourage or mandate cooperation on homework.
- Demonstrate the logical flow of individual course topics, but also point out connections between the current material and other relevant material in the same course, in other courses in the same discipline, in other disciplines, and in everyday experience.

Naturally, although teaching style can be influenced learning style, it is also influenced by the nature of subject being taught. It is hard to imagine learning watercolor painting by merely reading text books because the student is a “verbal learner”. And it is equally baffling to fathom studying theoretical math by only engaging group discussion among fellow learners simply because a student prefers peer interaction over reading text books. Before we explore the appropriate learning and teaching styles in the design field, we need to first understand what is design.

Design and Learning Design

What Is Design

Design was usually considered to be related to certain professions such as architects, engineers, industrial designers and others. Consequently design activity was naturally regarded as what these professions did in order to produce the physical products needed by their clients and by manufactures. However, since the global industrialization, attempts have been made by professional designers to redefine design activity and isolate the essence of design as a standard method, or recipe, that can be applied to most situations. Numerous literature dating to the tries to label the seemingly obvious and yet elusive definition of design activity. Here are some excerpts.

*“Finding the right physical components of a physical structure.”*¹² (Alexander, 1963)

*“A goal-directed problem-solving activity.”*¹³ (Archer 1965)

*“Decision making, in the face of uncertainty, with high penalties for error.”*¹⁴ (Asimow, 1962)

*“Simulating what we want to make (or do) before we make (or do) it as many times as may be necessary to feel confident in the final result.”*¹⁵ (Booker, 1964)

*“The conditioning factor for those parts of the product which come into contact with people.”*¹⁶
(Farr, 1966)

*“Engineering design is the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform prespecified functions with the maximum economy and efficiency.”*¹⁷ (Fielden, 1963)

*“Relating product with situation to give satisfaction.”*¹⁸ (Gregory, 1966)

*“The performing of a very complicated act of faith.”*¹⁹ (Jones, 1966)

*“The optimum solution to the sum of the true needs of a particular set of circumstances.”*²⁰
(Matchett, 1968)

*“The imaginative jump from present facts to future possibilities.”*²¹ (Page, 1966)

*“A creative activity – it involves bringing into being something new and useful that has not existed previously”*²² (Reswick, 1965)

It should be of no surprise that these definitions differ so much, since different writers see different design processes from different perspectives in different contexts. One thing common to all the above descriptions though, is that they all emphasize not the outcome of the design, but the ingredients of design. If we look at design from an end-to-end standpoint, the whole chain of events – beginning with the initial motivation, and moving through the actions of designers, manufacturers, distributors and consumers – leads to the ultimate result of something new, something different from what we have already. In that sense, Jones argues that the “definition of designing is the initiation of change in man-made things”. This definition expands the design activity from traditional design professions such as engineers and architects, to other professions including economic planners, legislators, managers, publicists, applied researchers, politicians etc.

However, the objective of this research is to explore the better approaches for design education practiced in university environment, focusing on fields such as mechanical engineering and architectural design. Therefore, for the sake of this research work, we limit the definition of design to the traditional sense of producing recorded ideas, 2-D drawings or 3-D models that are representations of physical products, or the physical products themselves, that serve pre-defined purposes.

Design Disciplines

The common design disciplines practiced in schools can include the following areas and combinations thereof:

- Architectural design. This includes interior design, urban planning, structural and civil engineering, commercial and residential construction, business services, etc.
- Electronic design. This includes microprocessor computer hardware, software engineering, consumer electronics, telecommunications, robotics, etc.
- Mechanical design. This includes mechanical engineering and manufacturing.
- Industrial design in industry. This includes consumer product design, fashion design, etc.
- Graphic design. This includes communications technology, environmental signage, broadcasting arts, animation, advertising, marketing, etc.

At the center of technology lies design. That “design is the very core of engineering” is affirmed by the requirement that all degree engineering courses should embody it. The design process in technology is a sequential process which begins with the perception of a need, continues with the formulation of a specification, the generation of ideas and a final solution, and ends with and evaluation of the solution.

The motivating factor behind all technological activity is the desire to fulfill a need. For this reason all designs should be made or realized – whether through prototype, mass-production or some form of three-dimensional or computer model – if the need is to be truly fulfilled, the design is to be legitimately evaluated, and the design activity is to have been purposeful and worthwhile.

Learning Theory and Teaching Design

Subconsciously, it is assumed that learning design is a very different activity from learning other subjects such as math, literature, or natural science. Learning can be thought of as “a relative permanent change in behavior that results from practice”²³. Objective/behavior learning theory believes in existence of reliable knowledge about the world²⁴. As learners, the goal is to gain this knowledge; as educator, to transfer it. Teachers serve as a pipeline and seek to transfer their thoughts and meanings to the passive students²⁵.

To the stark contrast of the behaviorism that emphasizes observable, external behaviors and, as such, avoids reference to meaning, representation and thought, the other branch of learning theory – the constructivism – takes a more cognitive approach. Under this model, learners construct knowledge for themselves – each learner individually (and socially) constructs meaning – as he or she learns²⁶. This seemingly subtle difference has profound implications for all aspects of the learning theory. The way in which the knowledge is conceived and acquired, the types of knowledge, skills and activities emphasized, the role of learners and teachers, how goals are established: all of these factors are articulated differently in the constructivist perspective. In particular, two issues stand out.

1. The focus of the learning theory is the learner as opposed to the subject/lesson to be learned.
2. There is no knowledge independent of meaning attributed experience (constructed) by the learner, or community of learners.

Design, as we have discussed before, is the activity of creating physical products that solve problems posed by a set of external parameters. Usually, there are a large number of possible paths a designer can choose, and hence large numbers of possible final outcomes a designer can produce, to reach the design objectives. It is unlikely for one designer to exhaust all the possibilities; and it is not desired to force a group of students come up with the exact same design. No one, not even the instructor, knows what the students' ultimate product will be like except that it is expected to function within a certain boundary set up by the initial parameters. Hence, the design product is not something that independently existed outside the designer/learners, nor is the ability to reach that final product. Teaching the students to understand the design parameters can take an objectivist approach, since after all, most of the design parameters are objective. "Clearance to the ceiling can't be more than 8 feet", "surface material of the interior wall should be mahogany"; or even fuzzier parameters such as "the entrance hall needs to convey a sense of solemnity", are independent criteria that can exist separately from the designers, and therefore can be managed and transferred between designers the same way the idea of "pure water is made of hydrogen and oxygen" is transferred from chemistry teacher to students. This type of knowledge is commonly regarded as explicit knowledge, or information. Accessing to knowledge, in this context, is the same as accessing information. However, the actual design happens when designer creates something that doesn't

exist before out of the given set of parameters. This takes a qualitative leap from the objectivist learning of the design parameters.

Polanyi introduced the notion of tacit knowledge to describe knowledge that we take for granted and is difficult to express²⁷, such as the ability to recognize a friend's face. It is irreducible to explicit propositional knowledge and it cannot be articulated. Therefore it cannot be taught the same way explicit knowledge is taught, although substantial evidence exists that it can be learned or acquired. Ryle gives an example of a surgeon. A man knowing little or nothing of medical science could not be a good surgeon, but excellence at surgery is not the same thing as knowledge of medical science; not is it a simple product of it. The surgeon must indeed have learned from instruction, or by his own inductions and observations, a great number of truths; but he must also have learned by practicing a great number of aptitudes²⁸. The same can be said to an architect or a mechanical engineer. Learning *how* or improve an ability is not like learning *that* or acquiring information. Surely, there exist common practices, or procedures that lead to guaranteed satisfactory, if not necessarily elegant, solutions to the design problem. But "truths can be imparted, procedures can only be inculcated, and while inculcation is a gradual process, imparting is relatively sudden"²⁹. It makes sense to ask at what moment someone became apprised of a truth, but not to ask at what moment someone acquired a skill. It is clear that design education needs to focus on facilitating students to acquire this tacit knowledge by repetitive practicing, or inculcation as pointed out by Ryle.

The objectivist teaching model is driven by "teacher-talk" and depends heavily on textbooks for the structure of a course. It assumes the knowledge that is expected to be internalized by students is actually information, and hence students are passive recipients of the information. Information is divided into parts and built into a whole concept. There is little room for student-initiated questions, independent thought or interaction between students. This is the basis of most e-education platforms that are put in use today, as it has been observed in Chapter 2. Clearly, it will not work well in a design-oriented learning environment.

Teaching styles in design-oriented learning environment

Having considered design as a special learning experience together with different dimensions of learning styles, we can match the skills required in learning design with desired learning styles to arrive at a set of teaching styles that would be most effective in design-oriented learning.

For sensory versus intuitive learning styles, design-oriented learning focuses on problem solving capabilities, and hence encourages direct interactions/perceptions with the world. But the unconscious activities such as speculation, imagination or hunch can play a vital role in creative thinking and hence should not be ignored.

For visual versus verbal learning styles, design-oriented learning heavily favors visual representation of physical object. 2-D drawings, 3-D scaled models or computerized models are such important tools to explore and express design ideas that in some curriculums themselves become independent training courses. On the other hand, verbal communication from instructors to students or among students themselves, and design theory text reading can be an effective auxiliary means to facilitate the visual representation.

Induction and deduction are interesting pairs in the human learning experience. Naturally we learn things by induction. We observe the world around us, draw inferences from past experiences, and generalize to arrive at principles and rules. Most of what we learn by ourselves originates in a real situation or problem that needs to be addressed or solved. On the other hand, we teach others naturally by deduction. Most of our teaching practices involve laying down the governing principles first and moving on to applications later. Consequently many design curriculums take the deductive approach which not only counters most learners' natural inductive learning style, but also suppress creativity by subconsciously suggesting similar if not identical design products. An effective design-oriented learning strategy should stimulate students' imagination and creativity by letting them explore possibilities within the design boundary and guide them with principles when necessary.

Active experimentation and reflective observation are two complex mental processes by which information is processed and converted into learner's knowledge. Again, by the nature of design,

“doing things” or “making things happen” is favored over merely “thinking about things”. Moreover, active learners perform better in a group context which is encouraged with today’s complex design problems, whereas reflective learners usually prefer working alone. Designers should be encouraged to engage initially in independent thinking for creative ideas, and subsequently collaborate with other team members and experiment with those ideas actively for assessment.

Sequential learning style follows linear reasoning process step by step to reach the solution of a problem. This would work well when the problem is clearly defined and there usually exist a limited number of, or some times the only, correct solution. Design tasks, however, are often ill-defined (except the simple samples given in class perhaps), and usually have no correct answer or unlimited amount of acceptable solutions. A global learning approach, given the freedom to device students’ own methods and paths, as opposed to faithfully adopting instructor’s pre-defined steps, is more likely to lead to an innovative and satisfactory result.

The design-oriented learning approach

Since design is a mix of understanding of explicit design parameters and conducting conscious and yet implicit creative activity, design-oriented learning takes a unique approach of a combination of objectivism/behaviorism and constructivism, with the emphasis on the following aspects.

Creative problem solving

Theoretical developments in cognitive science has provided us tools to analyze the information handling procedures that can be identified within the designer’s operations³⁰. These tools, despite different angles of view, all point out the characteristic of design is the creative problem-solving process under the conditions of bounded rationality³¹. Of course, here “problem” refers to a much wider definition of the literal meaning of the word itself, noted by Thorndike, as a situation where an organism wants something but the actions necessary to obtain it are not immediately obvious³². The problems can be well-defined – the goals are already prescribed and apparent,

their solution requires provision of appropriate means³³. They can be ill-defined – both the ends and the means of solution are unknown at the outset of the problem solving exercise³⁴. Or they can be wicked – they are problems without a definitive formulation, no explicit basis for the termination of the problem-solving activity, and no necessarily correct or incorrect answers³⁵.

During the late 1950s and the 1960s, attempts were made to describe the creative problem-solving process at work in design by way of the logical structure of overt activities that appears to take place. In other words, design was regarded as a series of stages characterized by dominant forms of activity. Commonly practiced procedures usually include the sequential steps required to reach the final design. Since “solving the problem creatively” is such an abstract concept, it is natural for design instructors to resort to a somewhat concrete and linear process that can be more tangible and easier for students to grasp. Constant feed-back following each step ensures design decisions are made towards the right direction. Classroom experience tells us that the “creation” step usually is usually the hardest one for students, simply because that is the step where “something comes out of nothing” as pointed out by Alexander. Much research effort has been focused in this area and a number of proven methods are proposed and practiced in the design teaching environment. Chapter 4 will address this issue in greater detail.

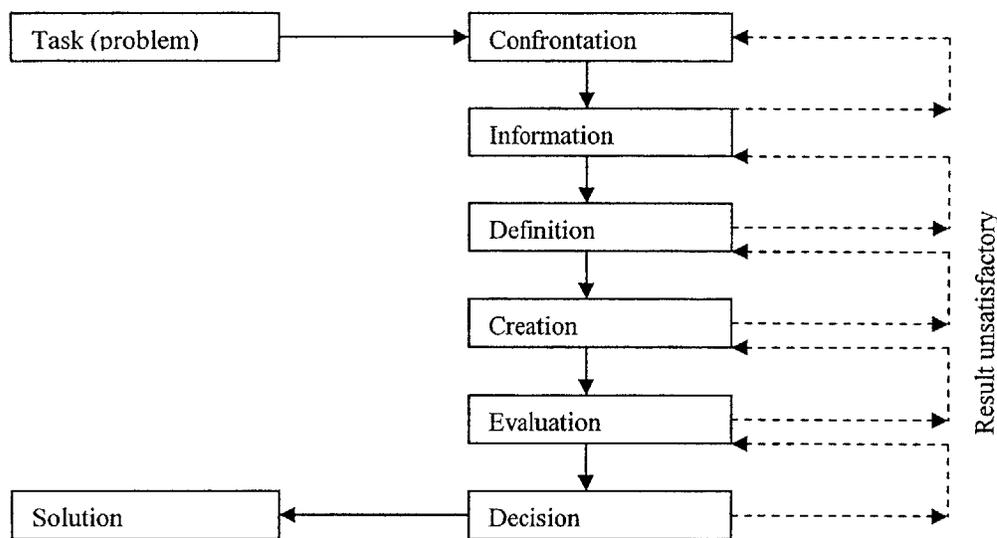


Figure 2 Mechanical engineering design process diagram

Team collaboration

Traditionally, great designers are regarded as highly talented individuals who are gifted with extraordinary creativities. Leonardo da Vinci, Michelangelo, or even someone as contemporary as Frank Lloyd Wright, are considered geniuses that possessed creativeness that an ordinary person or group could not possibly achieve. This is a myth that is difficult to disprove³⁶. Nonetheless, genius designers, by definition, are rare and hard to find, and hence can not be the targeted outcome of our design education. Meanwhile, with the advancement of social and technological environments, design tasks become invariably more and more complicated and require teams of designers to accomplish.

In a design group there is usually someone who takes the lead in a design task and assumes the responsibility for ensuring the process is right, but he/she can no longer be solely responsible for the final outcome of the design. The design education students acquired should release them from the tyranny of imposed ideas and enable each to contribute to, and to act upon, the best that he/she is capable of imagining and doing. This is not an easy task, and it can only be achieved by creating a collaborative environment where students can freely express themselves individually and yet the individual efforts can still be unified towards the common goal. In doing so, we have effectively transformed design education focused on individual designers to collaborative learning experiences for design teams that are more closely reflect the reality of design tasks today.

Project management

A complimentary and yet invaluable lesson students should take away with from design-oriented courses is project management skills. Because of the hands-on feature of design courses, design project is often the central if not sole task designed by instructors and handed out to students. With a given deadline, sometimes limited physical resources (e.g. mechanical kit, Lego, etc.), students, usually divided into design teams, and expected to turn in a physical design product, or 2D or 3D representation of in the product in the end. The whole project management would likely include but not limited to identifying key players in respective fields, allocating resources,

setting up realistic short-time milestones and long-time goals, dividing the task into independent modules, designing clean interfaces between modules, establishing a clear boundary of responsibilities, analyzing risks and returns, and on top of all, managing to deliver project on time. These are essential skills that determine the ultimate success or failure of the design project. Incorporating these elements in a project-drive design course can help students build a sense of how design project is handled in the real world.

Electronic platform support for design-oriented learning

Having discussed the unique characteristics of design-oriented learning, together with the understanding of capabilities and limitations of typical e-education platform, we propose that the following needs should be addressed when designing computer software to provide a successful e-education experience for a typical design course.

Encourage independent design thinking

Design thinking is the central task in teaching design courses. A successful electronic platform should encourage individual students to come up with unique design results. A proper example would be software agents that monitor student's progress in the class, draw reference from the background information about the student's past performance, and tailor customized learning material for the student. A design course typically sets the same goal of learning design methods in a specific discipline for all students. However, unlike most lecture-based classes where students go through the same materials and the same homework, design class students tend to exhibit much more variety in going about their projects. This is caused by the nature of design problems as we discussed before, and henceforth should be greatly encouraged and appreciated by the educators. By providing customized learning material and guidelines for individual students, student's creativity can be stimulated even further.

However, the realization of this attribute of design-oriented learning on an electronic platform remains a difficult task. A lot of artificial intelligence research has to be conducted before we can design a reasonably smart software agent as the virtual design advisor. A simplified version of

this, however, can be created by a tree structure that consists of a limited number of possible decisions that can be chosen by students and countered with learning strategies in respective context.

Facilitate communication

The creative problem-solving process is carried out through project-based learning in most design courses. This is fundamentally different from large classroom lectures given by instructors discussed before. In addition to being an online storage space for teaching materials related to the design project, the e-learning platform should also serve as a communication center for the teaching staff and the students. As mentioned before, design-oriented learning largely deals with transferring tacit knowledge from teachers' minds to the students'. Tacit knowledge, by definition, is difficult if not impossible to articulate with common mediums such as textbooks, lectures, or electronic web pages. The constant flow of communication, the exchange of ideas in forms of conversations, drawings, models, is what really bridges the teacher and the learner. This is why after thousands of years, from Baroque, Gothic, International, Modernism, till Post-modernism and Deconstructionism, from wood, bamboo, stone to glass, steel, concrete, the practice of teaching and learning architectural design in today's most architecture school remains its original form – apprenticeship. In engineering design courses that perhaps emphasize less on form and more on function, the interaction among students and teachers is still the essential ingredients to achieve the learning objective.

Thus, one of the focal points in building design-oriented e-learning platform is the emphasis on facilitating communications. Nothing can replace the face-to-face meeting that offers almost unlimited bandwidth with a huge amount of information going back and forth in very short time span with very little effort from the participants. However, what about geographically dispersed design teams? What about the teams that cannot get together at the same time? These are the areas that electronic platform can, and should, provide easy-to-use tools for asynchronous and synchronous interactions among team members.

Build the sense of community

Designers learn in a group environment. There is often no right or wrong answer to the particular design task. Everyone takes a different approach in dealing with the problem. Instructor may provide general guidelines or sample works, but it is only through articulating one's own idea and observing other's idea can one truly understand the reasons behind the design. Besides, to simulate the complex problem students will eventually face in the real world, it is usual for students to be required working in small design teams to tackle the problem. They comment on each other's work, brainstorm ideas, assign responsibilities, allocate works and collaborate with each other. The sense of community encourages them working closer among themselves, and promotes the sense of collective ownership of the project.

It is sometimes difficult to find designated physical spaces for individual design teams in most design teaching environments for practical reasons. However, creating an online community can be much more feasible. By restricting access to team members, providing communication platforms shared among team members, or staging contests between teams, students can be highly motivated to perform in a group environment.

In order to fulfill this requirement, the e-learning platform should provide easy steps to create community portals whenever needed. The portal should include material storage functions, group interaction mechanisms, and easy customization so that members can quickly turn it into their own home. A very flexible data model, componentized modules, and customizable user interface are the minimal prerequisites in building such a platform.

Provide project management functions

Last but not least, project management functions need to be built in the electronic platform for both the students as well as the teachers to monitor the progress of the project. Not only it can help students to better allocate their time and resources, it will be also easier for instructors to identify potential problems and help to correct them before it turns disastrous.

The specific functions can include group calendar where members can set milestone dates for all to view, automatically email reminder for deadlines, graphical view of the progress of the design cycle, etc. By helping students to manage their design projects, instructors are able to teach them a vital lesson that could mean success or failure in the real world.

Chapter 4 Peer Review Evaluation Process

Introduction

Having explored design-oriented learning in theory and proposed the features that an e-learning platform should support in order to create a successful design-oriented experience for students, this chapter will present an e-education platform – Peer Review Evaluation Process (PREP) system – that focuses on the peer review and evaluation process within the context of project-based design-oriented courses. We discuss the pedagogy, design philosophy, architecture and technology behind the system. The PREP system has been implemented and evaluated in the International Design Contest (IDC 2002) held at Massachusetts Institute of Technology (MIT) in August 2002, MIT's Mechanical Engineering courses 2.993 (Paths to Peace) in fall semester 2002, and 2.007 Introduction to Design in Spring 2004. The system is built with the web services technology using Microsoft .NET platform. As participants of Microsoft Tablet PC Rapid Adoption Program, we were able to equip the participating students with the exciting tools of tablet PC. They proved to be critical design resources which greatly enhanced the experience of using PREP system.

Project Background Information

Before we discuss the details of the PREP project itself, it is necessary to give some background information about the research project. PREP is part of RobotWorld project funded by iCampus at MIT. iCampus was initiated in October 1999 as a five-year research alliance between MIT and Microsoft Research to enhance university education through information technology. RobotWorld, funded by iCampus, is a collaborative project between Intelligent Engineering Systems Lab in Department of Civil and Environmental Engineering and Department of Mechanical Engineering. The main goal of RobotWorld is to systematize the deployment and running of courses that use design projects as a vehicle for teaching engineering, specifically robot design courses offered by Department of Mechanical Engineering. The PREP web

application, being phase I of RobotWorld, focuses on conducting the peer review process in teaching engineering on-line.

MIT's Department of Mechanical Engineering arguably offers some of the best design courses in the world. In fact, MIT invented the concept of robot challenge. RobotWorld, as a collaboration project between IESL (Intelligent Engineering Systems Lab) and Mechanical Engineering, gives us the opportunity to study the pedagogy practiced in these engineering design courses. We adopted the peer review design strategy adapted by Professor Alex Slocum in his 2.007 Introduction to Design course as the first building block for a complete e-education platform suite for project-based design-oriented learning. The beta version of PREP was created, tested in the International Design Contest 2002, revised based on the feedback, and launched to support the following 2.993 Designing Paths to Peace course. Some background information of these courses/events is provided as followed.

2.007 Introduction to Design

2.007 is one of the most successful project-based design-oriented courses taught at MIT. It has been offered to undergraduate students for the last 30 plus years. It aims at teaching students mechanical engineering design, and building students' competence, creativity and self-confidence via hands-on robot design. At the beginning of the term, students are given a kit of materials and asked to design a robot to accomplish a certain task. The kit usually only includes raw materials such as wind shield wiper motors, gears, pneumatic pistons, and lots of structural materials from which students have to design and fabricate remote controlled machines and compete in a final contest. Students go through the entire engineering design process, from developing concept by application of creativity and physics, making computer models and foam mockups of their concepts, to engineering the detail, and building their machine and having it ready for "shipment". At the end of the term, students compete in the robot contest that generates the winner of the year, attracting local and national media coverage.

International Design Contest (IDC) 2002

The 2.007 course was so successful that in 1990, it went international when a group of 2.007 students including the winner of the year traveled to Japan and participated in the International Design Contest (IDC). IDC was modeled on 2.007 but with teams of six students, each from a different country. Often times, students in the same team don't speak the same language, and the only way they can communicate is through language of design. The IDC was being held annually in different countries such as Japan, UK, Germany and Brazil.

In August 2002, IDC came to MIT. Forty students from Japan, USA, France, Germany, South Korea and Brazil participated in the two-week competition. They came to the MIT campus, worked together in 6 teams of 5 to 6, built a complex robot and competed in the final contest. It gave us a rare opportunity to have the contestants to test out our PREP system.

2.993. Designing Paths to Peace

2.993 was an introductory lecture/studio course designed to teach students the basic principles of design and expose them to the design process. They are required to design and build a major inlay tile whose elements are initially laid out using digital solid models and later manufactured on an abrasive water jet machining center. Again, the focus of this course is critical and visual thinking in addition to hands-on mechanical skills. 2.993 was offered in Fall 2002, in time for our updated version of PREP to be used in the process of the course.

Challenge: Taking Rohrbach Online

Engineering Design Process

The task of engineering design is to apply the scientific and engineering knowledge to the solution of technical problems, and then to optimize those solutions within the requirements and constraints set by material, technological, economic, legal, environmental and human-related considerations.

Because of the complex nature of modern technology, it is now rarely possible for an individual to tackle the design and develop of a product single-handed. Engineers often work in teams, which require highly efficient organization and communication. As pointed out in Chapter 3, the design process has to be broken down into phases and steps so that it can be planned carefully and executed systematically.

An essential part of the problem-solving method involves step-by-step analysis and synthesis. Figure 2 in Chapter 3 provides a diagram of operational guideline for finding solutions for general engineering problems. In the practice of teaching engineering design, the MIT Department of Mechanical Engineering teaching staff found that although most student are able to follow through the steps laid out in the diagram, the most difficult hurdle to overcome is the “creation” milestone where they are expected to come up with new ideas to solve the problem. Comparing to other steps that are relatively “mechanical” so to speak (no pun intended), generating novel ideas allows much greater freedom and proves more elusive for students to accomplish. To help the students, a number of idea-generation methods have been thoroughly studied.

Idea-generation Methods

Table 5 Classification of idea generation methods by representation

Graphical Representation	Textual Representation	Cross Representation
1. C-Sketch	1. Brainstorming	1. Morphological Analysis
2. Gallery Method	2. K-J Method	2. Synectics
	3. Checklists	
	4. Affinity Method	
	5. Storyboarding	
	6. Fishbone	
	7. Method 635	

Based on the design representation employed in expressing design ideas, a classification of idea generation methods is listed in above table.

Rohrbach 635

One of the widely practiced methods in the aforementioned MIT courses/events is Method 635 by Rohrbach³⁷. The method is described as following:

Designers are divided into 6-person teams. After familiarizing themselves with the task and after careful analysis, each of the 6 participants is asked to write down three rough solutions in the form of keywords. After some time, the solutions are handed to the participant's neighbor who, after reading the previous suggestions, enters three further solutions or developments. This process is continued until each original set of three solutions has been completed or developed through association by the five other participants. Hence the name of the method.

Rohrbach 635 is a very effective method in that it allows creative idea being developed more systematically. It encourages, and almost forces, team members' participation. It suppresses the dominating group member problem. It creates a democratic group atmosphere for every participant. It prevents censoring of such thoughts as might give offense to superiors or subordinates since all group members are equal. Introverted group members feel less intimidated in writing down their critics than openly discussing other people's ideas since no talking is allowed during the process. The method provides excellent documentation for itself along the way which is invaluable for the design process.

