

THESIS DOCUMENT

DUSAN A. KOLJENSIC
dusan@egodigita.com

Acknowledgements

Abstract ??

Foreword ??

Interests: Marriage of science, design and technology ??

Experience ??

Personal Background / From Programming To Graphic Design ??

Dot-Com Boom / Teaching Experience ??

Professional Work ??

Framework ??

Birth Of Internet: Web As A Platform For Exchange Of Ideas / Function Over Form ??

Popularization Of The Web: Graphic Web Browsers / Form Over Function / Initial Design Standards ??

Technological Advances: HTML / Server Side Scripting / Style Sheets / XML / Return To Content ??

Information Architecture: Content Organization / System Architecture / Central Role In Project Design ??

Internet Grows: User Perception Of The Medium / Content Management Systems / Modularity ??

Different Approach: Web Site As An On-Line Application ??

Market Forces: Collapse Of The .Com Bubble / Necessity For Content Management / Outsourcing ??

New Option: Application As Viable Replacement For Web/Multimedia ??

Conclusions: UI Standardization / Internet As Content Depository / Separation Of Form And Content ??

TABLE OF CONTENTS

Project 1: Web Application: Content management system for education	??
Why I chose this project	??
Dealing with a complex dataset	??
Integration with existing systems	??
Technology	??
Usability as design: different user practices (teacher vs. student)	??
The application	??
Conclusions	??
Project 2 (Thesis): Graphing Application for Molecular Biology	??
Initial Idea: Graph As Data	??
Existing Solutions	??
Dataset / Anatomy Of A Graph / Specifics Within Individual Laboratories	??
Target Audience / User Work Process	??
Development	??
Conclusions	??
Information	??
Networked Application	??
Focus On User Work Process	??
Moving The Interface Forward	??

Project 3 (Thesis): Graph creation tool	??
Initial Ideas	??
The Design Problem	??
Inspiration	??
Preliminary Design	??
Behavior	??
Integration Into The Application	??
Conclusions	??
Other Uses	??
Potential For The Future	??
Appendix	??
Paper: Object oriented information	??
Curriculum Vitae	??

ABSTRACT

ACKNOWLEDGEMENTS ABSTRACT FOREWORD EXPERIENCE FRAMEWORK PROJECT 1 PROJECT 2 PROJECT 3 APPENDIX

In November 2002 i spent a day in one of the molecular biology labs at the Boston University observing researchers working at their computers. I was surprised to see that the programs they were using to present their findings were standard graphic design applications: Illustrator, Photoshop and QuarkXpress. They explained that just about every published piece of scientific findings involved the creation of graphs using these design tools. No graphics software tailored to their specific needs – focusing on data instead of on visual formatting – existed.

My aim is to develop an application for molecular biologists that facilitates separation of form and content, as well as a visual interface that allows them to quickly and accurately build scientific graphs.

On the back-end, using XML, the application will provide a platform for embedding actual scientific data into the graph elements themselves, making them searchable and indexable, amongst other things. On the front-end, the application will aide in the separation of the science from design – choosing the right element vs. choosing what that element should look like.

FOREWORD

ACKNOWLEDGEMENTS

ABSTRACT

FOREWORD

EXPERIENCE

FRAMEWORK

PROJECT 1

PROJECT 2

PROJECT 3

APPENDIX

INTERESTS: MARRIAGE OF SCIENCE, DESIGN AND TECHNOLOGY

My grandfather was a teacher and a philosopher. He spent his life pondering the purpose of life and wrote numerous books about the topic. Although he died when I was only three years old, I grew up being told that we had a special bond, and that he had a grand plan for me. He was going to teach me everything he knew.

I grew up in an environment where knowledge was the most prized possession. My parents had a large library with books on all kinds of topics: philosophy, engineering, medicine, geography, history – too many to list. They often told me that they kept every book they ever purchased so that we, their children, could learn from them, and pass them on to our children.

As a kid, I used to love encyclopedias – I read them the way other kids read comic books. They provided the broadness of material that I found utterly fascinating. They provided the answers to all the ‘how’s’ I had in my head.

As a teenager, I was interested in philosophy and psychology. I read everything that I could get my hands on: Aristotle, Plato, Confucius, Descartes, Kant, Nietzsche, Adler, From, Jung, Freud. Soon, ‘why’ replaced the ‘how’ as the driving question. However, ‘how’ was absolutely necessary to explain (or to attempt to explain) the ‘why’.

My interest in computers started early. However, I never saw computer technology as a discipline in itself but more as an overarching service helping other disciplines. As a matter of fact, the first computer programs I ever wrote were problem solvers for mathematics, physics and engineering (for my dad – he is a civil engineer). This contact

FOREWORD

INTERESTS: MARRIAGE OF SCIENCE, DESIGN AND TECHNOLOGY

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

between computer technology, and especially web design later on, with other disciplines, was what kept my work interesting. In order to solve a particular problem, I first had to learn all I could about the problem itself. This act of learning, of analyzing the problem and the people involved, and then finding the best solution was the most exciting part of my job as an interface designer and later as an information architect.

For the last several years, technology aside, there have been two major topics I have been particularly interested in: quantum physics (with particular focus on string theory) and molecular biology (genetics). Both interested me more for their philosophical aspects than for the science itself: quantum physics deals with how reality works while molecular biology deals with how living beings (us humans being one of them) work.

As I started my graduate education, I wanted to find a way to combine my newfound interests with my design experience.

At the time when I was just starting with the graduate school, I spent a day in one of the molecular biology labs at the Boston University. As I was walking by researchers hunched over their workstations, I was surprised to see that almost every computer had one of the standard graphic design applications (Illustrator, Photoshop, QuarkXpress) running on the screen. In talking to them about it, i was told that just about every published piece of scientific findings involved the creation of graphs using these, primarily design, tools. There was no software tailored to their specific needs.

The idea was born. I was going to use my interest in molecular biology, together with my experience in information architecture and interface design, to develop a concept for the application that will provide a better environment for building scientific graphs for molecular biology.

EXPERIENCE

ACKNOWLEDGEMENTS ABSTRACT FOREWORD **EXPERIENCE** FRAMEWORK PROJECT 1 PROJECT 2 PROJECT 3 APPENDIX

PERSONAL BACKGROUND / FROM PROGRAMMING TO GRAPHIC DESIGN

I was born in Belgrade, Yugoslavia in 1974.

When I was 7 I accidentally got my hands on a British computer magazine. However strange this might sound, I just knew this was something I wanted to learn more about.

Although my parents did not know much about computers, they realized early enough the importance of this new technology.

I was 8 years old when my parents persuaded our cousin, a stewardess, to smuggle a Sinclair ZX computer into the country for me as a birthday gift. My father gave it to me with the following words: "When you learn everything you can about this one, I'll buy you a better one." I haven't spent a day without sitting in front of a computer since.

Pre-made software was scarce and expensive, so I spent most of my time programming. By the end of primary school – age of 15 – I won the national competition in computer programming, and decided to enroll in a special high school concentrating on computer technology.

I spent next three years learning about higher-level programming languages as well as computer architecture. I began finding real beauty in code. There is a certain elegance in a perfectly written set of commands that execute an operation in a most efficient way possible that I equated to poetry. This idea of perfectionism, stripping down all superfluous code until only the pure condensed information is left, is still the core principle in my work.

EXPERIENCE

PERSONAL BACKGROUND / FROM PROGRAMMING TO GRAPHIC DESIGN

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Another, seemingly irrelevant event, ended up guiding the way I think about problem solving to this day. In one of our programming classes the teacher gave us a problem that didn't seem to have much to do with programming: "You have a large rectangular grass field in the middle of the city. What is the best way to make pedestrian paths through that field?" Being technically inclined, everyone started calculating the shortest distances, paths with least number of turns, etc. When the teacher asked me about my solution, smart-alecky that I was, I said: "Just let the people walk through it and wherever they go, build a path." Apparently, that was the right solution.

By the end of the 3rd (junior) year, the civil wars were raging through former Yugoslavia, and since I was about to get of military age, I received a call to report to the local military office. Since I disagreed with all the wars and violence going on, I decided to pack my bags and leave the country instead.

I ended up in Watertown, MA, where I finished my senior year.

Being away from home, and without the influence of my peers and parents, I looked hard into what I wanted to be doing for the rest of my life, and, to my own surprise, realized that I did not want to be a programmer. I enrolled in art classes, and fell immediately in love with design. The fact that I could use a computer to pursue my new passion was just an icing on the cake. I immediately lobbied the Watertown High School art department to buy a computer (they didn't have one at a time), and, through a great help from wonderful Nancy Pippito (the chair of the art department), got a part-time job as a lab assistant at the Massachusetts College of Art. My path was clear.

I spent most of that semester building a portfolio, since I never did anything artistic before that. The portfolio was, of course, comprised solely of computer generated artwork. I applied to Massachusetts College of Art undergraduate school of design and was accepted. My undergraduate education in design started in September 1993.

DOT-COM BOOM / TEACHING EXPERIENCE

In 1995, I landed a freelance job as a package designer for Addison Wesley Longman – a large publishing company. The whole web phenomenon was just starting, and I knew very little about this new emerging technology. A couple of months after i started working with them, I received a call from one of the project managers. They were going to put all their books on line and they wanted to know if I knew anything about this new thing called Web Design. Of course, I said yes. I went home, bought books on HTML, didn't sleep for about a week, and learned everything I could about web site development. My first project was to create a comprehensive catalog web site for Addison Wesley Longman. This combination of programming and design immediately 'clicked' with me. I have been doing web site development since.

In 1996 the officials at the Massachusetts College of Art were starting to think about introducing a web design course. Although I was only an undergraduate senior, I wrote up a curriculum for the introductory web course and submitted it to the MassArt Continuing Education department. They allowed me to run the pilot course that same year. By next year, that course was made available to undergraduate students. That was the start of my web design teaching career at MassArt.

At the beginning of the .com boom, the technology behind the web was growing at an astounding pace. HTML was being expanded to meet the ever growing demands of the rapidly expanding audience. Server side technology was being developed that would make for a more dynamic user experience. The very idea of what the web was supposed to be and do changed with every new advancement. My teaching demanded that I try my best to be at the leading

APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

EXPERIENCE PROFESSIONAL WORK

edge of the technology. It was not enough just to know about all current trends and technologies. I had to look ahead and try to envision what would be the next thing available at the end of the semester.

PROFESSIONAL WORK

During those first couple of hectic years, I built over 300 static (HTML only) web sites for various clients. The sheer quantity of work allowed little space for errors, so I started concentrating on the process for easier project development. This brought forward ideas of standardization and modularity as well as really tight user-testing to avoid potential problems after the release of the site. Since, at the time, Information Architecture was emerging as a discipline of its own, and since it involved all of these concerns, in time, Information Architecture replaced general web design as my core passion.

From the onset, I saw Information Architecture as a field encompassing technology, design, as well as sociology, philosophy, and other sciences dealing with how people perceive, interact, and in general deal with complex information structures such as web sites, multimedia pieces and applications.

I believe that for someone to be a true Information Architect, that person needs to possess a formidable knowledge in all above mentioned disciplines. As a result, I continued developing my knowledge of computer programming, general networking and design. I also learned about business processes, management, as well as social implications of technology advancements and multicultural approach to web and software development.

Because of their limited scope (in terms of Information Architecture), i quickly lost interest in designing static web sites. Dynamic, database driven web sites posed much more of a challenge, so I moved in that direction. As the technology progressed, and the programming behind complex web sites became more and more similar to standard application programming, my interests shifted again. One idea from web development remained: modularity. Modularity in terms of distributed systems consisting of multiple components located on different servers (or computers) all working together to deliver a more enriching user experience. For me, there was an enormous beauty in an idea that you can have all of these separate little programs, existing and performing their functions all over the place, all assembled into a unified user interface. I used this concept over and over again in my subsequent work both for web sites and applications.

The separation of form and content, as a consequence of developments in database and content management technologies, as well as standardization in describing content (through XML for instance) expanded my thinking about how these complex systems should work.

An ideal application would consist of a number of smaller programs performing specific tasks existing on platforms best suited for the execution of that particular task, using the same set of standard formatted data, all brought together into a unified user interface. All that would need to be written is a conduit unifying the distributed pieces. The functionality of this kind of application could be easily expanded or modified by adding new programs, modifying existing ones, or changing the data.

This kind of thinking brought me a particular way of coding I call 'scavenger programming'. If an application is broken into these individual simple tasks, chances are that someone in the Open Source community already has the

EXPERIENCE PROFESSIONAL WORK

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

code. Since the platform (server, os, programming language) is irrelevant, the number of the preexisting solutions becomes vast. Programmers can now quickly create complex applications using pre-made code.

This idea is not original – programmers have re-used their code since the beginning of time. With the added benefit of distributed execution and the availability of Open Source code, the concept definitely rises to new heights. The reduction in time and costs this approach brings, combined with the advancement of ‘outsourcing’ allows us to develop on- or off-line applications that target very specific problems and concentrate much more on users’ needs than on the return on investment.

My goal was to eliminate the technology as the key factor (due to expense) in guiding development, and return the focus on user experience.

I would like to conclude this chapter with an example from my consulting work:

One of my clients, a PR company (‘the company’), invited me to a meeting to discuss a possible solution to a multimedia piece they needed to develop for their client. The final client had a large quantity of documents, all in Word, Excel and PowerPoint format, that they wanted to provide to their overseas offices. The company’s sales rep offered a solution they were familiar with – a flash program that would list all the content.

After the meeting, I could see that they were missing an opportunity. They were focusing on a solution they were familiar with, and not looking at the real problem.

First, in order to pour the content into a flash movie it would have to be converted into some format flash

would accept (text, XML, etc.) This would require unnecessary work, since someone already formatted those original documents. Second, since the final client was planning on updating the existing documents and adding new ones regularly, this would require more work every time the update was necessary. Third, all the overseas workers were required to use Microsoft Office (Word, Excel and PowerPoint) as a global corporate standard and this would introduce a completely new program.

The solution was simple. All Microsoft products are tightly integrated – they are designed so that each can open all Office formats. This means that all these documents can be opened using another overarching Microsoft product – Internet Explorer. Internet Explorer, a web browsing program, is also an integral part of the operating system used to format and display interface elements. Every time you open a folder, the operating system is using Internet Explorer to format the contents. Every time you look at your email, it is the Explorer that formats it for the screen. In the essence, the Explorer is one of those small programs that can be used by other programs to execute a particular task. As an added benefit, all the documents opened in Explorer are fully editable and searchable.

My solution was first to organize the existing documents into a logical folder structure. Then create a simple conduit application that would provide the browsing system (using the folder names as navigation ‘levels’), and the Explorer to actually ‘show’ the documents. Search is a built in os-level function that is automatically provided. This would solve all the above mentioned problems. Since the new application would open the original documents, there would be no need for any kind of reformatting. Adding or updating a document would be as simple as dropping a new one into the proper folder. It would immediately be available to the application. An added bonus was that this folder structure could reside on a single server, so that everyone is using the same set of documents. The last problem would be solved by simply not designing this application. Use the standard operating system GUI that resembles Office



EXPERIENCE PROFESSIONAL WORK

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

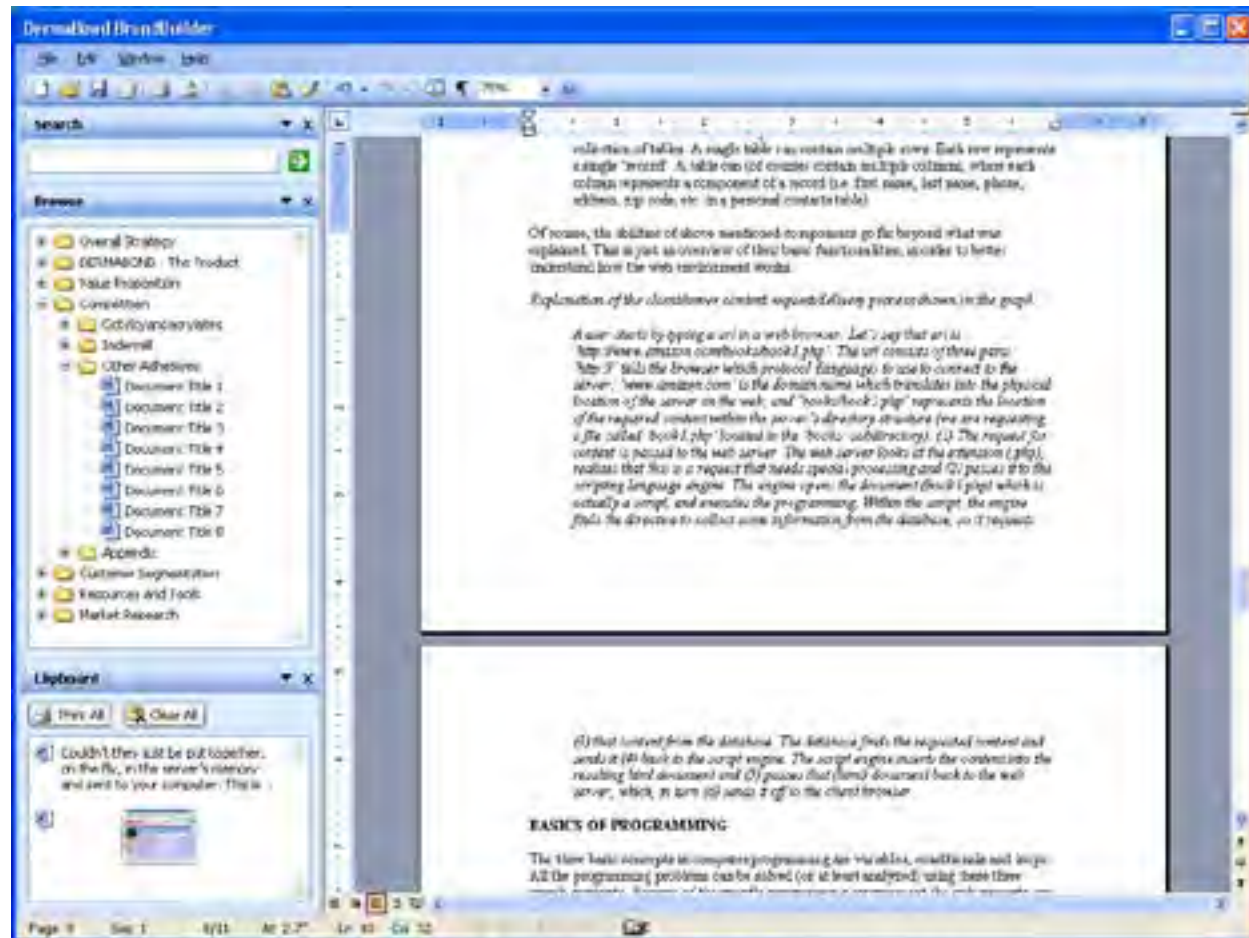
products all the workers are already using. All we needed to do would be to organize the functionality of the new application to make it easy and intuitive.

