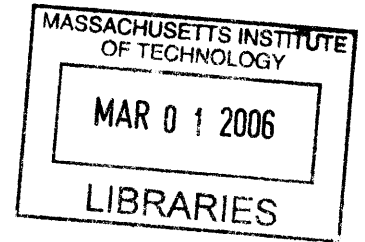


A Goal-Oriented User Interface for Personalized Semantic Search

by

Alexander James Faaborg

B.A. Information Science
Cornell University, 2003



Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

V.1
ROTCH

February 2006

© Massachusetts Institute of Technology 2005. All rights reserved.

Author _____ Program in Media Arts and Sciences
November 4th 2005

Certified by _____ Henry Lieberman
Research Scientist
MIT Media Laboratory
Thesis Supervisor

Accepted by _____ Andrew B. Lippman
Chair, Department Committee on Graduate Students
Program in Media Arts and Sciences

A Goal-Oriented User Interface for Personalized Semantic Search

by

Alexander James Faaborg

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning, on November 4th 2005,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Users have high-level goals when they browse the Web or perform searches. However, the two primary user interfaces positioned between users and the Web, Web browsers and search engines, have very little interest in users' goals. Present-day Web browsers provide only a thin interface between users and the Web, and present-day search engines rely solely on keyword matching. This thesis leverages large knowledge bases of semantic information to provide users with a goal-oriented Web browsing experience. By understanding the meaning of Web pages and search queries, this thesis demonstrates how Web browsers and search engines can proactively suggest content and services to users that are both contextually relevant and personalized.

This thesis presents (1) *Creo*, a Programming by Example system that allows users to teach their computers how to automate interactions with their favorite Web sites by providing a single demonstration, (2) *Miro*, a Data Detector that matches the content of a Web page to high-level user goals, and allows users to perform semantic searches, and (3) *Adeo*, an application that streamlines browsing the Web on mobile devices, allowing users to complete actions with a minimal amount of input and output.

An evaluation with 34 subjects found that they were more effective at completing tasks when using these applications, and that the subjects would use these applications if they had access to them.

Beyond these three user interfaces, this thesis also explores a number of underlying issues, including (1) automatically providing semantics to unstructured text, (2) building robust applications on top of messy knowledge bases, (3) leveraging surrounding context to disambiguate concepts that have multiple meanings, and (4) learning new knowledge by reading the Web.

Thesis Supervisor: Henry Lieberman
Title: Research Scientist

A Goal-Oriented User Interface for Personalized Semantic Search
by
Alexander James Faaborg

Thesis Committee

Advisor

Henry Lieberman
Research Scientist
MIT Media Laboratory

Reader

Pattie Maes
Professor
MIT Media Laboratory

Reader

Rob Miller
Assistant Professor
MIT Computer Science and Artificial Intelligence Laboratory

Reader

James Hendler
Professor
University of Maryland at College Park
Department of Computer Science

Acknowledgements

First, I would like to thank Anna, for bringing me food, for reminding me when to sleep, and for putting up with not having any furniture for ~~a month~~ two months while I finished writing this thesis.

I would like to thank my family for their motivation and support: my father Roger, for inspiring me to study computer science, my mother Diana for her constant encouragement and wonderfully accurate proof reading, and my brother Kevin, for his unconditional friendship, wisdom, and for understanding that the reason I wasn't done yet wasn't because I was lazy, but was because I had decided to write a "theses." I'm sorry I didn't get a chance to spend more time with you this past summer.

I would like to thank my readers, Pattie Maes, Rob Miller and James Hendler for their advice and guidance. Also, I would like to thank to Rada Mihalcea for her help with the natural language processing sections of my thesis.

For the past two years I have had the chance to work with some amazingly brilliant people at the Media Lab, including Tom Stocky, José Espinosa, Ashwani Kumar, Push Singh, Hugo Liu, Ian Eslick, Bo Morgan, and the rest of think-hackers.

I have been extremely lucky to work on the incredible vision of common sense computing being driven at the Media Lab by Marvin Minsky, Henry Lieberman, and Push Singh. Years from now I will still be immensely proud that I played a small role in "teaching computers the stuff we all know," and creating intelligent user interfaces.

Most of all, I would like to thank my advisor, Henry Lieberman. Henry has been overwhelmingly generous, and the two years I have spent as his student have been the most exciting and intellectually stimulating experience of my life. For example, a random side comment Henry made during a sponsor presentation at the Media Lab resulted in us flying to Helsinki and attending a rock concert at 10pm, with the sun still shining. The next day we ate dinner with Ora Lassila and James Hendler, (who co-authored The Semantic Web [1] with Tim Berners-Lee) at a restaurant on its own private island. It is experiences like these that many advisors do not afford their students.

Accompanying Henry on his academic adventures, both intellectually, and literally, has been a privilege I will never forget.

Table of Contents

- Chapter 1 Introduction 18**
- 1.1 Goal-Based Design 18
 - 1.1.1 The User Interface Design Process 18
- 1.2 Teaching Computers the Stuff We All Know 21
 - 1.2.1 Reversing HCI 21
 - 1.2.2 Creating a Large Knowledge Base of Commonsense Facts 21
- 1.3 As We May Think 25
- 1.4 A Goal-Oriented Web Browser 27
 - 1.4.1 Using Commonsense Knowledge to Detect Uncommon Goals 28
- 1.5 Taking End Users from the Present-Day Web to Semantic Web Services 29
 - 1.5.1 A Different Approach to Semantic Web Services 29
 - 1.5.2 Programming by Example: From Static to Dynamic Resources 30
 - 1.5.3 ConceptNet and TAP: from Syntax to Semantics 30
 - 1.5.4 Building New Interfaces on Top of the Web 31
- 1.6 Contributions 32
 - 1.6.1 Contributions to User Interface Design 32
 - 1.6.2 Contributions to Programming by Example 34
 - 1.6.3 Contributions to Data Detection 34
 - 1.6.4 Contributions to Artificial Intelligence 35
- 1.7 Towards the Semantic Web 36
 - 1.7.1 Future Need of Programming by Example Systems 36
 - 1.7.2 Future Need of Data Detectors 36
 - 1.7.3 A User Interface for the Semantic Web 37
- Chapter 2 User Scenarios and Discussion 38**
- 2.1 Introduction 38
 - 2.1.1 Example User Scenario: Pete Studies for an Exam 38
 - 2.1.2 Creo, Miro and Adeo 39
- 2.2 Creating Recordings 43
 - 2.2.1 Recording a Simple Procedure with Creo 43
 - 2.2.2 Recording a Procedure with Generalized Input with Creo 49
- 2.3 Using Recordings 55
 - 2.3.1 Using Miro to Match Recordings against the Semantic Context of a Web Page 55
 - 2.3.2 Putting Users in Control of their Data and Services 59

2.3.3	Creating a General-Purpose Recording with a Single Example	64
2.3.4	Personalized Semantic Searches	71
2.3.5	Adeo	75
2.4	Learning.....	84
2.4.1	Introduction	84
2.4.2	Learning in Creo	84
2.4.3	Learning in Miro - Learning From the User.....	92
2.4.4	Learning in Miro - Learning From the Web.....	107
2.4.5	Conclusion - Knowledge Capture Through the Use of Applications	121
2.5	Security and the Viral Spread of Recordings.....	122
2.5.1	Introduction	122
2.5.2	Creo's Detection and Storage of Personal Information.....	122
2.5.3	The Viral Spread of Recordings	128
2.5.4	Security in Adeo.....	132
2.5.5	Conclusion	133
2.6	Conclusion.....	134
Chapter 3 Design and Implementation		135
3.1	Introduction.....	135
3.2	Designing a User Interface Agent for the Web.....	135
3.2.1	Software Agents vs. Direct Manipulation Interfaces	135
3.2.2	Design Principles for Software Agents.....	138
3.2.3	Designing the User Interface	144
3.3	Implementing a User Interface Agent for the Web	150
3.3.1	Language and Platform	150
3.3.2	Monitoring and Impersonating the User	150
3.3.3	Basic Abilities of a Web Browser Agent.....	151
3.3.4	Advanced Abilities of a Web Browser Agent	153
3.3.5	Source Code and Documentation	160
3.4	Invoking Procedures on the Web from a Mobile Device.....	161
3.4.1	Design	161
3.4.2	Implementation	163
3.4.3	Source Code and Documentation	165
3.5	Limitations of the Current Implementation.....	166
3.5.1	Limitations of Creo	166
3.5.2	Limitations of Miro.....	168
3.5.3	Limitations of Adeo.....	169
Chapter 4 Background and Related Work		171
4.1	Introduction.....	171
4.2	Beating Some Common Sense into Interactive Applications	172
4.2.1	Common Sense and the Web.....	172

4.2.2	Using Commonsense Knowledge to Understand the Context of Text	176
4.2.3	Additional Applications of Commonsense Knowledge.....	178
4.3	Work Related to Creo.....	179
4.3.1	Programming by Example	179
4.3.2	End-User Programming for the Web	194
4.3.3	Extracting Information from the Web.....	196
4.4	Work Related to Miro	197
4.4.1	A Brief History of Data Detection.....	197
4.4.2	Present-Day Data Detectors	202
4.4.3	Back to the Future.....	202
4.4.4	Semantic Search.....	204
4.4.5	Learning from the Web	209
4.5	Work Related to Adeo	211
4.6	Related Semantic Web Research	212
4.6.1	"Semantic Web" Browsers	212
4.6.2	Semantic "Web Browsers"	213
4.6.3	Usability of Semantic Web Browsers	218
Chapter 5 Evaluation		221
5.1	Introduction.....	221
5.2	Evaluations of the User Interface.....	221
5.2.1	Evaluating Version 1	224
5.2.2	Evaluating Version 2.....	226
5.2.3	Evaluating Version 3.....	228
5.2.4	Evaluating Version 4.....	232
5.3	Evaluating the Software's Overall Effectiveness	232
5.3.1	Experiment Hypothesis	233
5.3.2	Experiment Design	234
5.3.3	Experiment Results	239
5.3.4	User Study Discussion and Summary.....	253
Chapter 6 Future Directions and Conclusion.....		257
6.1	Future Work	257
6.1.1	Passively Monitoring the User	257
6.1.2	Creating a Central Repository of Recordings	261
6.1.3	Building Different Types of User Interfaces on Top of the Web	262
6.1.4	Integrating Scan into Adeo.....	264
6.1.5	Ambient Intelligence and Context Aware Computing: Real World Data Detection.....	266
6.2	Conclusion.....	269
Appendix A User Study Recruitment Poster.....		270
Appendix B User Study Consent Form and Questionnaires		272