In essence, Rohrbach 635 is a peer-review evaluation procedure. The past 2.007 course adopted Rohrbach 635 effectively. Every student was required to go through a couple of iterations of this process before starting to build the computer model. It also provided a systematic means for sharing and compiling the ideas generated by all team-members, and also their opinions and comments on each other's ideas. Because of its proven effectiveness in past pedagogical practice, its importance in searching for solution for design problems which, to a lot of students, is a

significant barrier of entry to design-oriented courses, and its relatively rigorous step-by-step characteristics, Rohrbach 635 peer-review process became the preferred candidate for us to model our first software module for e-education platform supporting project-based design-oriented learning.

Technology Background

In building PREP application, not only did we try to implement a popular idea generation design method into a software application, we also wanted to experiment with a new programming model – service-oriented architecture, and a new kind of computing hardware – Tablet PC. A brief introduction of these new technologies is necessary before we analyze the system architecture of PREP.

Web Services

Web services has become the new buzz word in IT industry. We decided to build our PREP system based on web services model because of the flexibility and scalability it offers. Generally speaking, a web service is simply an application delivered as a service that can be integrated with other web services/applications using Internet standard protocols. Like any other component-based programming architecture, web services represent black-box functionality that can be used without worrying about how the service is implemented. It communicates with its consumer services/applications using ubiquitous Web protocols and data formats such as HTTP and XML. Any system supporting web standards will be able to leverage web services. Simple Object Access Protocol (SOAP), the web service protocol described in XML and transmitted over HTTP, is essentially a sole messaging protocol, which determines that web services model is completely language, platform and object model agnostic.

Traditional web applications usually take the 3-tier approach: presentation layer, business logic layer and database layer. The functionalities of the web application usually needs to be broken up into independent modules, but the tightly-coupled structure of the business logic layer and presentation layer makes it hard to divide the application into self-contained black-boxes which could have been especially valuable for team project where everyone's responsibility needs to be

clearly defined and thus problems can be easily identified. The problem exacerbates with the difficulty of exchanging information between web applications due to the lack of interface defined in standard protocol.

With a web services programming model, web applications can now fully implement the component-based architecture with each functional component being developed as an independent web service. Each web service has its own 3-tier structure: service contract layer, business logic layer and database layer. Each web service represents an isolated library capable of performing certain well-defined functions. The web application itself mainly contains presentation layer code that put together the data collected from various web services and sends it over to the client browser. If the web services are designed in such a way that they are very generic and well documented, they can be re-used by other applications as long as the format of the contract message is honored by the consumer application.

Service-oriented Architecture

XML web services have triggered the new programming model – service-oriented architecture – to emerge as the new flexible standard that has been endorsed and embraced by all software vendors. That fact that industry giants such as Microsoft, IBM, Oracle and Sun are all behind this move says more than enough about the future of this programming architecture.

The concept of breaking large software problems into smaller independent black boxes, or components, is hardly new. The object-oriented programming model has reigned in the software industry for the past 20 years. The leading component technology, namely CORBA (Common Object Request Broker Architecture), COM (Component Object Model), and Java RMI (Remote Method Invocation) have all been successfully adopted and played important role in various industries and organizations. They all work reasonably well within their defined boundaries, namely, a homogeneous computing environment.

Unfortunately, the computing landscape today in which we are living is just the opposite, a heterogeneous environment where a number of different mainstream operating systems, programming languages and development tools competing with each other. With the desire of

accessing computing resources from different organizations growing faster than ever, it has become clear that the existing component technology can not keep up with the pace. On the other hand, re-inventing an entirely new architecture to replace the existing infrastructure is not feasible, considering the huge investment corporations have made with their legacy systems.

In service-oriented programming model, XML is the core to bridge the gaps between various platforms and programming languages. Its popularity can be largely attributed to its simplicity. Unlike messaging formats used in CORBA or RMI, an XML document is text-based, human-readable, highly flexible and extensible. The requirements for either end of the communication channel using XML messages are very minimal. Although in most cases XML message is transported over HTTP protocol, largely due to the firewall issues in corporate networks, it can be transmitted over other protocols based on TCP/IP.

The greatest advantage for XML based service-oriented architecture lies in the fact that because of its simplicity and minimal requirements to implement, existing legacy systems, regardless of operations systems or programming languages can be integrated with a little effort spent in putting up a SOAP interface. Obviously in building a new system, following the service-oriented architecture will make the components of the systems truly extensible and reusable.

It is our belief that any E-Education platform will benefit immensely from the services-oriented programming model. It allows independent function units being developed, deployed, and integrated into complex systems while maintaining the flexibility, extensibility and re-usability. Furthermore, it can leverage existing content-rich education information systems and integrate them in new systems.

Tablet PC

An exciting new type of hardware for this project is the tablet PC from Microsoft and Acer. As one of the Microsoft Tablet PC Rapid Adoption Program participants, RobotWorld received 27 Acer TravelMate 100 tablet PCs for the use in the IDC 2002 and 2.993.

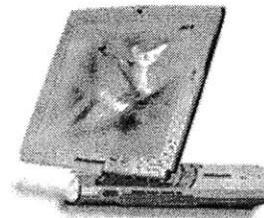


Figure 3 Tablet PC from Acer

Tablet PC is a significant step for Microsoft and hardware vendors to move from ubiquitous keyboard and mouse-based input computing environment today towards more natural interface of pen and voice computing. Unlike the pocket-size PDA, a tablet PC is, first and foremost, a fully functional portable PC. However, it is modified both in terms of hardware and software level to differentiate it from other portable PCs.

Tablet PCs are ultra-mobile laptop computers with “convertible” screens that can be used in normal laptop mode, or flipped around and used like a tablet, with stylus and on-screen keyboard input. “Pure tablet”, as it is called, is another type of hardware design that uses a docking station to provide keyboard and mouse access; when away from the desk, the tablet is used with the stylus and on-screen keyboard only. The Acer unit we used is a “convertible” tablet PC.

Tablet PCs ship with Windows XP Tablet PC Edition which is a super set of the Windows XP Professional operating system. In addition to the standard Windows XP features, Tablet PC Edition includes support for the active digitizer and stylus used by Tablet PC devices, instant display switching between landscape and portrait modes, and a small suite and Tablet PC-enabled applications, including Windows Journal, Sticky Notes and etc. Windows Journal is used as the primary software for sketching ideas in our peer-review process.

The motivation for getting these tablet PCs stems from the fact that peer review practices in the past robot design courses/contests were carried out in the form of paper and pen. Since we wanted to move the whole process into a web platform, digitizing the ideas sketched on pieces of paper became immediate problem. The tedious scanning, printing and re-scanning would quickly discourage students’ participation. With tablet PCs, everything is done digitally. Valuable time can be saved for students to conduct more efficient design process. The degradation of quality of the sketches caused by repetitive scanning and printing is also eliminated.

System Architecture

PREP web application, as the first phase of RobotWorld project, tries to implement the Rohrbach 635 method on-line using web services programming model. This section explains in detail of the design of the software itself.

Stakeholders and roles

As in designing any software application, the first step is always to understand who the users are and what their requirements are. The stakeholders in the PREP project includes a diverse group of users, each have a slightly different perspective in the development process, as well as a different role in the actual design activities.

Developers and development platform

PREP software development group consists of one professor as project supervisor, one graduate student as the project manager, two graduate students and one undergraduate student as developers, all from IESL. The development tool we used is Microsoft Visual Studio .NET version 7. The program language we used is C#. It was developed on Windows 2000 Server operating system with Microsoft Internet Information Services 5.0 and Microsoft SQL Server 2000 database application. ASP.NET is primary programming technology.

Users and client environment

The users in the PREP application are considered to be all people who use the system via its web browser client interface on a daily basis. The majority of them are students who participate in the Rohrbach process, as well as teaching staffs who use PREP to monitor the design project progress. PREP system contains two levels of groups: site group and project team. Site group contains all users that have access to the web site, where as project team contains only users within a particular project group. The same user identity could be assigned with different roles depending on which group context he is in. Following is a table contains all user roles and a simple description of his respective permissions.

Table 6 Groups and roles of PREP

Site Group	Anonymous
	Student
	Teaching Staff
	Web Administrator

Project Team	Team member
	Team Leader
	Supervisor

Users are expected to operate on Internet Explorer or Netscape Browser version 4 or above. There is no additional software required, and all user administrative functions have been built-in to be accessed from browser.

UML diagrams

Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. Here we present two sets of diagrams: use case diagrams for both levels of group, and a sequence diagram for the core functionality of the PREP system – the Rohrbach process.

Use case diagram for site group and project team

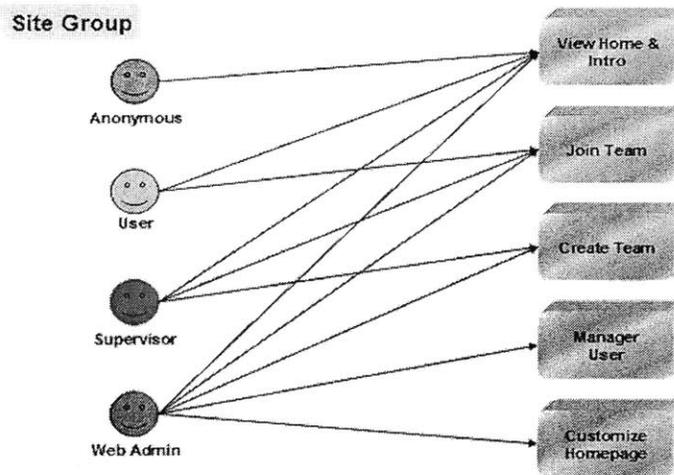


Figure 4 Use case diagram at site group level

Here four different user roles – *Anonymous*, *User*, *Supervisor*, and *Web Administrator* – can be associated with individual users. *Web Administrator* has the highest privilege and *Anonymous User* has the lowest privilege.

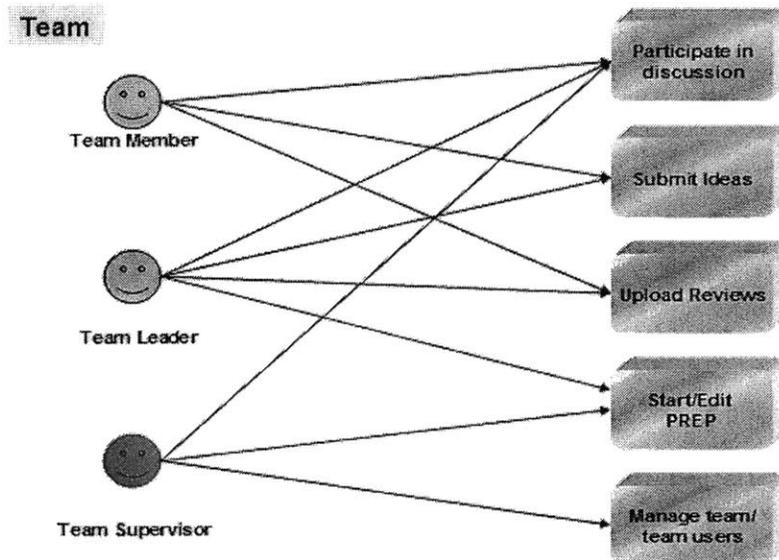


Figure 5 Use case diagram at team level

In a project team group, an authorized user can have one of the three different roles – *Team Member*, *Team Leader* and *Team Supervisor*. *Team Member* and *Team Leader* are students who participate in the Rohrbach process whereas *Team Supervisor* is more of a monitoring role.

Sequence diagram for Rohrbach process

The core functionality of the PREP web application is the Rohrbach process. It is our goal to simulate this face-to-face (F2F) process using software tool. It provides user interface for team leader to upload project definition, for students to submit sketched ideas, checkout other students' ideas, submit reviews, and for supervisor to easily monitor and advise on the proceeding of the peer review process.

Following is a sequence UML diagram that depicts the Rohrbach scenario. Note there are two actors described in the diagram symbolizing two types of actions that can be taken when downloading submitted ideas. One is for pure download and review; the other is for the normal Rohrbach review including download, review and submission. It is necessary to make the distinction between “download” and “checkout” for the submitted ideas. In the physical Rohrbach process, there is only one copy of sketched ideas per team member. If B is reviewing

C's ideas, A can not review them until B is done. This would require A to provide different views on C's ideas other than B's critics since A is expected to see B's review before he writes his own. In the virtual Rohrbach process, once B "checkout" C's ideas, a lock is put on C's original submission; A can only get a copy of C's idea by "download". Once B finishes the review, A would have to "checkout" the document with B's critic to start his own review.

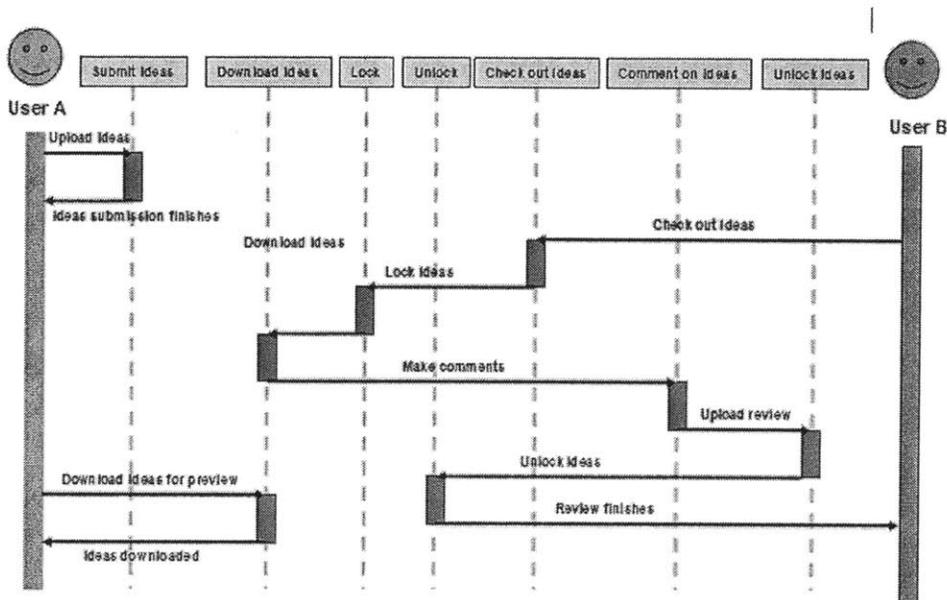


Figure 6 Sequence diagram of PREP

PREP architecture

PREP is written in C# on Microsoft ASP.NET platform and Common Language Runtime. Since it is built on the web services model, many of its web service components could be replaced with web services built with Java or other language/platform, or re-used for other web applications, as long as the standard SOAP contract is honored. This brings tremendous flexibility to the system, making it adaptable as time passes by and new functionalities or better algorithms for the existing functionalities need to be plugged in.

The overall structure of PREP is built with extensibility and flexibility in mind. The core rendering engine of PREP provides "sockets" to plug in any kind of ASP.NET user controls

which in turn may be backed up by web services which reside on different machines, which in turn may access database server on yet another different machine. This gives tremendous freedom for future functionalities to be easily and seamlessly integrated without affecting the existing modules.

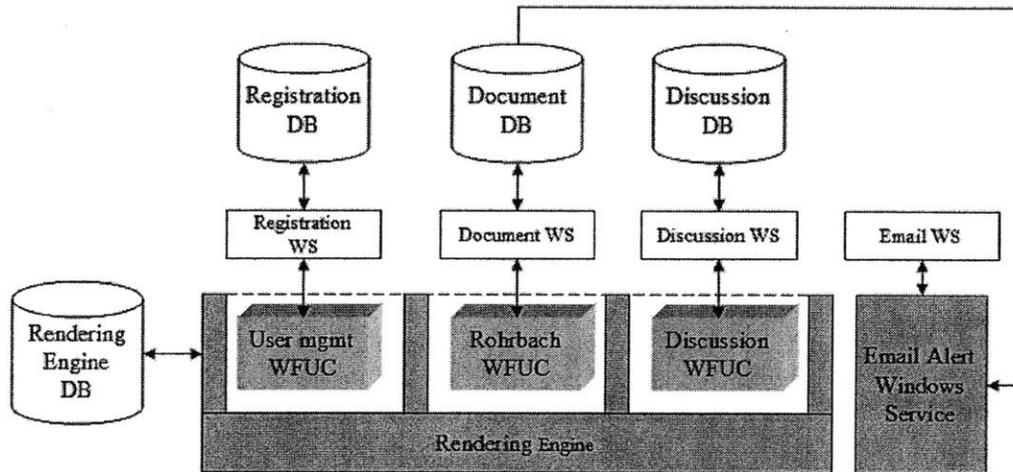


Figure 7 Web service architecture of PREP

Current PREP system consists of the following modules:

Rendering engine

The rendering engine is responsible for taking in user inputs, sending out requests to other service modules based on the user input, gathering all the information returned by the service modules, and laying out the information on a web page generated on the fly, and delivering it back to client browser in HTML stream. It uses its own database to store page configuration information.

The rendering engine leverages the ASP.NET web form user controls (WFUC). Developer can write small “page-lets” – web form user controls. They are organized to form various virtual tabs. Depending on the tab user clicks, and his role in the current group context which determines his permissions, the controls are loaded on the fly and laid out on the current tab.

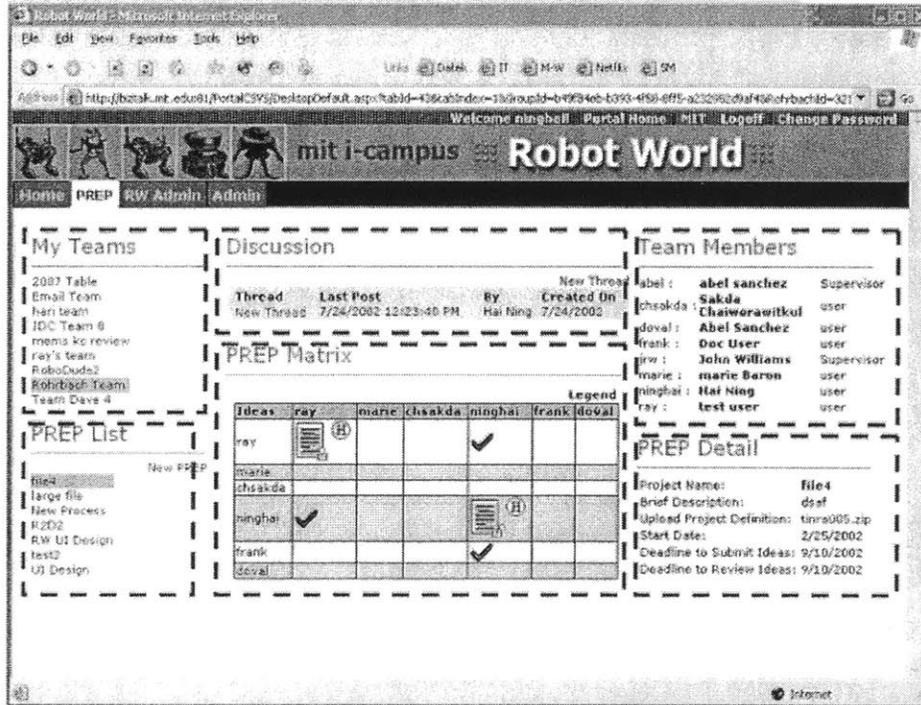


Figure 8 Web form user controls

User management WFUC

This is a set of user controls that handles administrative management of PREP site users. It provides UI to register new users, authenticating users, providing authorized permissions, create new teams, adding or removing users in teams and etc. It is backed up by the registration web service.

PREP Matrix WFUC

This is a set of user controls that handles peer review process work flow. It provides UI to start/edit/delete a Rohrbach process, upload/download documents, lock/unlock documents and etc. It is backed up by the document web service.

A unique feature of Rohrbach process web form user controls is the PREP matrix. This is the tool that let Rohrbach participants start their peer review process. The document icons appearing diagonally across the matrix indicate whether a particular user has uploaded his ideas for review. The green ticks indicate finished reviews. The small lock/unlock icons associated with document icons tell user if a document is checked-out for review at this moment. Just by clicking on the document icons, user can access the “downloading” function if the document is currently checked out and locked by another user, or the “check-out and review” function if the documents is unlocked and free for check-out. The supervisors and everyone in the team can get a clear, graphical view of the status of current Rohrbach process.

The other notable feature is the document history. In the physical Rohrbach process, it is not easy to tell which comments are made by whom unless it is carefully documented, which does not typically happen in a brain-storming process. With virtual Rohrbach process, the document web service keeps all versions of the document. By clicking on the “H” icon to the top-right hand corner of the document icon, user can get a document history that provides links to every single revision so that they can always refer back to the original document.

Idea History:

Username	Action	Date	Document
pjwst10	new	7/29/2002 2:12:15 PM	ideas for 20007 Tab
slocum	check-out	7/30/2002 11:20:41 AM	ideas for 20007 Tab
slocum	check-in	7/30/2002 11:30:25 AM	ideas for 20007 Tab
slocum	check-out	7/30/2002 4:58:54 PM	ideas for 20007 Tab
slocum	check-in	8/1/2002 7:35:44 AM	ideas for 20007 Tab
slocum	check-in	8/1/2002 7:35:45 AM	ideas for 20007 Tab
pjwst10	check-out	8/28/2002 10:24:38 AM	ideas for 20007 Tab
pjwst10	check-in	8/28/2002 10:26:40 AM	ideas for 20007 Tab
ninghai	check-out	9/6/2002 4:01:52 PM	ideas for 20007 Tab
ninghai	check-in	9/6/2002 4:02:23 PM	trace_b.txt
pjwst10	check-out	10/22/2002 2:14:20 PM	trace_b.txt
pjwst10	check-in	10/22/2002 2:14:29 PM	ideas for 20007 Tab
pjgl2	check-out	11/5/2002 11:04:28 AM	ideas for 20007 Tab
pjgl2	check-in	11/5/2002 11:09:35 AM	Review.txt

Figure 9 Idea history

Discussion WFUC

Within the context of a team, team members can participate in threaded discussion. The UI is handled by the discussion user controls, which access discussion web service to store and retrieve the content of the posts.

Web/Windows Services

A range of web services as well as a windows service contains the core business logic for PREP application, receiving input data from the WFUC web interfaces, retrieving output data from respective databases, and performing necessary calculations before put it back out for WFUC to display to client browsers.

Registration web service

Generic registration web service handles users, groups, as well as permissions. It is built to accommodate users from multiple sites. Numerous web sites are built around campus every semester. Many require user registration. People often forget usernames or passwords. Based on our experience with web system user support, a large number of user requests come from frustration with log-ins. If all these web sites use one central registration web service, there is only one username/password pair to remember for the user. It also becomes much easier to manage from a system administration's point of view. In the PREP system, the separation of user management user controls and registration web service allows us to re-use the registration service effortlessly for other web applications. This is the most extensible and most reused web service module of the system. Many other e-education web application we built leverage this web service in such a way that not only the functions do have to be re-written every time, all these web applications can now share a single user identity database. This has enormous impact on possibilities of building personalized solutions for individual users. Here is a list of the main web service calls of generic registration service.

LogOut
GetUsersInGroup
LogIn
SaveAttributeValue
UpdateGroupByGroupId

```

ChangeUserPassword
CreateAccount
GetUserAttributesByUserId
DeleteUserFromGroup
GetUserGroupFromGroupId
GetRolesInUserGroup
IsUserInGroup
GetUserAttributes
GetUsersFromGroupWithThisRole
SaveUserPicture
GetUserIdFromUsername
CreateGroupRole
AddUserWithUserIdToGroupWithGroupId
GetUserPicture
ListGroups
GetUserRoleInGroup
UpdateUserRoleInGroup
CreateUserGroupWithGroupName
GetUsersAttributes
GetUsernameFromUserId
LogInWithoutSiteId
AddUserToGroupWithGroupId
ListGroupsUserIsIn

```

Following is one sample SOAP request format the consumer needs to supply to call the LogIn function.

```

POST /GenericReg/GenericUserDatabase.asmx HTTP/1.1
Host: ken.mit.edu
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://deceiver.mit.edu/GenericReg/LogIn"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LogIn xmlns="http://deceiver.mit.edu/GenericReg/">
      <username>string</username>
      <password>string</password>
      <siteId>guid</siteId>
    </LogIn>
  </soap:Body>
</soap:Envelope>

```

The return message of this LogIn call is in the following SOAP format.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8

```

Content-Length: **length**

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LogInResponse xmlns="http://deceiver.mit.edu/GenericReg/">
      <key>base64Binary</key>
      <userId>guid</userId>
    </LogInResponse>
  </soap:Body>
</soap:Envelope>
```

Document web service

This web service handles storing and retrieving generic user documents. It provides a tree-like structure much like Windows file explorer, but actually stores data in SQL database.

Discussion web service

Discussion web service handles the threaded discussion data. Again, this is a common used function for all web sites that can be re-used.

Email alert Windows service

The Rohrbach process usually needs to be finished in a given period of time. Alert emails are sent out prior to the deadlines to remind teams about them. Also, after the deadline, emails need to be sent to supervisors informing them the status of the Rohrbach process. This is done by the email alert Windows service. It runs in the background as a Windows process, much like Unix daemons. Periodically, it queries the database to check if the deadline is approaching or if the deadline has passed and some team member failed to submit their ideas/reviews. If those conditions are met, it will subsequently connect to the email web service to send emails out to the team supervisors.

Email web service

Email web service is another generic web service that can send emails based on SOAP request made by authorized applications. This is web service in its simplest possible form. The only necessary incoming parameters it takes through SOAP calls are message title, body, recipient email address and a reply-to email address. Using the Windows 2000 built-in virtual SMTP (Simple Mail Transport Protocol) server, it fires off the emails to intended email servers. Again, this service although was built for PREP, it has been reused again and again by many other applications.

Chapter 5 Impact of PREP

PREP was ready for the IDC in August 2002. After a few bug fixing, it was used again in the fall 2002 semester's 2.993 Paths to Peace course. Also, a few smaller scale projects used PREP for design communication as well. This chapter intends to provide some preliminary user feedback to the system in both occasions, together with some observations and analysis.

IDC 2002

The IDC 2002 itself was a huge success. About 40 students from 7 leading universities of 7 different countries got together at MIT for the competition. They formed 8 teams, each has about 5-6 member. They overcame the language barrier, among other difficulties. And all teams successfully produced qualified robots to enter the final contest. The final contest was covered on TV stations such as CNN and NECS (New England Cable News).

The objective of the competition was to build a remote-controlled shoebox-size robot that, within 45 seconds, picks up hockey pucks and street-hokey balls that are stationed on a contest table, and puts them into a bucket. At the same time, the robot could to spin a ball-filled pendulum to get more balls onto the contest table. The mass of pucks/balls being put in the scoring bin is calculated; together with the angle in radianse the pendulum spins. The final score is determined by the following algorithm:

$$\text{Score} = (\text{Newtons} + 1) \times (\text{Radians} + 1)$$

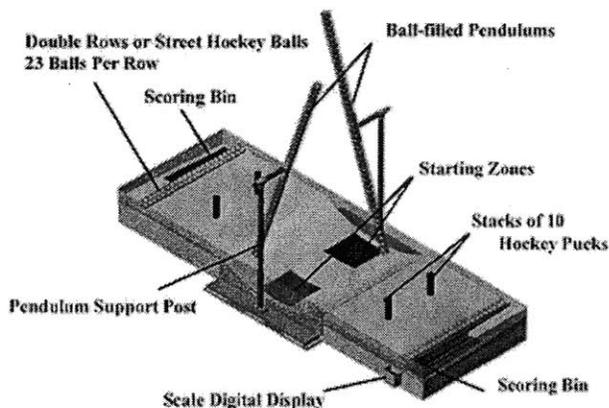


Figure 10 ICD Contest table

Students were highly motivated by the idea of being able to design the robot using their own strategies to achieve the highest score. Project-based design-oriented learning was put in thorough test in IDC 2002. Contestants have to go

through a full cycle of engineering product design in a short period of two weeks.