The solution was successful. The client requested one of these programs for each division within the company.

Instead on focusing on the product and it's content as a separate entity, it is necessary to analyze it within the broader environment where it will be used. Client's business practice as well as cross-corporate technology standards provide clues to how the product should be structured and developed.

As visual communicators, we often approach problems from the design perspective. Focusing on the look and feel of the product takes precedence over the intricacies of the inner-workings of the solution. In case of software applications, interface elements are readily available. The primary focus is then on the usability, achieved through the organization of existing interface components.

The intimate knowledge of available technologies – in this case the interoperability of Microsoft applications – allows us to develop comprehensive solutions that, at the same time, provide user friendly interfaces as well as reduce the production time and cost.



FRAMEWORK

ACKNOWLEDGEMENTS

ABSTRACT

FOREWORD

EXPERIENCE

FRAMEWORK

PROJECT 1

PROJECT 2

PROJECT 3

APPENDIX



BIRTH OF INTERNET:

WEB AS A PLATFORM FOR EXCHANGE OF IDEAS / FUNCTION OVER FORM

Note: The history and chronology that I give here is as I experienced it first hand except if otherwise noted.

The internet infrastructure was originally created for the U.S. military as a technology for sharing information. The technology, though, soon started gaining foothold for the civilian application. In 1989, Tim Berners-Lee, an employee of computer services department at CERN (the European Laboratory for Particle Physics in Geneva, Switzerland) came up with the idea of researchers from institutes all over the world sharing information quickly and easily. More importantly, this 'depository' for research findings should not be just a collections of individual papers. The content in one paper could be 'linked' with the relevant content in the other, thus enriching the quality of the information. The HyperText Markup Language (HTML), born out of this idea, was based on Standard Generalized Mark-up Language (SGML), an internationally accepted method for marking up text into structural units such as paragraphs, headings, list items etc. Taking an existing, proven, readily available and accepted technology and expanding on it, brought as radically new and different result as something developed from scratch. In this case, as well as many times since (think Napster), the difference in concept - how the technology was used - was much more important than which technology was used.

Markup languages use tags to define visual or contextual properties of the content. This is done by surrounding a piece of content with an appropriate tag pair – an 'opening' and 'closing' tag. For example: `IMPORTANT` would make the word IMPORTANT bold.

APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

FRAMEWORK POPULARIZATION OF THE WEB

Since the newly developed HTML was supposed to be a vehicle for exchange of information, the tags used to format this information were, naturally, content based. For instance, the tag used to represent an address was <address>, the tag represented emphasized text was <emphasize>, the tags represented headlines were <h1>-<h6> depending on a relevance of a headline, etc. The tags were describing the content nested within them. What the content looked like was much less relevant than what the content was. The whole purpose of the language was to describe the organization of the document and not its visual properties. This, however, did not last long.

POPULARIZATION OF THE WEB: GRAPHIC WEB BROWSERS / FORM OVER FUNCTION / INITIAL DESIGN STANDARDS

In 1992, the National Center for Supercomputer Applications (NCSA), a research institute at the University of Illinois at Champaign-Urbana, decided to start developing the first graphical version of a web browser – Mosaic.

Previous browsers (i.e. Lynx) were text-based, and had to be accessed using a terminal emulation program (Telnet). The Telnet program runs on the user's computer and connects it to a server on the network. User can then enter commands or run applications – such as the Lynx browser – through the Telnet program as if he was working directly on the server computer.

Marc Andreessen, who was later to start Netscape, was one of the programmers on the Mosaic team.

At the time, ideas about advances in the development of HTML were debated on an electronic discussion group called WWW-talk. It was understood that the standards for further development should be accepted by all participants. Members of the Mosaic team, and Marc Andreessen in particular, disagreed. The decision by consensus was too inefficient for the rapid progress of the new technology. One of the first tags implemented in the Mosaic browser without the acceptance of the WWW community was the tag (tag for placing images in documents) – beginning a new direction in HTML development.

In 1993, upon graduating, Mark Andreessen, together with an investor Jim Clark, formed a new company called Mosaic Communications. In November, 1994, the company changed its name to Netscape Communications Corp.

The aims of the new corporation were clearly commercial. Netscape understood that the new technology would grow only if general computer users accepted it, and for that to happen, the visual appearance of the web pages needed to be ‘sexed up’.

In order to try to impose some sense on the development of the web standards, World Wide Web Consortium (W3C) was formed in 1994. Although it did not have any actual power in having the standards implemented, it remained a central place for the discussions and innovations related to the advancement of World Wide Web.

1995 was, in many ways, the darkest year in the annals of internet development. At the same time, it was the year when the .com boom officially started, with the ever wider acceptance of the web as the largest information depository in history.

In 1995 Netscape added a large number of new tags (without discussing any of them with the WWW community)



FRAMEWORK POPULARIZATION OF THE WEB

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

to its now dominant web browser. The tags were clearly aimed at visual formatting of the documents. The tags, such as BGCOLOR (for background color), FONT (for font properties), and many others were added to Netscape's HTML implementation. These tags did not describe the content they surrounded – they were primarily there to make the content look a certain way. Since these tags offered much more control over the design of the content (and designers were flocking to this new medium like there was no tomorrow – literally), they quickly replaced the content-based tags in general use. Headlines, subheads, callouts, and other document elements were described using the new visual formatting tags, and thus, were impossible to recognize as such within a structure of a document. Later on, this proved to be a serious problem with search engines, trying to decipher relevant information in this jumble of content-irrelevant formatting. Although the World Wide Web community protested this kind of abuse of the original HTML idea, Netscape paid little attention, emboldened by the growing acceptance and excitement over the new technology.

In the same year, Microsoft released its Internet Explorer browser, starting the 'browser war' that still plagues the development of internet standards. Both companies believed that the supremacy in this war would be achieved with proprietary extensions of HTML. They each added tags and technologies to their browsers that were not supported by the competing one, thus rendering web designers' lives a living hell. Instead of using these advanced technologies that might not work on all browsers, web developers accepted the smaller number of properly implemented tags, in a sense, a lowest common denominator, and created sites that worked everywhere. Sadly, visual formatting tags were usually best implemented on all browsers.

The only bright spot in 1995 was the initial development of CSS (Cascading Style Sheet) standard. Realizing the need for a return to the original ideas of HTML as a document description language, style sheets (similarly to publishing programs) were to be used to apply visual formatting to content-based tags. Because of the poor implementation in

browsers, this technology, although vastly superior, did not become a full standard for years. It did, however provide a starting point for another crucial advance in web technology – the development of XML (eXtensible Markup Language).

Amidst the chaos created by the browser wars, unlikely standardization emerged in page design. In particular, the positioning of the page navigation systems. This was driven in part by the way we read – left to right, top to bottom - and in part by the technical limitations – since the bottom right corner of a window was not anchored, it seemed natural to position navigation close to the upper-right corner of the screen so that none of it is cut off. It also seemed driven by the users' (and designers') familiarity with another ever present graphical user interface – the operating system. In both Windows and Mac os, the navigation bars at the top and on the side are standard components of applications as well as the os itself. In a sense, since the beginning of the web design, the web sites were trying to simulate visual organization of desktop applications. Sites that strayed too much were quickly dismissed as disorganized or hard to navigate.

TECHNOLOGICAL ADVANCES:

HTML / SERVER SIDE SCRIPTING / STYLE SHEETS / XML / RETURN TO CONTENT

Initial requests by clients in the early days were very simple. Most wanted their brochures converted to this new medium and made available to their potential customers (thus the term 'brochureware' – the name given to these



FRAMEWORK TECHNOLOGICAL ADVANCES

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

simple small websites). Since, at the time, very few companies specialized in web design, they simply asked the same graphic designers who created the brochures if they could re-create them for the web (which is exactly how I started my web design work). Since HTML was simple, and thus did not require a large investment in time to learn, a lot of designers gladly accepted this new role as ‘web developers’.

This was an interesting, albeit a temporary, anomaly. Graphic designers were not, and still are not, trained to be programmers, nor did they ever have a real interest in becoming ones. Yet, for the first few years of web development, they had complete control over all the elements in the process of building web sites. Since their primary interest was in the ‘look and feel’ of these web sites, and not necessarily in the creation, organization and distribution of the content, the original idea of HTML as content description language was put aside and replaced with an idea of HTML as a design tool. This was, in a sense, the driving force for the inclusion of format-specific tags into the web browsers. The majority of effort was put into giving designers full visual control over all the page elements. Since text (actual content) allowed for the least amount of control, it was often replaced with images created with other design programs that provided better formatting options. This posed a problem – the text formatted in Photoshop, for instance, then placed on the page as an image, lost its information properties. This method was worse than visual formatting tags, which at least left text intact.

HyperText Markup Language (HTML), as the name implies, is a markup language. This means that its role is to describe properties of content nested within its tags. Since it lacks more advanced programming elements – loops, conditionals, etc. – once the content is formatted there is very little that can be done in terms of its response to user interaction. The ability of the web page to respond to user input (other than clicking on a link which doesn’t provoke a response – it just opens another page) was recognized from the beginning to be an important feature for further

development of the new medium. To fix this, Netscape and Microsoft decided to add basic programming capabilities to their browsers. Of course, just like in all other features, they could not agree on the standard language that would work the same on both browsers. Netscape implemented a more standard simplified version of Java called JavaScript, while Microsoft decided to go with its own version of Java-based scripting language called JScript. Both languages were supposed to exist intertwined with HTML and provide a level of interaction within a single HTML document. Since all the commands were supposed to be executed within the web browser itself, on the client machine, these languages were called 'client-side scripting languages'.

By the end of the nineties, Microsoft dropped their version of JScript in favor of a more standard JavaScript thus providing a solid platform for user interaction with the components within a web page. Although JavaScript provided quite a bit of programming capability, the whole script still had to be completely downloaded to the user's browser before it was executed. This meant that if the script was too large and complex, first, it would take too long to download and second, the potential for error due to browser differences was too great. Also, for security reasons, the script could execute only within the confines of the page that contained it, and it was prohibited from writing data to the hard disk of the user or the server the page came from.

Given these limitations there was clearly a need for another kind of technology that would compliment client-side scripting, but, by off-loading processing to the server, provide more flexibility and power in dealing with data.

Server-side scripting languages were in use long before the web was invented. These programming languages resided on server computers and aided in the automation of various back-end administration tasks. Because of their utility, these languages were adopted and extended by web developers. Pearl, a server-side scripting language found



FRAMEWORK TECHNOLOGICAL ADVANCES

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

on Unix and Linux platforms, was an early example of this move to incorporate these new technologies into the web.

It is hard to write about server-side programming without touching on database technology. Different database programs (Oracle, dBase, etc.) have existed long before the web as well. Databases are, essentially, content depositories – any kind of content: text, images, numbers, etc. Content is broken into individual pieces of data and organized in rows and columns within a table. A row represents a single content ‘entry’. Columns are different pieces of data within that entry. Multiple tables can be cross-referenced to form relations between different pieces of content (relational databases).

Server-side languages have long been able to connect to databases and perform operations on the content they stored (search, add, replace, etc.). Combining these two technologies fueled new internet development and changed the fundamental idea of how web sites should work and what their role would be for both clients and their general users.

The path was set for the separation of content and form. Previously, websites were built as collections of separate HTML files, where the content was built into and formatted within the pages themselves. This new approach introduced an idea of a site as containing three separate components: The page design built as one or more templates with dedicated areas for different type of content, the database containing the actual content, and the server-side program combining the two at the moment the user requests any given piece of content. The obvious benefit of this kind of organization was that, since content and page design were separate entities, either could be easily modified or expanded without affecting the other. Since the web sites were constantly changing (the average site’s lifespan before the first content change was often only a few months) this provided a huge benefit to the client.

The issue that emerged was that server-side programming was much more complex than HTML programming. It grew increasingly harder for a non-programmer (designer) to acquire the new skills necessary for implementation of these kinds of projects. The designer could no longer be in charge of all aspects of the project – the team needed to expand to include a programmer and, often, a database expert. The designer was put in charge of the visual implementation of the user interface.

From the programming perspective, the beauty of server-side scripting is in its coding methodology. A server-side program does not have to be a single chunk of code. It is a collection of smaller scripts (subprograms) executing individual tasks and exchanging information with other scripts. Since these scripts (subprograms) can communicate through the internet, there is no need for all of them to exist on the same computer. The database, which each of these scripts can access at any time, also does not need to reside on the same computer with the program, and neither do the design templates (built in simple HTML – still the main formatting engine in web browsers). All these pieces coming together in a user's web browser, forming a unified web site, present a truly modular, distributed on-line system.

In parallel with the advancements in server-side scripting and database technology, style sheets (CSS) were quickly gaining ground on the client-side. The idea behind the CSS was quite simple and logical. Instead of using the visual formatting tags to describe what the content should look like, revert back to the original HTML concept of using the content describing tags, and then use the style sheets to apply visual properties to those tags. The initial problems plaguing the acceptance of CSS were, again, inconsistent implementation of the technology across different web browsers. It wasn't until the turn of the century that the browsers achieved an acceptable level of standardization for the usage of this technology to become viable.



FRAMEWORK TECHNOLOGICAL ADVANCES

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Web developers, however, immediately recognized the power of CSS. If we use a standard set of HTML tags to describe the content of the web page and separate the visual formatting used throughout the web site in a different file, we can quickly and easily change the design of the whole web site by simply modifying the style sheet file. This added the final component to the ultimate modular approach to web development. The building blocks of web sites became: HTML framework template, CSS describing the visual properties, and JavaScript providing user interactivity within the page on the client side, and server-side programs and databases on the server-side.

HTML reverted to its original function – the framework describing the content of the document. The circle was finally closed – the philosophy guiding the internet development returned to its original path. However, there was more work to be done.

In 1996, at the time when the number of HTML sites was growing at an exponential rate, the World Wide Web consortium (W3C) was already starting to develop the next stage in the evolution of the markup language. The XML Working Group (formerly SGML Editorial Review Board) was started in order to create the new standard for content description.

It is impossible to write about XML without first touching on SGML (Standard Generalized Markup Language).

SGML represented an evolution (standardization) of GML (Generalized Markup Language) developed by Charles Goldfarb in 1969, at the time leading an IBM research project on integrated law office information systems, as a means of allowing the text editing, formatting, and information retrieval subsystems to share documents.

In 1978, the American National Standards Institute (ANSI) committee on Information Processing established

the Computer Languages for the Processing of Text committee. Its role was to develop a text description language standard based on GML.

The first working draft of the SGML standard was published in 1980. It described SGML as an international standard for the description of marked-up electronic text. SGML used markup codes which simply provided names to categorize parts of a document. Using this descriptive markup, the same document could be processed by many different pieces of software, each of which could apply different processing instructions to those parts of it which are considered relevant.

The similarities with HTML are obvious. As a matter of fact, SGML was used as a base for the development of HTML. But, where SGML was open – it allowed users to extend (create new) tags for describing the content – and rigorous in its formatting – the developers had to use Document Type Definition (DTD) to describe the document’s constituent parts and their structure, thus reducing the possibility of an error when the document was interpreted, – HTML had a pre-defined set of tags, and was too flexible in its formatting – allowing for numerous errors in its interpretation by web browsers (which was quite obvious looking at the same chunk of bad HTML being formatted differently on different browsers). There was an obvious need for a new markup language that would bridge the gap between the flawed HTML and the aging SGML. This new language was called XML – eXtensible Markup Language.

One of the driving forces behind this was, again, standardization - but of a different kind. One could argue that the ultimate standardization would assume that every developer uses the same set of pre-created tags, with exactly the same structure for each document. Then, there would be no way to make an error in re-interpreting this document. This would be true if we did not exist in a society where there is a near-infinite number of ‘types’ of information we use every



FRAMEWORK TECHNOLOGICAL ADVANCES

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

day. For example, if I was to describe a news article, I would use terms such as 'headline', 'subhead', 'callout', 'caption', etc. If, on the other hand, I was to describe a travel itinerary, I would have to use different terms: 'departure', 'arrival', 'vehicle', 'location', etc. Obviously, in order to create a set of tags that would encompass all the types of information processed every day, the list would be impossibly long. The solution had to be more flexible.

XML is, in the essence, a language for describing markup languages – a meta-language. It does not specify any particular tag set nor any particular structure. It does provide a facility to define tags and structural relationships between them. All of the interpretation and formatting is to be defined by the application that processes the XML document, or by a style sheet. The idea is beautiful in its simplicity. Provided the core rules for structuring any given set of tags, it is up to developers to create their own tag sets that will best define the content of the document.

This concept brought the idea of re-usable information to a whole new level. Where previously the information received from a database still needed to be 'tagged' within each individual web site, the generic tags themselves making the recognition of the relevant content imperfect, now, the content could be pre-tagged using XML and simply inserted into any given site (or an application), where the style sheet would format it to fit the site's visual guidelines.

Because each tag represents an actual type of information contained within the document, an XML file could also be viewed as a database in itself, free from the confines of any particular database program. The tag names replace columns in a database table, a group of tags represents a row of information. This has taken the power of a database as a content depository away from the cumbersome, often server-based, database software, and made it available to each on- and of-line application.

INFORMATION ARCHITECTURE (IA):

CONTENT ORGANIZATION / SYSTEM ARCHITECTURE / CENTRAL ROLE IN PROJECT DESIGN

As the complexity of the web sites was growing beyond the original 'brochureware' stage, a whole new set of problems was starting to emerge. Instead of being faced with a simple redesign problem (take this printed piece and put it on the internet), the web developers were presented with a pile of content and asked to organize it for the new medium. Lacking experience in content development and organization, it was getting increasingly harder for designers to adopt to yet another role in the web site development process.

At the same time, as the web users were moving a way from simply 'surfing' the web (to use a cliché) – browsing web sites without a particular goal in mind – toward seeking specific information, it became crucial that the site content was organized in such a way that the information the user was looking for was easy to find.

The discipline emerged that aimed to solve these problems: Information Architecture.

The basic definition of the discipline goes as follows: Information architecture (IA) is the art and science of structuring information, and defining user interactions. In the context of Web design, information architecture is the organization of information to aid in information retrieval.

Although, over time, many different descriptions of IA as a discipline and the role of an Information Architect were put forth, they all more-or-less revolved around this simple definition.

FRAMEWORK INFORMATION ARCHITECTURE

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

The concept of Information Architecture, as applied to graphic design and software development existed long before the internet, but it was the growth of the World Wide Web that finally brought it forward and turned it into a discipline of its own.

Within a web development process, the role of an Information Architect was an overarching one. Since all parts of the development team had their own areas of focus – designer was concerned with the visual representation, programmer with the execution, project manager with timeline and costs – that did not necessarily overlap, it was the Information Architect's role to act as a central figure that would bring all of the pieces together, with the primary responsibility to the client's – and, consequentially, client's clients' – needs. This meant that this person needed to have working knowledge of all aspects of the process – business, design, technical – as well as the ability to organize the content based on the client and user analysis.