6.2.1	Consent to Participate in Non-Biomedical Research.....	273
6.2.2	A Goal-Oriented User Interface for Personalized Semantic Search Pre-Experiment Questionnaire	278
6.2.3	A Goal-Oriented User Interface for Personalized Semantic Search Post-Experiment Questionnaire	279
Bibliography		280

List of Figures

Figure 1-1 A goal-oriented Web browser.....	27
Figure 1-2 Blueberry by Maria Eva.....	28
Figure 1-3 A different road map to Semantic Web Services.....	29
Figure 2-1 Creo.....	40
Figure 2-2 Miro.....	41
Figure 2-3 Adeo.....	42
Figure 2-4 Creo's <i>Player</i> tab.....	43
Figure 2-5 Creo, ready to start recording.....	44
Figure 2-6 Creo, recording a simple procedure.....	45
Figure 2-7 Creo, action list.....	46
Figure 2-8 A graduate student's "hypothetical" bank account.....	47
Figure 2-9 Training Creo to scrape information from a page by giving it an example.....	47
Figure 2-10 The user enters a name for the recording.....	48
Figure 2-11 The recording is complete.....	49
Figure 2-12 The user browses to FreshDirect.....	50
Figure 2-13 Creo correctly recognizes example input and generalizes it.....	50
Figure 2-14 Creo defaults "diet coke" to a type of "food brand".....	51
Figure 2-15 Generalizations of the concept "diet coke".....	52
Figure 2-16 Creo prompts the user for input when directly playing a recording.....	54
Figure 2-17 Without any context, Creo must ask the user for input (not so grrreat).....	55
Figure 2-18 Using Miro to scan a page for concepts.....	55
Figure 2-19 Miro suggests the Order Food recording.....	56
Figure 2-20 The user clicks the <i>Order Food</i> button.....	56
Figure 2-21 All of the foods in the recipe turn into hyperlinks to the Order Food recording.....	57
Figure 2-22 Two recordings match the semantic context of the page.....	57

Figure 2-23 The user selects the Nutritional Info recording instead of the Order Food recoding	58
Figure 2-24 An example of Smart Tags in Microsoft Word	59
Figure 2-25 Google's AutoLink feature.....	59
Figure 2-26 Types of information detected by Smart Tags	60
Figure 2-27 Types of information detected by AutoLink	61
Figure 2-28 Using Miro's Scan feature to link the titles of books to the user's favorite book store	63
Figure 2-29 Creo automatically generalizes the user's input	64
Figure 2-30 Miro matches concepts on a Web page against the current set of recordings	65
Figure 2-31 Miro generalizes musicians using a messy knowledge base	66
Figure 2-32 I want to buy an apple	67
Figure 2-33 Apple URIs	68
Figure 2-34 Telling Open Mind that Apple is a type of a computer.....	69
Figure 2-35 Miro's search box	71
Figure 2-36 Using Miro to search for "milk"	71
Figure 2-37 Searching Miro for "the lord of the rings".....	72
Figure 2-38 The results of Miro's search are weighted against the semantic context of the page (books).....	72
Figure 2-39 The results of Miro's search are weighted against the semantic context of the page (movies)	73
Figure 2-40 Adeo brings the functionality of Creo's <i>Player</i> tab to a mobile device	75
Figure 2-41 Adeo brings the functionality of Miro's semantic search to a mobile device.....	76
Figure 2-42 A WAP site for rare bird watchers (wonderfully ironic).....	78
Figure 2-43 Interacting with a Web page designed for viewing on a PC can be a slow and frustrating process on a mobile device	79
Figure 2-44 The user invokes a multi-step procedure with a single click	80
Figure 2-45 Using Adeo to check the balance of a "hypothetical" bank account.....	81

Figure 2-46 Adeo performs a semantic search, reducing the number of steps to complete a task82

Figure 2-47 A sample of the LifeNet network.....85

Figure 2-48 Many critical pieces of information are embedded in images86

Figure 2-49 Attempting to set up automatic billing with an electric company, the user is confronted with a reverse Turing Test87

Figure 2-50 A captcha test, also described as a reverse Turing Test.....88

Figure 2-51 The Debug Recording hyperlink.....90

Figure 2-52 The user steps through actions in a recording90

Figure 2-53 Creo's debug interface91

Figure 2-54 The *Train* hyperlink appears instead of results92

Figure 2-55 The *Train* hyperlink.....92

Figure 2-56 The user performs a semantic search on an unknown concept .94

Figure 2-57 Miro asks the user to generalize the unknown concept94

Figure 2-58 Miro does not know about one of the books because it is new ..95

Figure 2-59 The user tells Miro which concept should have been detected ..95

Figure 2-60 Miro performs word completion for the user, weighted against the generality of concepts.....96

Figure 2-61 The user was going for "fruit" but "food" is also true.....97

Figure 2-62 More general generalizations have a better chance of matching the recordings created with Creo98

Figure 2-63 The final step of the training wizard.....99

Figure 2-64 The Vogons would be very displeased with missing "Hitchhiker's Guide to the Galaxy" generalizations..... 100

Figure 2-65 Miro thinks "Hitchhiker's Guide to the Galaxy" is a book 101

Figure 2-66 The user instructs Miro that "Hitchhiker's Guide to the Galaxy" is also a movie 101

Figure 2-67 Miro does not know which recording should match the concept of "tylenol" 102

Figure 2-68 Tylenol is not a food, but you can order it from FreshDirect ... 103

Figure 2-69 Miro already knows what Tylenol is..... 104

Figure 2-70 Miro does not know which recording should have been activated	104
Figure 2-71 Miro now associates all types of medicines with the user's online grocery store	106
Figure 2-72 Why does Miro think "Elton John" is a food?.....	106
Figure 2-73 Miro learns a new piece of knowledge by reading the Web.....	107
Figure 2-74 1980s new wave era goodness.....	112
Figure 2-75 The user selects the activated recording, indicating that the new knowledge may be useful	113
Figure 2-76 Generalizations of Britney Spears	116
Figure 2-77 Creo recognizes the user's address.....	123
Figure 2-78 The user's profile.....	124
Figure 2-79 Users can control which pieces of input are personal information from the <i>Share</i> tab	125
Figure 2-80 Step 1: The user types their account number into a form	126
Figure 2-81 Step 2: Creo automatically identifies the input as personal information.....	126
Figure 2-82 The user clicks <i>Explore</i> and is taken to their recordings folder	128
Figure 2-83 Creo's confirmation interface	131
Figure 2-84 Adeo's thin client design.....	132
Figure 3-1 Direct manipulation interfaces do not scale well as functionality increases.....	136
Figure 3-2 Is this better than a Direct Manipulation Interface?	138
Figure 3-3 The Scan Page button is a debatable design decision	140
Figure 3-4 The record and play metaphor	144
Figure 3-5 The Submit Form Information window	145
Figure 3-6 Example of dynamically generated contextual help	146
Figure 3-7 Example of dynamically generated contextual help	147
Figure 3-8 Skittles, taste the rainbow™	148
Figure 3-9 Miro converts plain text into hyperlinks that invoke a recording	158
Figure 3-10 Adeo's list of recordings is automatically synchronized with the user's computer	161

Figure 3-11 Adeo provides the user with feedback	162
Figure 3-12 Adeo's client/server system architecture	162
Figure 3-13 Adeo_m performs semantic searches by querying Adeo_s for generalizations.....	164
Figure 3-14 Creo cannot parse bling-bling search fields implemented in Flash	166
Figure 3-15 This hyperlink shouldn't exist.....	167
Figure 3-16 Miro's inability to directly modify the page breaks many Web sites	168
Figure 4-1 Woodstein, by Earl Wagner	173
Figure 4-2 GOOSE, by Hugo Liu.....	174
Figure 4-3 Predictive Text Entry, by Alexander Faaborg.....	176
Figure 4-4 ARIA, by Hugo Liu.....	177
Figure 4-5 The User Solution interaction mode in Creo	180
Figure 4-6 The Collaborative Solution interaction mode in Creo	182
Figure 4-7 The Collaborative Solution interaction mode in Miro.....	183
Figure 4-8 The System Solution interaction mode in Miro	184
Figure 4-9 Grammex, by Henry Lieberman	188
Figure 4-10 TrIAs, by Mathias Bauer	190
Figure 4-11 Internet Scrapbook, by Atsushi Sugiura	191
Figure 4-12 LAPIS, by Rob Miller	193
Figure 4-13 Chickenfoot, by Michael Bolin.....	195
Figure 4-14 1995: IntelliSense	198
Figure 4-15 1998: Apple Data Detectors.....	199
Figure 4-16 1998: CyberDesk	200
Figure 4-17 1998: LiveDoc and DropZones	201
Figure 4-18 2001: Microsoft Smart Tags.....	202
Figure 4-19 2005: Google AutoLink.....	202
Figure 4-20 Activity Based Search by the Stanford Knowledge Systems Laboratory	206
Figure 4-21 The Matrix?	207
Figure 4-22 Using Haystack to categorize a recipe	213

Figure 4-23 Semantic Browsing, the Web Annotation Pane	214
Figure 4-24 Defining a layer of semantic metadata on top of the existing Web	215
Figure 4-25 Piggy Bank's <i>Data Coin</i>	216
Figure 4-26 Selecting an ontology with Magpie.....	218
Figure 4-27 Magpie's <i>Semantic Services</i> menu	218
Figure 4-28 URIs in Haystack.....	219
Figure 4-29 RDF triple icon in Piggy Bank	219
Figure 5-1 The evolution of Creo's design, Versions 2, 3 and 4	223
Figure 5-2 Creo Version 1: paper prototype	224
Figure 5-3 Early generalization interface.....	225
Figure 5-4 Creo Version 2: computer prototype.....	226
Figure 5-5 Creo Version 3: functional prototype	228
Figure 5-6 The generalization interface of Version 3.....	231
Figure 5-7 The generalization interface of Version 4, with contextual help	231
Figure 5-8 Creo Version 4: final interface	232
Figure 5-9 Experiment Part 1: experimental group, using the Miro toolbar	235
Figure 5-10 Experiment Part 1: control group, using a normal Web browser	236
Figure 5-11 Experiment Part 2: using Creo to associate foods with an online grocery store	237
Figure 5-12 Subjects' Web usage.....	240
Figure 5-13 Subjects' programming background, classes.....	241
Figure 5-14 Subjects' programming background, years	241
Figure 5-15 Mean time to complete Part 1	242
Figure 5-16 Mean time to complete Part 2	243
Figure 5-17 Did the subjects find Miro easy to use?	244
Figure 5-18 Would the subjects use Miro?.....	245
Figure 5-19 Did the subjects find Creo easy to use?.....	246
Figure 5-20 Would the subjects use Creo?	247
Figure 5-21 Did the subjects understand the software's overall utility?	248
Figure 5-22 Usability problem with saving recordings in Creo	252