Tablet PCs proved to be invaluable in the IDC 2002 project teams. The IDC contestants came from seven countries, speaking six different

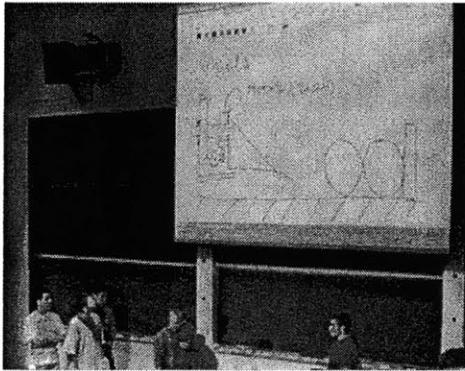


Figure 11 IDC student using tablet PC for sketch presentation

languages. Each team consisted of contestants from a mixture of countries, which meant most students had to communicate in a foreign language. Although the official language is English, not every one mastered it enough to express his or her ideas. Sketching on the tablet PC becomes the natural alternative. Although they could have done it with pen and paper, with the help of MIT's wireless enabled classrooms and machine shops, the tablet PCs allowed them to share their ideas quickly

without resorting to other devices such as scanners and printers.

However, the PREP software itself did not attract as much usage as we originally planned. There reason lies in a couple of facts.

First and foremost, the PREP software is designed to implement the Rohrbach process without the physical presence of team members. This would work very well with geographically dispersed teams or teams that have members operating on different time schedules. In the IDC, on the contrary, all team members were on-site and worked together closely in a central location. This proved to be a difficult challenge for PREP software. It became apparent after a few days into IDC, people would rather make design sketches on tablet PCs and simply copy files through the wireless network than go

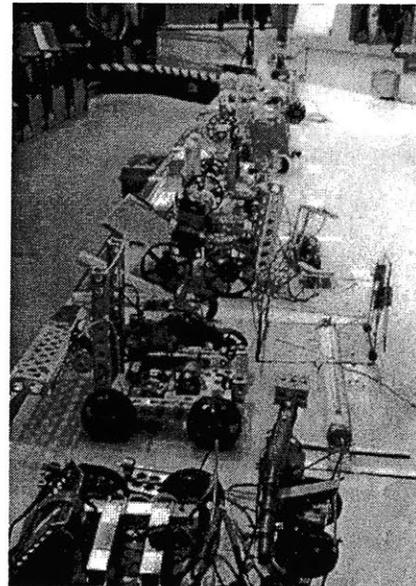


Figure 12 Robots ready for contest

through the upload and download procedure using PREP software, even though PREP software provides revision history track that file system does not. The problem lies in the fact that when teams are under a tight schedule and require real-time synchronous peer-review process, direct interaction between teammates is preferred over asynchronous-oriented facilitating tools.

Further more, because of the short time-span of the project, IDC teams did not have enough time to get themselves familiar with the system. Although the software is user friendly and self-explanatory to people who are familiar with the Rohrbach process, the concept of peer review is new to most of the students not from MIT. In addition to getting used to this new design thinking process, they have many other things to do in two weeks time. And learning how to use the PREP software, although simple enough, was not on their priority list.

In other words, the benefit of PREP software in this case did not justify the time spent learning and actually using the software. Interestingly enough, most students did accomplish a few rounds of Rohrbach process, with the help of tablet PCs, plain file system and the wireless network, meeting our expectation half-way. The PREP methodology proves to be efficient in producing rapid design ideas in a relatively short period of time. As to the hardware – tablet PCs, students quickly adopted them as a useful design tool and made extensive use throughout the competition.

2.993 Paths to Peace

2.993 Paths to Peace was taught in the fall semester of 2002 following the success of IDC. It aimed at developing students' design thinking and mechanical skills through the making of an inlaid or mosaic tile that is precise and can withstand the rigors of time. A final open exhibition of the tiles made was held for MIT community and Greater-Boston public.

As a pilot course using tablet PCs as well as PREP software, 7 enrolled students were divided into 2 design teams. Unlike IDC, this was an individual project course where team collaborations were limited to commenting other member's design ideas.

After the design teams were formed, students started the Rohrbach process with the help of PREP software. Each student came up with 3 ideas for his tile design, documented them with

tablet PCs and uploaded the design files into PREP system. Everyone then was responsible to comment on everyone else's ideas through the check-out, review and upload functions offered by PREP. This was the first round for design concepts. After the completion of concepts PREP, students decided on what they want to build, and then moved on to the next round of strategy PREP where they decided how to approach the design problem, how to convert the artistic theme into engineering manufacture, at the same time with risk analysis and counter measures in mind. After that students started setting up PREPs for various issues depending on their design, and they were encouraged to meet and brainstorm their ideas after finishing every round of PREP.

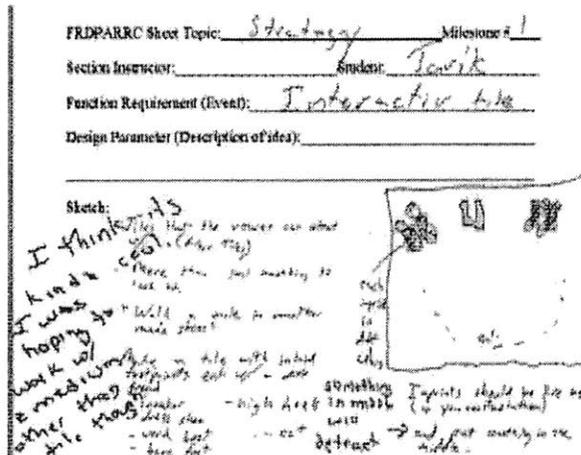


Figure 13 Rohrbach sheet in Windows Journal

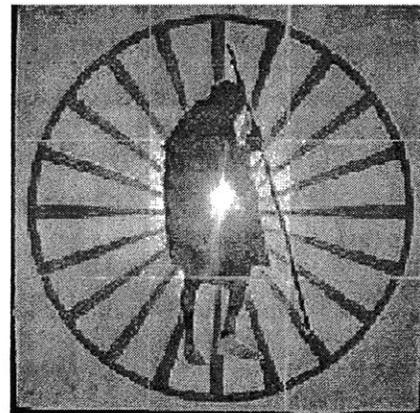


Figure 14 Finished tile

Students showed great interests in using the PREP software to privately comment on each other's ideas prior to the brainstorm sessions. Due to the relatively longer commitment in 2.993 (one semester) and the busy schedule each students typically had to handle, the asynchronous feature of PREP was greatly appreciated. They did not have to spend their already precious time in scheduling group meetings for Rohrbach sessions. Comments and reviews were done without the physical presence of either party. There were no "dominant" or "inert" members. Everyone's ideas were treated as equal contributions. Team moral was high and individual participation was more than active. Even the normally under-appreciated threaded-discussion board was filled with passionate posts regarding the design.

From the teaching staff point of view, the PREP software gave a very clear picture of how the Rohrbach process was carried out. They could find out the status of a particular round of PREP by just looking at the PREP matrix, and download a copy of any Rohrbach sheet submitted by any team member at any time. The system automatically sent out email to remind team member the approaching deadline for the current Rohrbach process he is in. The supervisors (TA or professors) also automatically received emails regarding each member's standing after one round of Rohrbach completed.

The course was finished with successful results. All students accomplished their goals set by their own design strategy and concept.

Design of a totally buried distraction device

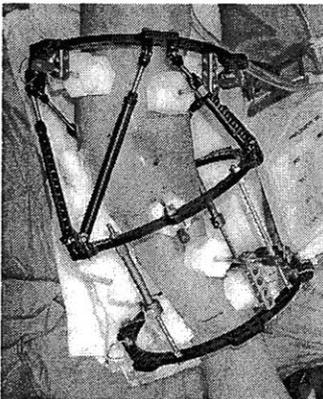


Figure 15 A typical skeletal distractor

In June 2003, MIT Mechanical Engineering Department started a collaboration project together with Massachusetts General Hospital. The goal was to “produce a miniature, completely buried, remotely activated distraction device capable of producing controllable (initially discrete and ultimately continuous) pre-defined multidirectional skeletal movements in the mandibular region.” This device is used in the surgical technique called distraction osteogenesis in which skeletal expansion is forced for the body's natural healing ability to generate bone in a gap. The existing devices are often too cumbersome, complex, and usually require frequent follow-up clinic visits. This new device would be socially acceptable for the patients to wear, would shorten treatment times, could reduce the incident of infection, and would provide highly accurate skeletal movements.

The design team is consisted of an engineer from Advanced Precision Engineering, a machine and subcontract assembly company, two clinical surgeons from Massachusetts General Hospital, Professor Slocum and graduate student Pat Willoughby from MIT's Department of Mechanical

Engineering. Because of the success of the PREP in MIT Mechanical Engineering courses, they decided to use PREP in this cross organization collaboration project. This is a perfect setup for PREP because unlike the first two cases, the team members are not in the same geographical location. And they all have their different work schedules. Getting together even for a phone conference was difficult. Online collaboration asynchronously using the Internet was the ideal way for this team to work together.

Although this was a design project, the first phase was actually drafting up a grant proposal for Small Business Innovative Research program from Department of Health and Human Services of National Institutes of Health. It turned out PREP also worked very well in this seemingly non-design task in a strict sense. Collaboratively working on the same document in the past has proven to be quite a hassle, usually involving mass-emailing everyone in the team the revised document and a lot time wasted in digging up files that is most up-to-date. With PREP, it becomes very clear as to where to download the latest document, who is making the changes at any particular moment. With the locking mechanism, it prevents the development of multiple parallel versions of the same document since it only allows one person to edit it at a time.

After the initial proposal, the joint team conducted multiple rounds of PREP step by step towards the design objective. They went through design strategies, functional requirements, schedules, deliverable milestones, budget, testing plans and etc. Although this was a relatively small team, unlike previous examples, each team member was experienced profession instead of first year or second engineering student, and their task at hand was a real design problem as opposed to an educational one. Thus not surprisingly, in relative short period of time of less than 3 months, they finished astonishingly 22 rounds of PREP in various aspect of the design process, and are well on their way in making the first prototype as of the writing. A CAD model of one of the proposed designs is shown in Figure 16.

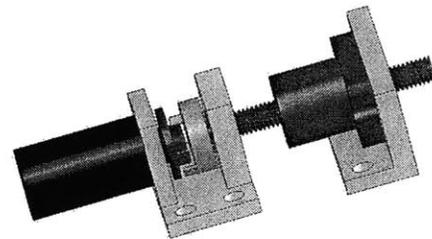


Figure 16 Distractor Prototype

Another interesting factor in this project is the absence of Tablet PC. Due to various reasons, this team was not equipped with this gadget to experiment. However, it didn't seem to affect the usage of the PREP tool. There were a few scanned-in free-hand sketches circling around, but the majority of the electronic documents were in the format of Microsoft Word, Excel or the 3D modeling software SolidWorks. This proves that the PREP platform is content neutral.

This embedded distractor project is still on-going. Nonetheless, it becomes clear that PREP has been, and will remain to be a potent tool for facilitating the design of the distractor through out.

Observations

Besides the aforementioned use cases, PREP also is, or will be, used in a number of smaller scale projects, such as a contest table design for the 2.007 course among the teaching staff, as well as future design classes. So far the result has been very satisfactory. Feedback from instructors and students are overwhelmingly positive. Especially with the Tablet PC, there are no more stacks of design sketches to collect. The whole design activity evolves around a chain of PREP events and becomes manageable even for inexperienced students. A lot of time that would've to be spent in coordinating face to face meetings can be saved and devoted to the actual design task. Even without the Tablet PC hardware and other type of team projects that are less design-oriented, PREP still works very well in supporting collaboration.

There is strong evidence showing that the combination of PREP software and tablet PCs can be very efficient tools for teaching engineering design courses at MIT. Both IDC 2002 and 2.993 enjoyed huge success with the help of these tools, although on different levels. The MIT-MGH collaboration project further demonstrated the feasibility of using PREP not just as in an educational environment, but also as a powerful tool in real world design tasks for experienced engineers.

PREP addresses the asynchronous communication needs in design activities, and it proves to be effective. However, design activities often require direct interactions between participants in synchronous, real-time fashion. There is a need for software tools that can facilitate this type of communication and deliver immediate response among team members. Since the majority of

design communication is conducted visually through sketches, Tablet PCs can become the designer's electronic sketch pad given the right software. Therefore, we created the second piece of the design-oriented educational software – InkBoard.

Chapter 6 Sketch and Design

Before we examine the InkBoard software design, this chapter explains the motivation of building InkBoard, as well as some of the fundamental theories behind sketching in design activities.

The relations of drawing to problem solving

In design and architecture studios, the first thing are often seen is rolls after rolls of drawings, preliminary sketches, construction documents, detail design drawings, computer renderings, etc. It is such a common knowledge that designers heavily rely on 2D drawings when engaging in design activity that we just simply take it for granted. But what exactly is the role of drawing in design? What do we achieve using drawings in the problem solving process particular concerning making physical products as the outcome? Understanding the importance of drawing during all developmental stages of the design process helps to produce better software for supporting design. Here, based on a study in Oregon State University where mechanical engineering students were videotaped during a design process in which they made a number of drawings, we summarize that drawings help design in the following aspects³⁸.

To archive the geometric form of the design

First and foremost, 2D drawing is the most convenient and effective way to expression geometric forms. Texture format used in describing shapes adds a layer of abstraction and hence loses some of the vital information. It is more or less like a one way encoding mechanism such that the lost information in the process can not be recovered by decoding it back. Therefore it can't really be used as a reliable communication tool in this context. At the other end of the spectrum, 3D models preserve more information than 2D drawings and are much easier to absorb for the viewer. And that is why it is not uncommon to see study models being made during the design process. However, the drawback of course is that the models are much more time-consuming to make than drawings. The differences of the level of commitment required in drawing lines on

paper versus making physical models are simply more than a few orders of magnitude. Spending an unnecessary amount of time in making the perfect model not only slows down the design project, it could also subconsciously blind the designer from other possible alternatives.

To communicate ideas

Since 2D drawing has the above advantage in describing shapes and volumes, it is often used to communicate the design. Mechanical engineers make drawings to send to shops or manufactures to produce; architects make drawings for the construction workers to build. To further eliminate the possible misunderstanding, we designed all kinds of rules and systems to make sure the symbols used in these drawings carry consistent meanings from the sender to the receiver.

To act as an analysis tool

Drawing is also used as an analysis tool during design process. The development of the drawing is often the process of refining of the design through constant deliberation. After the initial coarse framework is put down on paper, a lot of missing detailed information such as dimensions and spatial structure of a design is often resolved by careful calculation or trial and error process until they fit together. Thus, the drawing itself not only helps the designers to document the ideas, it also acts as an invaluable analysis tool to help them develop the design. It reflects the private dialog that takes place in the mind of the designer and on paper through drawings. It serves as a communication with the self. It is a form of internal debate leading to the choice and refinement of a design³⁹.

To simulate the design

The simplest form of simulation of the finished product can be performed with a pencil on a piece of paper too. Architects often make contextual drawings of the building's environment in which the newly designed building is situated in order to study the spatial relationship between them and the impact the new building brings to the environment. Mechanical engineers sometimes also use drawings to simulate the interaction between different parts of the machine. 3D model usually gives better sense of the real design, but as we discussed before because of the

low-cost and low-commitment, drawing is still preferred over other forms in the initial development stages of the design.

To serve as a completeness checker

The development of design drawings often follows the pattern from larger settings towards smaller details. The “big picture” is usually conceived and put down on paper before designer dives into details of particular parts. Once the framework is established, the designer starts to fill in the blank. As the drawings are being made, the details left to the designed become apparent to the designer. This, in effect, helps to create an agenda of design tasks left to accomplish.

To act as an extension of designer’s short-term memory

Designers often unconsciously make drawings to help them remember ideas that they might otherwise forget. That is why even after a design project is finished, all hand-made sketches are usually archived together with detailed design drawings. Often the origin of the design is regarded as even more valuable than the actual design product. In a collaborative design environment, understanding how the design ideas by the coworkers come into being through explanatory drawings can be essential to carry out the next step.

Role of Sketching

As the empirical study carried out by the Department of Mechanical and Aerospace Engineering at Arizona State University demonstrates, graphical (pictorial) representation is better than textual (sentential) for recording of design ideas⁴⁰. It also has been observed that engineers are visual learners and, therefore, operate better in a graphic mode⁴¹. The capability of making “back-of-the-envelope” or “cock-tail napkin” sketches to aid in solving a given design problem is greatly emphasized in architectural education. In mechanical engineering field where, perhaps to the outcome of the design, form is not as important as function, engineers are still expected to be able to capture and express ideas going through his/her mind with impromptu sketches. This type of quick sketches allow for clearer thinking, stabilizing generated ideas during the conceptual stage of design, and facilitating spatial and geometric reasoning.

Freehand sketch stands out among other types of drawing activity being much more than a means of communicating design information. Its function includes but not limited to the presentation of spatial information and relationships. Furthermore, it often acts as the link that bridges the gap between the cognitive process of design and the physical world in which the designed artifact eventually will exist. It is the first stage to realize and assess the design ideas.

Booker defines the wider role of engineering drawing as follows: “In its narrowest sense engineering drawing is a language used for communication. However, languages in general are not only useful for communication; they play an inherent part in our very thinking, for we tend to think in terms of the language we know. Drawing is of this nature, and he who can draw can think of, and deal with, many things and problems which another man cannot.”⁴²

Cognitive analysis

A number of different views by a cognitive psychologist can be found in design theory literatures. Some state that mental imagery is a complex linguistic representation with non-logical semantic parts describing a structure of object parts, properties and spatial relationships⁴³. Others believe that visual or graphical reasoning is conducive to creative problem solving and innovation⁴⁴. However mixed these views regard the visual imagery in human’s mind, visual communication is undeniably an equally important means for promoting both individual thinking and group information exchanges.

Sketching plays the most important role in visual communication because of its low commitment of effort and high volume of expressiveness with trained hands. It is an indispensable tool when one tries to convey ideas and information⁴⁵, and hence becomes the predominant activity by the designers⁴⁶. Study shows that a stunning 67% of all drawings done over the course of a typical design project are freehand sketches⁴⁷. Designers are constantly having a conversation with themselves through the cycle of sketching, inspecting, and revising⁴⁸.

Following is a model proposed by Shal, Vargas-Hernandez, Kulkarni and Summers⁴⁹. In this model the initial designer generating ideas is called an ORIGINATOR. The designer’s mind searches mentally through a broad (DOMAIN) knowledge base and through more specific base

related to the problem (CONTEXT). The mental representation is transformed into a physical representation. Depending on the quality and preference on the language skill of the designer the idea is transformed into some graphical solution (sketches) or textual solution (sentences).

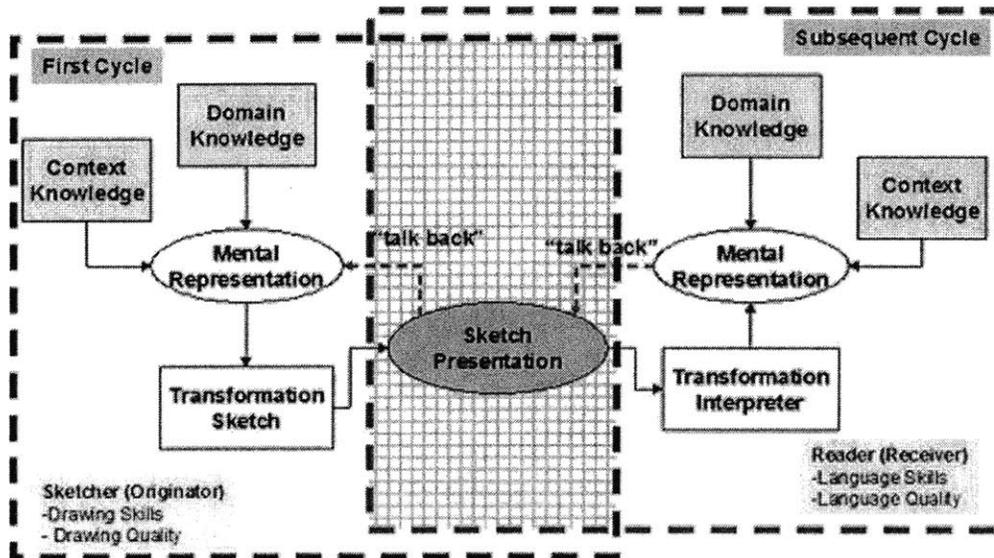


Figure 17 System model for generation and interpretation of ideas

A “talk-back” process generates new mental images when the designer sees his or her idea physically represented and realizes changes to the original idea. The “talk-back” process is repeated until the ORIGINATOR is satisfied with the resulting design solution.

Subsequently, the design solution is then passed on to another designer (RECEIVER). The transformation reverses its course where the physical representation “talk-backs” to the RECEIVER and, depending upon language skill and quality, and DOMAIN and CONTEXT knowledge, the RECEIVER creates a mental representation. The transformation cycle continues until the designer is content with his/her interpretation or understanding of the original design.

Visual Thinking

Visual thinking is loosely defined as the “creative process by which spatial, numerical and descriptive information is integrated to realize a design idea”⁵⁰. As the above cognitive analysis demonstrates, the process of sketching is not simply the hands-on skill of presenting spatial

information and relationships. It serves as an essential tool for the designer to engage a responsive conversation between the boundary of design parameters and the ideas in his/her mind. With today's advancement of computer technology and ubiquitous and powerful CAD software packages, freehand sketching retains a central place in the process of visual thinking. It forms the link between the cognitive processes of idea generation and the representation of design ideas in physical world. In the design process, the visual problem solving method helps the designer to abstract the elements of a problem. Through examining these elements the designer understands the underlying structure, makes judgment and arrives at a decision.

One other important function of sketch in the visual thinking process is provoking creativity. When making sketches on paper, designers consciously group related information together spatially such that they are presented in a concise and meaningful way. Not only faster processing of the information can be stimulated, connections between different pieces of information are also often more spontaneously made while designers study and internalize the sketch they just made. Research shows that pattern recognition occurs subconsciously in human's mind when visual information is obtained through observation⁵¹. This explains why designers are able to extract information beyond what was originally intended in a sketch.

Back to the Drawing Board

Computer technology has taken over the formal drawing mechanism like a storm since late 1990s. Comprehensive CAD (Computer-Aided Design) software packages spread widely in all design fields, from architecture design, urban planning, structure engineering, and mechanical engineering to industrial design. Today most of the architecture construction documents or mechanical design documents are drawn with CAD software. Powerful graphics workstations, versatile digitizers as well as giant plotters have become common sights in design studios and firms. They certainly have made a lot tasks that used to be extremely tedious much easier and less labor-intensive. Unfortunately however, this revolution also brings less favorable impact on some other aspects of the design.

Loss of Public Forum

In the past, large drawing boards served as forums for fellow designers to share their ideas and communicate the design. It was a perfect setup for informal face-to-face meetings where designers gathered spontaneously to examine and discuss design problems. Nowadays, designers tend to work on their own computers with less interaction and brainstorming around the evolving design. The sense of public forum that the drawing board used to supply now is lost in the transition to CAD software. Not only are computer screens perceived as private workspace and hence not appropriate to be studied by others for a prolonged period of time, the nature of “virtual desktop” usually limits the display to a small subset of the entire project. The existing CAD packages have not been able to help designer to reclaim this lost public space that was proven, and still should have been, very helpful in a collaborative design environment.

Lack of abstraction, ambiguity, vagueness and imprecision

It is interesting that in the early conceptual, creative phases of designing, designers almost always consciously reject using computer tools. Rather, they prefer making rough sketches with pencil and paper. One of the main reasons for this seemingly peculiar behavior is because of the ambiguity, vagueness and imprecision that is tremendously helpful in early conceptual design, easily obtainable with freehand sketch, and yet unavailable in today’s popular CAD systems. Current design software, which restricts visual representations to precisely drawn geometric elements, stifles the graphical conversation that designer has with herself. CAD drawings eliminates the suggestive power of the sketch.

Abstraction

Graphical abstraction is the technique in conceptual design where designer uses symbols to represent more detailed configuration of intended parts. By employing abstraction, designer can focus on a higher level of the system, or “The big pictures”, without specifying their internal structure. This is much like the “black box” concept in software engineering.

Ambiguity and vagueness

During the idea searching process, designers often put down several alternative options in a particular part of design. The intentional ambiguous representation in the sketch allows designer to keep the options open, delay the commitment of decision to a later stage when more conditions are gathered and explored. This ambiguity and vagueness should be preserved in the early design process instead of being forced for clarification as most CAD software would.

Imprecision

In the early conceptual design stage, designers only need rough dimensions to decide on a basic layout. Not only it gives designer maximum freedom in dealing with configuration, layout or other higher level aspects of the design, it also discourages them from dwelling on specific issues with too much details, or making overly fine-grained decisions. Over time, designers often develop skills in making freehand sketches that have close approximations of fixed scales without resorting to rulers or other measuring devices.

No incremental formation process support

From plotted CAD drawing one can hardly follow the incremental progression of the design, since it is more than often a “cleaned up” version of the initial sketch. The ease of erasing, duplicating, modifying geometric shapes in CAD software often subconsciously encourage designer to constantly remove, refine, or replace shapes drawn during the formation process of design. The final product may be a very clear and clean representation of the design, but the thought process that could have been revealed by the comments, lines or shapes drawn along the way is lost forever.

No easy way for rapid exploration of alternatives

Designers often use translucent tracing paper to make conceptual sketches, the benefit being that one can overlay a sheet on top of an existing design idea and quickly explore a different spatial configuration based on the previous drawing. If the outcome is unfavorable, he can simply discard it and start with a new sheet; or if it is worth exploring, he will add another alternative to

his solution collections. With CAD software, instead of pulling a sheet of tracing paper and instantly starting to draw, one probably has to get involved with a series of complicated operation such as saving the old file, opening a new file using the old file as template, creating new layers on top of existing drawings, turning layers on and off to see the current status, and etc. Again, in this case, freehand sketch is simply so much more efficient than CAD for exploring alternative solutions quickly and easily.

The promise of the Tablet PC

It is our hope that the emergence of the Tablet PC could mitigate, if not entirely eliminate, the aforementioned problems with traditional computers running CAD software packages, since most of these problems are caused by the awkwardness of the reigning hardware interface, namely, mouse and keyboard. Keyboard is the natural input device for textual information. With the invention of windows-like graphical user interface and mouse, the archaic text command lines, now only cherished by die-hard geeks, are replaced by simple action of pointing and clicking. However, it is sufficient to say that freehand drawing was never a task made easy by these input devices, not even with the appearance of digitizers or digital tablets. There is simply no substitute for the free flowing feeling of pen running on paper. Tablet PC, with its natural user interface, is trying to change all this. Following are some of other benefits Tablet PC promises to deliver.