It is important to recognize that, in this stage, the Information Architect dealt with the information that was available (or to be made available) to the web development team. How the information was generated and managed on the client end was most often not taken into consideration. Also, the technical involvement was superficial – all the decisions about the technical execution (hardware and software) of the project were the responsibility of the programmer.

As the sites grew in complexity, they were also getting more and more incorporated into companies' everyday business practices. Combined with the exponential growth of technology, it was obvious that the Information Architect's knowledge base needed to expand to encompass these changes.

I always assumed that the role of an Information Architect went beyond the initial definition of the discipline. The

'life span' of a piece of information begins with the creator of the information, follows its path through revisions, approval, format changes (plain text document > Word document > desktop publishing document > PDF file > database entry > web page) and continues into the future with more edits and updates. Understanding this 'life span' provides a key to creating systems that reduce unnecessary reformatting of the information as well as reduce the need to simultaneously update multiple format instances.

In order to accomplish this, an Information Architect needs to know much more about common business practices (business management) as well as all the technologies involved in every step of the way. Intimate knowledge of database technology, software and hardware, as well as networking became an important factor in developing good comprehensive on-line solutions.

Contrary to the general opinion, this is the domain of an Information Architect. If the goal, as the definition states, is to structure information and define user interaction, we can't exclude the information flow, nor can we exclude the 'users' and their interaction with this information before and after it is in its web site stage.

The new Information Architect retained his previous role – information organization and interaction design – and expanded it to encompass total information system design. The result of the work was no longer to be just a web site, but a complex application often consisting of both off- and on-line components.

INTERNET GROWS:

USER PERCEPTION OF THE MEDIUM / CONTENT MANAGEMENT SYSTEMS / MODULARITY

No other technological innovation in history has been accepted so thoroughly in such a short period of time as the internet. Within a few years, as the novelty of the new technology wore thin, passive web browsing was replaced with a more task-oriented approach.

It was clear that many 'live' services (such as shopping, banking, paying bills, renewing a driver's license, etc.) could be made available through the web. Applied in this way, the use of the Internet could dramatically reduce the cost of doing business. Since such services were integral part of business processes, this new Internet-based technology had to merge with existing information management systems within companies (inventory, accounts receivables, payroll, HR systems, etc.)

The users ever-growing demand for accurate and timely information required regular updates to the content. Making the user experience more personalized required programming that would adapt the provided content based on users browsing patterns. It is easy to recognize that these new on-line systems had very little in common with the simple static web sites that initially dominated the market.

Projects requiring this level of complexity were rare. Most Internet projects were focusing on the web site as an entity separate from other business systems. In addition, as the amount of content in these web sites grew, even these less complex web sites required a component that would make content management manageable.

The name 'Content Management System (CMS)' does not represent any single application or technology. It is a combination of different technologies put together to facilitate an easier deployment and maintenance of web sites requiring continuous content changes.

A simple Content Management System consists of:

- a database or an equivalent content depository,
- a set of design templates providing the formatting for the pages viewed by the user,
- the interface for content maintenance, and
- a server-side 'program' finding the requested information and formatting it using the appropriate template, as well as providing the rules for content management (access control, approval process, etc.)

Because a CMS is a combination of different technologies, the system required a modular approach. Server-side 'programs' were made up of different pieces of code, each performing various functions. These pieces were stored in separate files to make troubleshooting and upgrading easy. Templates were designed so that page elements (navigation, search functionality, and content) were formatted by different parts of the server-side program. This required for those elements to be HTML coded as separate units to make integration easier. The database structure needed to be flexible enough to provide easy expansion and modification.

Since information organization (database), flow (server-side program), and formatting (templates) was at the core of the problem, it was clear that designing these systems was the responsibility of an Information Architect. And because these systems depended on correct use of technology to function properly, it was obvious that the knowledge base of the Information Architect had to expand well beyond the one required for the initial role (organizing content).



APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

FRAMEWORK
DIFFERENT APPROACH

DIFFERENT APPROACH:
WEB SITE AS AN ON-LINE APPLICATION

With the developments in broadband – a relatively inexpensive, constantly connected, fast internet connection, the line between the data stored on a user’s home computer and the data stored on a remote server was becoming increasingly blurred. Clicking on a button in a regular desktop application could pull information or a piece of executable code (another program) from the internet and present it within that application without the user knowing that this kind of connection has been made.

This warrants a quick glance back to the beginnings of mainframe computing some 40 years ago (in April 7, 1964, IBM unveiled it’s System/360 mainframe computer). The initial computer systems consisted of one central server computer and a number of ‘dumb’ terminals (client computers with limited processing power and no storage) networked together. All the applications and data were stored on the server. When the user at one of the client terminals started an application, it would actually execute on the server. The client computer simply formatted the server’s output for the screen.

It is hard to miss the analogy with today’s web sites. A computer with nothing but a web browser on it could, essentially, be a ‘dumb terminal’, running applications (complex web sites) executed on web servers.

MARKET FORCES:

COLLAPSE OF THE .COM BUBBLE / NECESSITY FOR CONTENT MANAGEMENT / OUTSOURCING

The development of internet was followed by an unprecedented economic boom.

The potential of instantaneously reaching millions of people, of bringing together buyers and sellers, or advertisers and audiences for a fraction of the cost previously spent on advertising, mail orders, customer relationship management and other areas seemed to overturn the established dogmas in business practices.

The investors saw the fast rise in valuation of internet related companies, and moved in faster and with less caution, funding any web-based idea that came their way.

The business model of the majority of the new companies relied on monopolizing their respective sectors by expanding their customer base through the use of internet. As a matter of fact, the company's survival depended on expanding the customer base as quickly as possible, even if this created huge losses. This approach was inherently flawed. In each sector there could be, at best, a single winner, and in most sectors, such as clothing for instance, not even a single winner could emerge through the sole use of the internet (almost all of the currently operating internet clothing stores have a physical presence as well, and have been around long before the dot-com boom).

By the year 2000, the dot-com boom was over. It was clear that most of the internet companies will never turn the profit. The investments shrunk, and so did the budgets for new web-based product developments.

FRAMEWORK MARKET FORCES

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Although the companies' investments in infrastructure development did not fall directly under the dot-com boom, the falling stock values required serious cuts in operating budgets. Since, by that time, internet technology was an integral part of day-to-day operations, there had to be a way to lower the costs of new web development projects.

Such web sites had to be:

1. Valuable: Provide clear return on investment to the company.
2. Comprehensive: Integrate with existing systems and/or data depositories and utilize the existing technology and resources.
3. Flexible: (If there is no existing infrastructure) based on standard, inexpensive, easy to obtain technology.
4. Manageable: Easily updated, modified and expanded internally with no additional cost.
5. Inexpensive. Have low production and technical maintenance costs.

The key – and as such, the most expensive – components of those projects were Information Architecture – focusing on business and user analysis, technology utilization, content management, and general project planning and scheduling – and production.

Since the analysis and planning had to be done locally, in close collaboration with the client, the production (programming) was the area that had to endure the largest cost reduction.

One way to do this was to re-use the existing systems and code to the highest extent. A lot of web design firms developed their own content management systems that they could, with minor modifications, provide to multiple clients.

The Open Source community didn't sit idle either. There are dozens of free pre-made content management systems for various platforms available for download. For any custom programming needs, there are thousands of free scripts only a mouse click away.

Another way to save on production costs was outsourcing.

Outsourcing was in no way a novel idea. Manufacturers have been moving their production facilities to countries providing low cost work force for years – just look at the 'made in' label on your favorite shirt, or car for that matter. However, this was the first time that high-level high-paying jobs were en masse moved abroad. The programming 'factories' were being established all over Russia, India, China, Eastern Europe, etc. Although outsourcing a project to one of these outposts was a gamble – often, the quality of work as well as the approach to doing business in these countries was not up to American standards – the savings were just too great to be ignored.

NEW OPTION:

APPLICATION AS VIABLE REPLACEMENT FOR WEB/MULTIMEDIA

In the past, software application development (non-web) has been prohibitively expensive. Software companies needed to sell large quantities of their programs to recuperate the development costs and make profit. This meant that the software needed to accommodate the widest possible area of need for the targeted audience. This was accomplished by adding 'features' to programs. The theory was that the more features program had, the easier it was



APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

FRAMEWORK CONCLUSIONS

to differentiate from the competitors, and the larger was the potential audience. This resulted in software that aimed to do everything for everyone, making it confusing, hard to learn and, essentially, not perfect for anyone in particular (just think of all the functions in Microsoft word you have never used – or needed for that matter.)

The lowering of programming costs, as a consequence of outsourcing, allows us to return the focus on actual user needs. Essentially, we can develop applications that solve very specific problems without worrying about the size of the user base – the low production cost justifies the development of a complete software solution even for a single user. It also allows us to offer a software solution as viable alternative to web or multimedia.

CONCLUSIONS:

UI STANDARDIZATION / INTERNET AS CONTENT DEPOSITORY / SEPARATION OF FORM AND CONTENT

Is web site the 'killer app' of the internet, or is it just a temporary phase in the development of this new technology?

Web sites, in their present form, were necessary to bring the power of distributed information to the user. A web site is, however, just a formatting device and, as such, can be easily replaced with another device – be it a desktop application or something altogether different.

With that in mind, is there a real benefit to web site interfaces as they are being designed today – different for

each site? Granted, there is the branding issue, but that benefits the owner and not the user of the site. There is also the possibility that, since many different interface options are being explored, a new better approach to interface design could emerge. This, however, was hardly the case. Even the most advanced Flash sites use the same interface elements used in operating systems long before the emergence of the internet: buttons, pulldowns, drag-and-drops, scroll bars. The difference is in how these elements look, and not how they work.

This reminds of the pre-Windows days of personal computing. DOS (Disk Operating System) was an operating system running on PC machines before the emergence of Windows. As a text-based operating system, DOS did not provide user interface elements to application developers. As a result, each program had it's own interface. Since there was no interface standard, the learning curve for the adoption of such programs was very steep. The emergence of a graphical operating system (Windows, or, even earlier MacOS) provided a common interface platform for all applications, which facilitated rapid growth in software development as well as general adoption of the computer as a viable every-day household tool. Although there were other reasons for such quick adoption, such as lower price of home computers, the ease of use of the new technology was a very important one.

In the latest version of Windows operating system – Windows XP – internet access is a component of the OS itself. Since Internet Explorer is an integral cross-system content formatting program, we can open a web page directly on the desktop, or in Microsoft Word, thus completely circumventing the 'web browser'.

The separation of content and form and further standardization of content description in web site development provides us with a choice in how we will access the information available on the web. It turns the internet into an all-encompassing database where the entry is the information itself and not the web site that contains it.



APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

FRAMEWORK CONCLUSIONS

New content formatting applications are not bound to the information available through a single web site. We can now pull in content from multiple sites based on true interest and not the brand loyalty. There are numerous programs already taking advantage of this concept: Sherlock, a multi-site search tool; Trillian, an instant messaging program allowing simultaneous messaging through multiple systems (ICQ, AIM, MSN, IRC, YAHOO); NewsGator, News RSS feed aggregator.

The problem is that, currently, separation and standardization only applies to text-based content. Therein lies the core of my thesis idea. A graph, with its discrete elements and highly structured organization, seems an obvious choice for a first step towards adding contextual properties to visual information.

REFERENCES:

BIRTH OF INTERNET:

<http://www.w3.org/People/Raggett/book4/ch02.html>

POPULARIZATION OF THE WEB:

<http://www.webopedia.com/TERM/T/Telnet.html>

TECHNOLOGICAL ADVANCES:

<http://www.users.cloud9.net/~bradmcc/xmlstuff.html>

<http://www.w3.org/TR/2000/REC-xml-20001006#sec-origin-goals>

<http://www.isgmlug.org/sgmlhelp/g-index.htm>

<http://www.oasis-open.org/cover/sgmlhist0.html>

<http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58>

INFORMATION ARCHITECTURE (IA):

http://www.mywiseowl.com/articles/Information_architecture

<http://builder.com.com/5100-31-5074224.html>

DIFFERENT APPROACH:

http://search390.techtarget.com/originalContent/0,289142,sid10_gci958841,00.html?track=NL-79&ad=480169

NEW OPTION:

<http://en.wikipedia.org/wiki/Dotcom>

PROJECT 1

WEB APPLICATION: CONTENT MANAGEMENT SYSTEM FOR EDUCATION

WHY I CHOSE THIS PROJECT

The advances in internet technology created a platform for easier exchange of information (syllabus, class notes, various class materials, homework) between students and teachers.

There are many available on-line Content Management Systems for education, both commercial (WebCT, Blackboard) as well as free (Moodle, Sakai, etc.). Most of these systems, however, suffer from the same problem that affects the standard desktop applications: overabundance of features. They attempt to do everything a teacher, student or an institution would need in terms of on-line course management. This makes for a very steep learning curve.

Commercial systems are expensive, complex, and often require a large investment in hardware and continuous costs for system maintenance. Also, since they are created to accommodate as wide audience as possible, they rarely address the particular needs of specialized institutions such as art schools.

Free systems, on the other hand, are often developed by individual schools, and are thus best suited for that school's needs. They also lack the support required to run such important systems.

But, overall, the most significant issue is the user interface. Performing any operation, whether on teacher or student side, often requires drilling through multiple levels of poorly organized menus.

My goal in doing this project was to develop a concept for a simple course management system that would sacrifice broadness for the sake of usability, ease of deployment, and reduction in deployment costs.

PROJECT 1: WEB APPLICATION: CONTENT MANAGEMENT SYSTEM FOR EDUCATION DEALING WITH A COMPLEX DATASET

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

DEALING WITH A COMPLEX DATASET

Note: This project was based on my own experience as an educator.

The program's key purpose is to facilitate data exchange between the teacher and the student. The first step was defining what kind of data should be exchanged and who is in charge of which piece of the data.

Teacher:

- Class overview information (course description, class rules, grading criteria, general readings)
- Syllabus and supplemental material related to each lecture (readings, notes, links)
- Assignments and supplemental material related to each assignment
- Comments on submitted homework

Student:

- Homework

System:

- Integration with a larger school-wide systems (class list, registered students, school web site)

The system had to provide basic direct communication (messaging) as well:

Teacher:

PROJECT 1: WEB APPLICATION: CONTENT MANAGEMENT SYSTEM FOR EDUCATION DEALING WITH A COMPLEX DATASET

- Global and individual communication (announcements and messages)

Student:

- Individual communication (messages)

System:

- Warnings - timeliness of assignments
- Warnings - changes in syllabus

After further analyzing the data, I found that it could also be sorted as:

Data not related to day-to-day teaching activities:

- Class overview information (course description, class rules, grading criteria, general readings)
- Integration with a larger school-wide systems (class list, registered students, school web site)
- Global and individual communication (announcements and messages)

Data directly related to day-to-day teaching activities:

- Syllabus and supplemental material related to each lecture (readings, notes, links)
- Assignments and supplemental material related to each assignment.
- Homework
- Comments on submitted homework
- Warnings - timeliness of assignments
- Warnings - changes in syllabus

PROJECT 1: WEB APPLICATION: CONTENT MANAGEMENT SYSTEM FOR EDUCATION INTEGRATION WITH EXISTING SYSTEMS

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Looking at the data related to day-to-day activities, it is clear that all the information revolves around the syllabus. The syllabus is, as a matter of fact, the organizational structure for all data. Beyond the class material, which naturally falls under the syllabus entry, each student project and comments regarding that project are directly related to a particular syllabus entry as well. All the warnings are calculated based on the current state of the syllabus.

The application structure had to focus on the syllabus as the core data organization system.

INTEGRATION WITH EXISTING SYSTEMS

Since usability is a primary concern, it is important that the application uses the data already available from the existing school database system – class list as well as the list of students registered for a particular class – and thus reduce the amount of setup work required from the teacher.

The application, on the other hand, provides up to date general class information (course description, class rules, grading criteria, general readings) as well as examples of student work (class portfolio) for use in school's printed materials and for the school's web site.

TECHNOLOGY

In order for the application to be as platform independent as possible, I had to expand my view of what constitutes a database.

Using any specific database software would require that software to be installed on the server. Instead, I decided to use standard directory structure as a metadatabase with regular files as database entries. Since in a CMS system like this we are not dealing with an excessive amount of data, this kind of organization would provide enough speed for the program to run smoothly.

For the programming language, PHP was the perfect choice. It is available for all platforms, and it's simplicity allows for easy modifications and additions to the program.

PHP works particularly well with Linux – a free Open Source operating system. Since Linux requires much less processing power than, for instance, Windows, the school could use any old PC to set up and run the application. This means that this CMS could be set up, tested and ran at no cost for the school.

Following these conclusions regarding data and technology, I developed an application structure model that provides the blueprint for actual coding of the application:

USABILITY AS DESIGN: DIFFERENT USER PRACTICES (TEACHER VS. STUDENT)

To create the user interface for the application, I first had to establish the needs for two distinctly different types of users: teachers and students.

The needs are listed in order of importance. Each item is given a value (high, medium or low) that represents the frequency of information access.

Teacher:

- (High) Quickly and easily review the student's work, and have a clear overview of changes (new uploads, messages) since the last login
- (Medium) Update and modify the class portfolio
- (Low) Make updates to the syllabus
- (Low) Make updates to the course description

Student:

- (High) See what the previous and next lecture/assignment is, as well as access all the relevant course material
- (High) Submit their homework and review teacher's comments of their work.
- (Low) Review the complete class syllabus
- (Low) Review the course description

PROJECT 1: WEB APPLICATION: CONTENT MANAGEMENT SYSTEM FOR EDUCATION USABILITY AS DESIGN: DIFFERENT USER PRACTICES (TEACHER VS. STUDENT)

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

This classification was my guide in organizing the functionality of the interface.

I avoided designing the interface. I approached it as an application design problem. The core issue was functionality. I used only the simplest possible forms to address the usability issues. Essentially, this was supposed to be a procedural prototype, and not a fully developed product.

Even though the internet technology has progressed far beyond the initial 'web brochure' stage, the ideas governing the interface design haven't changed as much. This is mainly a consequence of the fact that the web site is still perceived as a structure that has multiple levels – access to information still requires linear 'drilling' through menu choices. Each menu choice along the path is assumed to be associated with some content.

In reality, menu choices represent organizational structure. This is clear in pulldown menus – for example, a site that has 'products' option in it's primary menu. Rolling over this button produces a pulldown menu that contains types of products: hardware, software and services. Each one of those list the actual products. These products are the only actual content. Showing a separate page for any of the 'parent' choices is unnecessary since those choices are just sorting elements. Yet people, and for the most part web designers as well, still perceive a site like this to have multiple levels.