Figure 5-23 Usability problem with submitting forms with Creo 252
Figure 5-24 The time to complete Part 2 added to the experimental group's
time in Part 1 254
Figure 6-1 The Cellular Squirrel, by Stefan Marti 263
Figure 6-2 ReachMedia, by Assaf Feldman 265
Figure 6-3 The Augmented Reality Kitchen 267

List of Tables

Table 1-1 Example of ConceptNet	22
Table 1-2 Example of ConceptNet	23
Table 1-3 Example of Stanford TAP	24
Table 2-1 Types of actions	46
Table 2-2 Using patterns and context to disambiguate concepts	70
Table 2-3 An example of how Creo encodes the user's actions with a Web site	89
Table 2-4 Miro's three-step training wizard	93
Table 2-5 Step 1 - Sentence boundary detection	108
Table 2-6 Step 2 - Isolation of nouns and words used for pattern matching	109
Table 2-7 Step 3 - Lemmatization.....	109
Table 2-8 Step 4 - Using regular expressions to extract IsA relationships .	109
Table 2-9 Example of Pattern 1	110
Table 2-10 Example of Pattern 2	110
Table 2-11 Example of Pattern 3	110
Table 2-12 Example of Pattern 4	111
Table 2-13 Example of Pattern 5	111
Table 2-14 Example of Pattern 6	111
Table 2-15 Example of Pattern 7	111
Table 2-16 Example of Pattern 8	111
Table 2-17 Example of Pattern 9	112
Table 2-18 Example of Pattern 10.....	112
Table 2-19 Example of Pattern 11.....	112
Table 2-20 Britney Spears is a three-headed alien	118
Table 2-21 Guava is a tree/fruit	119
Table 5-1 Changes to the <i>Recorder</i> interface between Version 3 and Version 4	230
Table 5-2 Study segment times.....	234

Table 5-3 Subject's demographic information..... 239
Table 5-4 Subject's Web usage 240

Chapter 1

Introduction

1.1 Goal-Based Design

1.1.1 The User Interface Design Process

A popular mantra in user interface design is to design for the user's goals, and not the user's tasks [2]. This is a very important guideline to follow when designing a user interface. For instance, digging through options dialog boxes to configure the Bluetooth settings on a Smartphone and on a PC to set up a wireless data transfer is not a goal, it is a task. The user's goal is to share a picture they took the night before with their friends. User interface designers follow a process of studying and understanding who their users are, and what their goals are. Then, the designer uses this knowledge of the user to shape and mold the interface accordingly. Often, user interface designers will evolve a user interface through an iterative process. They evaluate the effectiveness of their interface design with real users after every iteration.

In an Ecosystem of Interfaces, There is No Designer

Goal-based design and iterative design processes lead to highly usable interfaces. Unfortunately, these design techniques alone do not solve all of the challenges facing Human-Computer Interaction. This is because there are many situations where there is no team of interface designers in charge of shaping and molding the user experience. For instance, consider a home theater set-up containing a television, surround sound system, receiver, DVD

player, Xbox, GameCube, PlayStation, cable box, and TiVo. No single design team can be held responsible for the inherent complexity and confusion that emerges from dealing with so many different interfaces simultaneously. In fact, some of the devices in isolation have extremely well designed interfaces, like the TiVo. But the home theater experience, as a whole, is certainly not based around the user's goals [3].¹

In a similar manner, consider the Web. The Web contains billions of interfaces, some of which are usable. However, the user experience of interacting with the Web, as a whole, was not designed. The Web is not goal-based.

While Web browsers sit between the user and the Web, the very thin amount of interface they do provide (*Back, Next, Stop, Refresh, Home*) has little to do with the user's higher level goals. The same is true with search engines. Typing "milk" into Google Local does not result in the nearest grocery store.

Goal-Based Design, Without a Design Team

The reason a user's home theater, or the Web, are not goal-based is because there is no design team responsible for studying the user, and molding the interface appropriately. Traditionally, goal-based design has only applied to the creation of software applications with static interfaces. Any application that attempted to provide the user with a goal-based experience on top of the broad and changing collections of interfaces like a home theater, or the Web, would have to dynamically create the interface on its own. And the application would have to do this without the help of a designer's ethnographic aptitude. Creating a usable interface has traditionally been the designer's challenge, and not the application's challenge.

¹ Jose Espinosa at the MIT Media Lab has used software agents to specifically address this problem in his masters thesis *Reducing Complexity of Consumer Electronics Interfaces Using Commonsense Reasoning* [3].

Accurately predicting the user's goals is a difficult problem. This thesis demonstrates how software applications can begin to address this challenge through monitoring the user, and leveraging large knowledge bases of semantic information. These sources of information allow a software application to generate potential predictions of the user's goals. However, attempting to predict the user's goals at runtime requires software that knows a lot more about the user than the current generation of applications.

1.2 Teaching Computers the Stuff We All Know

1.2.1 Reversing HCI

The field of Human-Computer Interaction has predominantly focused on improving usability by simplifying user interfaces, making it easier for humans to understand computers. The Software Agents Group and Commonsense Computing Group at the MIT Media Lab have taken the opposite approach: improving usability by making it easier for computers to understand humans.

1.2.2 Creating a Large Knowledge Base of Commonsense Facts

Computers lack common sense. Current software applications know literally nothing about human existence. Because of this, the extent to which an application understands its user is restricted to simplistic preferences and settings that must be directly manipulated. Once software applications are given access to Commonsense Knowledge, hundreds of thousands of facts about the world we live in, they can begin to employ this knowledge to understand their user's intentions and goals.

Open Mind

Since the fall of 2000, the MIT Media Lab has been collecting commonsense facts from the general public through a Web site called Open Mind [4-8]. Currently, the Open Mind Common Sense Project has collected over 772,000 facts from over 16,000 participants. These facts are submitted by users as natural language statements of the form "*tennis is a sport*" and "*playing tennis requires a tennis racket.*" While Open Mind does not contain a complete set of all the common sense knowledge found in the world, its knowledge base is sufficiently large enough to be useful in real world applications.

Open Mind, lines 958298 to 958316
Something that might happen when you begin work is getting interrupted
Something that might happen while playing tennis is hitting a ball
The effect of seeing art is pleasure
Something that might happen while playing poker is losing money
Something that might happen when you wash clothes is you get clean clothes
Something that might happen when you learn is gaining new knowledge
The effect of going to a performance is seeing a show
The effect of becoming more clean is being more comfortable
Something that might happen while making sure you re healthy is going to the doctor
Something you might do while washing your clothes is rinse the soap out

Table 1-1 Example of ConceptNet

ConceptNet

Using natural language processing, the Open Mind knowledge base was mined to create ConceptNet [9], a large-scale semantic network currently containing over 250,000 commonsense facts. ConceptNet consists of machine-readable logical predicates of the form: (IsA "tennis" "sport") and (EventForGoalEvent "play tennis" "have racket"). ConceptNet is similar to WordNet [10] in that it is a large semantic network of concepts, however ConceptNet contains everyday knowledge about the world, while WordNet follows a more formal and taxonomic structure. For instance, WordNet would identify a "dog" as a type of "canine", which is a type of "carnivore", which is a kind of "placental mammal." ConceptNet identifies a "dog" as a type of "pet" [9].

ConceptNet, lines 15809 to 15818
<pre>(IsA "diamond" "rare stone" "f=2;i=0;") (MotivationOf "have money" "buy present" "f=2;i=1;") (SubeventOf "play baseball" "hit ball" "f=2;i=3;") (CapableOf "bus" "carry lot of person" "f=2;i=0;") (UsedFor "buy food" "prepare meal" "f=2;i=0;") (FirstSubeventOf "skate" "put on person 's skate" "f=2;i=0;") (LocationOf "rug" "under table" "f=2;i=0;") (CapableOfReceivingAction "play game" "enjoy" "f=0;i=4;") (IsA "subway" "form of public transportation" "f=2;i=0;") (LocationOf "bug" "in field" "f=2;i=0;")</pre>

Table 1-2 Example of ConceptNet

Stanford TAP

The Stanford TAP (The Alpiri Project) knowledge base was created to help bootstrap the Semantic Web [11-16]. Unlike the Open Mind knowledge base, which was generated through the contributions of knowledge from volunteers on the Web, TAP was generated by creating 207 HTML scrapers for 38 Web sites rich with instance data. These sites included AllMusic, eBay, Amazon, AOL Shopping, TicketMaster, People Magazine, Weather.com, Mapquest, Carpoint, Digital Cities, and Walmart.com. TAP has extracted knowledge from over 150,000 Web pages, discovering over 1.6 million entities and asserting over 6 million triples about these entities [16]. This knowledge covers a wide variety of topics, including: music, movies, actors, television shows, authors, classic books, athletes, sports, sports teams, auto models, companies, home appliances, toys, baby products, countries, states, cities, tourists attractions, consumer electronics, video games, diseases, and common drugs. The instance data found in TAP is a good complement to commonsense knowledge bases like ConceptNet or CYC [17]. For instance, "CYC knows a lot about what it means to be a musician. If it is told that Yo-Yo Ma is a cellist, it can infer that he probably owns one or more cellos, plays the cello often, etc. But it might not know that there is a famous cellist called Yo-Yo Ma" [13]. For this project, the TAP knowledge base has been modified to match the formatting of ConceptNet. From the software's perspective, there is a single, large, unified knowledge base.

Stanford TAP, lines 12525 to 12534
(IsA "star wars galaxies" "computer game")
(IsA "eastern college" "university")
(IsA "mario van peebles" "movie director")
(IsA "papasa" "baby products brand")
(IsA "tonga" "country")
(IsA "andrea gabrieli" "composer")
(IsA "citizen dog" "comic strip")
(IsA "columbine meadow rue" "perennial")
(IsA "leipzig" "city")
(IsA "how to write a blackwood article" "book")

Table 1-3 Example of Stanford TAP

Using Commonsense Knowledge to Associate Data with Goals

This thesis demonstrates how the knowledge in MIT's ConceptNet and Stanford's TAP can be used to create software that associates the content of a document with a user's mental model of information, and potential goals.