Pen-based natural user interface

Perhaps the most striking difference one feels one first time using a Tablet PC is the smoothness of the flowing ink when writing on the screen. Pen-based computer is not a novel idea. Many previous products, such as the GRiDPad from GRiD Computing in 1989 and the Newton MessagePad from Apple in 1993, had tried to commercialize this idea. Unfortunately they all fell short when it comes to simulate the natural feeling of using pen and paper. That is, of course, until the release of Tablet PC. Being able to use the stylus writing directly on the surface of the laptop computer screen and seeing the digital ink appearing from the tip of the stylus is a very satisfactory experience indeed. It removes the frustrating barrier of old input devices where users are forced to move their hands holding a digitizer on top of a pad resting on the desk while trying

to focus their eyes on the screen to see the outcome. The pressure-sensitive digitizer screen also adds the natural smooth feeling of real strokes of ink with varying width. Thus, designers will be able use Tablet PC just as if they are using pen and paper to make conceptual sketches. Combined with the ease of storing and retrieving digital sketch, perhaps even with shape recognition, at certain stage when the designer is confident enough about the idea, the sketch made can even be imported to CAD software for further detailed drawings.

Advanced operating system

Microsoft Windows XP Tablet PC Edition is a superset of Windows XP Professional version. It provides the power of Windows XP with no sacrifices. Tablet PC Edition has the full capabilities of Windows XP Professional, plus additional features for tablet pen-based computing. And, because it uses the Windows operating system, Tablet PC can run all Windows XP-compatible applications.

Windows XP Tablet PC Edition lets users interact with their PC in a more natural way by incorporating the convenient and intuitive aspects of pen and paper into the PC experience. Using a tablet pen and Tablet PC Input Panel, users can write directly on the screen and save their notes in their own handwriting – or convert them to typed text for use in other applications. The pen can also handle common mouse and keyboard tasks like opening applications, selecting text, and displaying menus. Or, if the users prefer, they can still use a mouse or keyboard with Tablet PC.

Enhanced mobility and feature-rich applications

The Tablet PC provides allows users to be productive in more situations – at the desk, in the hallway, at a meeting, or on the go. Tablet PC comes in two basic forms: the "convertible," with an integrated keyboard, and the ultra-slim "slate tablet," which has docking solutions for easy access to the keyboard at the desk. Tablet PC supports grab-and-go removal from a docking station and has a fast resume-from-standby time. These capabilities, combined with wireless network support, offers greater mobility and immediate access to the full power of the PC.

Windows XP Tablet PC Edition comes with Microsoft Windows Journal, a note-taking utility that lets user create and organize handwritten notes. Windows Journal makes it easy to capture the text and drawings users would normally create using pen and paper. Advanced handwriting recognition technology lets user search his/her handwritten notes to quickly find what he/she need.

Integration with Office suite

Windows XP Tablet PC Edition lets user integrate electronic "ink" into popular business applications such as Word, PowerPoint, Excel. Ink integration is native to Microsoft Office 2003 Editions. Users can then share these handwritten documents with other PC users – even if they are not using a Tablet PC. Non-tablet PC users with Windows 2000 or Windows XP operating systems can read handwritten documents with the free Microsoft Windows Journal Viewer.

Personalization

Using Tablet and Pen Settings controls, users are able to customize their Tablet PC: calibrate the pen, optimize the Tablet PC for left- or right-handed operation, and program the hardware buttons of the Tablet PC to complete specific actions, such as opening an application or changing screen orientation from landscape to portrait.

Extend Existing Applications with Digital Ink Handwriting

Windows XP Tablet PC Edition is a powerful platform for developers and the foundation for a new generation of applications with pen and ink capabilities. For example, Windows XP Tablet PC Edition has powerful but simple ink controls and application programming interfaces (APIs). These APIs allow software developers to extend existing applications with pen and ink capabilities and to develop new applications using this technology. We will talk about this in detail in the following chapter.

Summary

As the above discussion and aforementioned literatures demonstrated, freehand sketch remains to be the indispensable and powerful visual thinking tool despite the fact that CAD software have largely replaced the drawing board in design studios nowadays. It not only serves as communication medium for getting ideas across fellow designers, it also plays important roles in the self-critic conversation designers constantly have with themselves.

It is self-evident that most CAD software is not created for facilitating free-hand sketches. Rather, the goal is to replace drawing board and triangles to produce precise documentations of the design so that the manufacture/construction partners can make the product accurately based on these drawings. And the reason that the need for creating software for facilitating early conceptual design process is largely ignored by the software industry can be attributed to the limitation and awkwardness of existing user input devices. Keyboard and mouse are really not the best tools to make free-hand drawing on the screen.

The Tablet PC promises to change all this. The immense amount of effort goes behind the Tablet PC software and hardware design ensures an extremely natural pen-on-paper experience that has never existed before in consumer computing environment. With this UI barrier comes down, it is expected that sketch-oriented software that targets early conceptual design phase will become the new competing field for CAD software vendors. Already, the leading 3D modeling and rendering software vendor Alias has shipped its Tablet PC sketching software *Sketchbook Pro*, and the top tier graphics software company Corel released its own sketching software *Grafigo*, also on Tablet PC platform. Autodesk, makers of the industry standard design software such as *AutoCAD* announced that their leading software package *AutoCAD Architectural Desktop* would also support the Tablet PC in future releases.

In the next chapter, we will discuss the sketching software built on the Tablet PC platform that focuses on the collaboration and communication between fellow designers in real time.

Chapter 7 InkBoard Software Design

As we have seen the innovative Tablet PC should make a great hardware to support freehand sketch. It comes with a piece of simple note-taking software called *Windows Journal*, and a more complicated but similar application named *OneNote* is shipped as a component of the new *Microsoft Office* suite. Other graphics software vendors are also releasing sketchbook applications based on Tablet PC. However, one conspicuous blank field that is still waiting to be filled is in the area of collaborative sketching engaging multiple clients.

Motivation and Goals

Motivation

With the PREP software we created, Mechanical Engineering Department teaching staff are able to organize the Peer Review Evaluation Process among students within a design team efficiently. The PREP software provides an asynchronous process for students to exchange documents and make comments/critiques on each other's design sketches.

However, a lot of times design teams hold interactive brainstorming sessions where they make drawings, talk about their ideas, and expect instant feedbacks from their design peers. It is clear that PREP software is not setup to address this issue. Making a sketch locally (even if it is on a Tablet PC), submitting it to the PREP Matrix, downloading at the other end, making comments, saving it locally and submitting it back to the system... There are simply too many obstacles from the initiation of the process till receiving the feedback that it is not feasible to carry on an interactive design conversation using PREP. The Tablet PC, essentially a laptop with writable screen, is designed to be an electronic sketch pad, and would naturally be expected to fulfill this role. However, the missing piece of this was the lack of networking capability support that is essential for real time direct interaction among peers.

Goals

Our main goal is to create an Ink-enabled sketch pad application that connects multiple design team members simultaneously through network (preferably wireless network) and allows them to communicate their sketches in real time. With this overall objective in mind, there are three important objectives to achieve.

Design-oriented education needs

The InkBoard should address the common needs of design-oriented education. Although Mechanical Engineering Department courses are our co-investigators of the RobotWorld project and hence our initial targeted customer, we should keep in mind that InkBoard should be content neutral just as in the case of PREP. Students in all design fields, including but not limited to mechanical engineering, architecture design and urban planning, should be able to leverage this tool in their design process. However, this is not to say that we shouldn't consider the possibility of offering discipline-specific elements such as architectural or mechanical design symbol libraries like the ones that can be found in Microsoft Visio.

Scalability

The InkBoard software needs to be highly scalable. Since it targets small design teams, 2-5 design members are expected to participate in a given session. This is a reasonable estimation when you consider in face to face design meetings group larger than 6 usually find either lost in too many digressions from everyone or dominated by a few participants. This does not mean that InkBoard only deals with small amount of simultaneous clients. We want to provide one piece of server software that can handle multiple design sessions at the same time. For instance, 40 people in 8 meetings with each meeting having 5 participants. If this doesn't sound a lot, imagine all of them making simultaneous ink strokes on their Tablet PC. The design of the networking protocol and the way the InkBoard server handles the traffic is the key to accommodate large amount of users and yet still maintain reasonable response time on the client side.

Application usability

InkBoard needs to be Tablet-friendly. The Tablet PC is a revolutionary new device that has changed the computing experience for users. Of course there were other earlier pen-based devices such as PDAs and digitizers that provide similar interactions between human and computer. However, none of which were able to offer such an experience so close to real pen and real paper. When traditional mouse and keyboard input devices give ways to the Tablet PC, it is only natural to expect a different UI design for pen-based software. InkBoard, being sketch pad software, would be primarily used with the stylus touching the screen. Because pen and mouse operate differently from one another, it is a big challenge to offer new UI experience that better fits pen-paper type of interaction but at the same time establishes subconscious connections with traditional windows GUI design so that users would not feel totally disoriented. For example, dropdown menus are conventional windows application features, but when using a pen directly touching the screen right-handed user often find the menus are blocked from their eyesight by their own hand. Providing the ability for users to customize the windows menus from drawing themselves on the left or right side of the root menu greatly enhances the Tablet PC's usability and yet maintains the familiar UI convention to decrease the learning curve of this new computing hardware.

Related works

Before we discuss the design of InkBoard software in detail, two related research project that have similar goals but take on different approaches are examined.

Conference XP

Conferencing Experience Project, or Conference XP, is a research and development initiative of Microsoft Research's Learning Sciences and Technology group. It aims at providing an extensible foundation for interactive collaborative environments in the form of high quality audio/video conferencing and messaging over high-bandwidth of Internet 2 within a multicast-enabled network environment. It also provides support for transfer of other kinds of real-time

data. We are able to integrate the Conference XP components with InkBoard to add the audio and video conferencing capability to enrich the online design meeting experience. However due to the limitation of the hardware of popular Tablet PCs, the relative rarity in Internet 2 infrastructure, the video conferencing experience is not entirely satisfactory. But the audio worked very well which in our belief is largely what is needed in design meetings in addition to the graphical communication provided by

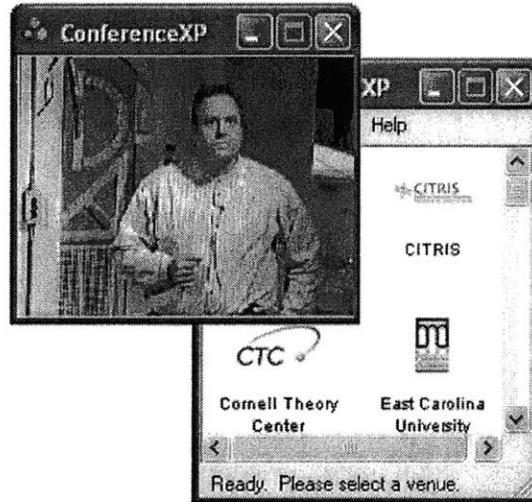


Figure 18 ConfereceXP client

InkBoard. One other intrinsic limitation of Conference XP, of course, is the relative unavailability of multicasting across different networks. This we will address in detail in the following chapter when we discuss the InkBoard Message Protocol design.

ReMarkable Texts

Another interesting project conducted closely in collaboration with Conference XP is ReMarkable Texts from Brown University. It is built upon some of the Conference XP

technology, and focuses on real time classroom experience with ink-enabled slides presentation application. The messaging protocol that ReMarkable Texts adopts is derived from the *RealTimeDocument* class developed as one of the Conference XP API components, which will be the subject of an interesting comparison study

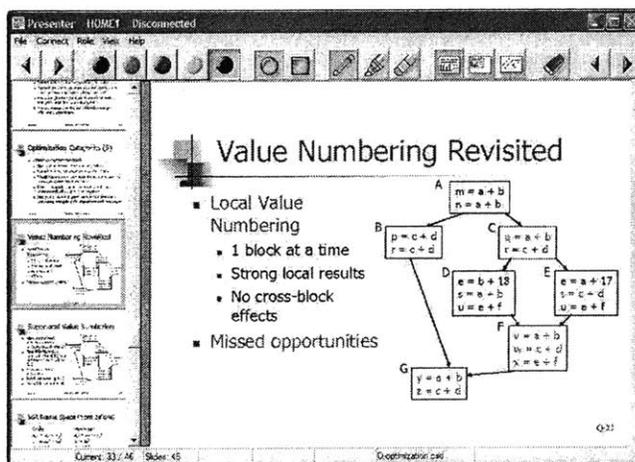


Figure 19 RemarkableText user interface

together with InkBoard Messaging Protocol in the next chapter. The functionalities of ReMarkable Texts are similar to InkBoard in that it provides networked ink sharing capabilities on Tablet PC platform. The limitation there again though, is that it only works on a multicasting network environment that is not commonly supported across different TCP/IP networks, precisely because it adopts the Conference XP messaging protocol.

Tablet PC SDK

Tablet PC SDK (Software Development Kit) is published by Microsoft for developers to create Ink-enabled desktop software on Tablet PC. It has gone through notable changes since its debut in 2001. Its ease of use and free availability attracted a large devoted developer community constantly providing constructive feedback, driving the SDK to better improvements.

Overview

The Tablet PC Platform SDK has several major conceptual components. The core of the SDK is composed of a set of APIs exposing pen and ink features that can be leveraged by Microsoft .NET applications. As a way of supporting simple and rapid integration of ink into applications, several .NET and Microsoft ActiveX controls are provided so that Ink-enabled components can be directly dragged and dropped on to Windows application panels the same way text labels or checkboxes are incorporated. There is also support for COM (Component Object Model) which enables the interactivity of Ink object with applications like Microsoft Office suite. (COM is a core Microsoft technology that has been used widely in the world of Windows programming. These APIs are provided as an alternative to the managed APIs, for use with C/C++ and Microsoft Visual Basic 6.) To make the development of Tablet PC applications possible on non-Tablet PC editions of Windows, the SDK installs the necessary runtime libraries. In other words, you don't need to have a Tablet PC to create or run Ink-enabled applications so long you have the SDK installed.

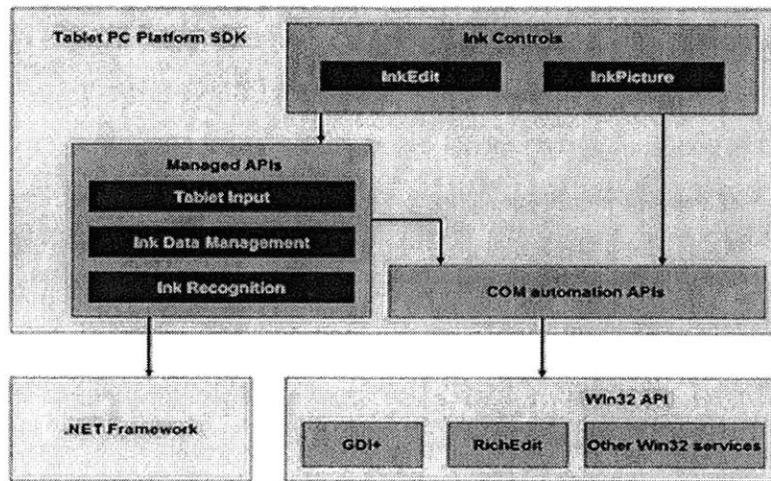


Figure 20 Tablet PC SDK API architecture

The high-level view of the Tablet Platform SDK architecture is illustrated in the above block diagram. The arrows in the figure represent dependencies between components. At the lowest level of the SDK are the COM automation APIs, which are implemented in C/C++ and directly use Microsoft Win32 calls. The managed APIs are built on top of the COM automation APIs, essentially providing a managed wrapper for that functionality. It's worth noting that the Tablet PC managed API is not merely a subset of the available COM automation features. The managed APIs are fully able to do all the things that the underlying COM automation APIs can, often in an easier fashion by using features made possible by the .NET architecture.

Managed APIs

At the heart of the Tablet PC Platform SDK is a set of managed APIs. A significant part of the Microsoft .NET Framework is the common language runtime (CLR), which controls and supports the execution of .NET-compatible applications written in various languages. The core Tablet PC Platform SDK APIs are “managed” because they are designed to run in, or be managed by, the .NET Framework’s common language runtime. Because the managed APIs target the CLR, you can call them from any .NET language, Such as Microsoft Visual C#, Visual Basic .NET, managed C++, and Microsoft JScript.

The managed APIs are divided into three subsets, each providing a specific portion of essential Tablet PC pen and ink functionality. The Tablet Input API is targeted at pen-specific features, such as the various buttons on a pen, and also at collecting digital ink and gestures from the movement of the pen. The Ink Data Management API provides functions for manipulation and storage of the Ink once it has been collected using the Tablet Input API. The bulk of ink-related features are exposed through this API. The Ink Recognition API is used to interpret ink intelligently by grouping and recognizing written ink.

Ink controls

The Ink controls are built with C/C++ and exposed via .NET wrappers for easy access through C#, Visual Basic .NET, and other .NET-capable languages. They can be drag-and-dropped in Windows Form Builder to easily assemble Ink-enabled applications. Unlike the managed APIs, the Ink controls expose only a subset of the available underlying functionality.

Although the Tablet PC Platform SDK is designed to create applications targeting at Tablet PCs, it can be installed and used on any PC running Windows XP. This allows for development and testing of Tablet PC software on whatever computer is most convenient for the user. However, some of the Ink controls are severely limited when used on any computer not running Windows XP Tablet PC Edition. Most notably, the Ink controls go into what is essentially a “render-only” mode, where new ink cannot be captured and existing ink cannot be further recognized. It may be helpful to have a digitizer pad if development is carried out on a desktop computer. Not having a pen-based input device makes ink entry and testing somewhat challenging.

COM Automation APIs

For developers more familiar with COM, the Tablet PC Platform SDK includes a set of COM automation APIs that are almost direct analogs of the managed APIs. The COM automation APIs allow development of Tablet PC applications in unmanaged code, using languages such as C++ and Visual Basic 6.

Ink – the new native data type

As we have discussed in the overview section, there have been a lot of attempts, some of them even commercial, in digital handwriting and recognition. One of the early misconceptions was that the computer had to do a perfect job in recognizing handwriting. The correct recognition rate was the single most important metric in measuring the success of the algorithm. However, Tablet PC designers realize and admit the limitation of machine. When sometimes the author herself cannot read what she wrote, how can we expect the computer do a better job, considering it does not understand the contextual meaning of what it tries to recognize? The approach Tablet PC takes is to let ink remain as ink. It treats digital ink as a naïve data type. The computer can offer its own interpretation of the ink when asked, but the original ink is preserved as an equally important data type of its own. It is the Tablet PC developers' hope that Ink will in the foreseeable future become as ubiquitous as text, image, audio or video.

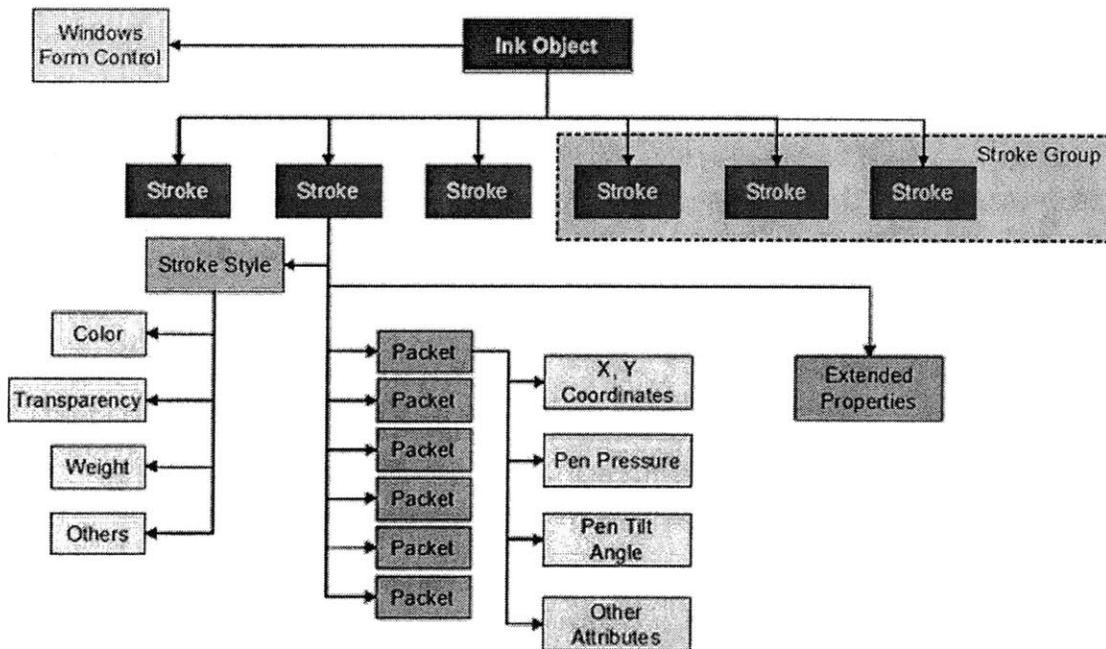


Figure 21 Ink data object model

Above is the data object model of Tablet PC digital Ink. An instance of the Ink class is automatically created when an InkCollector or InkOverlay object is created and attached to a Windows form object such as a panel. The panel then becomes the canvas for Ink Stroke objects. Each Ink object can contain multiple Stroke objects. The trace left from the pen touches the screen till the pen is lifted is considered to be one Stroke object. Stroke objects can be collected from the Tablet PC digitizer screen from the pen/mouse action on the panel object, or they can be created programmatically through API function calls.

Stroke objects can be grouped together as Stroke groups called Strokes. Strokes and Stroke objects are different concepts but can be confusing. Strokes class refers to a group or a set of Stroke objects. Different values can be assigned to properties of Stroke object such as color, transparency, weight and etc. Each Stroke object consists of multiple packets, or points. Packets are sampled based on time intervals during which the stroke is created by pen and digitizer screen. Each packet again has properties such as X Y coordinates, pen pressure, pen tilt angle and so forth. Customized properties can also be added to stroke object as extended properties.

This is indeed a very complex data type. Fortunately Tablet PC SDK provides a large set of APIs to manipulate this data type programmatically. With the API function calls, you can create Ink object, attach it to a windows panel, collect Ink strokes from user input, or create Ink stroke based on given coordinates, change Stroke properties on the fly, and etc.

Challenge: Building ink-enabled communication tools

With the complex Ink data type and rich Tablet PC SDK API functions, it is rather easy to create a Windows application that is capable of collecting freehand writing or sketching. However, the emphasis of InkBoard as we discussed before lies in the collaboration functions. At the very bottom of multiple client communication is the Ink networking support, which happens to be the one feature missing from the Tablet PC SDK. Other than providing a proprietary serialization method that turns ink stroke object into byte arrays, much is left for the developers to figure out how to transfer ink object through TCP/IP network in real time between Tablets. And this is the main technical challenge of InkBoard. In the following text, we will discuss the system

architecture of InkBoard in general, and its usability considerations. In the next chapter, we will focus on the network protocol designed for Ink objects and implemented by InkBoard.

System architecture

One of the early design decisions to make for InkBoard is whether to go with the traditional client/server architecture, or take the P2P (peer-to-peer) approach.

P2P or client/server?

P2P allows easier deployment without the hassle of setting up a central server. It has the advantage of having no single point of failure. If one node fails for some reason, it will not affect any other peers connected, whereas with client/server architecture, if the server fails, the whole session can not be continued.

The problem with P2P in this particular situation has to do with data storage. Since we are trying to create Ink-enabled software to communicate sketches through the network, it is necessary to store the sketches for future reference. With the P2P approach each client has to keep a copy of the drawing. And for every resumed session, the application has to make sure each copy from each client is identical, or there has to be some way to reconcile the differences. Adopting client/server architecture would avoid problem since there is only one master copy of the drawing on the server. Clients can choose to save their local copy on their hard drive. However when it comes to collaborate with others, everyone download the server copy to continue.

The other disadvantage of the P2P architecture is the large amount of network traffic it can generate. As we stated earlier, InkBoard sends every stroke each client makes to the participating peers. With P2P architecture, every peer becomes a server sending strokes to everyone else. This greatly impacts the complexity of programming and creates unnecessary network traffic that is hard to monitor and regulate. In client/server architecture, each client only deals with the InkBoard server. It sends its own stroke to the server and let server decides which peers to relay them to. And it only receives remote strokes from the server, which reduces the complexity of

network traffic. The added benefit is that all strokes go through the InkBoard server and makes it easy to store in a central database for later retrieval.

In the end, the client/server structure is chosen for InkBoard because of its ease to collect transferred data and the predicable and controllable network traffic pattern.

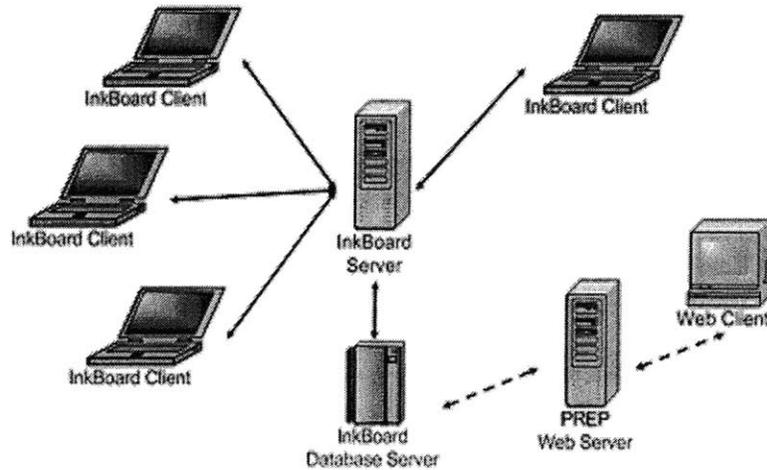


Figure 22 Client/server architecture of InkBoard

In the client/server architecture illustrated above, multiple Tablet PC clients running the InkBoard client application connect to the central InkBoard server via TCP/IP network. They send every stroke the users make, or any other type of actions, over the InkBoard server in the form of InkBoard Message (discussed in detail in next chapter). InkBoard Server collects the messages sent from every client, figures out the meaning of the messages, stores them in the InkBoard Database Server and follows the recipient list of each message and sends them out again to the respective clients. The clients receive these messages, restore them back to strokes and take the appropriate action to display them. All these steps happen in the real time.

InkBoard Client Architecture

InkBoard Client is an Ink-enabled Windows application that has very rich user interfaces and features. Inside the client Windows form class, there are three main components – InkBoard drawing area, UI tools and the network manager.

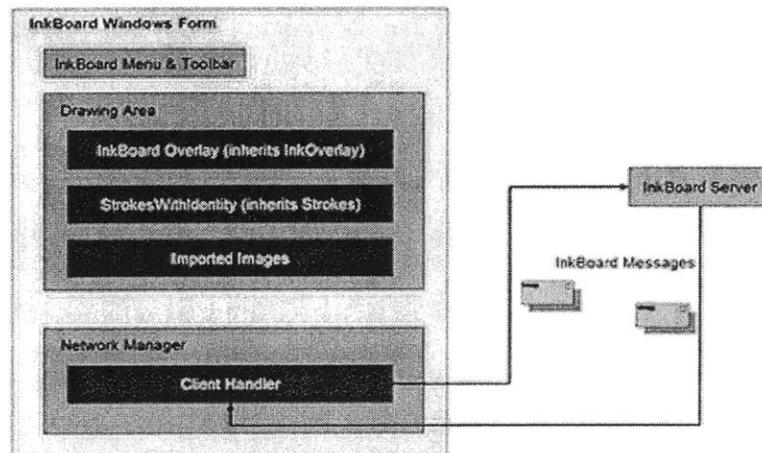


Figure 23 InkBoard Client Architecture

InkBoard drawing area

The main drawing area is realized using a class called *InkBoardDrawingArea* inherited from the *System.Windows.Forms.UserControl* class. It includes a Windows panel that attaches itself to an *InkOverlay* object to collect user Ink strokes, horizontal as well as vertical *Scrollbars* for moving the panning and zooming, an image *ArrayList* object to hold images uploaded by local user or transmitted over the network from other users and a range of other elements that deals with the collecting and drawing of Ink as well as images.