Contrary to the web, most application interfaces are flat. Generally, an application consists of a 'stage' – a main area where the action is performed – and a menu and palette system providing organization for all application's functions.

With the current level of technology available in web development, the same model can be applied to web sites.

The distinction is purely conceptual in so far that it doesn't affect the way the web programming works – choosing an option from the menu will still initiate opening of a new page.

For as long as the user is presented with a single 'environment' – consistent navigational system that, visually, presents the site structure as flat instead of multi-leveled, the chance for the user getting lost within the site structure is minimized and the focus moves from 'browsing' for a piece of information to accomplishing a required task.

THE APPLICATION

APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

STUDENT | OVERVIEW

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

└─ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

└─ semester.info (start/end; holidays)

└─ [classes]

| └─ [class.number]

| └─ class.info (name; day; teacher; ta; student list)

| └─ [syllabus]

| | └─ [syllabus.date]

| | └─ syllabus.date.flag (custom date; holiday; test)

| | └─ [syllabus.entry.type_uniqueNumber]

| | └─ syllabus.entry (definition; content; project end date)

| | └─ [material.type] (ftp accessible by teacher)

| | └─ document

| └─ [overview]

| | └─ uniqueNumber_overview.entry.name (definition; content)

| └─ [portfolio]

| └─ document

| └─ document_info (general info; access; comment)

└─ [users]

└─ [user.name] (teacher; ta; student)

└─ [class.number]

└─ user.log (last state)

└─ [messages] (sent: teacher to all students; student to teacher)

| └─ message.date_message.time

└─ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

└─ [documents] (ftp accessible by student)

| └─ document

└─ [comments] (from teacher)

└─ comment.date_comment.time

Schoolbook	Syllabus	Overview
Category	Description	
Course Description	Introductory course to basic concepts, methods, and procedures of Information Architecture focused on managing information complexity toward accessibility and understanding by targeted audience. The course addresses the issues of information structures and fundamental principles of organization in the context of both dynamic and traditional media.	
Process	The course content will be presented in the form of studio assignments, individual and class critiques, computer lab demonstrations and homework. While learning professional methods and procedures, the course will emphasize both conceptual and technical aspects of information architecture as they apply to interactive environments.	
Class Rules	The following are the simple rules and conditions you as a student will be bound by: Class attendance is mandatory; Students are expected to demonstrate systematic, week-to-week progress; Being late or leaving early on two occasions will count as an absence; Third absence will result in no credit; Students are responsible for any instruction or assignment missed because of absence; Incomplete for the course will be given only for medical or emergency reasons and only if 80% of coursework is completed.	
Grading Criteria	Individual project grades are based on the following criteria: concept + creativity = 40%; process + class participation + meeting deadlines = 30%; production + presentation = 30%; Good work habits and improvement over the term are expected.	
Readings	Information Architecture for the World Wide Web. Rosenfeld, Louis; Morville, Peter. O'Reilly, 1998. Don't Make Me Think. A Common Sense Approach to Web Usability. Krug, Steve. New Riders, 2000. Designing Web Usability. Nielsen, Jakob. New Riders, 2000. Information Anxiety 2. Wurman, Richard, Saul. QUE, 2000. Information Architects. Wurman, Richard, Saul. Graphis Publications, 1996.	

01

02

01 | The entry name, content and associated links are read from the file named uniqueNumber_overview.entry.name where the uniqueNumber defines entry's position in the list.

02 | If an overview.entry.item has an associated link, it will become clickable and connect the student to the remote location. In this case, it could link directly to an on-line book seller.

NOTE | The required overview entries are not highlighted in the student area because that is not important to a student.

STUDENT | SYLLABUS

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

___ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

___ semester.info (start/end; holidays)

[classes]

| ___ [class.number]

| ___ class.info (name; day; teacher; ta; student list)

| ___ [syllabus]

| ___ [syllabus.date]

| ___ syllabus.date.flag (custom date; holiday; test)

| ___ [syllabus.entry.type_uniqueNumber]

| ___ syllabus.entry (definition; content; project end date)

| ___ [material.type] (ftp accessible by teacher)

| ___ document

| ___ [overview]

| ___ uniqueNumber_overview.entry.name (definition; content)

| ___ [portfolio]

| ___ document

| ___ document_info (general info; access; comment)

[users]

___ [user.name] (teacher; ta; student)

___ [class.number]

___ user.log (last state)

___ [messages] (sent: teacher to all students; student to teacher)

| ___ message.date_message.time

___ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

___ [documents] (ftp accessible by student)

| ___ document

___ [comments] (from teacher)

| ___ comment.date_comment.time

The screenshot shows a web interface for a syllabus. At the top, there's a navigation bar with 'Schoolwork', 'Syllabus', and 'Overview' tabs. Below is a table with columns for '#', 'Date', 'Description', and 'Material'. The table contains several rows of lecture and assignment entries. Annotations 01 through 06 are placed around the table to highlight specific features: 01 points to the 'Material' column, 02 points to the 'Date' column, 03 points to the 'Description' column, 04 points to the 'Assignment' rows, 05 points to the 'Date' column, and 06 points to the 'Material' column.

#	Date	Description	Material
01	09.10.01.	Lecture: Introduction to Information Architecture. Definition and placement.	3 0 4
01	09.10.01.	Assignment: How things work - sequential information.	2 1 1
02	09.17.01.	Lecture: Algorithmic thinking. How things work. Linear representation of action. Static Information Design issues.	3 0 4
01		Assignment: How things work - sequential information.	
03	09.24.01.	Lecture: Recognizing and reflecting patterns of meaning and logical structures in information.	3 0 4
01		Assignment: How things work - sequential information.	
04	10.01.01.	Lecture: User expectation and audience definition. Introduction to other disciplines that help information organization. Demographic and Ethnographic considerations.	3 0 4
01		Assignment: How things work - sequential information.	
02	10.01.01.	Assignment: Information in motion.	2 1 1
	10.08.01.	Columbus Day	
05	10.15.01.	Lecture: Technology and it's place in information architecture. Prototyping tools. Dynamic Information Design issues.	3 0 4
01	10.15.01.	Assignment: How things work - sequential information.	
02		Assignment: Information in motion.	
06	10.22.01.	Lecture: Review	0 0 0
02		Assignment: Information in motion.	

01 | Content of the syllabus.entry. A text description of the lecture, project, or other syllabus.entry object.

02 | syllabus.entry.type.

03 | syllabus.date.

04 | Warning related to the appropriate date. Created by the information stored in syllabus.date.flag. Depending on the type of flag (reschedule, holiday, etc.) it will either add an icon to the <!> column or modify the formatting of the whole row.

05 | Start/end date warnings. For better readability of the list, for entry objects that have start/end date, those dates are color-coded and re-enforced with 'play' and 'stop' icons.

06 | Material pop up. It allows students to download material uploaded by the teacher (handouts, worksheets, links, etc.). Student chooses type of material, then the document from the list for download or browser viewing. A script analyzes the document and reformats it (or uses appropriate plug-in) for web browser presentation. The numbers show number of documents of each respective type.

NOTE | Since the student can't change the syllabus, the only active links on the page are ones for class material. The student can use the syllabus to download material for entries not shown in the 'Schoolwork' area.

STUDENT | SCHOOLWORK

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

___ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

___ semester.info (start/end; holidays)

[classes]

| ___ [class.number]

| | ___ class.info (name; day; teacher; ta; student list)

| | ___ [syllabus]

| | | ___ [syllabus.date]

| | | ___ syllabus.date.flag (custom date; holiday; test)

| | | | ___ [syllabus.entry.type_uniqueNumber]

| | | | ___ syllabus.entry (definition; content; project end date)

| | | | ___ [material.type] (ftp accessible by teacher)

| | | | | ___ document

| | | | ___ [overview]

| | | | | ___ uniqueNumber_overview.entry.name (definition; content)

| | | | ___ [portfolio]

| | | | | ___ document

| | | | | ___ document_info (general info; access; comment)

[users]

| | ___ [user.name] (teacher; ta; student)

| | | ___ [class.number]

| | | | ___ user.log (last state)

| | | | ___ [messages] (sent: teacher to all students; student to teacher)

| | | | | ___ message.date_message.time

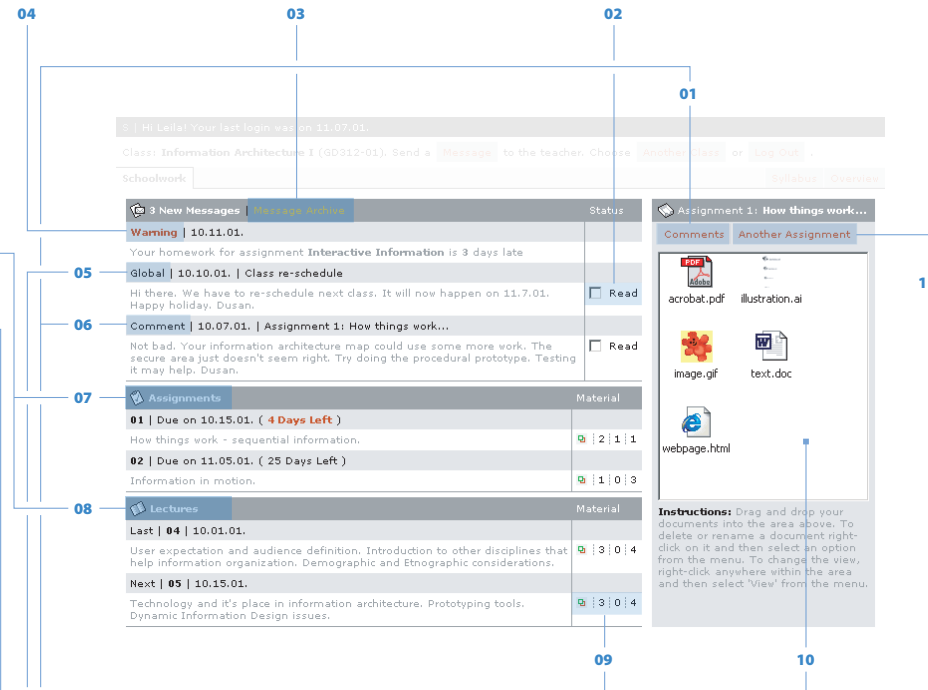
| | | | | ___ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

| | | | | ___ [documents] (ftp accessible by student)

| | | | | | ___ document

| | | | | ___ [comments] (from teacher)

| | | | | | ___ comment.date_comment.time



01 | Teacher's comments associated with currently selected assignment. Opens a pop up window with a list of comments.

02 | The message remains in the 'New Messages' area until 'Read' is checked off. Then it moves to the 'Message Archive'. Information about new and read messages is written to the user.log.

03 | List of all messages (without assignment comments) received in the current semester. The list is sorted by type and/or date.

04 | 'Warning' message type. This type of message is created automatically by the system when there is a problem with an assignment. The message cannot be moved to the archive and will be repeated until the problem is corrected.

05 | 'Global' message type. This is a message sent by a teacher to the whole class. It could be a reschedule, a reminder, etc. This message can be moved to the archive.

06 | 'Comment' message type. The subject of this message is the assignment the comment is attached to. When 'Read' is checked, it moves to the 'Comments' area linked to the selected assignment.

07 | Lists all currently active assignments. If the time to the due date is less than a week it shows a warning.

08 | Since students will access this tool between and (hopefully) not during the classes, it lists the last and the next lecture.

09 | Material pop-up. It allows students to download material relevant to the current class or assignment. Student chooses type of material, then the document from the list for download or browser viewing. A script analyzes the document and reformats it (or uses appropriate plug-in) for web browser presentation. The numbers show number of documents of each respective type.

10 | Assignment drop box. Students can upload their assignments by dragging and dropping their work into this area. The teacher can see documents as soon as they are uploaded.

11 | Menu that lists all projects up to the current date. Choosing another project will change current drop box and comments to its own.

TEACHER | OVERVIEW

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

___ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

___ semester.info (start/end; holidays)

___ [classes]

| ___ [class.number]

| ___ class.info (name; day; teacher; ta; student list)

| ___ [syllabus]

| | ___ [syllabus.date]

| | ___ syllabus.date.flag (custom date; holiday; test)

| | ___ [syllabus.entry.type_uniqueNumber]

| | ___ syllabus.entry (definition; content; project end date)

| | ___ [material.type] (ftp accessible by teacher)

| | ___ document

| ___ [overview]

| | ___ uniqueNumber_overview.entry.name (definition; content)

| ___ [portfolio]

| ___ document

| ___ document_info (general info; access; comment)

___ [users]

___ [user.name] (teacher; ta; student)

___ [class.number]

___ user.log (last state)

___ [messages] (sent: teacher to all students; student to teacher)

| ___ message.date_message.time

___ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

___ [documents] (ftp accessible by student)

| ___ document

___ [comments] (from teacher)

___ comment.date_comment.time

Nav	Category	Description
+		
•	Course Description	Introductory course to basic concepts, methods, and procedures of Information Architecture focused on managing information complexity toward accessibility and understanding by targeted audience. The course addresses the issues of information structures and fundamental principles of organization in the context of both dynamic and traditional media.
+		
•	Process	The course content will be presented in the form of studio assignments, individual and class critiques, computer lab demonstrations and homework. While learning professional methods and procedures, the course will emphasize both conceptual and technical aspects of information architecture as they apply to interactive environments.
+		
•	Class Rules	The following are the simple rules and conditions you as a student will be bound by: Class attendance is mandatory; Students are expected to demonstrate systematic, week-to-week progress; Being late or leaving early on two occasions will count as an absence; Third absence will result in no credit; Students are responsible for any instruction or assignment missed because of absence; Incomplete for the course will be given only for medical or emergency reasons and only if 80% of coursework is completed.
+		
•	Grading Criteria	Individual project grades are based on the following criteria: concept + creativity = 40%; process + class participation + meeting deadlines = 30%; production + presentation = 30%; Good work habits and improvement over the term are expected.
+		
•	Readings	Rosenfeld, Louis; Morville, Peter. Information Architecture for the World Wide Web. O'Reilly, 1996. Krug, Steve. Don't Make Me Think. A Common Sense Approach to Web Usability. New Riders, 2000. Nielsen, Jakob. Designing Web Usability. New Riders, 2000.

01

02

03

04

01 | Add new overview.entry object. It will open a text form pop up where you can enter the name and content of the new entry. Entry content consists of one or more items. A link can be associated with any item within an entry.

02 | Move or Delete object buttons. Move buttons will move the overview.entry to the next or previous position in the list. Delete button would remove the appropriate entry.

03 | The entry name, content and associated links are saved to the file named uniqueNumber_overview.entry.name where the uniqueNumber defines entry's position in the list.

04 | The institution can define required overview entries. In this case 'Course Description' is required because the school's web site draws its course information directly from it, and 'Readings' is required because the list of books goes directly to the bookstore for ordering or to the on-line seller for discounts. The required entries are color coded.

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

___ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

___ semester.info (start/end; holidays)

[classes]

| ___ [class.number]

| ___ class.info (name; day; teacher; ta; student list)

| ___ [syllabus]

| | ___ [syllabus.date]

| | ___ syllabus.date.flag (custom date; holiday; test)

| | ___ [syllabus.entry.type_uniqueNumber]

| | ___ syllabus.entry (definition; content; project end date)

| | ___ [material.type] (ftp accessible by teacher)

| | ___ document

| ___ [overview]

| | ___ uniqueNumber_overview.entry.name (definition; content)

| ___ [portfolio]

| ___ document

| ___ document_info (general info; access; comment)

[users]

___ [user.name] (teacher; ta; student)

___ [class.number]

___ user.log (last state)

___ [messages] (sent: teacher to all students; student to teacher)

| ___ message.date_message.time

___ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

| ___ [documents] (ftp accessible by student)

| ___ document

| ___ [comments] (from teacher)

| ___ comment.date_comment.time

Nav	#	Date	Description	Material
+	01	09.10.01.	Lecture: Introduction to Information Architecture. Definition and placement.	3 0 4
+	01	09.10.01.	Assignment: How things work - sequential information.	2 1 1
+	02	09.17.01.	Lecture: Algorithmic thinking. How things work. Linear representation of action. Static Information Design issues.	3 0 4
+	01		Assignment: How things work - sequential information.	
+	03	09.24.01.	Lecture: Recognizing and reflecting patterns of meaning and logical structures in information.	3 0 4
+	01		Assignment: How things work - sequential information.	
+	04	10.01.01.	Lecture: User expectation and audience definition. Introduction to other disciplines that help information organization. Demographic and Ethnographic considerations.	3 0 4
+	01		Assignment: How things work - sequential information.	
+	02	10.01.01.	Assignment: Information in motion.	2 1 1
+		10.08.01.	Columbus Day	
+	05	10.15.01.	Lecture: Technology and it's place in information architecture. Prototyping tools. Dynamic Information Design issues.	3 0 4
+	01	10.15.01.	Assignment: How things work - sequential information.	

01 | Content of the syllabus.entry. A text description of the lecture, project, or other syllabus.entry object.

02 | syllabus.entry.type. Clicking on this will open a text form pop up where you can enter/edit syllabus.entry.content and/or change the definition of that object.

03 | syllabus.date. Since there can be multiple syllabus.entry objects with the same date, the date is not a part of entry object - the entry object receives the date property from the directory it resides in. Also, using a date as directory name allows for easy sorting directly from the directory listing.

04 | Add new syllabus.entry object. It will open a pop-up similar to 02 where you can enter new syllabus.entry.content, choose the type of that object or create a new type (in which case you will have to define properties of this new object).

05 | Warning related to the appropriate date. Created by the information stored in syllabus.date.flag. Depending on the type of flag (reschedule, holiday, etc.) it will either add an icon to the <!> column or modify the formatting of the whole row.

06 | Move or Delete object buttons. Move buttons will move the syllabus.entry to the next or previous scheduled day, and push the syllabus.entry of the same type that previously occupied that date into it's old date. Delete button would remove the appropriate entry. If the entry has a begin/end date, all would be removed.

07 | Start/end date warnings. For better readability of the list, for entry objects that have start/end date, those dates are color-coded and re-enforced with 'play' and 'stop' icons.

08 | Material pop-up. It allows teachers to define types of material that will be available for their class (i.e. handouts, worksheets, links, etc.) and then upload documents to the server. The pop up would operate like students' assignment drop box. Teachers would choose the type of material (or new type), and then drag and drop documents into the box. The numbers show number of documents of each respective type.