1.3 As We May Think

Matching a user's mental model and goals to the content found in a document is a powerful idea. The origin of this idea can be attributed to the scientist Vannevar Bush. In his 1945 article *As We May Think*, published in the *Atlantic Monthly*, he describes the idea of hypertext [18]. However, he takes this idea farther than the links we are familiar with on the present-day Web [19, 20]. In the following passage he relates machine created semantic networks to a user's mental model of information. This matching between a user's thoughts and goals and the content of the document is referred to as *Selection by Association*.

The real heart of the matter of selection, however, goes deeper than a lag in the adoption of mechanisms by libraries, or a lack of development of devices for their use. Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. It can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.

The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature.

Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. In minor ways he may even improve, for his records have relative permanency. The first idea, however, to be drawn from the analogy concerns selection. Selection by association, rather than by indexing, may yet be mechanized [18].

Vannevar Bush is clearly speaking in this article about the user's cognitive association of information. However, hypertext as we know it today is not based on the user's associations [19, 20]. Hypertext is based on the author's associations. Vannevar Bush's original vision of *Selection by Association* is centered around the user. His notion of *Selection by Association* is more powerful than the hypertext we are familiar with.

This thesis presents software that has been designed to enable this type of *Selection by Association*, from performing semantic searches on a mobile device, to a Web browser that can dynamically generate personalized, semantic hypertext.

1.4 A Goal-Oriented Web Browser

Using the knowledge in ConceptNet and TAP, Microsoft Internet Explorer has been modified to match the semantic context of a Web page to potential user goals. For instance, imagine a user is viewing a Web page that contains a recipe for Blueberry Pudding Cake. The user's browser will notice a pattern of foods on the page, and present the user with two suggestions: order the foods, or view their nutritional information. When the user selects one of these buttons, all of the foods on the page turn into hyperlinks for the selected action. By pressing the "Order Food" button, each food in the recipe will be converted into a hyperlink for that food at the user's favorite online grocery store. Alternatively, the user can view the nutritional information for each of the foods at their favorite Web site for nutritional information.

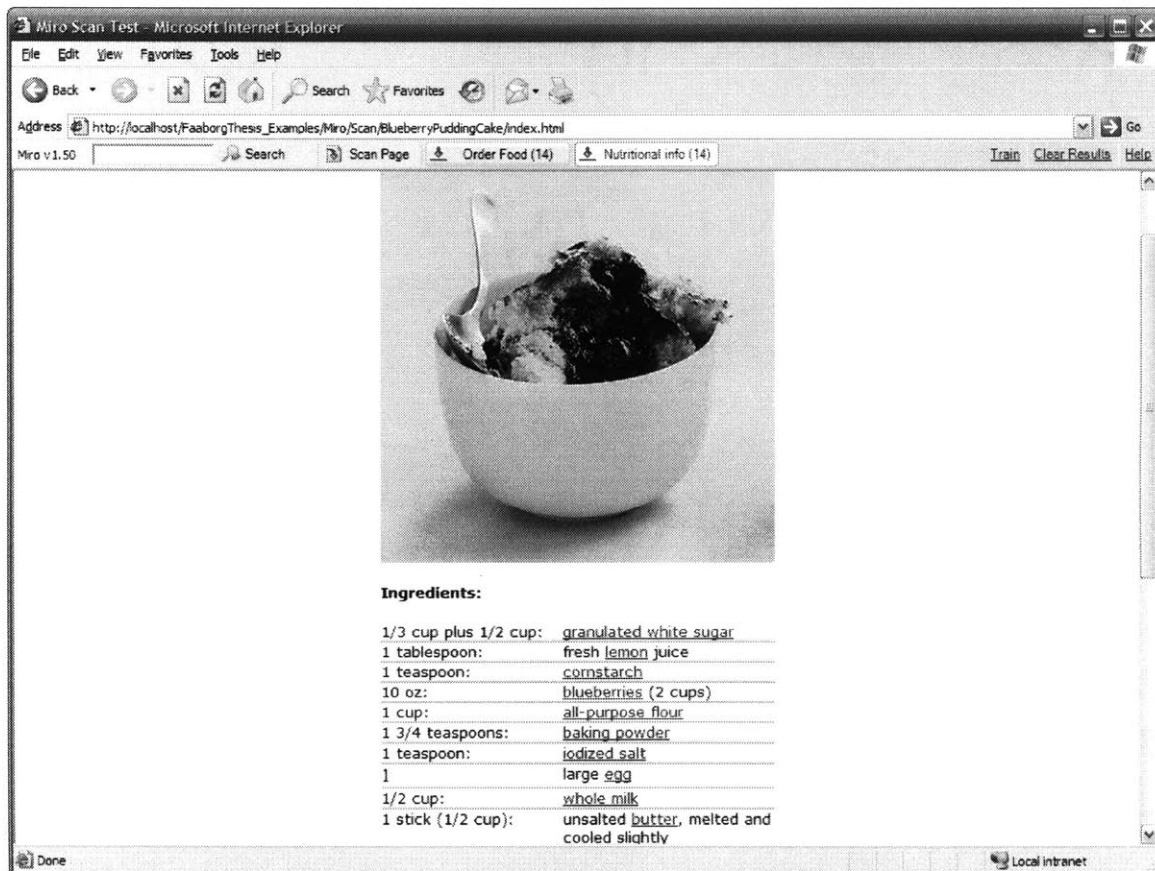


Figure 1-1 A goal-oriented Web browser

After being presented with this example, a critical reader likely has two significant questions: (1) *How does the browser know how to interact with the user's favorite grocery store?* And (2) *How does the browser know which of the terms in the recipe are foods?* The answer to the first question is by enabling users to train a Web browser to interact with their favorite sites using a Programming by Example system named Creo (Latin, "to create, make"). The answer to the second question is by leveraging the knowledge bases of ConceptNet and TAP to create a next generation Data Detector named Miro (Latin, "to wonder").

1.4.1 Using Commonsense Knowledge to Detect Uncommon Goals

Ordering foods from a grocery store is a very common thing to do. However, users often have very uncommon goals. The goal-oriented actions that Miro proactively associated with the previous page, like "Order Food" or "Nutritional Information," were created by the user using a Programming by Example system named Creo. The user could have just as easily used Creo to create a less common association. For instance, the user could associate the names of foods with a search of the CDC's food safety database, a Web site that lists if foods are Kosher, or even an image search of the foods in modern art paintings. Miro is able to use the commonsense knowledge in ConceptNet and TAP to associate data with the user's particular goals, regardless of how common or uncommon their goals may be.

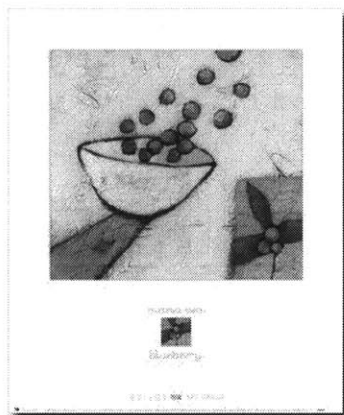


Figure 1-2 Blueberry by Maria Eva

1.5 Taking End Users from the Present-Day Web to Semantic Web Services

The Semantic Web gives information well-defined meaning, better enabling computers and people to work in cooperation [1, 21-25]. The software developed for this thesis is designed to achieve the same goal. However, this thesis approaches the problem from a different direction; it attempts to bridge the rift between the Web and Semantic Web Services by giving the user a more powerful Web browser.

1.5.1 A Different Approach to Semantic Web Services

The current road map to Semantic Web Services relies on organizations adopting standards like RDF, SOAP, OWL and OWL-S, and then recreating their content and services to conform to these standards. This thesis takes a radically different approach, based on leveraging Programming by Example systems, and using the large-scale independent knowledge bases ConceptNet and TAP.

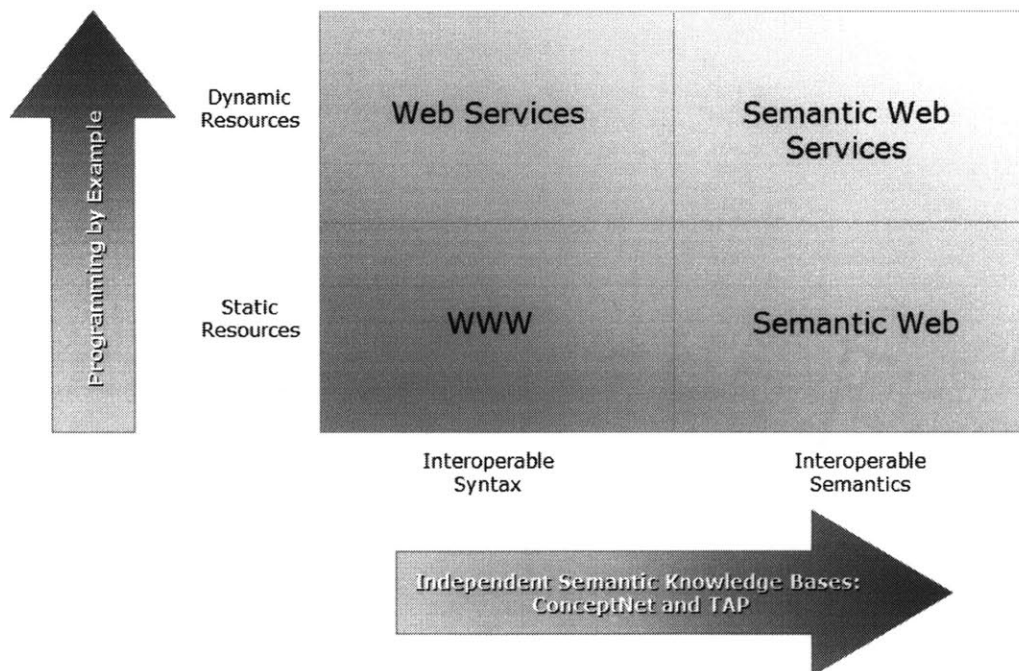


Figure 1-3 A different road map to Semantic Web Services

The aim of this thesis is not to argue against standards like RDF, SOAP, OWL and OWL-S. In an ideal world, every content and service provider on the Web would instantaneously embrace these standards. The purpose of this thesis is to explore how far software agents can get towards a world of Semantic Web Services, given the Web as it exists today. The solution is based on two vectors of research: (1) Programming by Example, and (2) leveraging large knowledge bases of semantic information.

1.5.2 Programming by Example: From Static to Dynamic Resources

Creo is a Programming by Example system that allows users to convert services on the Web into *Web Services*, allowing their browser to automate interactions with their favorite Web sites. The result is a *Favorites* menu for actions, in the same way that current browsers' *Favorites* menu displays a list of commonly viewed documents. Once created, these actions can be manually invoked by the user, or proactively suggested by the user's Web browser or Smartphone.