One of the most noteworthy and yet seemingly invisible features is the double buffering mechanism implemented in the Windows *Panel* object. For a standalone sketch pad application, there is no need for double buffering since the Window would only be redrawn when it is reactivated. And usually a flicker is expected by the user at this moment. However, it is entirely

different with networked sketch book. Imagine you are working on one area of the drawing surface, and simultaneously your colleague sends a stroke through the InkBoard Server that is supposed to be displayed in a different visible area. To maintain the real time sense, you would expect the application updates the screen as soon as it receives that stroke. Unfortunately, this would cause the whole drawing area to flicker every time the application receives a stroke, which makes your drawing experience quite frustrating. Of course, one solution is to clip out the region unaffected by the new stroke received from the server and only update the bounding box of the new stroke. But what if the bounding box of the new stroke happens to intersect with the current stroke you are making at the same moment? The only feasible solution then, is to create memory maps to handle the drawing before displaying it on screen, i.e., double buffering.

The .NET managed *Windows.Form.UserControl* object has a built-in double buffering flag. When it is set to true, Windows is supposed to be smart enough to create memory maps automatically and handles the double buffering for it. Unfortunately it is not supported for controls that have *InkOverlay* object attached to it. Therefore, we have to manually build a double buffer graphics object called *DibGraphicsBuffer* from unmanaged code and supply it to the managed rendering function of the *Ink* object.

Thanks to the classes built in *System.Runtime.InteropServices* namespace we are able to call Win32 system functions directly from managed C# code. Thus, we can conduct direct dialog with Win32 to have the system allocate blocks of memory for a bitmap the same size of the drawing area sitting in the video RAM. Using the bit-block transfer technique, the Tablet PC can take the advantage of its graphic acceleration hardware to manipulate bit blocks on the screen independently of the contents of the rest of the screen. This not only eliminates the annoying screen flickering, it also speeds up the redraw process of the drawing area.

After the double buffer graphics object is built, we turn off the automatic rendering flags of the *Ink* object, and overwrite the *OnPaint* event handler of the *InkOverlay* that holds the *Ink* object. Following is an excerpt of the overwritten paint event handler. A *DibGraphicsBuffer* object is created based on the drawing panel's graphics object, then all the drawings including background, imported images and mostly the collected strokes are painted directly on the

DibGraphicsBuffer object. After the painting is finished, the bitmap is then copied into the panel's graphics object, and the *DibGraphicBuffer* object is destroyed.

```

public void HandlePaintEvent(Graphics g, Rectangle rect)
{
    try
    {
        using (Graphics panelGraphics = g)
        {
            // create the double buffering graphics object
            Graphics bufferGraphics = dbGraphics.RequestBuffer(
                panelGraphics,
                Width, Height);

            // set the clipping region
            bufferGraphics.SetClip(clientRect);
            lock (bufferGraphics)
            {

                using (Brush b = new SolidBrush(pnInk.BackColor))
                {
                    bufferGraphics.FillRectangle(b,
                        new Rectangle(0, 0, rect.Width,
                                    rect.Height));
                }
                // paint the background first
                background.Draw(bufferGraphics);
                DrawImportedImages(bufferGraphics);

                // draw strokes
                ... ..
                inkOverlay.Renderer.Draw(bufferGraphics,
                                        strks.Strokes);
                ... ..

                // paint the buffered image on to panel's
                // graphics object
                dbGraphics.PaintBuffer(panelGraphics, rect, 0, 0);
                dbGraphics.ClearBuffer();
                bufferGraphics.Dispose();
                bufferGraphics = null;
            }
        }
    }
    catch (Exception e)
    {
        mainForm.SetStatus(e.ToString());
    }
    return;
}

```

The drawing area component also includes an array of *StrokeWithIdentity* objects. In order to identify the owners of each individual stroke collected by the *InkOverlay* object, each

participants of a particular session is given a GUID (Global Unique Identifier). The GUID is recorded together with the stroke object as one of the extended properties. Other customized extended properties include time stamps that record the time when the stroke is made or deleted, an string object indicating the stroke sequence number, another *DateTime* object recording the timestamp when user send out the stroke.

```
| public static readonly Guid GUID_USER_ID; |  
| public static readonly Guid GUID_DELETE_TIME_STAMP; |  
| public static readonly Guid GUID_STROKE_ID; |  
| public static readonly Guid GUID_STROKE_SEQ; |  
| public static readonly Guid GUID_SENT_AT; |
```

Add an extended property to the stroke object:

```
| stroke.ExtendedProperties[GUID_STROKE_ID] = "12"; |
```

Retrieve an extended property from the stroke object:

```
| Stroke_Id = stroke.ExtendedProperties[GUID_STROKE_ID].Data.ToString(); |
```

Other UI tools and thread safety

These include the menus, toolbar, as well as Windows controls on the side panel that handles server name, port, as well as audio/video conference controls. Many of them are designed with the consideration of Tablet PC friendliness in mind and hence have some interesting features. They will be discussed in detail in the following usability section.

The InkBoard client application has very rich user interface elements intended to provide a friendly experience for sketching and other related activities. The network component as we will discuss in the next section, operates in its own thread space. When an InkBoard message comes in, the client application needs to translate that into some sort of action visible in the front UI. However, this poses an interesting problem. The network thread cannot directly access the UI thread because the UI thread has its own message loop. For instance, the InkBoard client application implements a Microsoft Messenger-like presence icon list. When a new peer joins the online design session, the network thread gets an *IBM_Update* (to be explained in the next chapter) message telling that a new peer has come online. The network thread cannot simply call for an update of the *ListBox* in the UI thread to add the new client's icon to the existing team

members' icon list. The UI thread has to provide a delegate function for the network thread to call in order to carry out this function.

```
private delegate void UpdatePeerListDelegate();
private UpdatePeerListDelegate UpdatePeer;
.....
public ClientHandler (ServerNetworkManager caller,
                     Socket socket, FrmMain creator)
{
    .....

    // delegate to call main form's functions
    UpdatePeer = new UpdatePeerListDelegate(creator.UpdatePeerList);
    .....
}

public void ProcessClientRequest()
{
    .....
    frmCreator.Invoke(UpdatePeer, new Object [] {});
    .....
}
```

In the above code, *UpdatePeerList* is created as an instance of the delegate function *UpdatePeerListDelegate*. Using the *Invoke* method, the network manager component is able to call on the *UpdatePeerList* function that belongs to the main form of the application, and hence running in the UI thread.

Network Manager

The third piece in the InkBoard Client application is the Network Manager class that handles the network transactions of user's strokes. There are quite a few challenging issues in building a real time networked sketching application.

First of all, user's strokes are in the form of instances of *Stroke* types and need to be serialized into binary streams before they are sent over the network. Although Tablet PC does provide a serialization mechanism for the *Ink* object, its format is proprietary. In order to add other meta data about the particular stroke, or to represent actions taken by the user other than making a stroke, a new message class is constructed to handle this. In the next chapter, we will closely examine the structure of this new InkBoard Message class.

Once we have the serialized InkBoard Message object, the next step is to send them to the server using TCP/IP protocol. Again, we didn't choose the P2P approach of sending it directly to the other clients to avoid generating large amount of network traffic and bringing down the performance of the client. The asynchronous writing to the network stream is adopted because we don't want the client hanging until the sending is completed. The client UI should be responsive during the sending process. A callback function is implemented so that the client application is notified when the message is sent.

Besides sending InkBoard Message objects, the client also has to be able to receive them from the InkBoard Server. The reading stream function is also implemented in asynchronous fashion so that the client UI maintains the responsiveness while receiving messages. Thus, the ink-collecting user experience will not be interrupted by the arrival and display of the new strokes sent from the InkBoard Server.

InkBoard Server Architecture

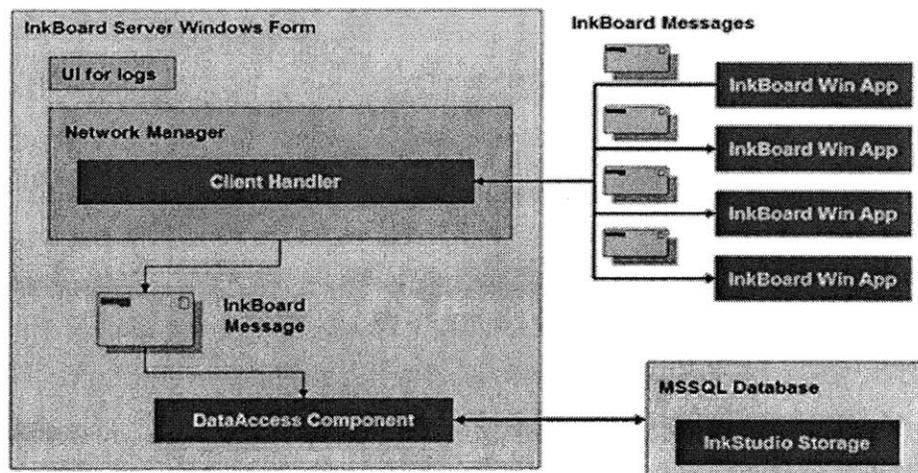


Figure 24 Inkboard server architecture

The major component of the InkBoard Server is the Network Manager that handles the network traffic of incoming and outgoing InkBoard Messages. It also includes some UI element for server logs, and a MSSQL database called InkStudio that stores all the InkBoard Messages.

Network Manager

The network manager component of InkBoard Server is a multi-threaded TCP/IP server. A client handler object is created every time a new client connects to the server, and consequently a new thread is spun off to handle the message transaction between that client and the server.

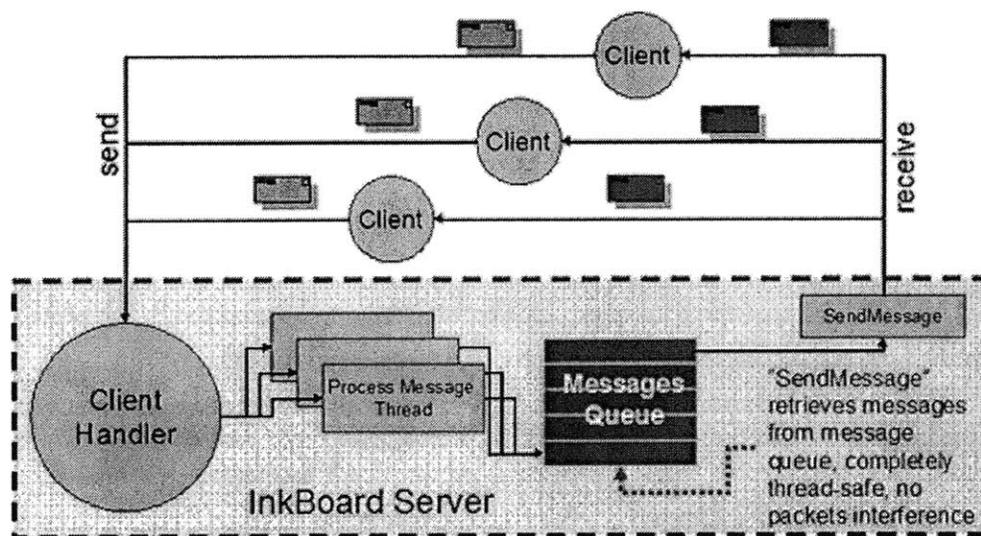


Figure 25 Inkboard network manager

One of the interesting problems occurred in the message sending process is packet interference. The server passes on every message it receives to the respective client handler, and depending on the nature of the message, the client handler takes appropriate action. In most cases, the message would be a new stroke message and hence needs to be broadcast out to other participants of the same drawing. Consequently, multiple client handlers operating on different threads could potentially compete for the network socket in order to send out the message. This would cause a problem because the network write function is implemented in asynchronous way. When one message is still being written to the channel, another thread could take over and start to write a

new message. This packet interference leads to the confusion of the client receiver and cause it unable to decode the received binary stream. To prevent this from happening, the messages are pushed into a message queue implemented on top of *System.Collections.Queue* class.

```
public class MessageQueue
{
    private System.Collections.Queue _msgQ;
    private System.Collections.Queue _clientQ;

    private ServerNetworkManager nm;
    private DataAccess da;

    public System.Collections.Queue MessageQ { get { return _msgQ; } }
    public System.Collections.Queue ClientQ { get { return _clientQ; } }
    public MessageQueue(ServerNetworkManager mgr)
    {
        nm = mgr;
        _msgQ = new Queue();
        _clientQ = new Queue();
        da = new DataAccess();
    }

    public void DispatchMessages()
    {
        ...
    }
}
```

Periodically, the *DispatchMessages()* function retrieves messages from the queue instead of being called by spun-off client handlers. It will not process the next message until the current one is finished sending and notified by the asynchronous callback function. This ensures the integrity of the messages being sent out by the server, and hence the integrity of the messages received by the respective client.

Ink Studio database

Ink Studio is the MSSQL database that is designed to store InkBoard messages. It is a SQL database consisting of four tables. Basically four entities are constructed – studio, drawing, message, and user. Multiple drawings belong to one studio, and multiple messages belong to one drawing. For each drawing there can be multiple users as well. Of course some messages may not directly link to any particular drawing.

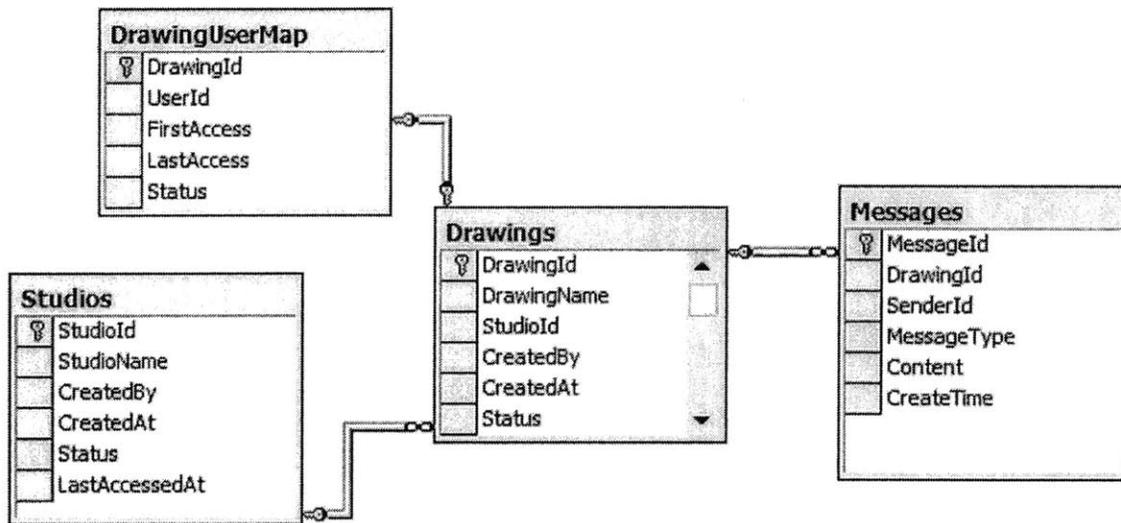


Figure 26 Ink Studio database E-R diagram

InkBoard messages base class implements a *SaveToDB()* function to serialize itself into the messages table. Figure 26 is an E-R diagram of the Ink Studio database structure. The column *Content* takes in binary data type, and is used for storing the serialized message. The column *MessageType* ensures the message can be de-serialized back to appropriate message class.

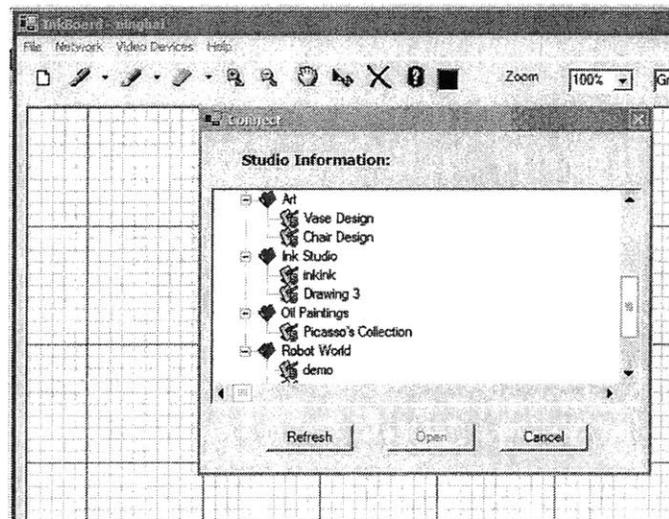


Figure 27 Connecting to Ink Studio from InkBoard

Every time a client connects to the InkBoard server, a dialog box (Figure 27) is presented with all the existing studios and drawings they include. The client can simply choose to open an existing drawing, in which case all messages related to that particular drawing will be retrieved in chronicle order and sent back to the client. The InkBoard client application, upon receiving these messages, faithfully executes the respective actions and thus recreates the drawing for the user the same way it was drawn before. Alternatively, the client can choose to create a new studio or start a new drawing, in which case the drawing or studio tablet will get updated and a blank drawing area will be presented to the user.

InkBoard Usability

One of the things that we paid special attention to when we designed InkBoard client application is the issue of usability. Pen on digitizer screen is such a different way of interacting with

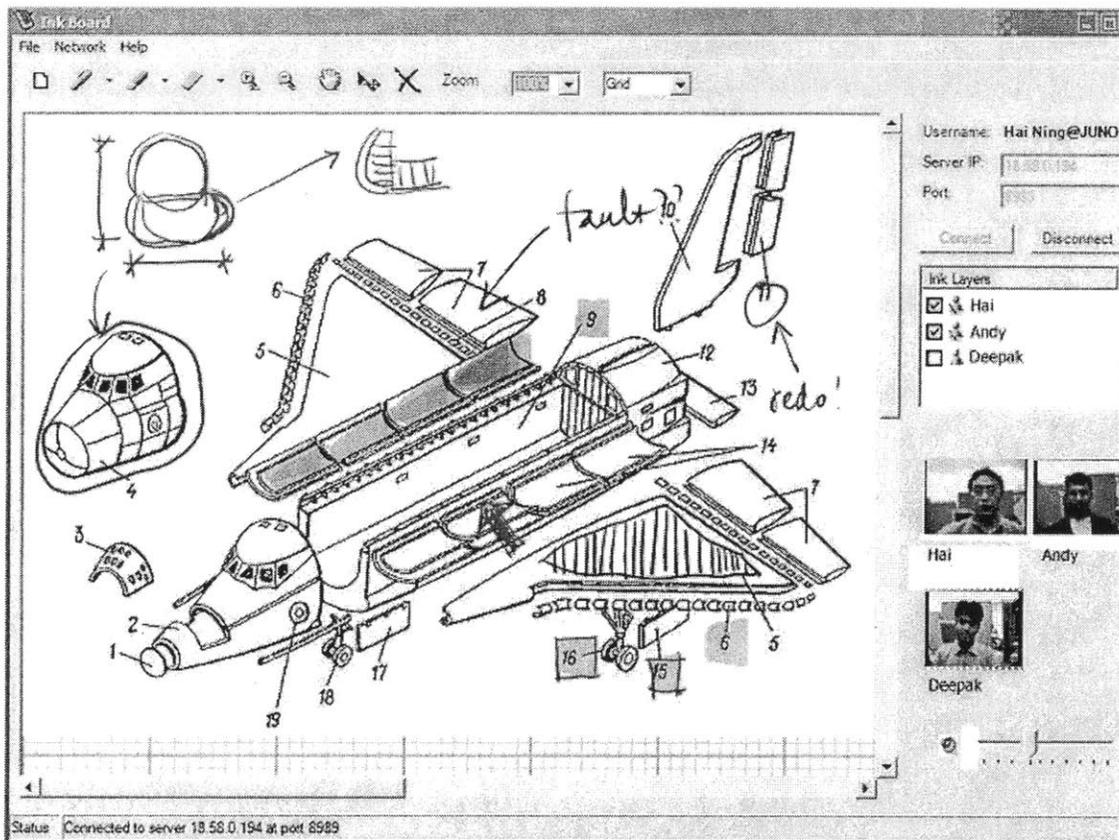


Figure 28 Inkboard main UI

computers from keyboard and mouse that many user experiences that are taken granted for do not work as well in the new hardware platform.

The main interface of InkBoard (Figure 28) resembles very much a traditional sketch pad application with central drawing area and drop-down menus. However, there are many subtle and yet important unconventional features specifically targeted for Tablet PC user experience.

Sign-in dialog box

Since InkBoard is a network application, each user needs to identify herself when she connects to the server. Of course IP address can be used, but in order to give each user a more human-recognizable name, it is often common to have user sign in using a user name of their choice. Normally this is done with a text box presented to the user at the time of connecting. With the advance handwriting recognition algorithm, it is easy and great fun to create a sign-in dialog box (Figure 29) that automatically converts user's handwriting into username. And yet at the same, we don't want to introduce any frustration caused by this unconventional UI change.

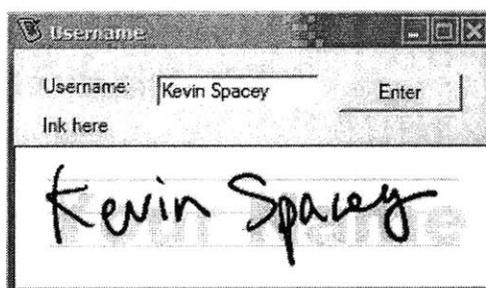


Figure 29 Sign-in dialog box with handwriting recognition

To accommodate users preferring typing, we keep the text box for the traditional input method. And we add an Ink-collecting space under the text area for users preferring using a stylus. Tablet PC SDK actually provides an ActiveX object called InkEdit that implements the function of collecting ink and converting to text. However it looks just like the traditional text box, and in this case would have certainly created confusion had we used it. Instead, we manually

implemented the above functions in a panel object that has 3 guide lines and grayed out word “Your Name” as the background and the text “Ink here” as the title of the control. These small details help user to clearly understand their expected action here is inking instead of typing.

InkBoard toolbar

On the top of the main drawing area, ten different buttons lined up as the most common tasks performed by an InkBoard user. From left to right, they are:

1. New drawing: erase everything user has drawn so far and start a new drawing
2. Pen: change pen color, width and etc.
3. Highlighter: a special pen with transparent color and wide brush width that imitates a highlighter. Again, you can change its color, width and etc.
4. Eraser: erase ink strokes.
5. Zoom in: zoom in on the drawing area.
6. Zoom out: zoom out on the drawing area.
7. Pan: moving the view port on the drawing area
8. Move: move image objects
9. Delete: delete image objects
10. Stroke Info: display information about selected ink stroke object.

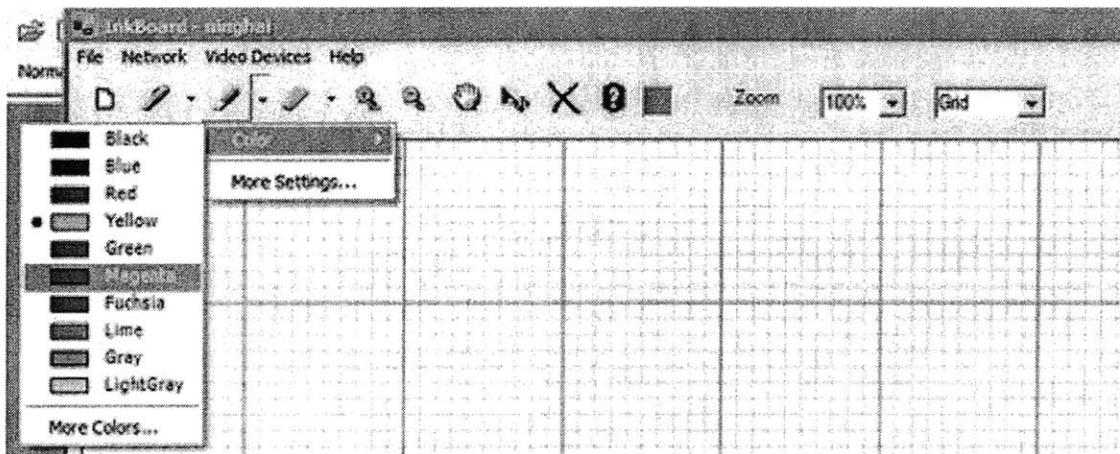


Figure 30 Inkboard toolbars

All of these functions can be performed with a touch of the stylus on the respective buttons so users never have to resort to keyboards. The pen and highlighter buttons also contain a drop down menu that includes some popular colors and widths of the pen for easy access. It is also worth noting that the drop down menu appears on the left side of the root menu as it shows in the above screen capture. This is to accommodate right-handed users so that their hand will not block their view when operating the stylus. This setting can be changed to the other direction for left-handed users as well.

Color picker

At the immediate right along the ten buttons on the InkBoard toolbar is the color picker. Users can use the pen or highlighter buttons to change current pen color, but that involves one click on the arrow besides the button, another click on the color root menu, and a third click to choose a color from the list of 10 colors. This is ok with a mouse but not optimal when using a stylus pen. The InkBoard color picker looks just like a rectangle filled with current color, but when you “pen-over” it (not clicking it but just moving the stylus over the rectangle), a swatch of 512 colors appears on top of the drawing area. When you move your pen around within the swatch, the size of the color box nearest to the pen tip is magnified two times to indicate this is the candidate for the current color. With one touch on the screen at this moment, this color is selected. And as you move the pen outside the swatch area, the whole swatch disappears, revealing the drawing area. Thus, with a single touch users are able to select a color from 512 choices. And it does not occupy any valuable screen real estate.

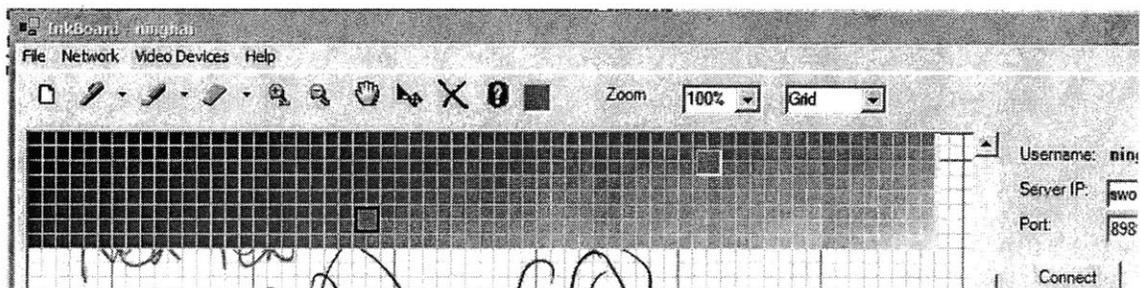


Figure 31 Inkboard color picker

Timeline

Another interesting UI feature is the time line. Since we have recorded every message sent by every client to construct a particular drawing, we have the capability to describe each stroke made by each user using a time line whose length is proportional to the time intervals between the strokes. A VCR like interface (Figure 32) is provided to the user so they can move the time marker along the time line. Any stroke made after the time marker becomes the “future strokes”, and thus grayed out. Dragging the marker back and forth will cause the drawing appears or disappears just as if it was being drawn or erased. User gets a clear picture of how the drawing came into being by observing this process. This greatly helps them to understand the design process, enhance the design experience immensely.