TEACHER | STUDENT WORK

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

└─ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

└─ semester.info (start/end; holidays)

└─ [classes]

| └─ [class.number]

| └─ class.info (name; day; teacher; ta; student list)

| └─ [syllabus]

| | └─ [syllabus.date]

| | └─ syllabus.date.flag (custom date; holiday; test)

| | └─ [syllabus.entry.type_uniqueNumber]

| | └─ syllabus.entry (definition; content; project end date)

| | └─ [material.type] (ftp accessible by teacher)

| | └─ document

| └─ [overview]

| └─ uniqueNumber_overview.entry.name (definition; content)

| └─ [portfolio]

| └─ document

| └─ document_info (general info; access; comment)

└─ [users]

└─ [user.name] (teacher; ta; student)

| └─ [class.number]

| └─ user.log (last state)

└─ [messages] (sent: teacher to all students; student to teacher)

| └─ message.date_message.time

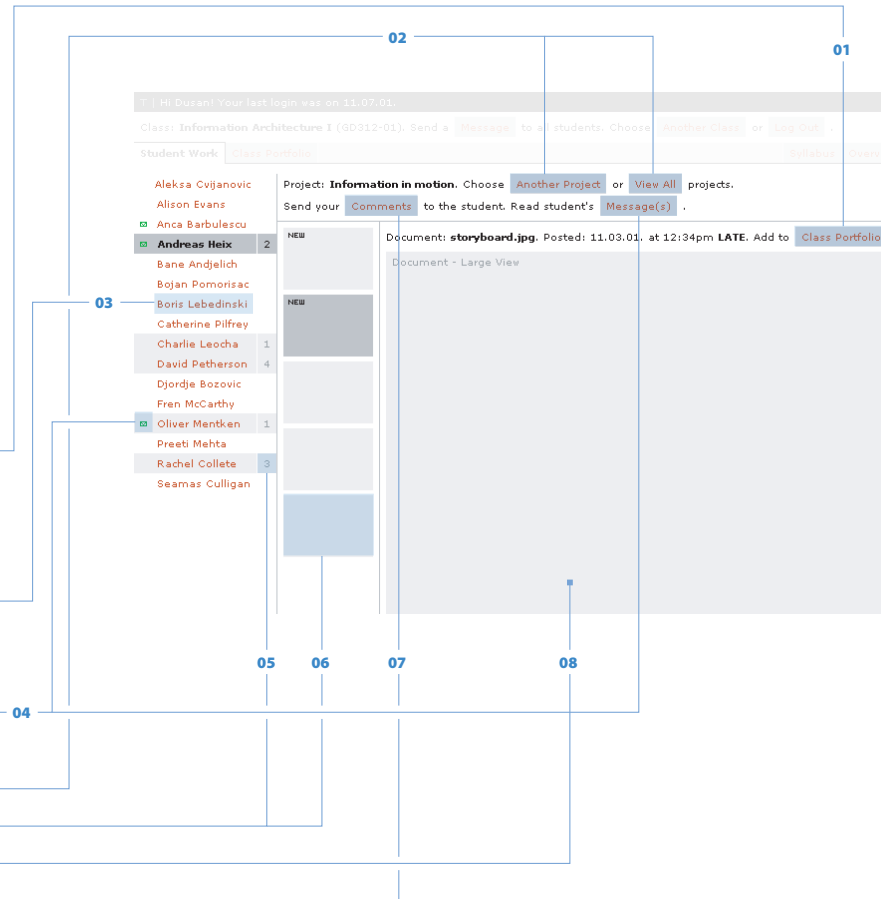
└─ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

└─ [documents] (ftp accessible by student)

| └─ document

└─ [comments] (from teacher)

| └─ comment.date_comment.time



01 | Copy currently selected document to [portfolio] directory. Create document_info file with document information (project, date, student name) in the same directory.

02 | Menu that lists all projects up to the current date. Choosing 'All Projects' shows all documents. Available projects are read from [user.name] / [class.number] directory.

03 | Student names are read from the class.info file. Location variables are created from these names, so the program can make the student > directory association.

04 | 'New Messages' warning. Calculated by comparing user.log with content of the [messages] directory. The 'Message(s)' button opens a pop-up with a list of student's messages.

05 | 'New Uploaded Documents' warning with number of new documents uploaded. Calculated by comparing user.log with content of the [documents] directories for all projects up to the current date.

06 | List of thumbnails of documents in currently selected project's directory. A script analyzes documents and creates appropriate thumbnails.

07 | Mail form pop up. The message is saved in [comments] directory as a text document. Since [comments] directory is located within the project directory, they are always associated.

08 | Large view of a chosen document. A script analyzes the document and reformats it (or uses appropriate plug-in) for web browser presentation.

[folder] | single file/folder | multiple files/folders | optional | (note) | uploaded files/links

[global]

└─ (scripts; object definitions; visual settings; passwords; etc.)

[semester]

└─ semester.info (start/end; holidays)

[classes]

| └─ [class.number]

| └─ class.info (name; day; teacher; ta; student list)

| └─ [syllabus]

| | └─ [syllabus.date]

| | └─ syllabus.date.flag (custom date; holiday; test)

| | └─ [syllabus.entry.type_uniqueNumber]

| | └─ syllabus.entry (definition; content; project end date)

| | └─ [material.type] (ftp accessible by teacher)

| | └─ document

| └─ [overview]

| └─ uniqueNumber_overview.entry.name (definition; content)

| └─ [portfolio]

| └─ document

| └─ document_info (general info; access; comment)

[users]

└─ [user.name] (teacher; ta; student)

└─ [class.number]

└─ user.log (last state)

└─ [messages] (sent: teacher to all students; student to teacher)

| └─ message.date_message.time

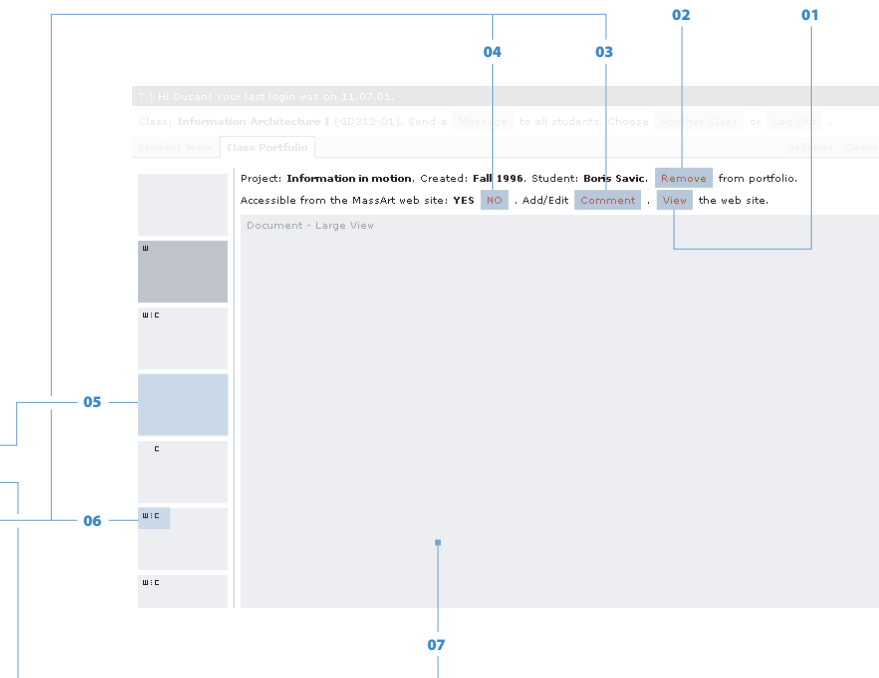
└─ [syllabus.date_syllabus.entry.type_uniqueNumber] (student only)

└─ [documents] (ftp accessible by student)

| └─ document

└─ [comments] (from teacher)

└─ comment.date_comment.time



01 | Link to the institution's public web site's class page.

02 | Physically delete the document and it's associated _info file from the [portfolio] directory.

03 | A text form pop up. Custom text message associated with the portfolio document. This message will appear on the web site as an explanation of the document. Written in document_info file.

04 | Toggles document visibility on the web site. State is written in document_info file.

05 | A list of thumbnails of documents in [portfolio] directory. A script analyzes documents and creates appropriate thumbnails.

06 | Mark showing if that document is accessible from the web site and/or has a comment associated with it. It would be embedded into the thumbnail itself by the script that creates it.

07 | Large view of a chosen document. A script analyzes the document and reformats it (or uses appropriate plug-in) for web browser presentation.

CONCLUSIONS

I did not intend to 'design' the user interface. I used the desktop application interface analogy and focused primarily on the information. As a consequence, simple visual choices I made to promote the usability resulted in a good looking, as well as functional, product.

This has re-enforced my belief in usability as the best branding device for on- or off-line application interfaces.

REFERENCES:

WHY I CHOSE THIS PROJECT:

<http://www.webct.com>

<http://www.blackboard.com/>

<http://www.opensourcecms.com>

PROJECT 2 (THESIS)

GRAPHING APPLICATION FOR MOLECULAR BIOLOGY

- APPENDIX
- PROJECT 3
- PROJECT 2
- PROJECT 1
- FRAMEWORK
- EXPERIENCE
- FOREWORD
- ABSTRACT
- ACKNOWLEDGEMENTS

INITIAL IDEA: GRAPH AS DATA

There are two distinct problems that needed to be resolved in order to successfully execute the project. The first deals with the graph content: developing the platform that would allow the scientific information to be embedded into the individual graph elements as well as provide for their visual formatting. The second deals with the behavior of the application itself: creating the user interface that would facilitate easy generation of such content-aware graphs.

A graph produced using a standard design application loses its content-related information. Essentially, the graph becomes an 'image' consisting only of visual elements. Such a graph, in its existing visual form only, can be distributed, printed, posted on the web, but the contextual properties of its content are irreversibly lost.



PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY

INITIAL IDEA: GRAPH AS DATA

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Separating content and form, or, in this case, first including the content in the graph generation process and then separating it from the form, would produce obvious benefits: Such graph would be content-searchable; Such graphs could be easily re-formatted to suit various presentation forms



The solution is obvious as well, and is a direct consequence of my web development work: use a form of markup language to describe the scientific properties of individual elements, and then use a separate style sheet to describe their visual properties. This provides a perfect platform for generation of content-aware, form independent graphs.

XML, already an established standard for generation of industry-specific markup languages, was a clear choice for the task. In fact, there is a number of existing XML schemas for molecular biology data model (see reference at the end of this section).



Note: One of my ideas at the beginning of the project was to create a standard for visual representation of all possible graph elements. This would have completely removed the need for formatting by the scientist generating the graph. The size of the field of molecular biology made this task impossible to accomplish. Also, as the concept of separation of form and content developed – which made it possible to generate and distribute the visual style sheet independently from the graph content – I realized that the visual standards, if at all necessary, could be created by the people in the field much better suited for development of such standards.

EXISTING SOLUTIONS

Although some graphing software, such as Microsoft Visio, already use a form of markup language in their native formats, the markup is used to describe the physical properties of the graph object – such as shape, size, text, color – and not its actual content.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DATASET / ANATOMY OF A GRAPH / SPECIFICS WITHIN INDIVIDUAL LABORATORIES

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

There are currently no available graphing applications tailored for scientific use. For a software company, the costs of developing such highly specialized application would be higher than the potential sales revenues. For the scientists programming their own tools, solving the visualization issue does not represent a priority. Most programs written by scientists attempt to solve actual scientific problems.

My graph-as-data concept, and the application implementing it, does not in any way attempt to affect the quality of the created material. It is not meant to be an educational or a scientific tool. Its only purpose is to provide a better way for storing and distribution of the generated material, as well as to provide the environment that makes the graph creation easier, faster and better suited to user's work process.

DATASET / ANATOMY OF A GRAPH / SPECIFICS WITHIN INDIVIDUAL LABORATORIES

Note: Data analysis as well as user analysis and testing were done in collaboration with the researchers at the Hansen Lab at Boston University, Boston, MA. This particular lab specializes in regulation of transcription (process of converting genetic information from DNA to RNA).

The events in molecular biology consist of chain reactions between various elements. Everything that happens has a clear, although not always known, cause and effect. For instance, a hormone interacts with the receptor protein on the cell membrane changing its physical properties (shape). The receptor protein then passes a phosphate group to another protein located next to it which activates it. This protein, in turn, passes the phosphate group to the next

one, and so on. All events are parts of continuous biological cycles. This is why a graph is one of the best methods for representing research findings.

Because of the vastness and the complexity of the field, individual labs are highly specialized. They deal with very specific issues, such as regulation of transcription in the case of the Hansen Lab, or even more localized events within these larger issues.

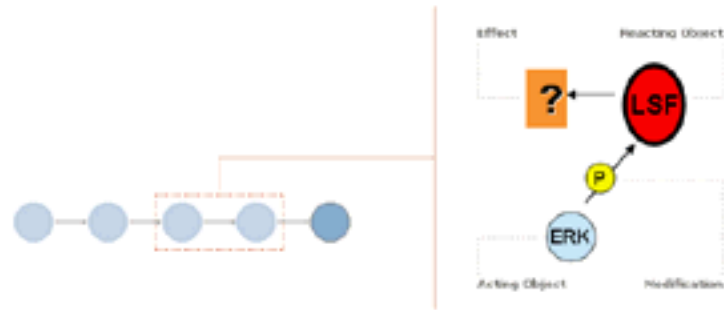
This chain-reaction nature of the events, as well as high level of specialization – very narrow data subset – make graphs a perfect medium for representation of scientific findings.



There is a standard 'template' for the representation of the interaction between two elements in a chain. The Acting Object interacts with the Reacting Object. In the course of interaction, it changes some properties of the Reacting Object (Modification). The Effect is, essentially, the next element in the chain or, to be more precise, the modification that the next element will undergo as a consequence of previous reactions.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY
 DATASET / ANATOMY OF A GRAPH / SPECIFICS WITHIN INDIVIDUAL LABORATORIES

APPENDIX
 PROJECT 3
 PROJECT 2
 PROJECT 1
 FRAMEWORK
 EXPERIENCE
 FOREWORD
 ABSTRACT
 ACKNOWLEDGEMENTS



Data analysis required defining the scientific properties attributed to the elements as well as to their interaction with the other elements.

ELEMENT PROPERTIES:

- Accession Number: The number under which the element is registered in the genbank database. For example, protein LSF has an accession number NP_005644. Entering the number would automatically insert all known information for other fields.
- Name: Element name (i.e. LSF for type of protein)
- Type:
 - Protein
 - DNA
 - Organelle
 - Cell (plant/animal/bacterial)
- Subcellular Location:
 - Membrane Bound
 - Soluble

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY
DATASET / ANATOMY OF A GRAPH / SPECIFICS WITHIN INDIVIDUAL LABORATORIES

Oligomerization State:	Monomer	
	Oligomer	<i>(a complex element assembled from multiple elements. It is necessary to include the number of component elements as well as their names.)</i>
	Homomer	
	Heteromer	
	Unknown/Irrelevant	
Function:	Description (to aide contextual search)	
INTERACTION PROPERTIES:		
Effect:	Activate	
	Deactivate	
	Unknown	
Modification:	Phosphorylation	
	Methylation	
	Acetylation	
	Fatty Acid Addition	
	Glycosylation	
	Oligomerization	
	None	
	Unknown/User Define	
Function:	Description (to aide contextual search)	

TARGET AUDIENCE / USER WORK PROCESS

Initially, in order to keep the application simple and effective I decided to narrow the target audience to advanced researchers (PhD candidates and above versus i.e. undergraduate students).

I made this choice because researchers at that level provide a high level of expertise and experience that would remove the need for the application to be involved in the content creation process. Because of the size and the complexity, as well as the ever changing and growing nature of the field, any attempt to create an application that would 'guide' the content development would be, at worst, futile, and, at best, require someone with much more knowledge of molecular biology than I have.

The initial analysis of the user behavior was very simple. I observed a researcher in the lab creating a graph. I found that:

- Graphics programs play no role in the actual development of the content for the graph. The graph is the consequence of research and is formed and conceptualized (with pen and paper) long before it reaches any graphing application.
- The step-by-step process of creating two chained elements was:
 1. Draw the container (box, circle, etc.) for the first element
 2. Type the name for the first element
 3. Play with the look of the container for the first element
 4. Draw the container for the next element
 5. Type the name for the next element

6. Play with the look of the container for the next element
 7. Draw the connecting line between the two elements
 8. Draw the container for the modifier
 9. Type the name of the modifier
 10. Play with the look of the container for the modifier
 11. Group and position the last element and the modifier (if necessary)
 12. Repeat steps 4 through 11
- The user would spend significant time formatting the visual properties of each element. Since researchers are neither graphic nor information designers, the visual formatting of elements was often detrimental to understanding the graph, as well as inconsistent across multiple similar elements.

The first finding re-enforced my initial choice in target audience.

The second made me re-think the process for actually building graphs in graphing applications. Since the graph represents a chain reaction, we could streamline the building process by asking the user to 'drop' the next element in the chain on top of the previous one, thus stating that the two should be connected (and drawing the required line between them). The modifier could be generated automatically from the embedded data within an element (since it is already defined in the 'Interaction Properties').

The third, although important, I completely omitted in the first iteration of the application. The way the user approached visual formatting should have had a direct influence the structure of the interface. It was clear that I needed to separate the process of placing actual elements on the page from the process of visual formatting of those elements.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Instead, i made an assumption that it was enough to make the visualization easier. The assumption was proven wrong after the first user testing.

DEVELOPMENT

Opposite page, following spread: Application, Version 1. Method for building graphs and interface elements.

The first version of the application included a drag-and-drop model for generation of the graph, as well as a method for inserting the scientific data for the elements.