1.5.3 ConceptNet and TAP: from Syntax to Semantics

ConceptNet and TAP are used in this thesis for two purposes: (1) to automatically generalize the user's input when they are creating recordings with Creo, and (2) to recognize data that can be used in these recordings in the future.

Miro is a Data Detector that uses the ConceptNet and TAP knowledge bases to dynamically markup unstructured HTML documents, or search queries, with semantic metadata. By understanding the meaning of text, Miro can proactively suggest contextually relevant actions to the user.

1.5.4 Building New Interfaces on Top of the Web

By using Creo, users are able to train their computer to automate interactions with their favorite services, like their local bank or local grocery store. Once the computer understands how to invoke these services on its own, the functionality of these services can be exposed in a variety of user interfaces. This thesis introduces two types of interfaces: (1) Miro, a Data Detector that matches the content of a Web page to high-level user goals, and (2) Adeo, an application for Smartphones that streamlines mobile browsing. Additional user interfaces constructed on top of the Web are discussed in the Future Work section of Chapter 6.

1.6 Contributions

This thesis provides the following contributions to the research areas of user interface design, programming by example, data detection, and artificial intelligence.

1.6.1 Contributions to User Interface Design

A User Interface for Recording Actions on the Web

Creo is a Programming by Example system that allows novice users to train their computers to interact with Web sites. Programming by Example systems often extol the virtues of enabling novice users to train their computers, but then end up being simply GUIs for computer scientists. To prevent this from happening, Creo was designed through an iterative design process, and its user interface design was formally evaluated four times.

A Goal-Oriented Web Browser

Miro associates the context of a Web page with a users' high-level goals.

Putting Users in Control of their Data and Services

Creo empowers users to define their own services, and Miro automatically associates them with the users' data. Because users are in control, Creo and Miro avoid the ethical and legal arguments surrounding similar user interfaces like Microsoft's Smart Tags and Google's AutoLink. But more importantly, Creo and Miro can be used to create a much broader range of services than those available by Microsoft and Google.

Personalized Semantic Search

The search feature that Miro provides users is very different from a traditional Web search. Miro's semantic search feature is personalized to each user, and based around their favorite Web sites. Miro is able to

associate the user's search with potential actions by understanding the meaning of the search query. Additionally, Miro leverages what the user is currently looking at to disambiguate search queries. For instance, if the user searches for "the lord of the rings" (both a series of movies, and a series of books), and the user has been looking at books, Miro may suggest purchasing the books as its first result. Conversely, if the user has been looking at movies, Miro may suggest looking up the movies at IMDB as its first result.

Dynamically Generated Contextual Help

This thesis presents a user interface technique referred to as "dynamically generated contextual help." This technique aids the user in creating a mental model of what effect their actions will have in the future.

Intelligent Defaults

This thesis shows how large knowledge bases of semantic information can be leveraged to provide the user with intelligent defaults, streamlining a variety of user interfaces.

A New Approach to Browsing the Web on Mobile Devices

Mobile browsing has not yet caught on in North America and Europe, where WAP was a colossal failure. This thesis presents Adeo, a Smartphone application that has been designed based on the idea that users do not want to "browse" on mobile devices, they want to complete specific tasks in the least number of steps possible. Adeo solves both of the major problems facing mobile browsing: (1) the lack of content on the Web designed specifically for mobile devices, and (2) the constrained user interface that makes completing complicated actions difficult.

1.6.2 Contributions to Programming by Example

Using Semantic Knowledge Bases for Generalization

Knowing how to correctly generalize is crucial to the success of Programming by Example. Past systems have either depended on the user to correctly supply the generalization, or they have attempted to guess the proper generalization using a handcrafted ontology, representing knowledge of a particular, usually narrow, domain. This thesis solves the problem of generalizing procedures by leveraging large knowledge bases of semantic information. This enables users to create a general-purpose recording with a single example.

Automatically Detecting the User's Personal Information

When creating a recording with Creo, the user's personal information is automatically detected and stored in their profile. This enables users to share the functionality of their recordings, without sharing any of their personal information.

1.6.3 Contributions to Data Detection

Increasing the Breadth of Data Detectors by an Order of Magnitude

The Data Detectors created by the research community, like Apple Data Detectors, the Intel Selection Recognition Agent, or CyberDesk, as well as Data Detectors released in commercial products, like Microsoft's Smart Tags and Google's AutoLink, all detect a range of three to thirteen different types of data. This thesis leverages large knowledge bases of semantic information to detect thousands of different types of data.

1.6.4 Contributions to Artificial Intelligence

Learning Semantic Knowledge from the User

Miro is able to learn new information through a simple wizard-based training interface. Miro provides intelligent defaults for as many steps as possible, enabling it to collaborate with the user while learning.

Learning Semantic Knowledge by Reading the Web

Miro is also able to learn potentially new knowledge by reading the Web, and then confirm this knowledge by passively watching the user's actions. This ability enables Miro to steadily increase its knowledge base over time.

Linking Knowledge Capture to the Use of Applications

The impact of linking knowledge capture to the use of an application could have significant implications for the artificial intelligence community if the application became immensely popular. For instance, if as many people used Miro as the Google toolbar, its ability to learn by reading the Web could potentially result in the creation of a very large knowledge base in a very short amount of time.

Semantics with a Messy Knowledge Base

The knowledge in ConceptNet and TAP (to a much lesser extent) is imprecise. This thesis shows that perfect data is not required to successfully match the semantic context of a term to the user's recordings. Creo is able to solve this problem by creating the longest list of generalizations possible, without being inaccurate. This effectively creates a larger target for Miro to hit when matching the context of a page to the user's goals.

Semantics without URIs

Miro performs data detection on Web pages that do not contain semantic markup. Because of this, Miro must deal with the ambiguity of natural language. Miro is able to disambiguate concepts based on their surrounding

context, using a system of patterns and thresholds. This enables Miro to provide semantic markup to unstructured text.

1.7 Towards the Semantic Web

Many of the challenges facing this thesis, including messy knowledge bases, the ambiguity of natural language, and detecting semantic information without URIs (Universal Resource Identifiers), are alleviated by Semantic Web technologies and standards. As the Semantic Web becomes increasingly mainstream, many of the hurdles addressed by this thesis, like an agent's ability to extract information from text or automate a user's actions, may no longer be relevant issues.

1.7.1 Future Need of Programming by Example Systems

In particular, the need for a Programming by Example system will no longer be required if organizations decide to forgo the need to control the user's experience (and provide banner ads for revenue), and describe the functionality of their services using an ontology language like OWL-S [26].

1.7.2 Future Need of Data Detectors

However, extracting semantic information from unstructured text may still be important in the era of the Semantic Web, in particular when dealing with pages on the Web that are not dynamically generated. Tim Berners-Lee states:

The Semantic Web is not about the meaning of English documents. It's not about marking up existing HTML documents to let a computer understand what they say... The vast bulk of data to be on the

Semantic Web is already sitting in databases... It's not as if every page on the Web will be retrofitted with Semantic information. What we are likely to see though, is the wrapping of existing data stores, such as data in relational databases. [25]

While the Semantic Web may not be about the meaning of English documents, this thesis is.

1.7.3 A User Interface for the Semantic Web

Regardless of the potential future adoption of Semantic Web technologies, the user interfaces presented in this thesis will remain relevant. Users do not speak in URIs. Consequently, at an interface level, Semantic Web applications will still have to deal with natural language. This is especially true in the domain of search. This is also true for bodies of text that may not contain semantic markup, like blog and wiki postings.

Additionally, the ideas presented in this thesis regarding streamlined user interfaces for mobile browsing are relevant regardless of the underlying technology the agent uses to invoke services for the user.

Chapter 2

User Scenarios and Discussion

2.1 Introduction

2.1.1 Example User Scenario: Pete Studies for an Exam

The stereo was belting out The Killer's "Somebody Told Me" when Pete's² Smartphone rang. Pete hit pause and answered the call. Turning to his friend Ian, he said, "It's Sara. She's going to come over and study with us." Ian nodded with a smirk and looked down at his textbook. "I'm getting hungry, let's order a pizza." "Sure," Pete said, typing *food* into his Smartphone. "How about we order from the pizza place down the street?" Pete clicked *Go* on his Smartphone, instructing his agent to take care of the details. Meanwhile during the quick pause in studying, Ian was using his Smartphone to obtain the song that was playing. "Are you still getting music from that illegal Russian MP3 site?" Pete said disapprovingly. "No," Ian responded smiling, "my agent is." Pete let out a sigh and said, "You know, one of these days you should train your agent to buy music from a site that is actually legal." A moment later they were back to studying. The pizza had already arrived by the time Sara showed up. "How much do I owe you for the pizza?" she asked, taking out her Smartphone. "Like \$3." Sara typed *pete@mit.edu* into her Smartphone and her agent displayed a list of actions

² As Pete grows older, the Web will transition into the Semantic Web [1]. Pete will have a more advanced mobile agent, and a considerably fancier entertainment system.

that could be performed on *Pete*. "PayPal ok?" "Sure," Pete responded. After a long pause Ian figured out that they were waiting for him to pay for the pizza as well. "I have a PayPal account...but I never showed my agent how to interact with their Web site. Mind if I use your computer?" "We need to get started studying," Sara replied. "I'll have my agent train yours." "Yeah, I suppose I can trust you," Ian said. "Ok, got it." Sara pulled up a chair, and they continued preparing for the exam.

Pete, Ian and Sara were able to carry out these tasks using the Web as it exists today, along with mobile software agents running on their Smartphones. The agent introduced in this example is able to automate interactions with conventional Web sites by monitoring and learning from the user's actions, being programmed by example. The agent is also able to anticipate a user's actions by using a large knowledge base of commonsense facts about the world to understand the semantic context of their user's commands. This chapter will discuss how the agent accomplished these tasks in detail.

To be useful, an agent must be able to (1) carry out actions for its user, and (2) understand the semantic context of information. In the following sections, providing an agent with these two core abilities will be explored, given the Web as it exists in its present form.

2.1.2 Creo, Miro and Adeo

To achieve the types of interactions discussed in the preceding user scenario, three pieces of software have been created: Creo, Miro and Adeo.

Creo

Creo (Latin, "to create, make") is an explorer bar for Internet Explorer, designed to monitor a user's interactions with a Web site. By monitoring the way a user interacts with a site, Creo can wrap the functionality of the site

into a service that an agent can later invoke on its own, without the user's guidance. Creo uses a metaphor of recording and playback, similar in style to the macro recorders found in other applications.