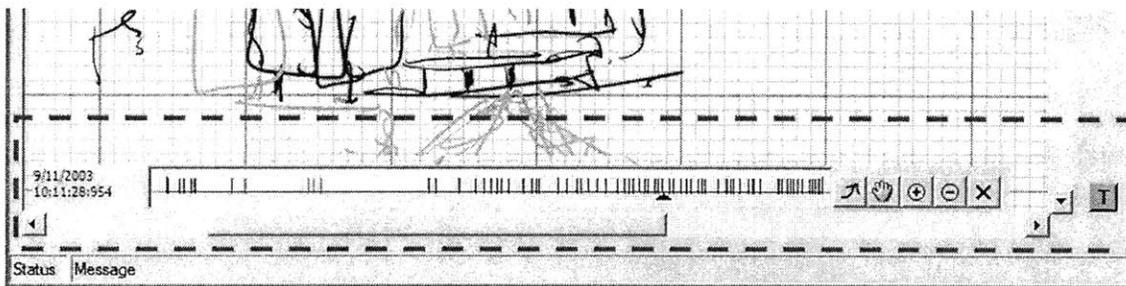


Figure 32 Inkboard timeline component

Presence and ink layers

Like many network collaboration program, InkBoard adopted an UI element that imitates the instant messenger buddy list (Figure 33). When a user starts the client application, he will see an icon representing himself in the list box. After he joins a collaborative sketch session, everyone who is working, or has worked on the same drawing appears too as icons. In consideration of simplicity, the icon has only three states right now. Green head means online, red head means offline, and green head plus a tiny flashing red pen means this person is right now making strokes. This last state has been particular useful since it indicates the actions being taken from

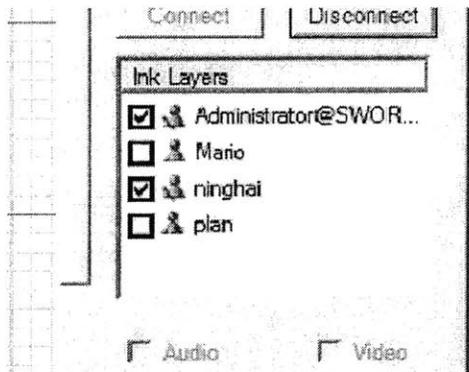


Figure 33 Presence and ink layers

the other end of the network. This gives the user a very satisfying sense that someone is working simultaneously with her, and it stimulates responsive actions from her own end too.

The drawing area of InkBoard stores ink stroke objects in layers according to their owner. Again, this is done by adding an extended property to the stroke object identifying the user who initially makes the stroke. Since the rendering function has been hijacked by a custom function, we can decide

at the run time which strokes should or should not be drawn on the canvas. Thus, we are able to use an checkbox in front of the presence icon so that by clicking on it, user can turn on or off the layer of – and thus all the strokes made by – a particular user. This is very helpful for identifying individual contributions of the drawing.

Chapter 8 InkBoard Messaging Protocol

The technical challenge in developing InkBoard application largely lies in dealing with networking capability of ink strokes. The messages sent and received by InkBoard clients and server are largely new ink strokes created by participating clients. But that is not always the case. The client could be issuing command of erasing a stroke he made earlier, clearing all strokes he sent earlier, or sending an image to the peers, or simply wanting to open an existing drawing stored on the server. The possibilities are not endless, but it can be in large quantities. A networking mechanism needs to be able to handle all these situations. And hence a basic message structure and a messaging protocol have to be put in place. Following sections discuss the anatomy of InkBoard Messaging Protocol (IMP), and present a comparative study of IMP against another similar research effort from Microsoft Research.

The anatomy of IMP

Clearly in order to handle various different types of ink related messages, the sensible thing to do is to implement a base class structure that captures the fundamental elements of similarities among these messages, and then build specific type of messages using class inheritance from the object-oriented programming concept. The advantage of doing this is that we would only need to implement one set of network mechanism to be able to handle the base class message type, instead of having to write different networking code for each of these different message types. Since all other message types inherit from the base message class, the networking code would be able to handle all of them. Plus, this leaves room for future expansion of the message type. In case there is ever a need to implement a new type of message, we can simply create a new message class inheriting the base class without even touching the networking module.

Base class InkBoardMessage

Figure 34 is a diagram of all the existing message types that inherit from the base class *InkBoardMessage*. This base class has a few member variables that are used across the board,

such as an enumeration of *InkBoardMessage_Type*, three GUID (Globally Unique Identifier) respectively representing *UserId*, *DrawingId*, and *MessageId*, and finally a *TimeStamp* variable. The only member function it implements is *SaveToDB()*. As discussed in the last chapter, it is used to serialize all messages into the *InkStudio* database for storage and retrieval.

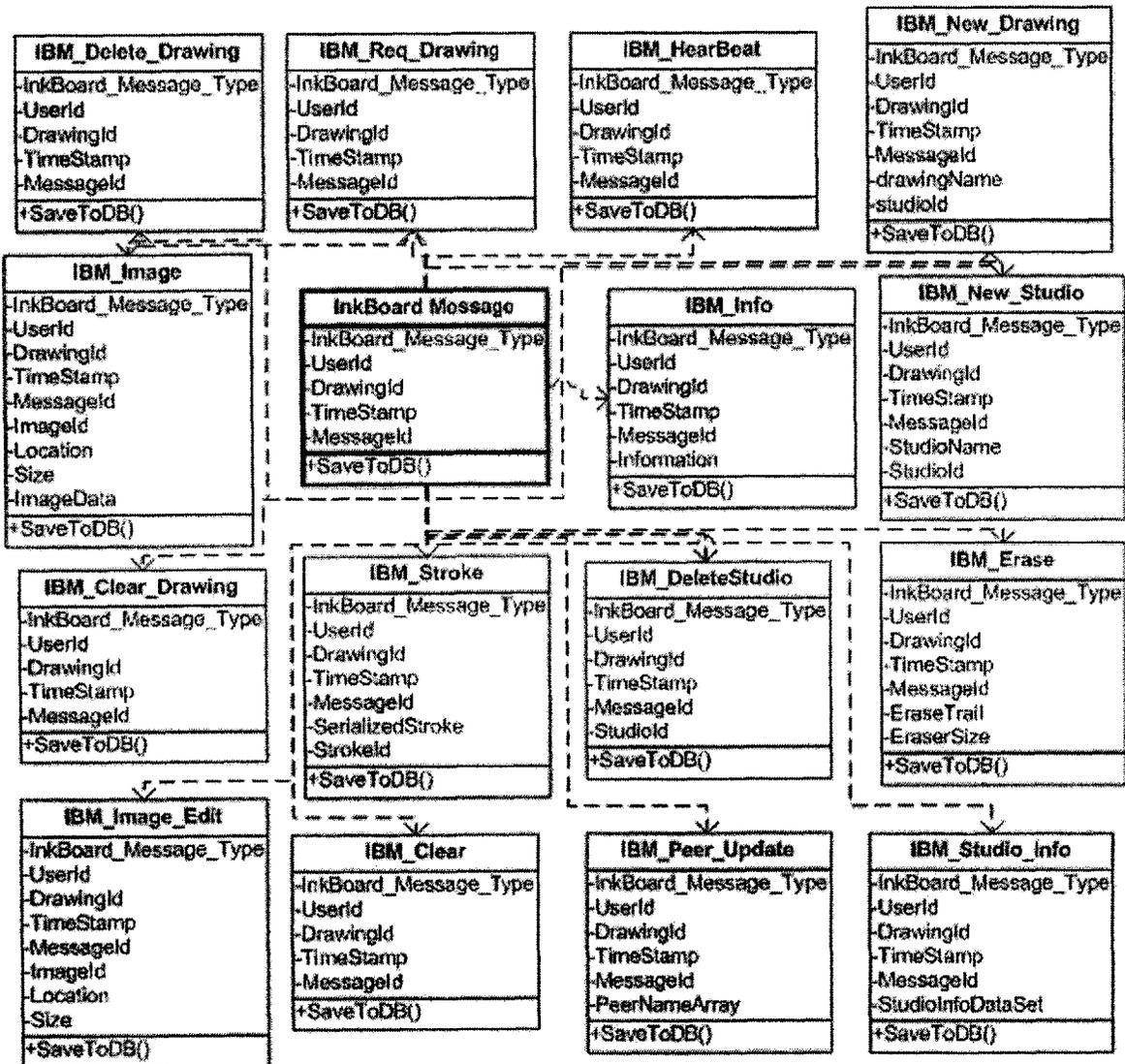


Figure 34 InkBoardMessage base class and derived message classes

One of the caveats for building the *InkBoardMessage* class is that all of its member variables have to be *serializable* in order to be saved into a network stream object. Microsoft.NET Framework Class Library provides an attribute class called *SerializableAttribute*, which can be applied to a type to indicate that this type can be serialized. The CLR (Common Language Runtime) throws *SerializationException* if any non-primitive type in the graph of the declaring object being serialized doesn't have *SerializableAttribute* applied. In order to realize the serialization, all member variables in the type labeled *Serializable* must be either primitive data types that CLR knows how to serialize, or custom defined types that contain custom serialization implementations. In the case of *InkBoardMessage* class, all member variables have primitive data types that can be serialized automatically by CLR.

```
[Serializable]
public abstract class InkBoardMessage
{
    public enum INKBOARD_MESSAGE_TYPE
    {
        IBM_CONNECTION_CLOSING,
        IBM_STROKE,
        IBM_ERASE,
        .....
    }

    public abstract INKBOARD_MESSAGE_TYPE MessageType {get;}

    public InkBoardMessage(string strUserId)
    {
        userId = strUserId;
    }

    public InkBoardMessage (string strUserId, Guid drId)
    {
        userId = strUserId;
        drawingId = drId;
    }

    public void SaveToDB()
    {.....}

    private string userId;
    public string UserId
    {
        get { return userId; }
    }

    private Guid drawingId = new Guid(
        "{00000000-0000-0000-0000-000000000000}");
    public Guid DrawingId
    {
        get { return drawingId;}
    }
}
```

```

    public readonly DateTime TimeStamp = System.DateTime.Now;
    public readonly Guid MessageId = Guid.NewGuid();
}

```

Derived class *IBM_Stroke*

Right now there are 17 derived message classes (the diagram only shows 14) inherited from *InkBoardMessage* base class. A couple of them are worth closer scrutiny. *IBM_Stroke* is the most important one in that it is the one sent by a user when she make a ink stroke on her end of the InkBoard client application. Besides the inherited member variables and function, an object called *SerializedStroke* of byte array type is added in. Although the class structure of *Ink* object is published by Microsoft, the actual implementation of it is still proprietary information. Fortunately the Tablet PC Platform SDK provides a serialization function for *Ink* object called *Save()*. This function returns the *Ink* object as a binary array object which is a primitive data type that can be serialized automatically by CLR, and thus making the derived class friendly for network stream serialization.

```

[Serializable]
public class IBM_Stroke : InkBoardMessage
{
    public IBM_Stroke (string strUserId, Guid drId)
        : base (strUserId, drId) {}

    private const INKBOARD_MESSAGE_TYPE msgType =
        INKBOARD_MESSAGE_TYPE.IBM_STROKE;

    public override INKBOARD_MESSAGE_TYPE MessageType
    {
        get { return msgType; }
    }

    public byte[] serializedStroke;
    public string strokeId;
}

```

Derived class *IBM_StudioInfo*

Another interesting derived message class is *IBM_StudioInfo*. This is the type of message sent by InkBoard server to the client application when the client asks for a list of existing sketches under a certain studio portfolio stored in the *InkStudio* database. Besides the inherited member variables, a *DataSet* type object called *dsInfo* is added. This particular object contains all the values returned by the SQL database stored procedure that query the value stored in the *Drawings* and *Studios* table of the *InkStudio* database. The type *System.Data.DataSet*, built in the Microsoft .NET Framework Class Library, represents an in-memory cache of data that is also labeled with attribute *Serializable*. It implements interfaces *ISerializable* and *MarshalByValueComponent*. This again means CLR knows how to serialize *DataSet* object and our code would not have to worry about it.

```
[Serializable]
public class IBM_StudioInfo : InkBoardMessage
{
    public IBM_StudioInfo(string strUserId)
        : base(strUserId) {}

    private const INKBOARD_MESSAGE_TYPE MsgType =
        INKBOARD_MESSAGE_TYPE.IBM_STUDIO_INFO;

    public override INKBOARD_MESSAGE_TYPE MessageType
    {
        get { return MsgType; }
    }

    public DataSet dsInfo;
}
```

InkBoard message helper class

Having established the base class and the derived message classes, a message helper class is put in place to handle the serialization and de-serialization of the message objects. The serialization and de-serialization functions are implemented in two static functions for easy access.

Here with the help of *System.Runtime.Serialization.Formatters.Binary.BinaryFormatter* object, we are able to take an *InkBoardMessage* object and serialize it into a *MemoryStream* object that is ready to be fed to the TCP/IP network channel.

```
public static MemoryStream Serialize(InkBoardMessage ibm)
{
    MemoryStream ms = new MemoryStream();
    BinaryFormatter formatter = new BinaryFormatter();
    formatter.AssemblyFormat = FormatterAssemblyStyle.Simple;
    formatter.TypeFormat = FormatterTypeStyle.TypesWhenNeeded;
    formatter.Serialize(ms, ibm);

    return ms;
}
```

The reverse process of de-serialization happens in similar fashion. We use the *Deserialize()* function of the *BinaryFormatter* object to rebuild the *InkBoardMessage* object out of a *MemoryStream* object, which in turn is built from a byte array.

```
public static InkBoardMessage Deserialize
    (byte [] bytes, int offset, int count)
{
    MemoryStream ms;

    ms = new MemoryStream(bytes, offset, count);

    BinaryFormatter formatter = new BinaryFormatter();
    formatter.AssemblyFormat = FormatterAssemblyStyle.Simple;
    formatter.TypeFormat = FormatterTypeStyle.TypesWhenNeeded;
    InkBoardMessage msg = new IBM_Null("error");
    try
    {
        msg = (InkBoardMessage)formatter.Deserialize(ms);
    }
    catch (Exception e)
    {
        .... // process the exception
    }
    ms.Close();
    return msg;
}
```

Transferring InkBoardMessage

After the *InkBoardMessage* type object is serialized into byte array in the form of *MemoryStream* object, the next step is to send it over the network to the server or client through

TCP/IP socket. A number of interesting issues need to be solved before this seemingly simple task can be accomplished.

Asynchronous client socket

After a network connection is established through TCP/IP socket, a *NetworkStream* object is created attached to the connected *Socket* object. By default, the *NetworkStream* class sends and receives data over *Stream* socket in blocking mode, which means the application suspends while waiting for network operation to complete. This is of course unacceptable for InkBoard client application. To maintain the responsiveness to the user interface, we need to use an asynchronous client socket to send and receive data in non-blocking mode. This way, the application will process the network operation on one thread while the application itself continues to run on its own original thread. *NetworkStream* object provides *BeginWrite()* and *BeginRead()* functions for asynchronous reading and writing. By using these methods, we are able to spin off new threads when network operation is executed. And consequently, a callback function is required to notify the original application thread that the network sending and receiving activities are finished and the data writing is finished, or received data is ready to be further processed.

```
public void SendInkBoardMessage(InkBoardMessage ibm)
{
    MemoryStream ms = InkBoardMessageHelper.Serialize(ibm);
    MemoryStream msLen = InkBoardMessageHelper.Serialize(ms.Length);

    byte[] msg = new byte[msLen.Length + ms.Length];
    Array.Copy(msLen.GetBuffer(), 0, msg, 0, (int)msLen.Length);
    Array.Copy(ms.GetBuffer(), 0, msg, (int)msLen.Length, (int)ms.Length);
    ns.BeginWrite(msg, 0, (int)msg.Length, writeCallback, null);
    ms.Close();
}
-----
public void ProcessClient()
{
    string strRemoteClientIP =
        ((IPEndPoint)clientSocket.RemoteEndPoint).Address.ToString();
    string strRemoteClientPort =
        ((IPEndPoint)clientSocket.RemoteEndPoint).Port.ToString();

    strClientIP = strRemoteClientIP;
    strClientPort = strRemoteClientPort;

    ns.BeginRead(byteBuffer, 0, BufferSize, readCallback, null);
}
```

```

clientInfo.strIP = strClientIP;
clientInfo.strPortNo = strClientPort;
clientInfo.dtStart = DateTime.Now;
}

```

Preserve message boundary

The decision to use TCP/IP over UDP is an easy one to make. UDP doesn't guarantee the delivery of the packets, and TCP/IP does. Unlike an audio or video stream where missing packets won't affect the overall application performance if they are controlled within a tolerable range, sketches missing strokes will create a severely negative user experience. However, unlike UDP, TCP/IP is a connection-oriented protocol in that Windows OS TCP subsystem uses buffers to aggregate packets before sending them out, and thus does not preserve data message boundaries. In other words, the remote device won't necessarily receive the data the same number of message units. The TCP Subsystem will place all of the individual messages of data into the TCP buffer. Depending on the rate at which the application is sending data and the rate at which the receiving device is receiving data, those messages may get pushed together, or separate from each other, in the data stream. Two common solutions are used to solve this invisible message boundary problem:

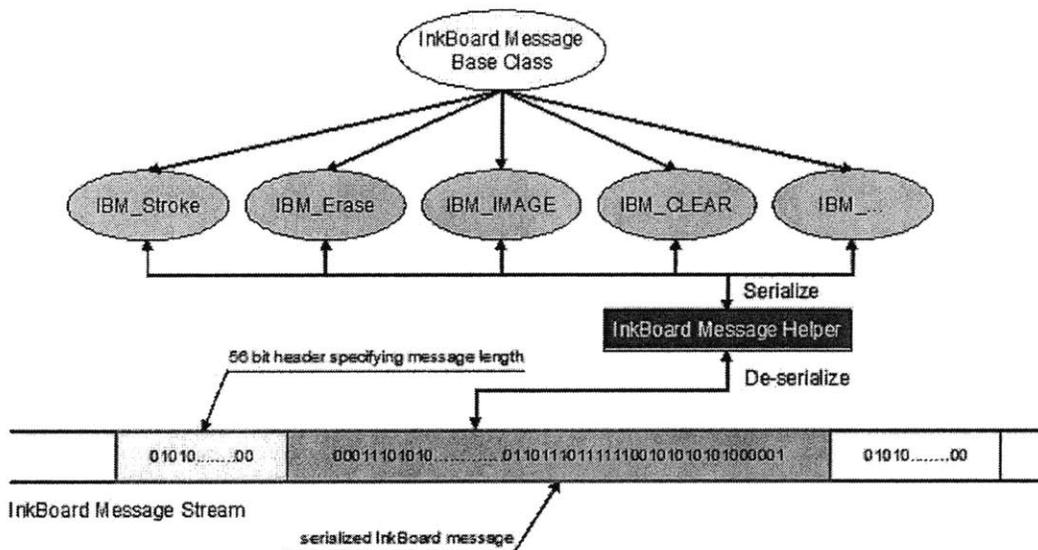


Figure 35 InkBoardMessage protocol

1. Create a protocol that requires a one-for-one response to each data message sent from the host.
2. Design a data message marker system to distinguish data message boundaries within the data stream.

InkBoard message protocol adopts the second strategy, because a send-acknowledgement mechanism is more error-prone in implementation, generates more network traffic, and potentially creates more delays. On the other hand, a message marker is relatively easy to implement. As shown in Figure 35, we prefixed the serialized *InkBoardMessage* object with another serialized *Integer* that indicates the length of the *InkBoardMessage* object. This could create a chicken-and-egg situation because distinguishing the length *Integer* object itself in the message stream becomes the next problem. Fortunately, the *BinaryFormatter* object we mentioned in the *InkBoardMessageHelper* class has an undocumented peculiar property. It always serializes a four-byte *Integer* into a 56-element long byte array. Thus, from the receiving end, we can always chop off the first 56 bytes of received binary array, and de-serialize it into an *Integer* *L*, which tells us the length of the immediately following *InkBoardMessage* object. Then we will continue to receive the binary data and store them in a buffer array, and keep a incremental counter until the counter reaches the number *L*, at which point we know for sure that this is the last byte of the serialized current message object, and the next byte is the start of the next *Integer* object indicating the length of the next *InkBoardMessage* object. We will then de-serialize the byte array in the current buffer, restore the *InkBoardMessage*, clear the content of the buffer array and wait for the next object.

A comparative study of IMP vs. ReMarkable Texts

InkBoard has gone through thorough test within a small research group. IMP (InkBoard Messaging Protocol) proves to be a viable solution for the network exchange of *Ink* objects since the application receives favorable feedbacks from the testing team. However, a viable solution is not the same as a good solution. A feasible protocol is not the same as an effective protocol. We want to compare IMP with other similar protocols dealing with the same problem. And the metric we chose is the messaging mechanism adopted in ReMarkable Texts.

ReMarkable Texts

As we have mentioned in Chapter 6, ReMarkable Texts is a research project carried out by Brown University in collaboration with Microsoft Research Conference XP team. It focuses on PowerPoint slide sharing across networked clients. However, it is its implementation of ink annotation sharing capabilities that this study is most interested in.

RTP

Being a spun-off research effort of Conference XP project, ReMarkable Texts leverages the messaging mechanism devised by Conference XP. The targeted application for Conference XP is real time video and audio conference through high-speed broad-band network that supports multicast. The protocol it adopts is Real-time Transport Protocol, or RTP. RTP is the key peer-to-peer standard for audio/video transport in IP networks, designed for scenarios where preventing latency is more important than guaranteeing delivery. The services it provides include timing recovery, loss detection and correction, payload and source identification, reception quality feedback, media synchronization, and membership management.⁵² It was designed for use in multicast conferences.

The RTP Stack is implemented using C# managed code with .NET object model and DirectShow filters by the Conference XP project. It provides a library called *MSR.LST.Net.RTP v1.0*, also known as *C# RTP v1.0*, that implements the RTP protocol section of the IETF Internet Draft for RTP for IP Multicast. The detailed specification can be found on the Conference XP web site⁵³.

RTDocs and RTInk

Implemented over RTP by the Learning Experience Project, of which Conference XP is the initiative, is a set of classes that describes formats and protocols collectively known as *RTDocs*. It includes the following classes.

RTDocument: Describes the format and protocol for describing and broadcasting, LXP documents, for example: Presenter slides and Student Experience documents.

RTInk: Describes the format and protocol for describing and broadcasting ink strokes used to annotate LXP documents.

RTNavigation: Describes the format and protocol for describing and broadcasting navigation events, for example: slide transitions, page turns.

RTStorage: Describes the schema for persisting RTDocs information.

The one that we need to pay particular attention to is *RTInk* type. Here is the description of the *RTInk*, or Digital Ink as it is called in the *RTDocs* specification⁵⁴.

1. the ink stroke itself
2. some transform/origin for where to position the stroke (in case it's been moved)
3. color (rgba)
4. width
5. user assigned labels (variable numbers of them; presumably done through another table that has as many entries per stroke guid as there are labels assigned to it)
6. layer (guid)
7. page number of the document
8. timestamp
9. deletion timestamp or 0 if not deleted
10. author id (guid)
11. hyperlink id (guid) or 0 if not a hyperlink
12. a pointer (presumably a guid) to the previous annotation entry if this entry is a modification of an earlier one (or 0 if it is not). Presumably this implies that all ink annotations have guides.

Comparing this specification with the *IBM_Stroke* type derived from *InkBoardMessage* class, they are almost identical except for some minor difference in member variables.

Multicast network

IP multicast enables a sender to transmit data to many receivers at the same time. It has the useful property that the network creates copies of the packet as needed, such that only one copy of the

packet traverses each link. IP multicast can be used for extremely efficient group communication, provided it is supported by the network, making the cost of sending data to a group of receivers independent of the size of the group.

Unlike unicast where sender has to send the data to each intended client explicitly, in multicast sender simply sends the data to a pre-defined fake IP. Any client who wants to receive the data will subscribe to this IP address through network router. And the router is intelligent enough to make copies of the data and send it to the subscribers.

This is why a lot of video/audio conferencing systems, including Conference XP, leverage this protocol in favor of unicast. In other words, *RTInk* objects in ReMarkable Texts applications are sent around using multicast mechanism. This would not be a problem for the Conference XP intended users who are explicitly required to either have “High-speed (100baseT or better) connection that supports multicast, such as local area network (LAN), or Internet2 network.”⁵⁵

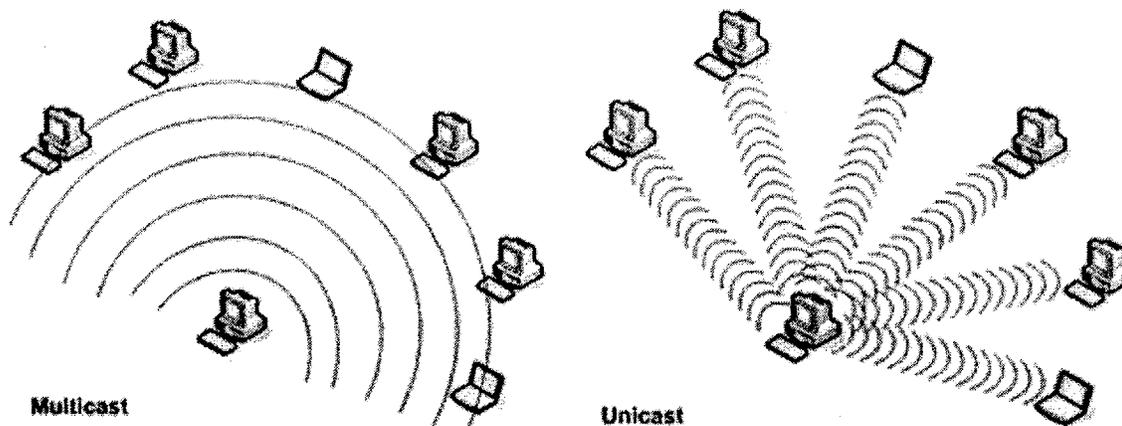


Figure 36 Multicast vs. unicast

However, multicast is an optional and relatively new feature of IP networks. It is usually only found in large research and educational institutions and in the network backbone, but it is uncommon in many commercial settings and service providers. Internet2 is even less popular in

ordinary network environment. This means that applications built with Conference XP API, including ReMarkable Texts, will most like not run between two different LAN.

IMP Assessment Tool

After the qualitative analysis of the similarity and difference of *InkBoardMessage* and *RTInk*, it is necessary to carry out a quantitative study to measure the performance of these two different implementations. IMP (InkBoard Messaging Protocol) Assessment Tool is built for this purpose.