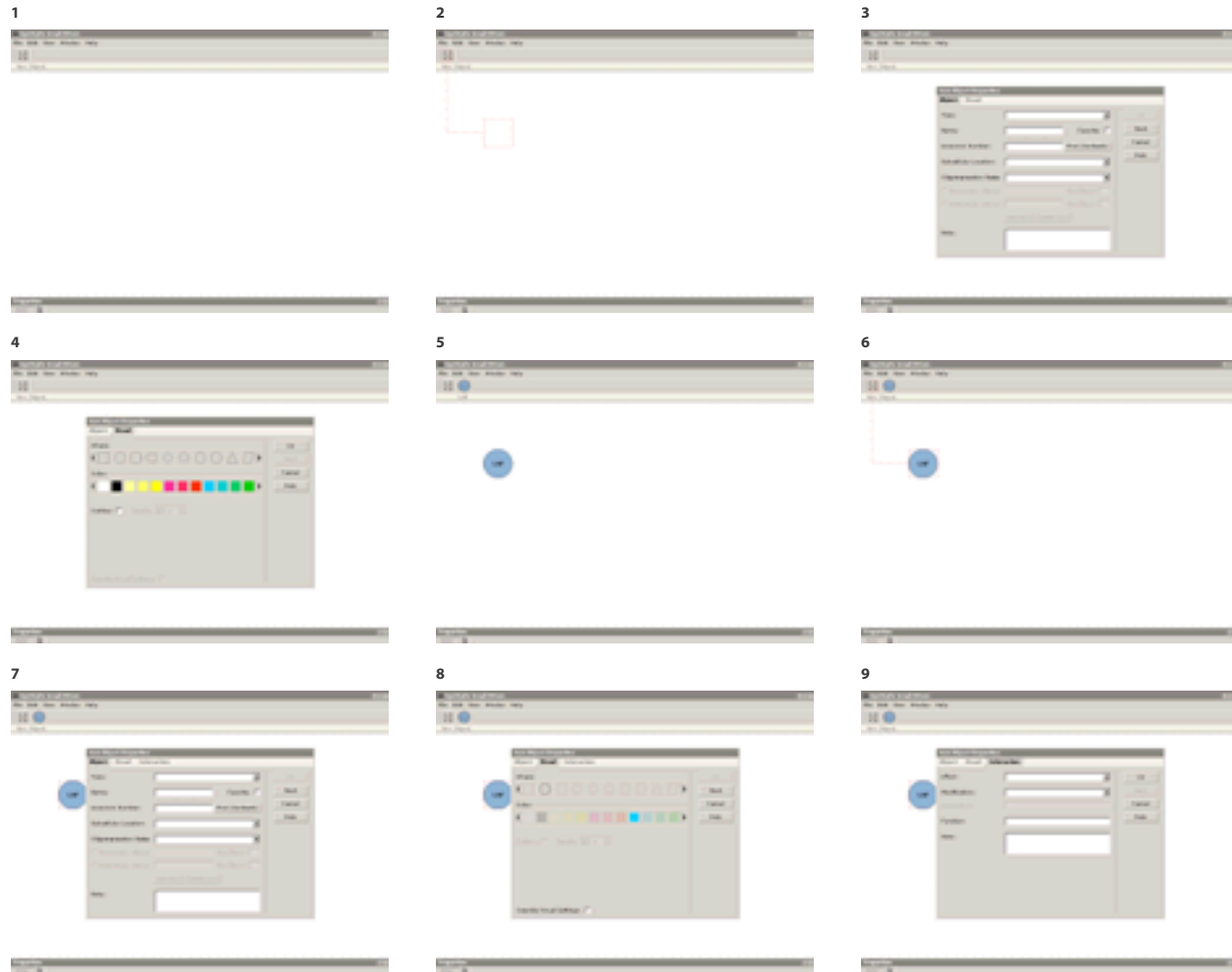
The visual properties of the elements were defined at the same time as the content properties (screens 3 and 4).

I also realized that it was necessary to be able to group multiple elements and then embed scientific data into the group (contextual grouping). This makes it possible to add 'titles' that would encompass multiple elements, as well as to define visual representation of such groups (screens 11, 12 and 13).

Since elements are often reused, I decided to add a lineup of all already defined elements (all screens – line of icons at the top of the page).

User testing proved this version of the application to be unsuccessful.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT



PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

10



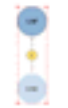
13



16



11



14



17



12

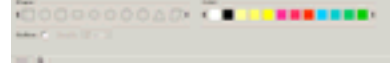


15



18





PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

Instead of making graph creation faster, the need for entering the information for each element (see Screen #??) made the whole process much more tedious.

Common remarks were:

“Why do i need to put in all this data when it’s already available.”

“Of course I don’t know the accession number by heart.”

“Can I skip all this and just make it look better.”

And the particularly problematic:

“I could do this faster in Illustrator.”

The benefit for the distribution of these content-aware graphs could not out-weight the extra effort needed for the graph creation. The concept needed a major overhaul.

The primary realization was the importance of the existing molecular biology databases. Instead of requiring the user to input all the information related to the element, it is safe to assume that this information is already present. The application should either contain the database with all the elements and their properties, or be able to connect to one of the existing databases and retrieve the required information in real time.

The user can be provided with a search functionality that would allow him to easily find the ‘next’ element, and then add it to the graph by dropping it on top of an existing one.

This brings us to visual formatting of individual elements. Looking back at the user behavior, it is clear that

the process of placing actual elements on the page and the process of visually formatting those elements should be separated.

Instead of requiring the user to format each element, the program should provide a general style editor. After completing the building of the graph content, the user can switch to the style editor, accept the default style (provided by the program) or modify it to suit his needs. The user should be able to save and load different styles independently from the graph content. By distributing separate style sheets, users could establish visual standards for graph representation where they were needed.

These modifications to the interface provide the following workflow for the user:

1. Type the name of the element
2. Drag the fully defined element on top of the existing one
3. Program automatically creates the connecting line and the modifier from the embedded data
4. Repeat steps 1 through 2

Since the graph content is fully developed before it gets to the 'design' stage (see User Analysis), this workflow provides quick and easy transfer of the graph to digital format.

At this point, though, I decided to push the idea of simplification of the graph building process as far as possible. I wanted to find the way to eliminate the need to type in the name of the next element.

This required another look at the information provided by the data attached to an element.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

The element properties, as well as its interaction options define which other elements it can connect to. Looking at the whole field of molecular biology, the number of possibilities is too high. However, since their research is highly focused, the individual labs deal with a significantly smaller, more manageable subset of elements.

Even more, multiple researchers within a single lab generate graphs using the same elements and, quite often, the whole parts of the chain. By keeping the log of all the graphs produced within a single lab, as the number of generated graphs grows, the possibility for predicting the next element in the chain based on the previous one gets higher.

This would, of course, require the application to expand on-line. Each workstation would have an installed copy of the software, and the centralized server would keep track of the generated graphs.

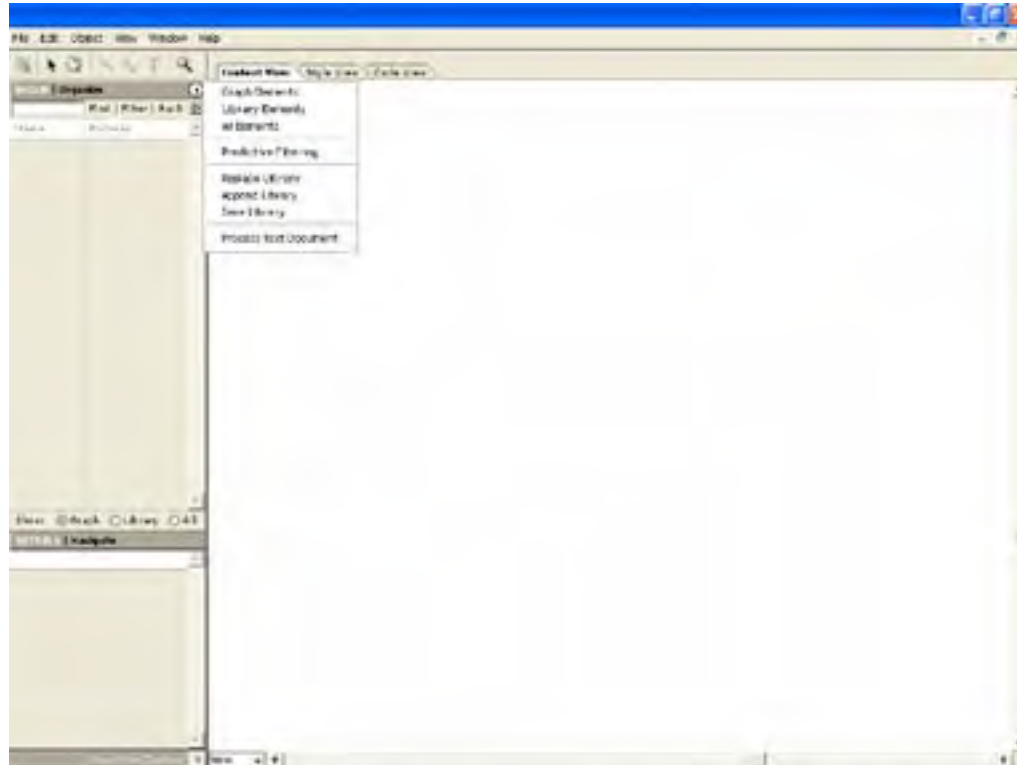
To push the idea even further, we could move the server collecting the graphs outside the individual lab. With the server connected to labs doing similar type of research all over the world, the amount of collected information would grow exponentially.

In terms of application behavior, the idea was for each chosen element to represent a search query for the next element in the chain. The user could then simply select one of the results from the list.

Opposite page, following four spreads: Application, Final version.

For the actual interface design, I chose the generic application look. Two primary components – graph building and graph formatting – were separated using tabs attached to the main stage. Another primary-level tab was added

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT



The user can start building the graph by dragging the 'new element' onto the stage, or by loading an element library and dragging in a pre-formatted element.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

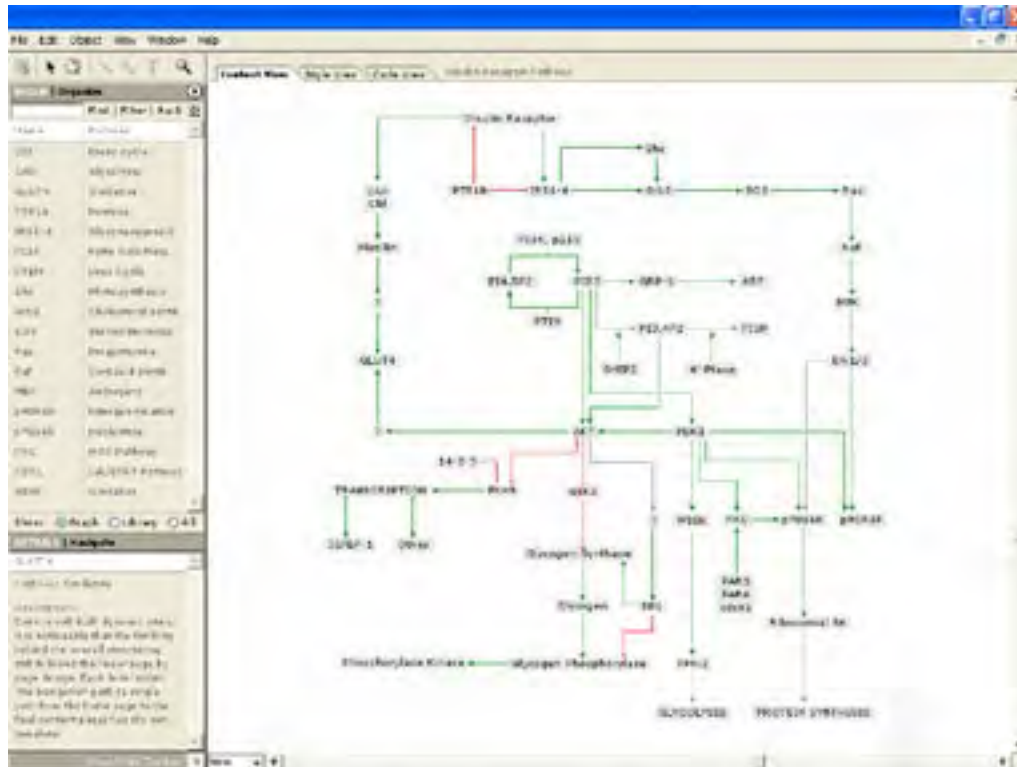
FRAMEWORK

EXPERIENCE

FOREWORD

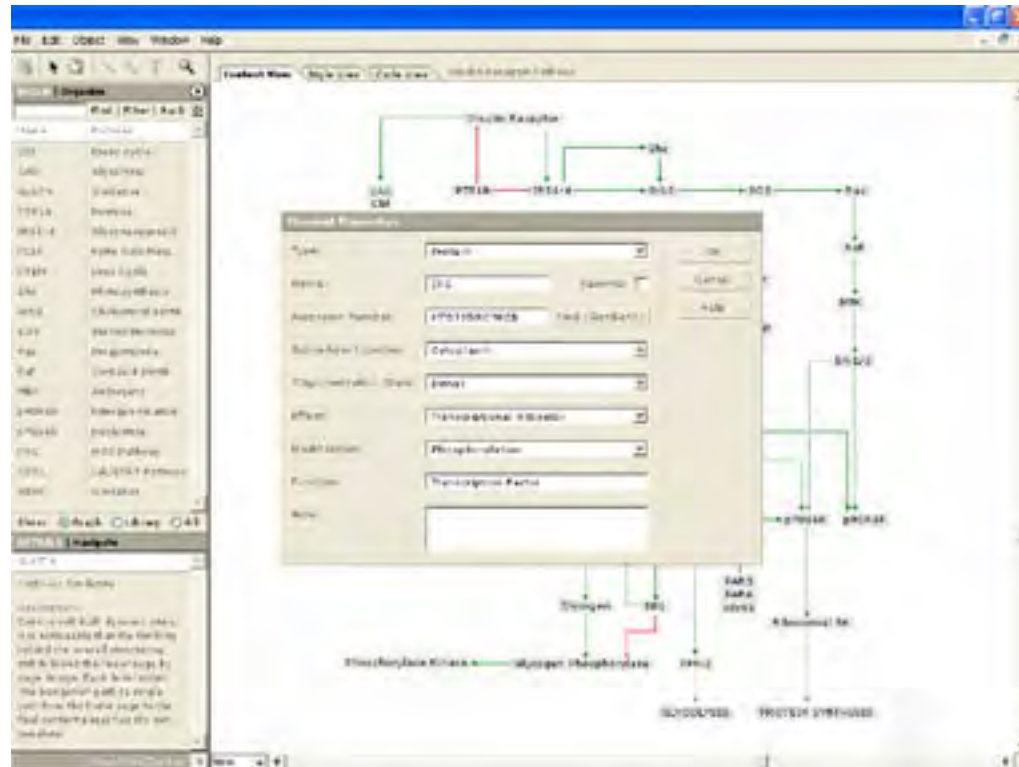
ABSTRACT

ACKNOWLEDGEMENTS



'Content View' shows the graph without visual formatting. 'Build' palette provides search results based on the currently selected element. User can add new 'links' to the graph by dragging the elements from the 'Build' list and dropping them on existing elements.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT



The 'Element Properties' menu contains the scientific data related to the element. It can be accessed by double clicking on an element.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

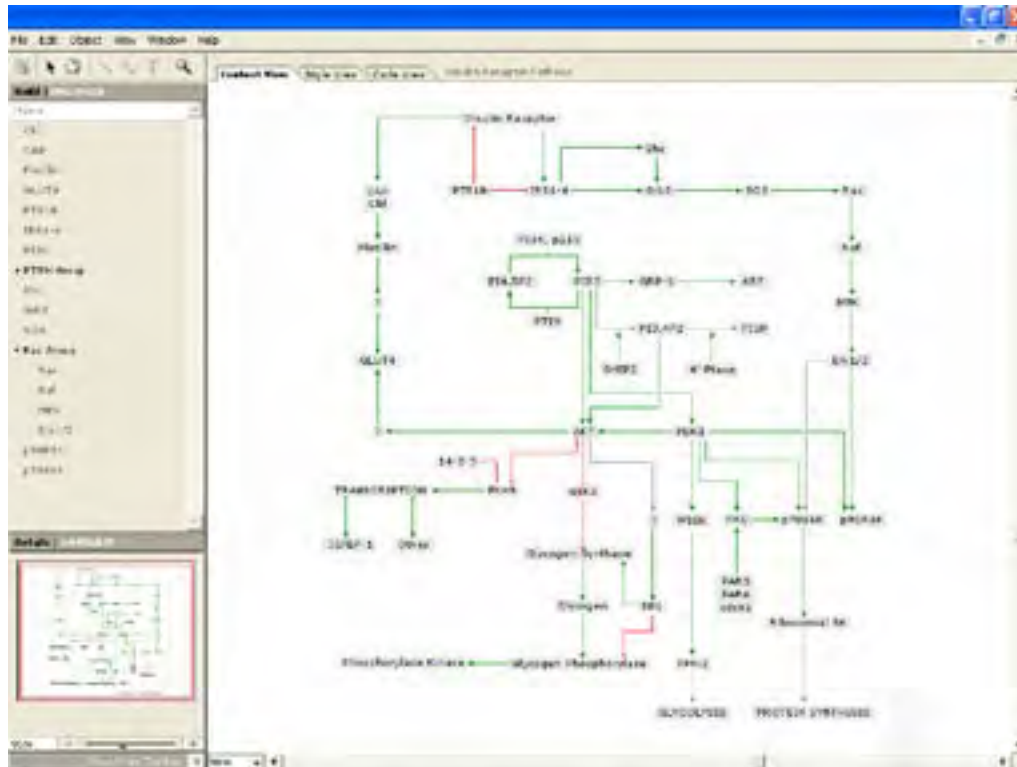
FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS



'Organize' palette shows the list of all used elements. The unfolded items represent complex elements built up of multiple single elements. 'Navigate' palette provides pan and zoom functionality - important when dealing with complex graphs.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