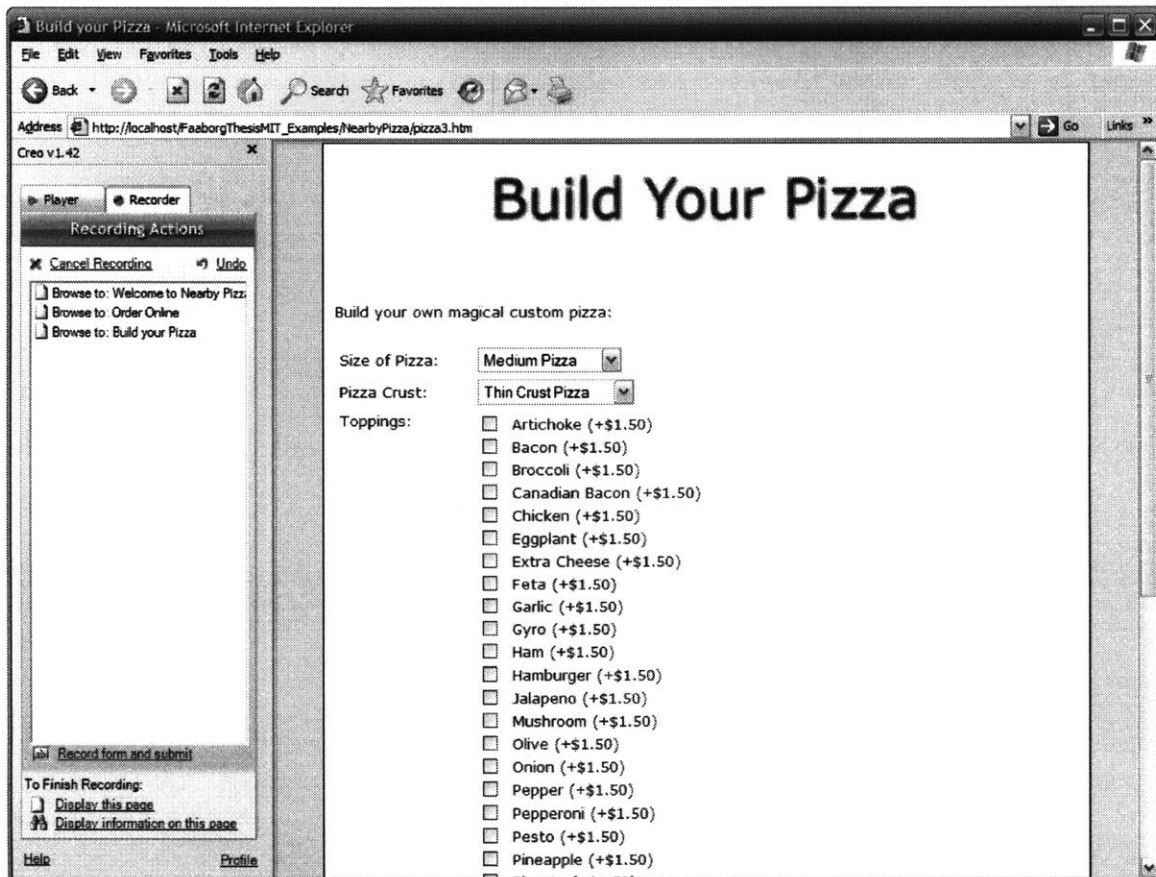


Figure 2-1 Creo

Miro

Miro (Latin, "to wonder") is a toolbar for Internet Explorer, which attempts to match the semantic context of the user's current Web page to potential user goals. Miro provides the user with two ways to invoke recordings created by Creo: (1) the ability to scan a Web page for information that can be used with recordings created by Creo, and (2) the ability to perform semantic searches against the recordings created by Creo. Miro also has the ability to learn the relationships between concepts by reading the Web. This learning is performed in cooperation with the user, and is discussed in detail in Section 2.4.



Figure 2-2 Miro

Adeo

Adeo (Latin, "to undertake") is a mobile application that runs on Smartphones. Adeo provides the user with the same type of semantic search features found in Miro, except on a mobile device. Adeo's ability to reduce procedures to only the absolutely necessary input and output allows users to quickly and efficiently complete actions on the Web using a mobile device, without ever having to "browse." The advantages of this type of streamlined interface were demonstrated several times in the earlier example.

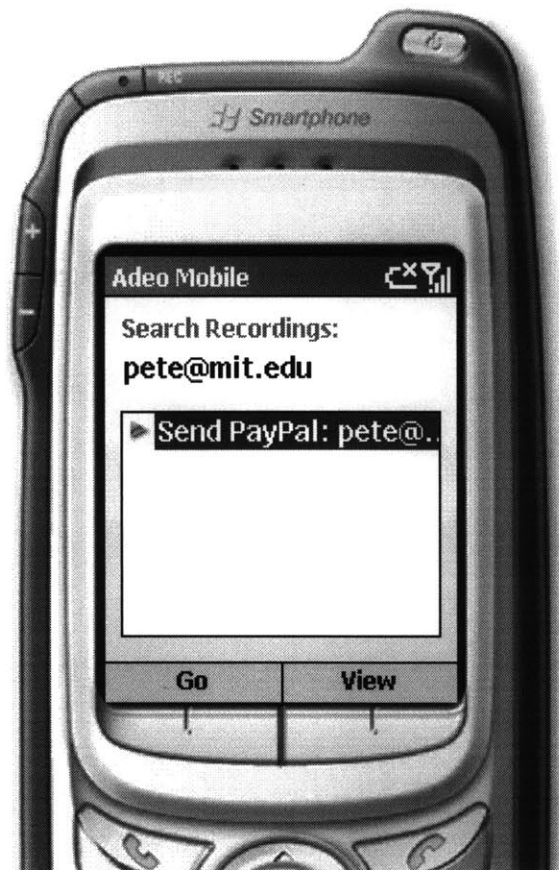


Figure 2-3 Adeo

2.2 Creating Recordings

2.2.1 Recording a Simple Procedure with Creo

Creo's interface appears on the left side of Internet Explorer, and it has two tabs: *Player* and *Recorder*. The *Player* tab provides a list of recordings, and serves as the user's *Favorites* menu for actions, in the same way that current browsers' *Favorites* menu displays a list of commonly viewed documents.

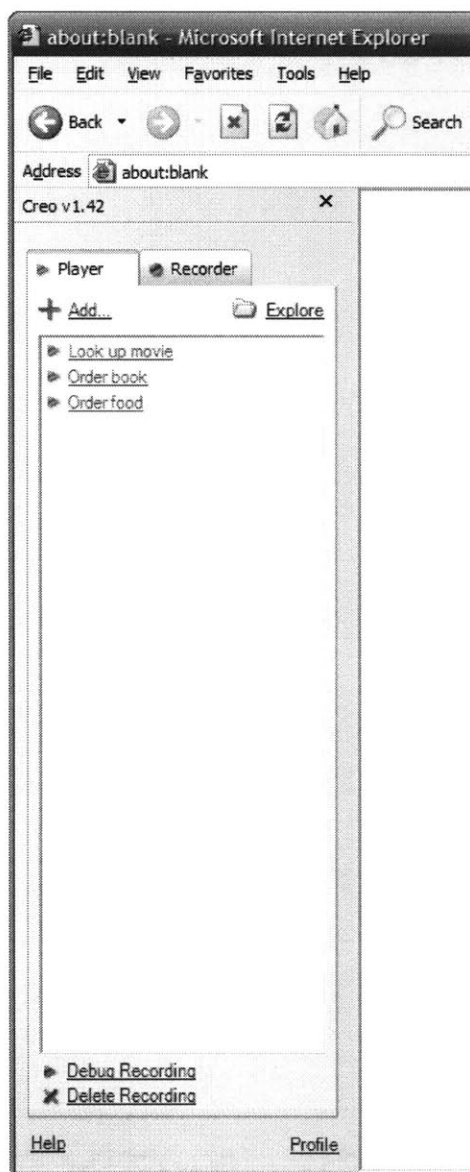


Figure 2-4 Creo's *Player* tab

To create a new recording, the user can either switch to the *Recorder* tab, or click the *Add...* hyperlink.

To demonstrate the functionality of Creo, we will start by recording a simple procedure: checking the balance of a bank account. First, the user switches to the *Recorder* tab.

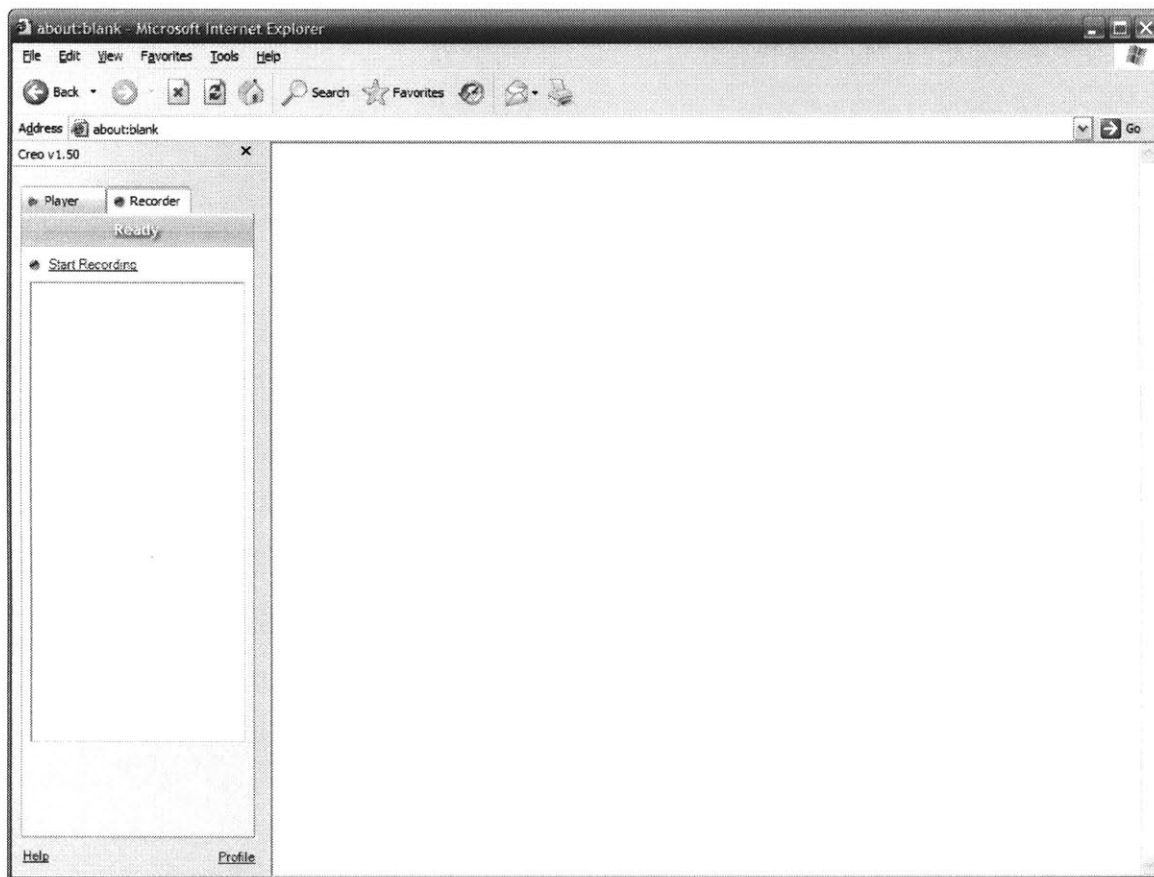


Figure 2-5 Creo, ready to start recording

The user then clicks *Start Recording*, and navigates to the home page of their bank.