Software architecture

The goal of IMP Assessment Tool is to measure the performance difference by comparing the latency of receiving the ink-loaded messages of these two different applications. It is built so that it can send both *InkBoardMessage* objects to InkBoard Server to be distributed to InkBoard Client, and *RTDocument* to ReMarkable Texts at a preset time interval. Both applications will then record the time they receive the sent messages. After gathering the sending time and the receiving time, we can calculate the difference to compare the performance of these two ink-enabled messaging mechanisms.

In the following diagram, it is clear the in the ReMarkable Texts route, the *RTDocument* message goes to the client with multicast protocol, where as the in InkBoard route, the *InkBoardMessage* goes to InkBoard Server and it is in turn sent to the client.

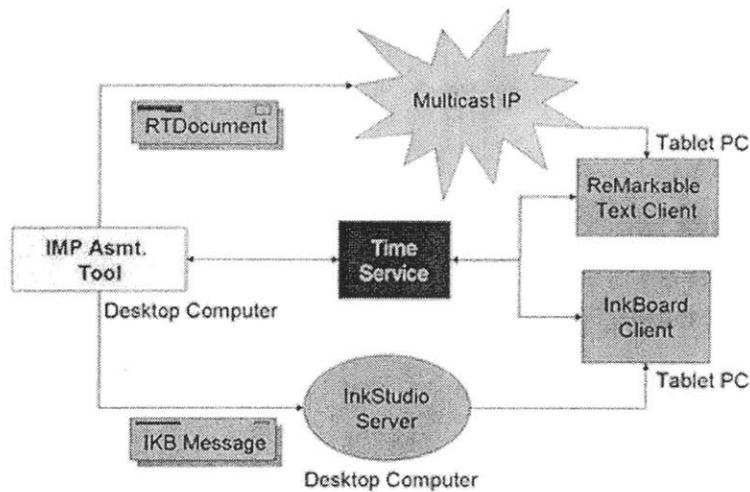


Figure 37 InkBoard Assessment Tool architecture

The synchronization of time

One of the challenging problem IMP encounters is the synchronization of time stamp on the messages. As the IMP architecture diagram demonstrates, the sender (IMP application) sends out the message and naturally attaches a timestamp generated by its own clock. When the Tablet PC (the receiver) running either ReMarkable Texts or InkBoard receives the message, another timestamp, generated by the receiver’s clock, is attached to it. However it is not possible to accurately calculate the time span between these two timestamps unless the computer clocks of both the sender and the receiver have been synchronized to the millisecond level.

After trying several commercial “Time Synchronizer”, we find that it is simply not feasible to synchronize computer clocks to millisecond. The structure of PC hardware design determined that the precision of its timing circuitry is worse than a \$10 watch. So a work-around that focuses on relative time instead of absolute time is devised to solve this problem.

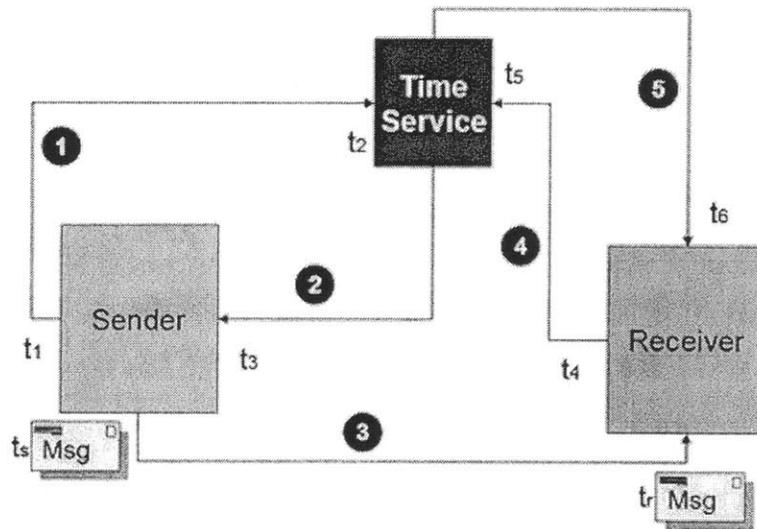


Figure 38 Synchronization of time

In the Figure 38, a web service called Time Service is introduced into the picture. The Time Server has only one web method – *GetTime()*, which returns a timestamp from its own local clock whenever queried by a consumer. Following are the steps to calculate the latency between sender and receiver.

1. Sender records a timestamp t_1 from sender's clock.
2. Sender makes a web service call to the Time Service.
3. Time Service returns a timestamp t_2 that is registered from Time Service's clock.
4. Sender receives the timestamp from the Time Service and again records a timestamp called t_3 from sender's clock.
5. Now sender is ready to send the message. Assuming the network traffic is symmetric for the sender when querying and receiving the timestamp from the Time Service, we can attach a timestamp t_{sent} to the message where:
$$t_{sent} = t_2 - (t_3 - t_1) / 2.$$
6. Sender sends the message to the receiver.
7. Receiver receives the message, and records a timestamp t_4 from receiver's clock.
8. Receiver makes a web service call to the Time Service.
9. Time Service returns a timestamp t_5 that is registered from Time Service's clock.

10. Receiver receives the timestamp from the Time Service and again records a timestamp called t_6 from receiver's clock.
11. Now receiver finishes receiving the message. Again, assuming the network traffic is symmetric for the receiver when querying and receiving the timestamp from the Time Service, we can attach a timestamp $t_{received}$ to the message where: $t_{received} = t_5 - (t_6 - t_4) / 2$.
12. Finally we can calculate the latency: $T_{latency} = t_{received} - t_{sent}$

The trick here is that we assume it takes the same amount of time for the incoming SOAP message for querying the Time Service to travel through the network as the outgoing SOAP message containing the timestamp. Thus we are able to obtain the time when the sender sends the message and the time when the receiver receives message both based on the clock of the machine running the Time Service.

Collected data

Using the above setup, we are able to obtain the following latency data.

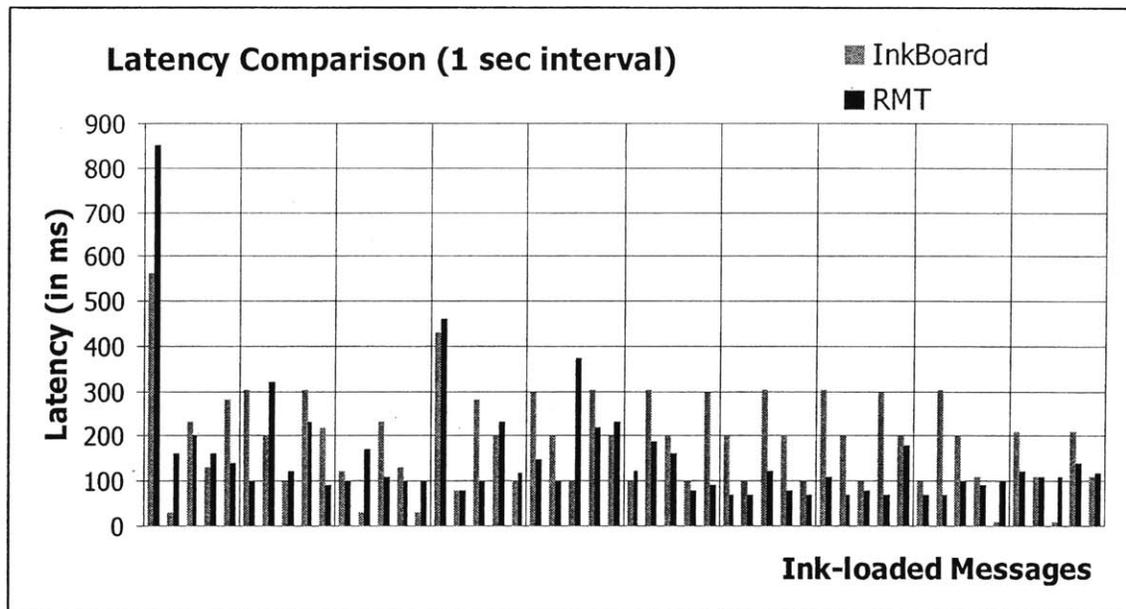


Figure 39 Latency comparison with 1 second interval

In the Figure 39, IMP Assessment Tool sends 50 messages, at the interval of 1 second apart, that include a randomly generated 200-point ink stroke object to InkBoard and ReMarkable Texts. The bar chart represents number of milliseconds from the time the messages are sent to the time the messages are received. The initial spikes are caused by the first time calling the Time Service on the remote web server. It usually takes some time for the CLR to load the web service component into memory when it is called the first time. After that, it resides in the physical memory of the server and executes very fast. The graph suggests slight smaller latencies for ReMarkable Texts. And we can also observe a 3-second pattern developed in the case of InkBoard where as the ReMarkable Texts latencies curve over time seem fairly random and smooth without noticeable patterns.

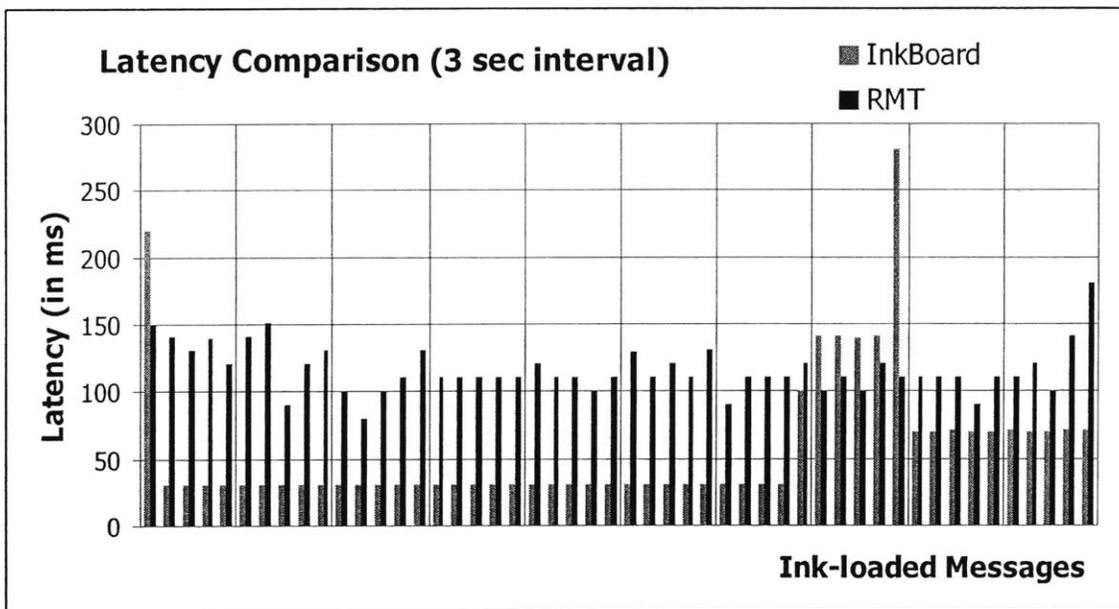


Figure 40 Latency comparison with 3 seconds interval

Figure 40 shows the result of the same experiment, the only difference being that the messages are sent at the interval of 3 seconds apart. In this case InkBoard performs substantially better than ReMarkable Texts with nearly one third of the latency time, and it demonstrates surprising uniformity. Besides the initial spikes, the first 40 messages or so all take about 35 milliseconds to deliver; then the rest of them take a hike jumping to the 140 milliseconds range before they settle

at about 70 milliseconds. This is rather peculiar but coincides with the interesting 3-second pattern we observed with the 1-second interval experiment. The only plausible explanation is that Microsoft.NET CLR implements the TCP/IP buffer stacks for the asynchronous socket calls such that they get flushed to the network pipe at a time interval that somehow “resonates” with the IMP Assessment Tool’s 3 second interval. The RTP implementation used by ReMarkable Texts, which is not part of the released CLR library, takes a different approach and thus does not demonstrate such peculiarity.

One more experiment using 5-second interval confirms this hypothesis (Figure 41). The messages seem to be stacked together before get sent out at a 15-second interval in the case of InkBoard.

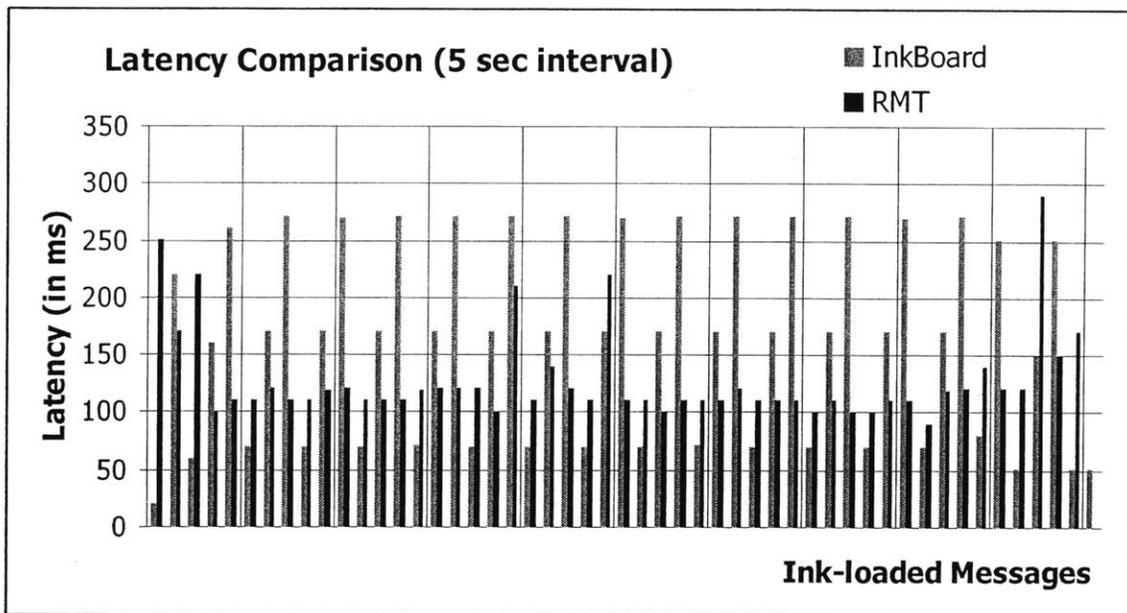


Figure 41 Latency comparison with 5 seconds interval

Figure 42 shows an overall latency comparison between InkBoard and ReMarkable Texts in different time interval situations.

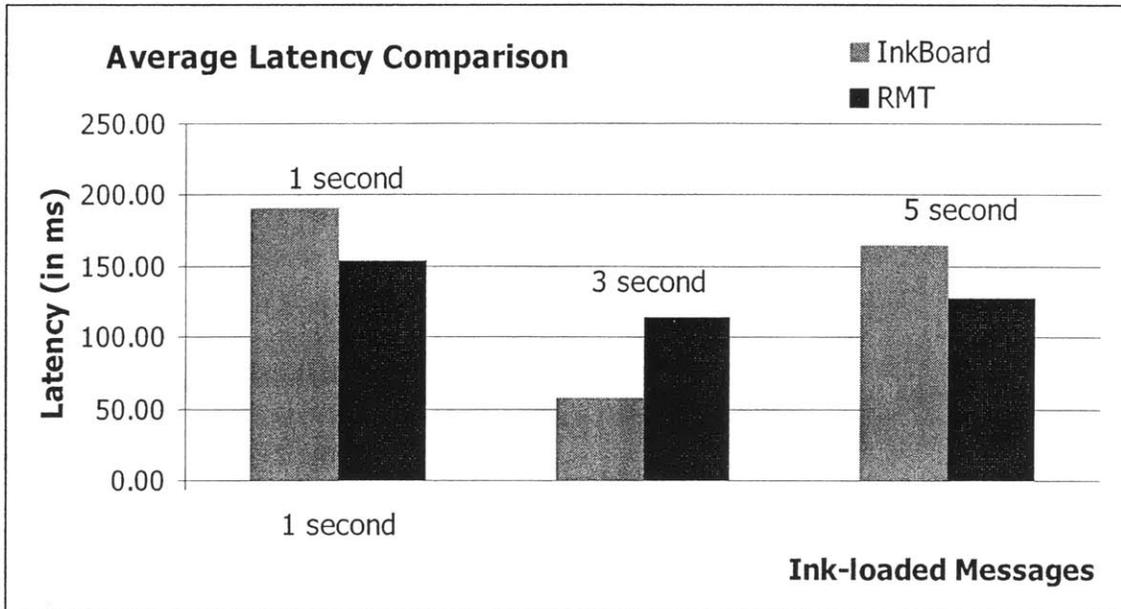


Figure 42 Average latency comparison

Analysis of the IMP in comparison with RTDocument + RTP

Advantages

The advantage IMP has over RTDocument lies in its substantially less restrictive requirements for the network environment. So long as the server and the clients are on TCP/IP network, InkBoard works well. This makes real time graphical communication available not only within the range of the campus network, but also possible across wide-area network (WAN). Designers can “talk” to each other with their sketches from all over the world. On the other hand, RTDocument plus RTP approach is designed for high-speed broad-band network that supports relatively new multicast feature. This is ideal for video/audio conference setup. But for network sketching applications that typically do not require such high connection speed, it seems to be an unnecessary luxury and an obstacle that restricts its potential deployment outside a local area network (LAN) or relatively hard to find Internet2 network.

As far as the message latency goes, both IMP and RTDocument on RTP perform well in a comparable range around 100 to 200 millisecond in a LAN setup. Because of the implementation of asynchronous socket calls in Microsoft.NET CLR, IMP appears to be at its most efficient level when messages are sent at a regular of 3 seconds apart or its multiples.

The other advantageous feature of IMP is the central server architecture that ensures the operation of IMP message flow. It makes the application much easier to manage both in terms of deployment and support. The client application is relatively light with fewer dependencies than ReMarkable Texts requires. Above all, the central server has a single copy of the shared sketch safely stored in database, and readily for retrieval and replay.

Disadvantages

The disadvantages that IMP has in comparison with RTDocument on RTP, unsurprisingly, is also related to its central server architecture. As we have discussed before, multicast network makes network traffic less congested when large number of peers join in the collaboration. The responsibility of scalability issue, a software tasks in the case of IMP, is instead assumed by the hardware of network routers.

The peer-to-peer architecture also avoids single point of failure and increases the reliability of the system. However, it comes with a price. Without a central server, the client is responsible for storing the sketch in its own storage. This increases the complexity of deployment since ReMarkable Texts client requires user to install a copy of Microsoft SQL Desktop Engine (MSDE), a stripped-down version of the SQL database server. The reconciliation of different copies of data stored in different client machine can also create headaches.

Leveraging the multicast feature of the network, the RTDocument message latency is more evenly distributed and slightly lower in average than IMP in the comparison study against InkBoard. But then again, this slightly higher performance may not justify the network environment restriction it requires to run.

In summary, we believe that InkBoard and ReMarkable Texts have similar performances in terms of ink-loaded message network throughput. However, the lower barrier of network environment

requirement makes InkBoard a more practical and easily-deployed solution for the Tablet PC based sketch application.

Chapter 9 Integrating Conference XP with InkBoard

In this last chapter before the conclusion, we discuss the strategy and implementation in integrating Conference XP components into InkBoard application for audio/video conferencing capabilities.

Conference XP & InkBoard

Microsoft Research's Learning Sciences and Technology group started Conference XP as an initiative of the Learning Experience Project, aiming to bring high-quality audio/video conferencing and messaging capability to multicast-enabled broad-band network. InkBoard, being a graphical communication tool through networked sketches, has real incentive to integrate Conference XP technology to provide A/V conference functions on top of the ink sharing interactions among designers. When peers can see each other's facial expression, hear each other's voice while studying each other's freehand sketch on a Tablet PC, this increases the effectiveness of the collaboration immensely. Of course we also have to bear in mind that by using Conference XP technology, we limit the A/V conference functions to users that are on multicast-enabled networks because of the RTP mechanism it adopts. Generally it will not work across different LAN, although the ink sharing part will still continue to function because of the InkBoard Messaging Protocol adopts the unicast approach as we have explained in the last chapter.

Technology overview of Conference XP

Conference XP is an open source research project of Microsoft Research, and there have been a few documentations available on the Conference XP project web site. The objective is that not only people can use the downloadable package directly without modification, but they can also try to integrate the necessary components into their own application to make it conference-enabled without too much effort. In that sense, Conference XP was designed with extended

development possibility in mind. The components are well-structured and provide many hooks – APIs – for users to leverage its functionalities, which makes our effort in integrating with InkBoard relatively less difficult.

Conference XP can be thought of as three layers bundled together – Conference layer, DirectShow layer and the RTP (Real-time transport protocol) layer.

Conference layer

This layer communicates with the interfaces exposed by the DirectShow layer to retrieve multimedia data, transmits compressed multimedia data through the network using the RTP layer and receives compressed multimedia data from the network through the RTP layer and decompresses it using the DirectShow APIs.

Managed DirectShow layer

As an interfacing layer between managed C# code and unmanaged VC++ code, the DirectShow layer is responsible for retrieving audio/video data from the attached multimedia devices by communicating with the unmanaged APIs of DirectShow. DirectShow is an application programming interface for client-side playback, transformation, and capture of a wide variety of data formats. It uses codecs to compress and decompress multimedia data before transmitting and after receiving respectively.

RTP layer

This is the layer that uses the real-time transport protocol to send and receive multimedia data through the network through multicasting. (Although Conference XP claims that this RTP layer supports unicast as well, they have yet to release an A/V application other than the Conference XP for Windows Messenger that has close ties with Windows Messenger APIs and making it even more restrictive by not only being dependent on hardware configurations but also on other software installations.)

Integration with InkBoard

In the following sections, we discuss the integration of Conference XP components with InkBoard application, both in terms of software architecture, and UI design considerations.

ActiveX controls

To make Conference XP components an isolated module so that it can be easily plugged in to any program that needs A/V conference support, we wrapped the video components and audio components together with their user interface into two Windows ActiveX control objects. The lower-level RTP APIs were retained with minimal changes made to the Managed DirectShow Layer. Using the Microsoft Visual Studio.NET integrated development environment (IDE), these two Windows ActiveX controls can be readily dragged from the control library, and dropped on to a standard Windows form object. The integration code is then automatically generated by the IDE and all the necessary API calls are exposed as public methods of the objects, making it very easy to wire them with the user interface event handling mechanisms of the host application, in this case, InkBoard.

Following is a list of public methods exposed by the A/V Windows controls.

- `SendAudio (IPEndPoint destIP)` – send audio data packets to a multicast IP.
- `SendVideo (IPEndPoint destIP)` -- send video data packets to a multicast IP
- `LaunchPlayer (IPEndPoint sourceIP)` – listen to a multicast IP and launch video/audio player to play incoming streams
- `StopDevice ()` – stop sending audio/video stream
- `StopPlayer ()` – stop the video/audio player
- `SetVolume(int iVol)` – adjust the audio volume
- Additional functions were exposed to control the display

UI consideration

The video conference control that can contain up to four small video windows is placed directly under the participant list box, making it easier for the local user to recognize remote users' faces and sketch strokes. Each video window has its user name shown as well to make easy visual connections. The audio control is placed below the video control. There is no limit to the number of users that can talk simultaneously as far as the UI is concerned. It is only restricted by the network bandwidth and the Tablet PC's hardware configuration. A slider bar is provided for adjusting the audio volume using the digital pen.

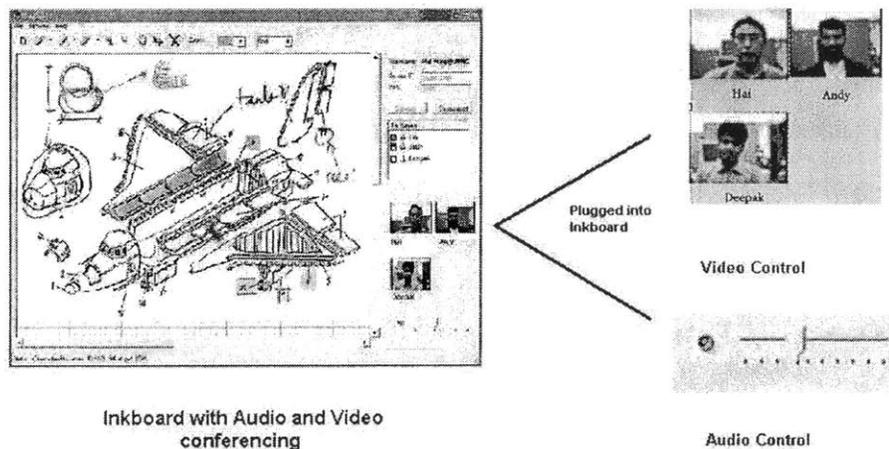


Figure 43 Conference XP components incorporated in InkBoard

When user connects to the InkBoard Server, they can choose to have A/C conference enabled by selecting two checkboxes. The InkBoard client will spin off two new threads at that moment. One goes out to make the socket connection with InkBoard Server; the other starts the services to subscribe to a multicast IP, send out and receive the A/V packets.

Future implementations

The integration of InkBoard and Conference XP is a rather ad-hoc solution. There are many more options that can be pursued. Here are a few of them.

Provide support for unicasting

Conference XP promises that unicast support is going to be built in their future releases. This could be a major improvement for InkBoard. With the current mixed model, InkBoard users can only enjoy the A/C conference functions if they are on a multicast-enabled network. If A/V conference can be supported on unicast, InkBoard can truly become a widely used application.

Record conference sessions for future playback

The entire interactive sketch session is recorded in the central InkBoard Server InkStudio database, and can be played back sequentially in the same order as it is constructed. The similar experience is also expected from the A/C conference. Building a storage-retrieval system that captures the video and audio of each peer and replay it back along side with her sketch will make a terrific tool for documenting design sessions.

Chapter 10 Conclusion

In this chapter, we review the pedagogical and technological contribution of this research effort in the context of design-oriented learning. Future directions of the research are also discussed in detail.

Summary

As information technology gets more and more integrated with education, software platforms that support in-class as well as outside-of-class activity have become more and more popular practice. Many traditional teaching methodologies are being transformed into electronic format to facilitate both the teacher and the learner.

However, as teaching styles and techniques vary from subjects and goals, we are witnessing the opposite in their electronic counterparts, the majority of which are primarily content storage and retrieval systems.

This thesis explores the unique characteristics of one type of teaching – design-oriented pedagogy, and proposes requirements for the e-education platforms that target at supporting this type of learning activity. Two exemplary systems are built to demonstrate the application of these principles and their respective impacts are assessed. Service-oriented software architecture and a unique network protocol designed for transmission of Tablet PC ink data objects are both presented too. The user feedback and the impact of these two systems demonstrates that following the pedagogical principles, e-education platforms can be designed and implemented to support learning activities that are much more interactive than traditional lecture-based teaching.

Contributions

The contribution of this research work is two-folded, including both a pedagogical and a technological side. Primarily it explores the e-education platform characteristics required for

supporting design-oriented learning activities, and demonstrates two exemplary systems that effectively fulfill those requirements.

Pedagogical contribution

Generic e-education platforms serving mainly as content storage and retrieval systems have been extensively studied and implemented by many academic institutions as well as commercial corporations. However, there was little work done in the area of building e-education platforms specifically targeted at particular learning activities – in this case design-oriented learning – that are very different in pedagogical practices than traditional lecture-based education.