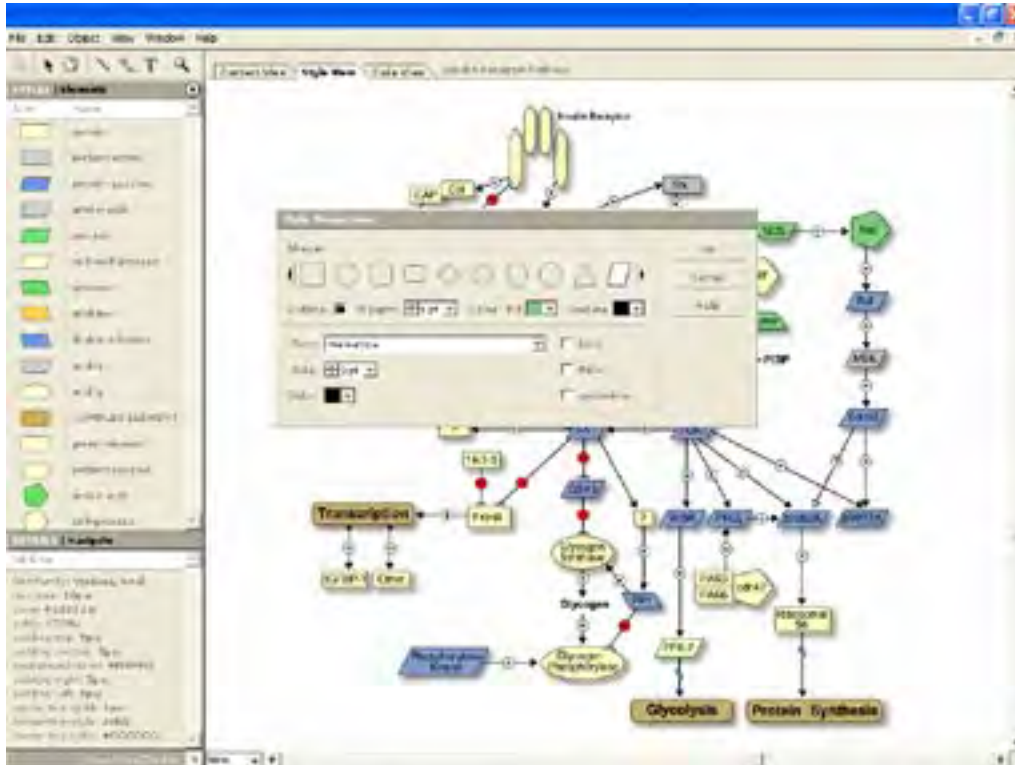
FRAMEWORK

EXPERIENCE

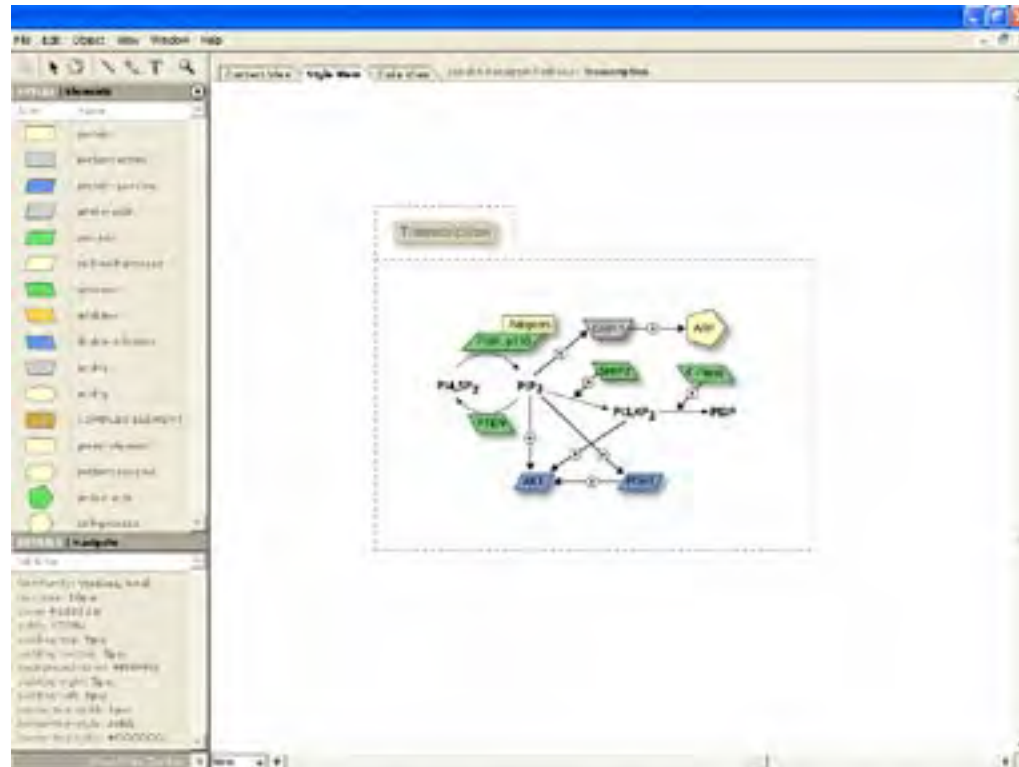
FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS



'Style Properties' menu contains the visual formatting related to the element or a group of elements. It can be accessed by double clicking on an element.



Double-clicking on a complex element brings the graph stored in it to the foreground. The 'bread crumbs' line next to the main tabs shows that we are now one level removed from the actual graph.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY DEVELOPMENT

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

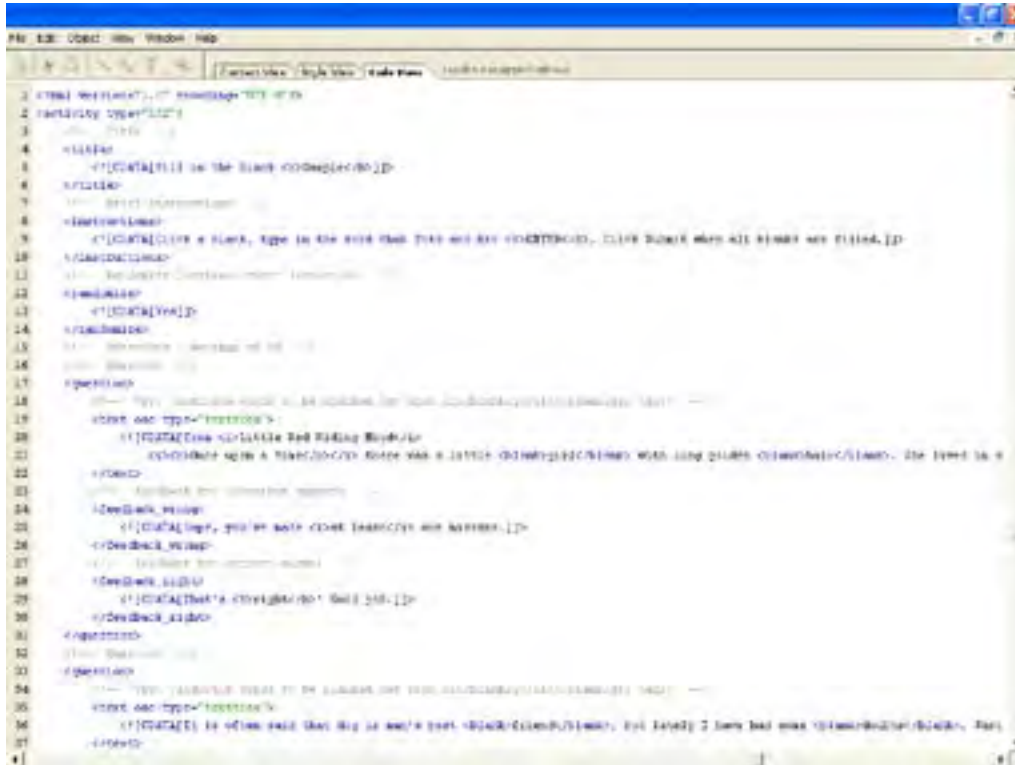
FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS



```
1 <!-- WebForm1.cs -->
2 <!-- WebForm1.cs -->
3
4 <!--
5 <!--
6 <!--
7 <!--
8 <!--
9 <!--
10 <!--
11 <!--
12 <!--
13 <!--
14 <!--
15 <!--
16 <!--
17 <!--
18 <!--
19 <!--
20 <!--
21 <!--
22 <!--
23 <!--
24 <!--
25 <!--
26 <!--
27 <!--
28 <!--
29 <!--
30 <!--
31 <!--
32 <!--
33 <!--
34 <!--
35 <!--
36 <!--
37 <!--
```

'Code View' provides direct access to the XML created by the application.

for direct access to the XML. These tabs provide different 'views' of the same content: contextual, visual and code. All the functions related to the primary components were organized using tabbed palettes on the side of the stage area.

Note: For the inspiration I looked at programs providing environments for editing of markup language documents. Two that most influenced the organization of interface elements were Macromedia Dreamweaver and Adobe Golive.

The new application was much more successful than the first version. Separation of design from the content as well as the removal of data input panels made the graph generation process quick and easy.

CONCLUSIONS

Information

Content does not begin and end within the graphing application. It is crucial to analyze how it is developed before it reaches the formatting state, as well as which forms it can take and how it can be used after it leaves the application.

In existing design applications, once the formatting is done, the information is limited to representational use. It can be printed, posted on the web, but its contextual properties are irreversibly lost. The true value of information, however, is not in its visual formatting. Although the formatting can make it easier to understand, the appropriate visual style depends on the user accessing the information.

PROJECT 2 (THESIS): GRAPHING APPLICATION FOR MOLECULAR BIOLOGY
CONCLUSIONS
NETWORKED APPLICATION

APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

The separation of content and formatting has been done with text-based content. A good example is Really Simple Syndication (RSS) – an XML format for syndicating news and the content of news-like sites. Anything that can be broken down into discrete items can be distributed via RSS.

Applying this model to graphs creates a multitude of possibilities for the distribution of visual content. The applications affecting different ‘states’ in the information life span – such as Desktop Publishing, Web Design, Database, and other programs – can act as style builders or content processors leaving the original information intact.

Networked Application

With the advancement of always-on connections, whether at work or at home, the line between web sites and desktop applications is becoming increasingly blurred.

Most of the existing desktop applications use the internet for the mundane tasks of product registration or updates. We can learn a lot from the functionality of web sites and apply this knowledge to the desktop/network hybrid software design.

In example, Amazon.com collects various information about user behavior. It then uses this information to personalize user experience. The more people use the web site, the more accurate the prediction for each user’s interests gets.

Connecting multiple instances of an application in a network and allowing them to share information about users’ actions provides a platform for predictive behavior of such applications.

Focus on user work process

Just as the molecular biologists use graphic design applications to format their results, people often use programs that are not created specifically with their fields in mind.

Applications offering customization based on the user experience (novice, mid-level or advanced) achieve customization by hiding menu options. Other applications, such as Macromedia Flash and Dreamweaver, offer customization based on the type of user (developer or designer). This is done through different organization of palettes on the screen, but the menu options remain the same.

These applications, although somewhat customizable, are still task- and not user-oriented. True user-oriented customization would require restructuring of both the menus and the general application organization to follow the work process of the user. The semantics used to describe menu options would have to be adjusted to the user as well.

This can be achieved without modifying the core programming of the application. What is necessary is researching all possible uses for the software, and then performing the in-depth analysis of each user group. The software could then offer much more meaningful customization based on the real needs and requirements of the user.

Moving the interface forward

Although my application achieved its goal of easy deployment of content-aware graphs, the interface used to build these graphs was not explored far enough. In designing the interface, I chose the generic method for graph generation: drag-and-drop.

The idea of the application predicting the next element in the chain opens up new possibilities for the interface. The repetitive nature of the graph-building task provides us with a possibility of integrating such interface into the 'stage' area of the application, thus streamlining the production process. I will further explore this idea in the next chapter.

REFERENCES:

INITIAL IDEA: GRAPH AS DATA:

<http://www.xml.com/pub/rg/Bioinformatics>

EXISTING SOLUTIONS:

<http://www.microsoft.com/office/visio/prodinfo/facts.mspx>

DATASET / ANATOMY OF A GRAPH / SPECIFICS WITHIN INDIVIDUAL LABORATORIES:

<http://www.ncbi.nlm.nih.gov/entrez/viewer.fcgi?db=protein&val=21361278>

<http://www.ncbi.nlm.nih.gov/>

CONCLUSIONS:

<http://www.webreference.com/authoring/languages/xml/rss/intro/>

<http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html>

PROJECT 3 (THESIS)

GRAPH CREATION TOOL

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

INITIAL IDEAS

The graph creation tool was envisioned as a component of the Project 2 application. Its aim was to streamline the graph creation process.

My initial assumption was that this tool should allow the user to build the whole graph without ever leaving the stage area. Once the tool was activated, the user would not have to return to standard application menus until the graph is finished.

After further analysis of this assumption, I realized that: The tool had to be located inside the stage area where it could directly facilitate the graph building; The tool had to be visually related to the graph

The application assumed each new element in the graph 'chain' to represent a search query for the next element. Consecutively, the tool should facilitate visual search result browsing.

An interactive tool – versus a standard application interface – allows for a wider range of user actions. These actions can be used to modify the tool behavior to provide personalized user experience and more accurate search results.

Since the tool is part of the Project 2 application, all the conclusions regarding the dataset and user definition remain the same: The target audience are advanced researches; The dataset is a focused subset of molecular biology data used within a single lab.

PROJECT 3 (THESIS): GRAPH CREATION TOOL
THE DESIGN PROBLEM
INSPIRATION

THE DESIGN PROBLEM

Inspiration

For inspiration, i looked at existing tools that focused on information organization and, in their visual form, mimicked the look of a graph.

One of the tools that matched the requirements was TheBrain created by TheBrain Technologies Corporation.

TheBrain is an easy-to-use system for organizing information. It enables users to link files, documents, and Web pages across applications and network boundaries. TheBrain illustrates how information is related and provides a visual context for documents and data.



Although this particular tool focused on representation and not creation of linked data, it provided a starting point for visual development of graph building tool.

Even though the quantity of information mapped onto TheBrain could be vast, the tool never ‘run out of space’ – following the links never took the user out of the boundaries of the tool. This was one of my concerns related to the graph building. I did not want the user to have to ‘leave’ the tool and use other interface elements – including the scrollbars. In TheBrain, clicking on a linked piece of data moves that piece to the center of the tool, and reorganizes the new linked elements around it. I used this method to control the positioning of the graph building tool.

The visual look of TheBrain, linked elements ‘fanned’ around the current selection, provided me with the visual organization for my tool. The results of the search based on the current element can fan around it.

Preliminary Design

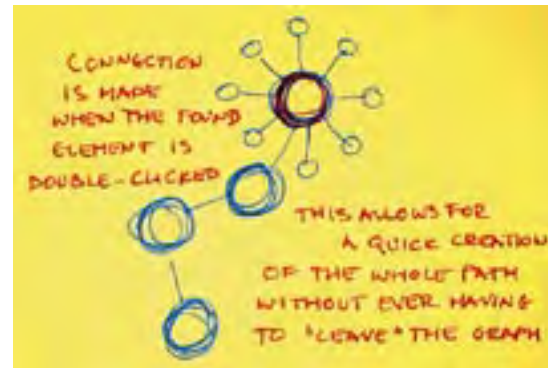
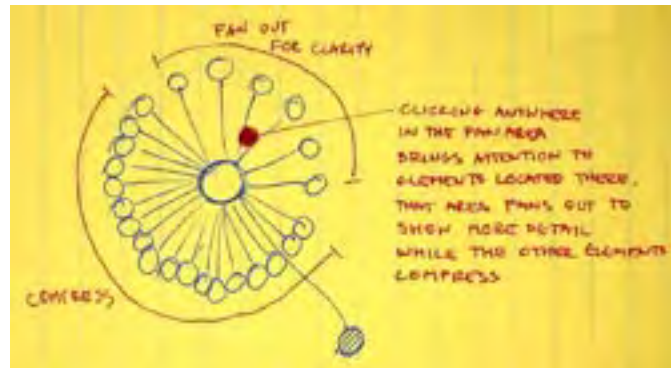
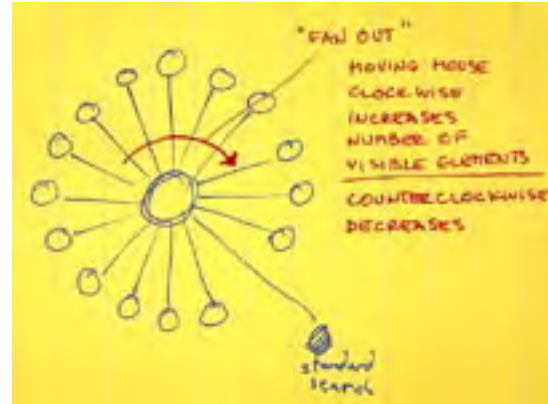
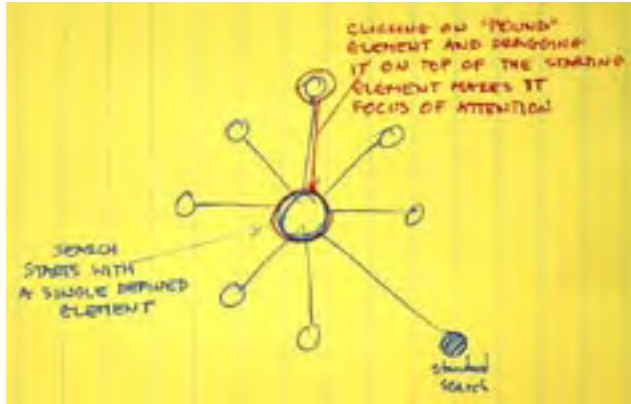
As a result, I did a sketch of how the tool could look and work:

Next page: Initial sketches of the greaph building tool

The sketches show basic ideas regarding the behavior of the tool.

PROJECT 3 (THESIS): GRAPH CREATION TOOL
THE DESIGN PROBLEM
PRELIMINARY DESIGN

APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS



Behavior

- After analyzing the actions envisioned in the initial sketch, I developed a new set of basic actions:
- Move the mouse around the current element:
 - Clockwise: increase the number of visible search results
 - Counterclockwise: decrease the number of visible search results
 - Hover within the 'fan' area:
 - Brings the attention to the result under the cursor – enlarges the result
 - Click within the 'fan' area:
 - 'Bookmark' the result
 - Click on the current element:
 - Reset the 'fan'
 - Drag the result on top of the current element:
 - Explore the path without committing to the result,
 - The 'fan' shows new search results based on the selected result
 - Drag the previously selected result back to the 'fan' area:
 - Return to the current element,
 - The 'fan' shows search results based on the current element
 - Double-click on the result:
 - Result becomes current element and the query for the new search
 - Double-click outside the tool:
 - Exit the tool

APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

PROJECT 3 (THESIS): GRAPH CREATION TOOL

THE DESIGN PROBLEM

BEHAVIOR

These basic actions provide the functions necessary for the graph building. However, the functionality of the tool could be pushed even further.

There are certain actions that could provide the tool with the information about the level of satisfaction of the user. The user frustrated with the search results could, for instance, grip the mouse tighter, put more pressure on the mouse button, or move the cursor on the screen faster. Although it would be interesting if we could collect the physical information from the mouse – and maybe this kind of input device would be an interesting thing to build – we can not. We can, however, measure the speed of the cursor.

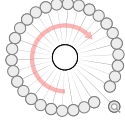
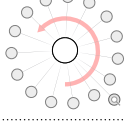
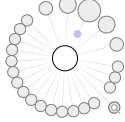
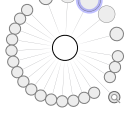
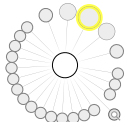
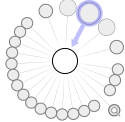
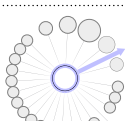
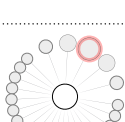
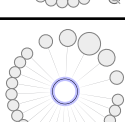
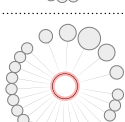
Assuming that the speed of the cursor indicates users dissatisfaction with the search results, we can add the new function to the tool:

- Rapid moving of the cursor around the current element:
 - Clockwise: Expand the result set / Replace the complete current set of results with a new one
 - Counterclockwise: Return the previous set of results

The application can be enabled to log both the graphs previously created by the user, as well as the actions a user took in developing those graphs. Because of the focused nature of research performed in individual labs, there are often similarities in graphs developed by a single user. The tool can then use this logged information to provide better search results and thus further streamline the graph building process.