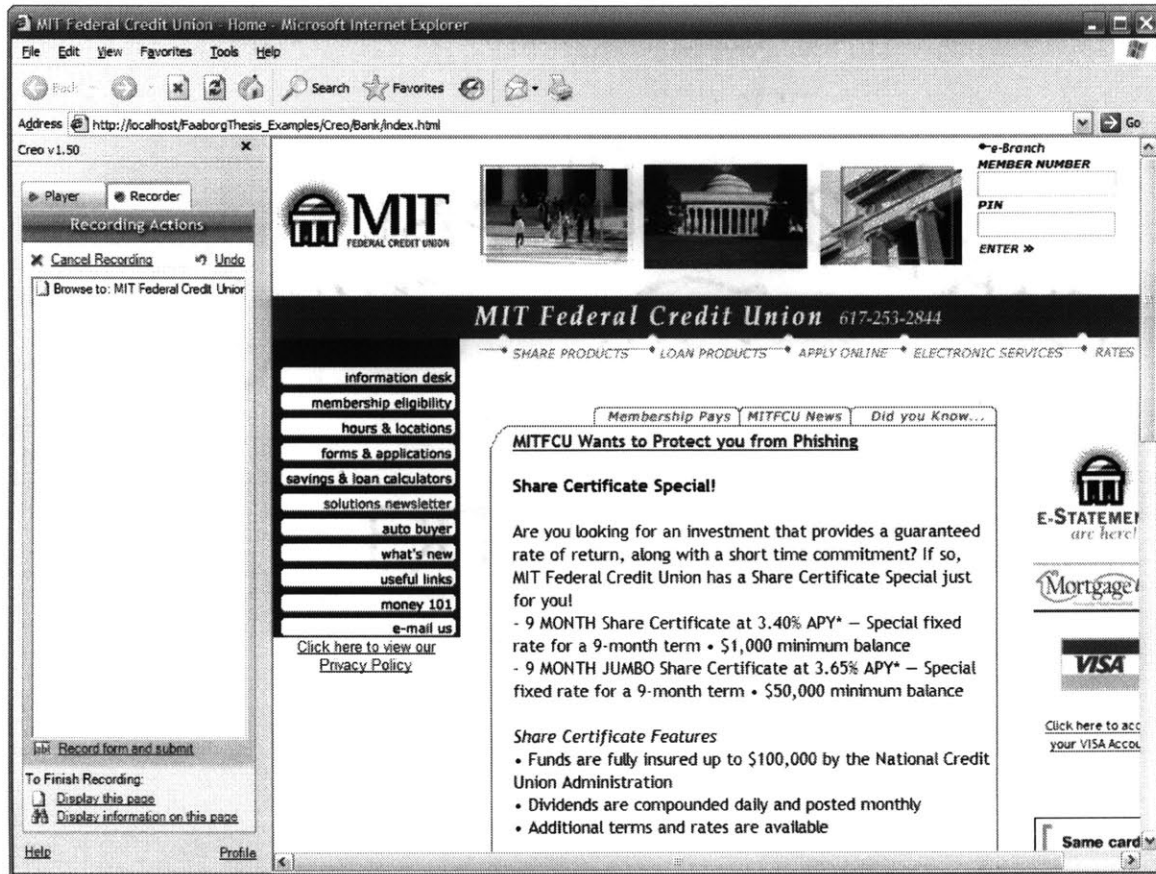


Figure 2-6 Creo, recording a simple procedure

Creo is now recording the user's actions. The *Recorder* tab has turned red, and the first action has been added in the action list: *Browse to: MIT Federal Credit Union*. As the user interacts with the site, each action the user takes will appear in the actions list.

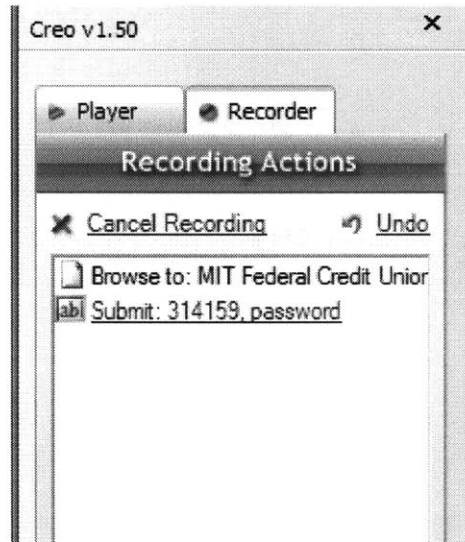


Figure 2-7 Creo, action list

The icons next to each item in the list represent the type of action. Creo supports four kinds of actions:





Icon	Action Type	Description
	Navigate	The user clicks on a hyperlink, or enters a URL in the browser's address box.
	Form Submit	The user submits information in a Web form.
	Return Page	Final action, the recording ends by displaying the current page.
	Return Value	Final action, the recording ends by displaying a piece of information on the current page.

Table 2-1 Types of actions

The user has logged into their bank account, so they are now ready to complete the final step of this simple three-step procedure.

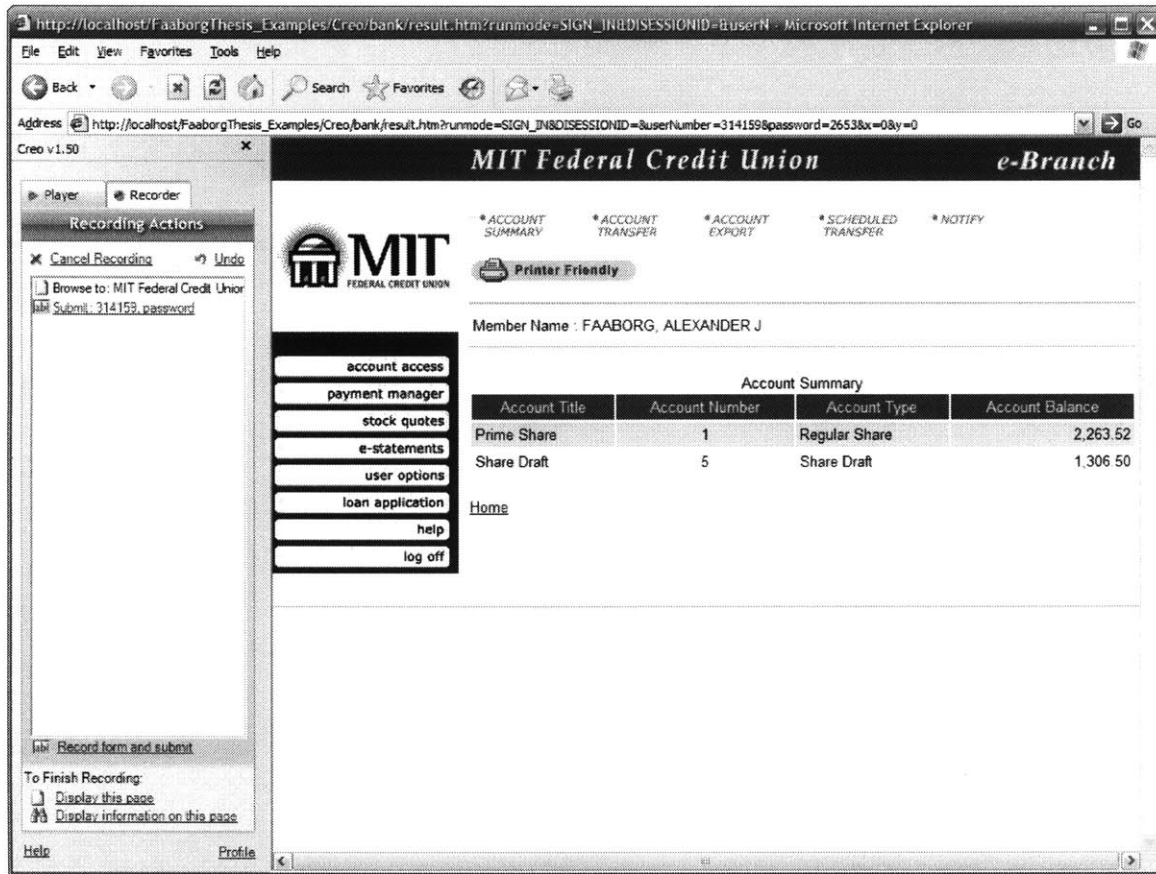


Figure 2-8 A graduate student's "hypothetical" bank account

The user can complete the recording by selecting *Display information on this page* at the bottom of the *Recorder* tab. A window appears asking the user to enter the information they wish to display. This trains Creo to scrape information from the page. When the value changes, Creo will return the changed value.

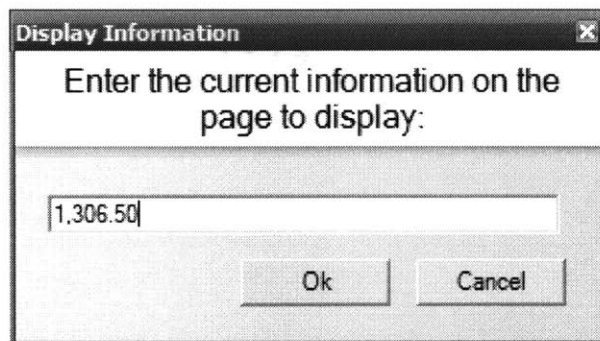


Figure 2-9 Training Creo to scrape information from a page by giving it an example

The user is then asked to enter their goal for using this recording.