It is true that the generic e-learning platforms have broader applicability, and can serve to reduce the cost of developing and deploying educational courseware and supporting systems. However, design-oriented learning activities, among other special type of learning, can not be taught effectively using these kinds of systems.

To build an e-education platform suitable for the design education needs, the system has to meet the following requirements.

1. Stimulate creative problem solving.
2. Encourage independent thinking.
3. Build team collaboration, and a sense of community.
4. Facilitate design communication.
5. Provide tools for project management.

PREP

The experimental PREP system is built based on the design-idea-generation method of the Rohrbach process which has been practiced for many years in mechanical engineering design education at MIT. Leveraging the extensive experience of the teaching staff and the advanced technology, PREP is a great success. It provides a private space for design teams to share design documents and make comments, sketches. The use of Tablet PC also great enhances the usability of the system, making the online version of the Rohrbach process completely paper-less.

PREP was adopted in a number of projects/courses of various scales, including the International Robot Design Competition with students from all over the world, Robot Design courses locally at MIT, as well as collaborative medical devices design projects between MIT and Massachusetts General Hospital. The results and feedbacks have been overwhelmingly positive.

The PREP effort was also recognized in the Microsoft Faculty Summit 2003 held in Microsoft headquarter in Seattle, and demonstrated as one of the major components of the iCampus RobotWorld project.

Being part of the design process, the virtual Rohrbach system could potentially have greater impact than in the design-teaching classrooms. Industry product designers, architects, or any other design-oriented teams can leverage the power of tablet PCs as well as the PREP software. It is an effective tool for asynchronous collaboration between geographically dispersed design teams or teams that operates on different time schedules.

Microsoft is very impressed by our effective usage of the tablet PCs through the PREP application that they build a case study for their promotion at the tablet PC official launch in 2002. Being the first adopter of tablet PC in the education field, we have gained value experience that can be applied to other educational institute who wants to adopt tablet PCs as an effective teaching and learning tool.

InkBoard

InkBoard, is the Tablet PC software tool built for addressing the synchronous interactive needs between design team members, also tries to fulfill the requirements of the design-oriented pedagogy. It is the very first sketch-sharing application built on tablet PC, leveraging the powerful Microsoft Tablet PC Platform SDK.

InkBoard allows members of multiple design teams to participate in real time online collaborative sketch sessions. It has a central server backed-up by database to record all the ink strokes and other data at the same time the design session is being carried on. The recorded session can later be retrieved and played back on client's machine exactly the same way it was created. There are many user interface considerations that make InkBoard very friendly with

digital pen and writable screen. Real time video/audio conferencing capability is also added on to InkBoard as an integral piece to further enhance the interactive design experience.

InkBoard is recognized to be a very useful tool in teaching design courses, and was also showcased at the Microsoft Faculty Summit 2003, along with other major research efforts of Learning Experience Project conducted by Microsoft Research, University of Washington and Brown University. InkBoard is featured as one of the “Partner Downloads” from the Conference XP project website.

Hoping to broaden the application of InkBoard in real world teaching practices, we have made InkBoard freely available from the Internet, along with part of the source code hoping to promote collaboration and to make improvements with the collaborative effort from the academic community. So far, we have got very positive responses, with openly expressed interest for collaboration from various parties such as the Computer Science Department of Rice University, and the Computer Science Department of University of Indiana.

Technological contribution

In building these new tools for facilitating design education, we also explored a number of challenging technical issues that constitutes part of the contribution of this research.

Service-oriented software architecture

The design of the PREP application takes a new and unique approach – service-oriented architecture. Leveraging XML-based SOAP messaging protocol, PREP web site is able to pull in data from web services backed up by distributed databases into one central location, organize the information flow, and present the information with consistent look-and-feel to the users. The loose coupling structure makes it possible to share and reuse the functionalities of existing systems.

Service-oriented architecture is an architecture style whose goal is to achieve loose coupling among interacting software agents. Adopting the service-oriented architecture, PREP truly achieved componentized web application structure, which is a big step forward comparing to

traditional three-tier web application structure. With the help of standardized WSDL documents and SOAP messages, we are able to cleanly separate data, service and operation such that each service component can exist independently, be used by different application, and can be replaced without ever changing the consumer's code.

The flexible, movable web form user control module is another unique contribution on the web application user interface design. Without going back to the code on the web server, we can customize the user interface layout right using the manageable and secure web interface. It also makes it easy to extend and plug in new modules effortlessly.

InkBoard messaging protocol

The other important technological contribution of this research work is the InkBoard application itself and the InkBoard messaging protocol designed for real time ink data sharing across TCP/IP networks. InkBoard is the very first Tablet PC application aimed at enabling ink-sharing capability in a heterogeneous network environment for design collaboration. Although a lot of ink data collection and management functions are provided by the Tablet PC Platform SDK libraries, there is no networking function readily available for streaming ink data.

The InkBoard messaging protocol takes an object-oriented approach in building the InkBoard message base class and derived classes containing different types of actions regarding to ink stroke objects. Asynchronous sending and receiving mechanisms, as well as multi-threading consideration are all incorporated in components handling the network traffic.

Upon quantitative comparison study of InkBoard messaging protocol against RTDocument + RTP mechanism adopted by ReMarkable Texts, a joint research effort between Microsoft Research and Brown University, InkBoard messaging protocol proves to have the same level of performance, and in some situations even better than the opponent. In addition, the ability of transmitting ink-loaded message over regular TCP/IP network makes InkBoard a more likely choice when the restrictive requirements of ReMarkable Texts on multicast-enabled high-speed network are not easily met.

Many user interface design considerations are also given to the InkBoard client application, such that it makes the user experience much friendlier with digital pen and writable screen equipped on Tablet PC. Incorporating Conference XP components for supporting audio/video conferencing capability also makes InkBoard a very useful tool for real time collaborative design sessions.

Future Research

There are many exciting directions for future research in the area of building e-education platforms to support design-oriented learning.

Design advisors

A very challenging approach is the concept of intelligent design-advisor. This is especially useful for extending the PREP system. Imagine after students submit their design ideas to PREP, the system can identify the progress each individual student makes, and makes intelligent recommendations based their performance. And if some one is not doing so well early in the design process, the system will be able to raise a red flag and perhaps remind the course instructor that a particular student needs special attention.

To achieve this goal, we of course can take the simple form of designing a pre-defined set of criteria and having students fill out survey forms to assess the situation. But we can also incorporate complicated artificial intelligence functionalities into PREP system to analyze student's online activities as well as the document they submit to determine the customized process of learning procedures.

Shape recognition

InkBoard provides freehand collaborative sketch capabilities across the network, which as we have seen in Chapter 6, primarily helps in the preliminary stages of conceptual design. After the designers are ready to commit with the ideas however, they would have to go back to their CAD tools to redraw them. The missing link here is the intelligent shape recognition and symbol

interpretation capabilities that can enable direct transformation from freehand sketch to CAD drawings.

There have been many research efforts in this area, but with multiple peers involved, the decision process became an order of magnitude more difficult. Not only do we need to decide the shape based on one person's sketch, we might need to take in consideration other designer's annotations too. Assessing a single designer's intention is hard enough; imagine assessing collective intention of a group. And if there are conflicts among them, a resolution mechanism has to be put in place.

To efficiently help the computer translate coarsely drawn lines into meaningful shapes, a set of rules have to be defined for the user to guide the way they draw these lines. On the other hand, the rules can not be too complicated. Otherwise it would make the learning curve so steep that people simply would stop using them. Finding the right balance is the key here for a practical and reliable shape recognition algorithm.

Unicast conferencing

As we have seen in Chapter 9, the audio/video conferencing component from Conference XP doesn't support non-multicast networks. This is a conflict with what InkBoard has promised to be, i.e. a collaborative system suitable for IPC/IP networks without multicast restrictions.

Conference XP is evolving and the unicast solutions will be rolled out soon in the near future. We hope to take the advantage of the new Conference XP modules and make InkBoard truly free of multicast restrictions.

The other approach we can take is try to separate the network management layer where RTP is currently employed from the streaming data layer that carries the audio/video packets, and replace it with a customized network layer that supports unicast. This of course would be a much harder route if Conference XP does not deliver what it promises.

Case studies

PREP has been thoroughly tested and used in quite a few major collaboration projects as we have mentioned in earlier chapters. However, due to some logistic problems, InkBoard haven't been used extensively in a real world design class, although it has been tested internally and demonstrated publicly. We plan to take advantage of the MIT 2.007 Introduction to Design class in 2005 to make a carefully designed case study. A group of students will have the opportunity of using tablet PCs and InkBoard software, while the other groups will be using the traditional medium. With surveys, assessment tools, exams, plus data collected by InkStudio database, we will be in a better position to judge the successfulness of InkBoard, not only from a technological perspective, but also from a users' perspective.

Deployment and integration

Integrating with our web services strategy is another important step we can take. Specifically, we would like to incorporate the Generic Registration web services to manage the InkBoard users, design teams as well as documents. The other interesting possibility is to offer the drawing document management process as a series of web service. That way, we could serve client application written in any other languages and platform as long as they talk to the server using pre-defined WSDL contracts.

We want to explore the deployment more strategy of InkBoard at the same time. Right now InkBoard 1.0 is freely available at <http://ken.mit.edu/portalfactory> for download. It requires user to install both the server software and the client desktop application. But it might make sense for us to launch a powerful InkBoard server to serve all the clients since the hardware and software requirement for the server can be high for ordinary users. That way, users only need to install the client desktop application and simply connect to our server to start InkBoard collaboration.

We also would like to explore the possibility of integrating InkBoard with the PREP software. One possible scenario is to use standalone InkBoard software to make the sketches when creating the initial ideas for circulation for PREP. The current application Windows Journal and OneNote doesn't record the sequence of the strokes and hence cannot produce documents that can be

played back. Starting the PREP process with InkBoard can produce a replayable document that can later be seamlessly integrated with an InkBoard collaboration session for group brainstorming.

Appendices

Acknowledgement

This work cannot be finished without the help and support from many people and organizations. I would like to express my sincere gratitude to all those who made this accomplishment possible.

Professor John R. Williams, my thesis advisor, for his support throughout my study at MIT for six years, for his confidence in granting me the opportunity to pursue this research, for his encouragement and guidance.

Professor Alex H. Slocum, the “picky customer”, for his invaluable feedback which provides constant drive for improvement of the thesis. Professor Jerome Connor and Professor Kevin Amaratunga, the rest of my thesis committee members, for their constructive criticisms and enlightening suggestions.

Dave Mitchell and Paul Oka, former and current program manager for Microsoft’s iCampus program at MIT, for their introduction of advanced technologies and researches from Microsoft and many occasions of thought-provoking discussions.

Center of Innovative Product Development (CIPD), Civil and Environmental Engineering Department, System Design and Management (SDM), Microsoft Corporation, for their generous financial contribution during my PH.D. study at MIT.

Abel Sanchez, Deepak Ravichandran, Hariharan Lakshmanan, Anamika Agarwal, Anand Rajagopal, Sakda Chaiworawitkul, Ching-Huie Tsou, Dan Robey, Steve Xiaohan Lin, and many other friends and colleagues from Intelligent Engineering Systems Lab, for their kind help and friendship that makes this journey enjoyable.

My parents Yuanzhong Ning and Zhanglin Zeng, for giving me an inquisitive mind, for their love and support from the other side of the earth.

And last but certainly not least, Bing Wang, my girlfriend who gave me unconditional love and support for the past ten years, through the most difficult times, and shared the happiest moments.

Selected Bibliography

- Alexander, Christopher. (1963). "The Determination of Components for an Indian Village". In Conference on Design Methods. New York: The MacMillan Company.
- Archer, B (1973). The Need for Design Education. Royal College of Art.
- Archer, L. Bruce. (1965). Systematic Method for Designers. London: Council of Industrial Design.
- Asimow, M. (1962). Introduction to Design. New York: Prentice-Hall.
- Atkinson R. L., Atkinson R. C., Smith E. E., and Bem D. J. (1993). Introduction to Psychology (11th edition). Fort Worth TX: Harcourt Brace Jovanovich
- Bazjanac, Vladimir. (1974). "Architectural Design Theory: Models of the Design Process." In William R. Spillers, ed., Basic Questions of Design Theory, pp. 8-16. New York: North-Holland.
- Beavers, J. and Moffatt, C. (2002). RTP Specification. Found at Conference XP project website as of 1/22/2004.
http://www.conferencexp.net/community/uploads/RTP%20Specification.htm#_Toc8666796
- Beavers, J., Moffatt, C. and Anderson, R.. (2002). Found at Conference XP project website as of 1/22/2004
<http://www.conferencexp.net/community/uploads/RTDocs%20Specification.doc>
- BlackBoard Inc. New Features and Functionality in the Teaching and Learning Environment. BlackBoard Learning System Data Sheet.
<http://www.blackboard.com/docs/datasheets/newfeaturesLS.pdf> (10/6/2003).
- Booker, P. J. (1963). A History of Engineering Drawing. Pub Cahtto and Windus, Lodon.
- Booker, P. J. (1964). Written Contribution appended to Conference on the Teaching of Engineering Design. London: Institution of Engineering Designers.
- Bruner, J. (1966). Towards a Theory of Instruction. Cambridge, MA: Harvard University Press.
- Churchman, C. West. (1967). "Wicked Problems", Management Science, 4, no. 14, pp. B-141, and B-142.
- Conference XP Getting Started Guide. Found at Conference XP website as of 1/22/2004.
<http://www.conferencexp.com/community/Default.aspx?tabindex=8&tabid=73#SystemRequirements>
- Dede, C. (2000). "Emerging Influences of Information Technology on School Curriculum". Journal of Curriculum Studies. Vol.32, No.2.
- Farr, Michael. (1966). Design Management. London: Hutchinson.

- Felder, Richard M. & Silverman, Linda K. (1988) "Learning and Teaching Styles in Engineering Education". *Engineering Education*, 78(7), 674-681 (1988)
- Felder, Richard M. (1993). *Reaching the Second Tier: Learning and Teaching Styles in College Science Education*. *J. College Science Teaching*, 23(5), 286-290 (1993)
- Felder, Richard M. (1996). *Matters of Style*. *ASSE Prism*, 6(4), 18-23 (December 1996).
- Fielden, G. B. R. (1963). ('The Fielden Report') *Engineering Design*. London: Her Majesty's Stationery Office.
- Finke, R., Ward, T., Smith, S. (1992). *Creative Cognition Theory, Research and Applications*, MIT Press, Cambridge, MA.
- Finkelstein, J., Pittinsky, M. "The Evolving Role of Course Management System Providers in the Transformation of Education: An Interview with Blackboard's Matthew Pittinsky". *The Technology Source*, January/February 2003. <http://ts.mivu.org/default.asp?show=article&id=1039> (10/6/2003).
- Gregory, S. (1966). *In a contribution to The Design Method*. London: Butterworths.
- Hayes, J. R. (1978). *Cognitive Psychology: Thinking and Creating*. Homewood, IL: Dorsey.
- Henderson, K. (1999). *On Line and Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering*, MIT Press, Cambridge, MA.
- Jones, J. Christopher. (1966). *Design Methods Reviewed*. In *The Design Method*. London: Butterworths.
- Jones, J. Christopher. (1980). *Design Methods*. New York: John Wiley & Sons.
- Kolb, D.A. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall.
- Larkin, J., Simon, H. (1987). "Why a Diagram is (Sometimes) Worth Ten Thousand Words", *Cognitive Science*, Vol. 11, pp. 65-99.
- Matchett, E. (1968). *Control of Thought in Creative Work*. *The Chartered Mechanical Engineer*, 14, 4.
- McGown, A., Gree, G., Rodgers, P. (1998). "Visible Ideas: Information Patterns of Conceptual Sketch Activity", *Design Studies*, Vol.19, No. 4, pp. 421-53.
- McKoy, L. Felicia et al. (2001). "Influence of Design Representation on Effectiveness of Idea Generation", *Proceedings of DETC'01: ASME 2001 Design Engineering Technical Conference and Computers and Information in Engineering Conference*, Pittsburgh, PA.
- McQueen, R. (1998). "Four Views of Knowledge and Knowledge Management". In *Proceedings of ACIS 98*, pp. 609-611.
- Miller, A. (1984). *Imagery in Scientific Thought: Creating 20th Century Physics*. Birkhauser, Boston, MA.
- Newell, Alan, and Herbert A. Simon. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

- Page, J. K. (1966). *Contribution to Building for People*. 1965 Conference Report. London: Ministry of Public Building and Works.
- Pahl G. & Beitz W. (1996). *Engineering Design: A Systematic Approach*. London: Springer-Verlag.
- Palmer, S. R. and Bray, S. L. (2001). "Longitudinal Study of Computer Usage in Flexible Engineering Education". *Australian Journal of Educational Technology*. 17(3), 313-329.
- Polanyi, M. (1966/1997). *Knowledge in Organizations*. Newton, MA: Butterworth-Heinemann.
- Radcliffe, David F. Lee, Tat Y. (1990) "Models of Visual Thinking by Novice Designers". *Design Theory and Methodology*, pp 145-12.
- Reswick, J. B. (1965). *Prospectus for Engineering Design Centre*. Cleveland, Ohio: Case Institute of Technology.
- Riel, M. (2000). "New Designs for Connected Teaching and Learning". White Paper at Ed Tech Conference 2000, Alexandria, Virginia, U.S.A., 11 -12 September.
- Rowe, Peter G. (1995). *The Design Thinking*. Cambridge, MA: MIT Press.
- Ryle, G. (1949/1984). *The Concept of Mind*. Chicago, University of Chicago Press.
- Ryle, Gilbert. (1949). *The Concept of Mind*. Chicago: The University of Chicago Press.
- Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. Internet Engineering Task Force, Work in Progress (Update to RFC 1889).
- Shah, J., Vargas-Hernandez, N., Kulkarni, S., and Summers, J. (2001). "Collaborative Sketching (S-Sketch) – An Idea Generation Technique for Engineering Design", Accepted to appear in *Journal of Creative Behavior*.
- Spender, J. C. (1988). "Pluralist Epistemology and Knowledge-based Theory of Firm". *Organization*, Vol. 5, No. 2, pp. 233-256.
- Suwa, M., Tversky, B. (1997). "What Do Architects and Students Perceive in Their Design Sketches? A Protocol Analysis", *Design Studies*, Vol. 18, No. 4, pp. 385-403.
- Thorndike, E. L. (1931). *Human Learning*. Cambridge, MA: MIT Press.
- Ullman, David G, et al. (1990). "The Importance of Drawing in the Mechanical Design Process", *Computer & Graphics*, Vol. 14, No. 2. pp. 263-274. Pergamon Press, Great Britain.

Foot Note

- ¹ Archer, B (1973). *The Need for Design Education*. Royal College of Art.
- ² Palmer, S. R. and Bray, S. L. (2001). "Longitudinal Study of Computer Usage in Flexible Engineering Education". *Australian Journal of Educational Technology*. 17(3), 313-329.
- ³ Dede, C. (2000). "Emerging Influences of Information Technology on School Curriculum". *Journal of Curriculum Studies*. Vol.32, No.2.
- ⁴ Finkelstein, J., Pittinsky, M. "The Evolving Role of Course Management System Providers in the Transformation of Education: An Interview with Blackboard's Matthew Pittinsky". *The Technology Source*, January/February 2003. <http://ts.mivu.org/default.asp?show=article&id=1039> (10/6/2003).
- ⁵ BlackBoard Inc. *New Features and Functionality in the Teaching and Learning Environment*. BlackBoard Learning System Data Sheet. <http://www.blackboard.com/docs/datasheets/newfeaturesLS.pdf> (10/6/2003).
- ⁶ Ryle, G. (1949/1984). *The Concept of Mind*. Chicago, University of Chicago Press.
- ⁷ Riel, M. (2000). "New Designs for Connected Teaching and Learning". *White Paper at Ed Tech Conference 2000*, Alexandria, Virginia, U.S.A., 11 -12 September.
- ⁸ Felder, Richard M. & Silverman, Linda K. (1988) "Learning and Teaching Styles in Engineering Education". *Engineering Education*, 78(7), 674-681 (1988)
- ⁹ Kolb, D.A. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall.
- ¹⁰ Felder, Richard M. (1996). *Matters of Style*. ASSE Prism, 6(4), 18-23 (December 1996).
- ¹¹ Felder, Richard M. (1993). *Reaching the Second Tier: Learning and Teaching Styles in College Science Education*. *J. College Science Teaching*, 23(5), 286-290 (1993)
- ¹² Alexander, Christopher. (1963). "The Determination of Components for an Indian Village". In *Conference on Design Methods*. New York: The MacMillan Company.
- ¹³ Archer, L. Bruce. (1965). *Systematic Method for Designers*. London: Council of Industrial Design.
- ¹⁴ Asimow, M. (1962). *Introduction to Design*. New York: Prentice-Hall.
- ¹⁵ Booker, P. J. (1964). Written Contribution appended to *Conference on the Teaching of Engineering Design*. London: Institution of Engineering Designers.

-
- ¹⁶ Farr, Michael. (1966). *Design Management*. London: Hutchinson.
- ¹⁷ Fielden, G. B. R. (1963). ('The Fielden Report') *Engineering Design*. London: Her Majesty's Stationery Office.
- ¹⁸ Gregory, S. (1966). In a contribution to *The Design Method*. London: Butterworths.
- ¹⁹ Jones, J. C. (1966). Design Methods Reviewed. In *The Design Method*. London: Butterworths.
- ²⁰ Matchett, E. (1968). *Control of Thought in Creative Work*. *The Chartered Mechanical Engineer*, 14, 4.
- ²¹ Page, J. K. (1966). Contribution to *Building for People*. 1965 Conference Report. London: Ministry of Public Building and Works.
- ²² Reswick, J. B. (1965). *Prospectus for Engineering Design Centre*. Cleveland, Ohio: Case Institute of Technology.
- ²³ Atkinson R. L., Atkinson R. C., Smith E. E., and Bem D. J. (1993). *Introduction to Psychology (11th edition)*. Fort Worth TX: Harcourt Brace Jovanovich
- ²⁴ Spender, J. C. (1988). "Pluralist Epistemology and Knowledge-based Theory of Firm". *Organization*, Vol. 5, No. 2, pp. 233-256.
- ²⁵ McQueen, R. (1998). "Four Views of Knowledge and Knowledge Management". In *Proceedings of ACIS 98*, pp. 609-611.
- ²⁶ Bruner, J. (1966). *Towards a Theory of Instruction*. Cambridge, MA: Harvard University Press.
- ²⁷ Polanyi, M. (1966/1997). *Knowledge in Organizations*. Newton, MA: Butterworth-Heinemann.
- ²⁸ Ryle, Gilbert. (1949). *The Concept of Mind*. Chicago: The University of Chicago Press.
- ²⁹ Ryle, Gilbert. (1949). *The Concept of Mind*. Chicago: The University of Chicago Press.
- ³⁰ Hayes, J. R. (1978). *Cognitive Psychology: Thinking and Creating*. Homewood, IL: Dorsey.
- ³¹ Rowe, Peter G. (1995). *The Design Thinking*. Cambridge, MA: MIT Press.
- ³² Thorndike, E. L. (1931). *Human Learning*. Cambridge, MA: MIT Press.
- ³³ Newell, Alan, and Herbert A. Simon. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.
- ³⁴ Bazjanac, Vladimir. (1974). "Architectural Design Theory: Models of the Design Process." In William R. Spillers, ed., *Basic Questions of Design Theory*, pp. 8-16. New York: North-Holland.

-
- ³⁵ Churchman, C. West. (1967). "Wicked Problems", *Management Science*, 4, no. 14, pp. B-141, and B-142.
- ³⁶ Jones, J. Christopher. (1980). *Design Methods*. New York: John Wiley & Sons.
- ³⁷ Pahl G. & Beitz W. (1996). *Engineering Desing: A Systematic Approach*. London: Springer-Verlag.
- ³⁸ Ullman, David G, et al. (1990). "The Importance of Drawing in the Mechanical Design Process", *Computer & Graphics*, Vol. 14, No. 2. pp. 263-274. Pergamon Press, Great Britain.
- ³⁹ Radcliffe, David F. Lee, Tat Y. (1990) "Models of Visual Thinking by Novice Designers". *Design Theory and Methodology*, pp 145-12.
- ⁴⁰ McKoy, L. Felicia et al. (2001). "Influence of Design Representation on Effectiveness of Idea Generation", *Proceedings of DETC'01: ASME 2001 Design Engineering Technical Conference and Computers and Information in Engineering Conference*, Pittsburgh, PA.
- ⁴¹ Ullman, David G, et al. (1990). "The Importance of Drawing in the Mechanical Design Process", *Computer & Graphics*, Vol. 14, No. 2. pp. 263-274. Pergamon Press, Great Britain.
- ⁴² Booker, P. J. (1963). *A History of Engineering Drawing*. Pub Cahtto and Windus, Lodon.
- ⁴³ Miller, A. (1984). *Imagery in Scientific Thought: Creating 20th Century Physics*. Birkhauser, Boston, MA.
- ⁴⁴ Finke, R., Ward, T., Smith, S. (1992). *Creative Cognition Theory, Research and Applications*, MIT Press, Cambridge, MA.
- ⁴⁵ Henderson, K. (1999). *On Line and Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering*, MIT Press, Cambridge, MA.
- ⁴⁶ McGown, A., Gree, G., Rodgers, P. (1998). "Visible Ideas: Information Patterns of Conceptual Sketch Activity", *Design Studies*, Vol.19, No. 4, pp. 421-53.
- ⁴⁷ Ullman, David G, et al. (1990). "The Importance of Drawing in the Mechanical Design Process", *Computer & Graphics*, Vol. 14, No. 2. pp. 263-274. Pergamon Press, Great Britain.
- ⁴⁸ Suwa, M., Tversky, B. (1997). "What Do Architects and Students Perceive in Their Design Sketches? A Protocol Analysis", *Design Studies*, Vol. 18, No. 4, pp. 385-403.
- ⁴⁹ Shah, J., Vargas-Hernandez, N., Kulkarni, S., and Summers, J. (2001). "Collaborative Sketching (S-Sketch) – An Idea Generation Technique for Engineering Design", Accepted to appear in *Journal of Creative Behavior*.
- ⁵⁰ Radcliffe, David F. Lee, Tat Y. (1990) "Models of Visual Thinking by Novice Designers". *Design Theory and Methodology*, pp 145-12.

⁵¹ Larkin, J., Simon, H. (1987). "Why a Diagram is (Sometimes) Worth Ten Thousand Words", *Cognitive Science*, Vol. 11, pp. 65-99.

⁵² Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Work in Progress (Update to RFC 1889).

⁵³ Beavers, J. and Moffatt, C. (2002). *RTP Specification*. Found at Conference XP project website as of 1/22/2004. http://www.conferencexp.net/community/uploads/RTP%20Specification.htm#_Toc8666796

⁵⁴ Beavers, J., Moffatt, C. and Anderson, R.. (2002). Found at Conference XP project website as of 1/22/2004 <http://www.conferencexp.net/community/uploads/RTDocs%20Specification.doc>

⁵⁵ *Conference XP Getting Started Guide*. Found at Conference XP website as of 1/22/2004. <http://www.conferencexp.com/community/Default.aspx?tabindex=8&tabid=73#SystemRequirements>