Effectively, through collecting and processing the information about user's previous work and user's behavior, the tool can 'learn' and adjust it's responses to the user. Incorporating these new ideas into the tool behavior led me to develop the final tool functionality matrix:

GRAPH BUILDING TOOL FUNCTIONALITY MATRIX

ACTION	SPEED	IMPLICATION	RESULT	LEARNING	+/-	VIEW
rotate: clockwise	fast	dissatisfaction with results	increase number of results; diversify results	reduce accuracy rating for the current result set	-1	
rotate: counter-clockwise	fast	too many results	decrease number of results; diversify results	user preference for number of results presented		
rotate: clockwise/ counter-clockwise	slow	correct result is within the result subset surrounding cursor position	increase information resolution for the item under the cursor	increase accuracy rating for the result subset surrounding cursor position	1	
pause over the result	pause	increased interest in the result	further increase information resolution for the item under the cursor	increase accuracy rating for the result under the cursor	3	
click on the line connecting initial item and a result		potential for a correct result	bookmark the result	increase accuracy rating for the selected result	3	
click on the line connecting initial item and a result (after it has been clicked on once)			reverse previous	reverse previous	-3	
click on the result		user wants to explore the path without committing to the result	mark the selected result; re-search the database using the selected result as a starting point and load the new results into the tool	increase accuracy rating for the selected result	3	
click on the result (after it has been clicked on once)		user did not like the path	unmark the clicked result; reload the original search results into the tool	decrease accuracy rating for the selected result	-3	
drag the result on top of the initial item		user wants to explore the path further without committing to the result	replace the initial item with the result; re-search the database using the selected result as a starting point and load the new results into the tool	increase accuracy rating for the selected result	2	
drag the initial item and drop it in the results area (available only if user previously dragged the result on top of the initial item)		user did not like the path	'back' button: return the tool to the last search	decrease accuracy rating for the selected result	-2	
double-click on the result		correct result	make the result an initial item for the next search	increase accuracy rating for the selected result	4	
click on the initial item		user got lost in the search	reset the tool/results	unsuccessful search experience - adjust behavior of the tool: speed/results		
double-click outside of the tool or graph			exit the tool			

Integration into the application

Since the tool was designed to be self-containing, integrating it into the application posed very little challenge.

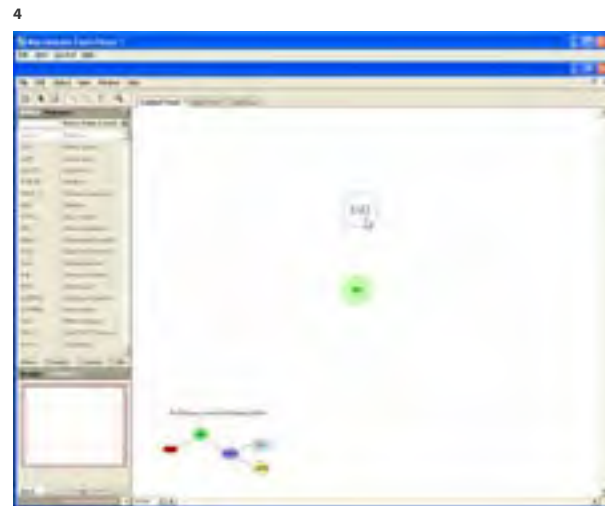
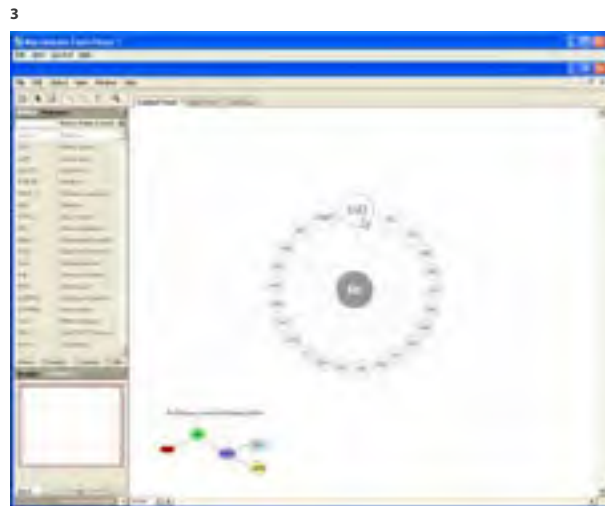
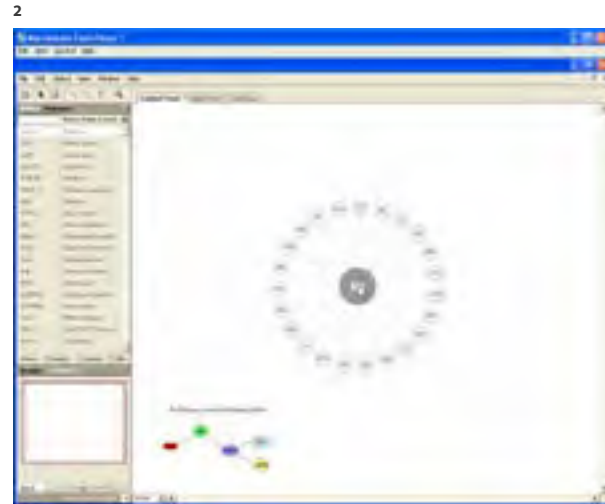
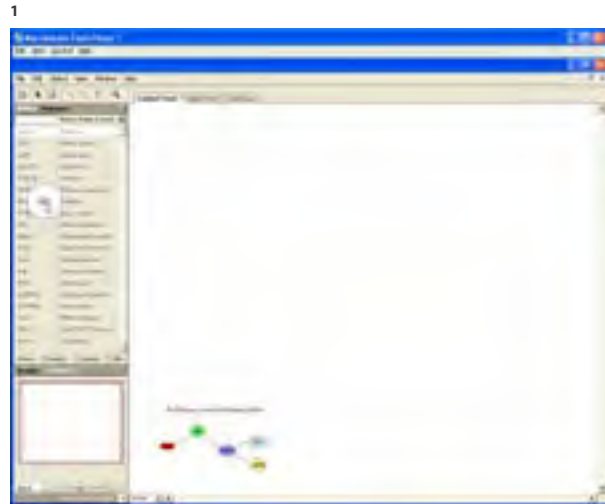
The only issue that emerged was dealing with large graphs. Since it is beneficial to the user to be able to see the arrangement of the whole graph at once, the zoom feature had to be integrated into the tool.

Right clicking outside of the tool brings up the zoom slider. This way the user can zoom in or out of the graph without leaving the stage area.

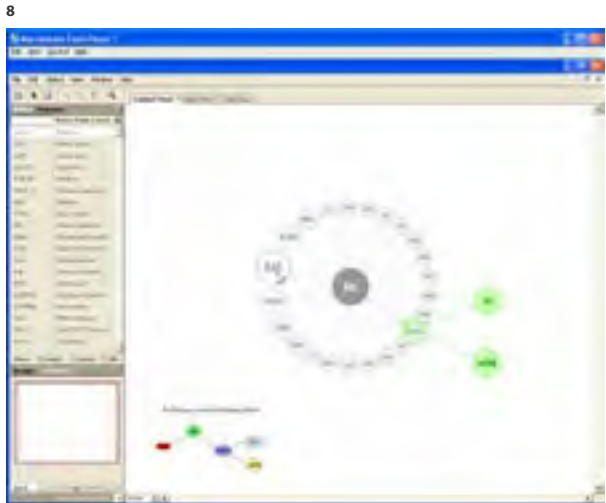
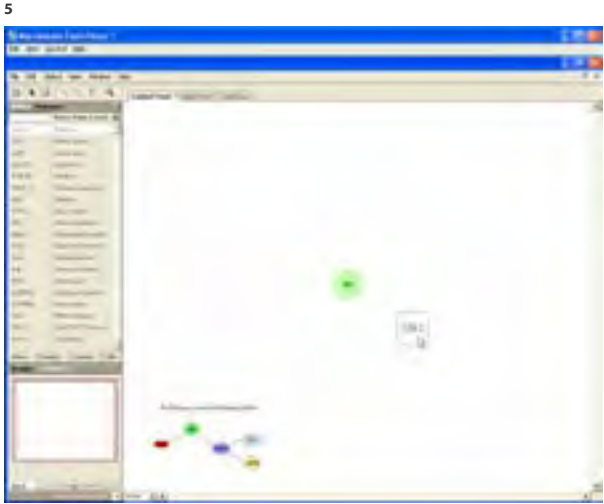
Next three pages: Screenshots of the graph building tool integrated into the application:

PROJECT 3 (THESIS): GRAPH CREATION TOOL
THE DESIGN PROBLEM
INTEGRATION INTO THE APPLICATION

APPENDIX ○
PROJECT 3 ●
PROJECT 2 ○
PROJECT 1 ○
FRAMEWORK ○
EXPERIENCE ○
FOREWORD ○
ABSTRACT ○
ACKNOWLEDGEMENTS ○



PROJECT 3 (THESIS): GRAPH CREATION TOOL
THE DESIGN PROBLEM
INTEGRATION INTO THE APPLICATION



PROJECT 3 (THESIS): GRAPH CREATION TOOL
THE DESIGN PROBLEM
INTEGRATION INTO THE APPLICATION

APPENDIX ○
PROJECT 3 ●
PROJECT 2 ○
PROJECT 1 ○
FRAMEWORK ○
EXPERIENCE ○
FOREWORD ○
ABSTRACT ○
ACKNOWLEDGEMENTS ○

9



10



11



12



CONCLUSIONS

Other uses

The graph building tool could be used in any industry where the events described graphs form chains – such as chemical engineering, circuit design, etc.

Potential for the future

The graph building tool shows a possibility for creating interfaces that go beyond the standard application structure and focus directly on the task execution. Most applications require the user to select an object, and then hunt through multiple, often confusing, menus to modify it. Although contextual menus provide some form of direct action, the options given are often very limited – group/ungroup, copy/paste, etc.

New interfaces, such as the graph building tool, integrated into the application ‘stage’ and tailored to the needs and workflow of the user, could significantly streamline the production process.

Almost all application log user actions – recording information required for multiple undos represents a form of action log – but very few use this information to enhance user experience. If an application could ‘learn’ from those actions, and adjusted it’s behavior or even the overall structure accordingly, this could translate into a truly personalized user experience.

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

PROJECT 3 (THESIS): GRAPH CREATION TOOL

CONCLUSIONS

POTENTIAL FOR THE FUTURE

Furthermore, recording the way actions are performed, and not only the actions themselves, could provide valuable information about the user's 'state of mind'. Simple feelings, such as anger, frustration, decisiveness amongst others, could be mapped onto various motions of the cursor and provide even more insight into how the application could modify itself to better suit the user.

To better facilitate this, the input devices could be built that would provide user 'state of mind' information to applications. For instance, a mouse could collect pressure or temperature information and send it to the active program.

REFERENCES:

INSPIRATION:

<http://www.thebrain.com>

APPENDIX



APPENDIX
PROJECT 3
PROJECT 2
PROJECT 1
FRAMEWORK
EXPERIENCE
FOREWORD
ABSTRACT
ACKNOWLEDGEMENTS

OBJECT ORIENTED INFORMATION

I would like to start by saying that the concept of object oriented information is neither new or original. It is simply a different way of approaching a particular information structuring problem.

The proposition is fairly simple. Instead of focusing on a volume of information (content) as a unit and trying to fit it into a 'standard' (i.e. Wurman's LATCH method) filtering system, the content needs to be broken down to the 'unit' of information – a single piece of information that is being offered to the user as a result of the filtering. That unit of information then needs to be analyzed. Finally, parallels need to be drawn between the result of the information analysis and the result of the user analysis.

User analysis has become an integral part of any information architecture or design process. It provides us with the social, economic, educational and other 'properties' of the user. It gives us insight into the real goals of the user and the tools available to the user to reach this goal (tools being language, technical proficiency, previous experience with certain navigational systems as a result of users education, employment, interests, etc.) It can also provide us the users' probable state of mind at the moment of accessing an interface – environment (home, office, school, museum kiosk), stress level (urgent access to information, leisurely browsing, etc.). A factor that is often overlooked are the users areas of interest indirectly related to the information we are trying to provide the structure. Such knowledge can provide us with valuable insight into potential ways of presenting the information.

In depth analysis of the information 'unit' is, however, seldomly (if ever) performed. The information unit itself

APPENDIX OBJECT ORIENTED INFORMATION

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

contains certain properties (thus the 'object oriented information'). The information contains two types of properties: first the general physical properties derived from the piece of information itself. Lets take this statement for example: "In the 15th century Guttenberg printed the Bible on the first moveable type press." From this example we can extract the following: time when the event occurred, who was involved, what was done, which tool was used. There is a whole other layer of properties of this piece of information that cannot be simply extracted from it's content. It deals with the information's placement within the cultural, socio-political, economic and other environments and it's connection to other events that preceeded and followed it. These properties of information can provide starting point to establishing more meaningful connections between the user and the information.

The key is finding the patterns and connections between the non-physical properties of the user and the information. This can provide a more intuitive and, even more important, learning oriented user experience.

Most of the user interfaces use "standard" drill-down filtering systems. The user starts with fairly general high level choices, and the available choices become more specific as the user comes closer to the information. At any given step in the process, the user is presented with more choices and only in the end is presented with the needed relevant information. If we take a standard corporate website as an example, we can find that the information is organized in a following way: the first level choice is, for example, a product, then type of product, than a specific product from the list and in the end the user is presented with the information he or she is looking for. In the process of filtering, the user is not being offered any new information, but is simply being presented with options. If the user is, by chance, presented with information, it is usually related to the previous choice and not the actual area of interest. This kind of system works well if you are targeting general audience and you cannot predict what their interests are and which particular task they are trying to accomplish. Since this is a standard way of sorting the information, everyone will eventually find what they

are looking for, even if they do not find it in the easiest and most intuitive way. The problem with this kind of system is that it does not foster a learning process. Instead it assumes that the users know exactly which information they are seeking. The question then becomes: is the information they are looking for really the information they need (there is a reason why the back button is the most used button on the web)?

Establishing deeper connections between the user and the information creates a different filtering experience. If the filtering system is set up according to users interests and their intersection with information properties, at any given point in the filtering (drilling) process, users can be presented with useful information related to their interests as well as indirectly related issues related to the information sought. This approach can educate (or at least give an idea) the user as to what possible other spheres of information relate to the current interests on that filtering level, thus creating a better chance that the users will find the information they need (as opposed to the information they think they need). This approach to information organization would be useless for general audiences, but it could be extremely useful in certain specific isolated environments, such as corporate or governmental knowledge bases, scientific libraries (field/research specific), etc.

Although this approach would not necessarily be used in most professional situations, it has tremendous potential for education of information architecture. It brings pure information from the abstract level down to something that can be observed and analyzed according to given rules. The only problem that remains is that the in-depth analysis of information's non-physical properties demands at least a basic knowledge of history, sociology, psychology, world events, and, pretty much everything, or at least a strong desire to learn about those things. It basically demands an interest in the world that surrounds us in general, and that broadness is something very few educational institutions (or educators) pay much attention to.

APPENDIX
CURRICULUM VITAE

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

CURRICULUM VITAE

TEACHING EXPERIENCE [01.1993 – Present]

05.1996 - Present : Massachusetts College of Art, Boston, MA

Visiting Lecturer, Communication Design

Developed curriculum for Introductory Web Design and Advanced Web Design.

Helped develop a sequence of courses in Information Architecture - adopted as mandatory.

Developed curriculum for a graduate level course in Server-Side Programming for the Web.

Classes Taught: Information Architecture I
Information Architecture II
Web Design I (Introductory)
Web Design II (Advanced)
Graphic Design II
PHP: Server-Side Programming for the Web
Computer Imaging; Digital Publishing Tools

Program Coordinator, Design for Interactive Communication

Member of the committee for development of the new Design for Interactive Communications Program.
Developed general curriculum for all courses in the Design for Interactive Communications Program.

- Classes Taught:
- Principles of Dynamic Design
 - Algorhythmic Thinking
 - Information Architecture I: Static
 - Information Architecture II: Dynamic
 - Branding I: User Interface – Static
 - Branding II: User Interface – Dynamic
 - Production I: Static/Client Side Programming
 - Production II: Dynamic/Server Side Programming

01.1993 - 05.1996 : Massachusetts College of Art, Boston, MA : Teaching Assistant

Teaching assistant for computer graphics classes: Photoshop, Illustrator, QuarkXpress for Mac; Deluxe Paint and Scala for Amiga; Design CAD for IBM. Involved in preparing scripts for classes. Developing manuals for different hardware and software packages used by MassArt students. Maintenance of IBM/Macintosh/Amiga laboratories.

APPENDIX
CURRICULUM VITAE

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

BUSINESS VENTURES [05.1996 – Present] *Chronological*

05.1996 - 03.2000 : Visualwaste, Cambridge, MA

Started a Web Design Studio. Over the course of 4 years designed and developed over 200 small and medium size web sites. Consulted small companies on communication as well as technical issues.

Duties: Developing multimedia and web design solutions for clients
Communication/technical consulting

03.2000 – 08.2002 : BitmapLogic, Cambridge, MA

Outgrew the name of the starting venture - Visualwaste. The new company became a Web Development and Communication Consulting Firm specializing in Information Architecture and Dynamic Web Site Design. The goal was to combine the educational, technical and design experience and offer it to corporate clients as a unified approach to web site and on-line application development.

Duties: Developing multimedia and web design solutions for clients
Information architecture consulting



08.2002 – 08.2003 : EgoDigita, Cambridge, MA

To offset the raising productions costs, I established an outsourcing outfit in Belgrade, Yugoslavia. The new company – EgoDigita, provided multimedia production and development services to a variety of clients ranging from small businesses to large educational publishing companies.

Duties: Developing multimedia and on- and off-line application solutions for clients
Management of the overseas development team

08.2003 – Present : Artmedialab, Cambridge, MA

The last business venture – Egodigita – grew into a partnership. The name changed to reflect the change in the company structure. The offer expanded to include full application development services as well as corporate education services.

Duties: Developing multimedia and on- and off-line application solutions for clients
Developing strategic business objectives of the company
Oversight management of the technology implementation and software development
Management of the development teams in the US and overseas
Information architecture consulting
Corporate education

APPENDIX
CURRICULUM VITAE

APPENDIX

PROJECT 3

PROJECT 2

PROJECT 1

FRAMEWORK

EXPERIENCE

FOREWORD

ABSTRACT

ACKNOWLEDGEMENTS

OTHER PROFESSIONAL EXPERIENCE

09.2002 – 05.2003 : i4Synergy, Inc., Bedford, MA : Art Director

Managing web development, corporate identity development as well as interface development for i4Synergy's main product - an online project management system for home builders.

05.1999 - 12.2000 : Pearson Education, Reading, MA : Information Architect / Web Developer

Principal interface developer for Pearson's largest educational web sites ('Places' series: Biology, Chemistry, Psychology, History). Streamlined and simplified information architecture and user interface. Built consistent visual identity for all 'Places' web sites.

05.1994 - 05.1996 : Essinger Design Associates, Newton, MA : Graphic Designer

Worked on various design projects as a member of a team. Computer graphics / image manipulation. Preparation of projects for printing. Administration work.