Figure 2-10 The user enters a name for the recording

The text they enter here will appear in the list of recordings on the *Player* tab, as well as in the Miro toolbar (if they search for the recording), or in Adeo's list of recordings (if they wish to invoke the recording while they are away from their computer). The *Recorder* tab enters the *Recording Complete* state, and the user now has the option to test, debug, or delete the recording.

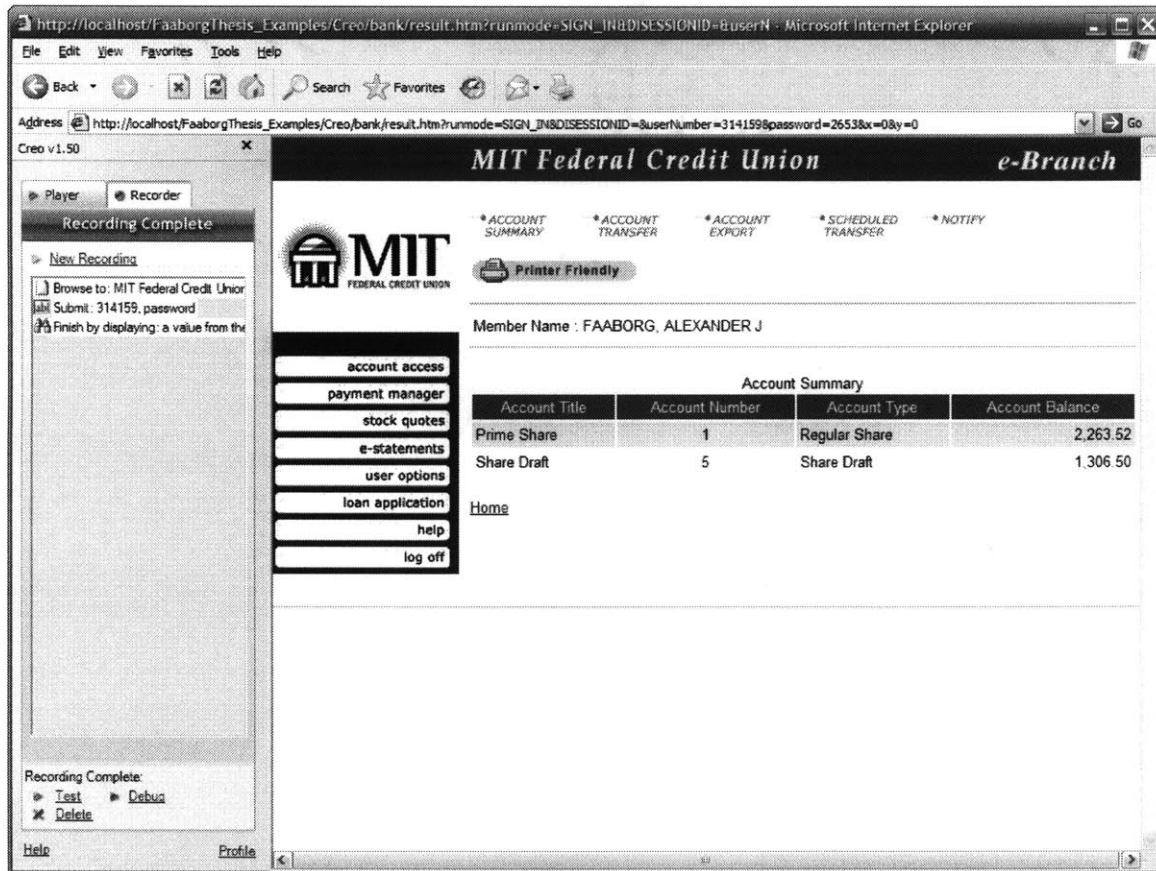


Figure 2-11 The recording is complete

2.2.2 Recording a Procedure with Generalized Input with Creo

In the previous example, the only information that changes when playing the recording is the recording's output. When the recording is called, it will return the varying (and often declining) balance of the user's bank account. In this next example, a piece of input (and output) will change each time the recording is played. We will see how Creo intelligently tells the difference between static and dynamic input, and how it uses MIT's ConceptNet and Stanford's TAP to automatically generalize the variable input.



Figure 2-12 The user browses to FreshDirect

The user begins this recording by browsing to FreshDirect, their favorite online grocery store. Next, the user enters “diet coke” into the search box, and submits the form. The user does not want to create a recording that always orders “diet coke;” they are simply providing an example. Creo is able to both correctly detect that “diet coke” is an example (rather than a piece of static information) and that it generalizes to a type of “food brand.”



Figure 2-13 Creo correctly recognizes example input and generalizes it

Submit information actions always appear as red hyperlinks in the *Recorder* tab's action list. By clicking on one of these links, the user can view more information about how Creo will treat each piece of information in the form.



Figure 2-14 Creo defaults "diet coke" to a type of "food brand"

The *Play* tab displays options to control of a piece of information should be generalized. On the *Play* tab, the user is given two options: *Always enter the text:* "diet coke," and *Ask me to enter a:* "food brand." For this particular recording, the generalization is correct. However, if for some reason the user did want the information "diet coke" to remain static, they could make that selection with the radio buttons.

Creo provides an intelligent default on this choice using a number of parameters, including:

1. If the text is a piece of static personal information about the user, like their name or email address.
2. The number of generalizations of the text may determine if it should remain static. For instance, "diet coke" generalizes to five concepts, while the text afaaborg (a username) does not generalize to any concepts.
3. If the name of the text field matches a predetermined list of fields that should remain static. This list includes field names like "user," "username," and "userNumber."

The *Scan* tab contains options for when the agent should proactively suggest this recording to the user, while the user is browsing the Web. By selecting the *Scan* tab, the user can see how Creo has automatically generalized the concept "diet coke." These generalizations appear in a list, and they can be activated or deactivated with checkboxes. The list is ordered with the most general concepts at the top.

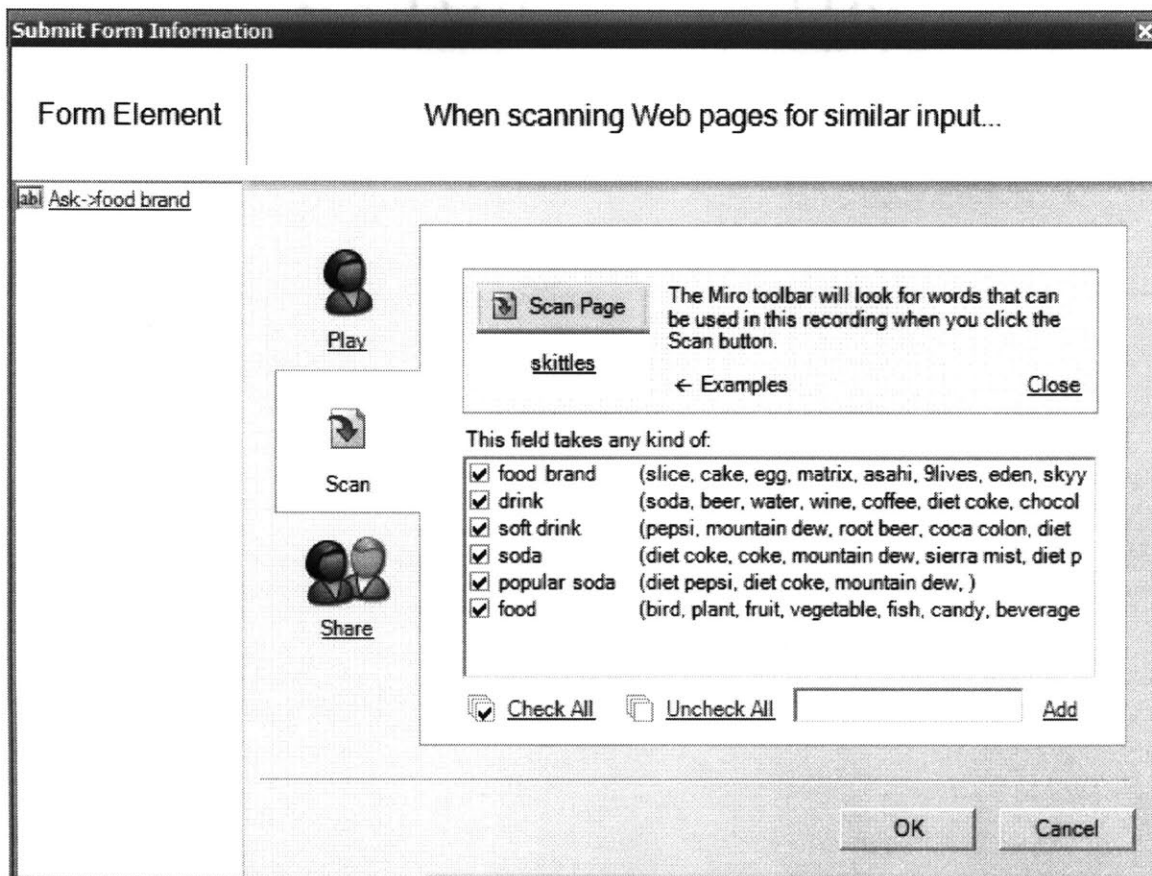


Figure 2-15 Generalizations of the concept "diet coke"

The Miro toolbar will look for concepts that can be used in this recording. Examples of the types of concepts Miro will activate are displayed to the user along with an explanation. The example displayed cycles every 500 milliseconds, so the user can quickly get a sense of if Creo has correctly generalized the information. Additionally, examples are provided in the list after each concept. By automatically providing users with lots of instances of the generalizations of the input they provided, users do not have to think abstractly. Instead of asking the user to generalize the input themselves, they simply have to decide if terms like "cake," "egg" and "skittles" also make sense.

Ideally, users will never need to view the options displayed in the *Submit Form Information* window. Creo makes every effort to provide intelligent defaults for each piece of information. For each piece of input in a recording, Creo attempts to provide intelligent defaults to the questions:

1. Should this information remain static?
2. Is this piece of information an example of information that will change in the future?
3. What other types of information could replace this information in the procedure?
4. Is this a piece of personal information about the user? (the implications of this decision are discussed in Section 2.5)

Creo will obviously not always come to the correct conclusions when attempting to answer these questions on the user's behalf. However, when Creo can answer these questions correctly, the task of creating recordings becomes as simple as completing a task on the Web and allowing Creo to passively watch the actions.

Once the user has created the recording, they can select it from the *Player* tab. If the recording requires any input, the user is then asked to provide it.

In this case, the user is asked to provide a "food brand," the most general description of their example, "diet coke."

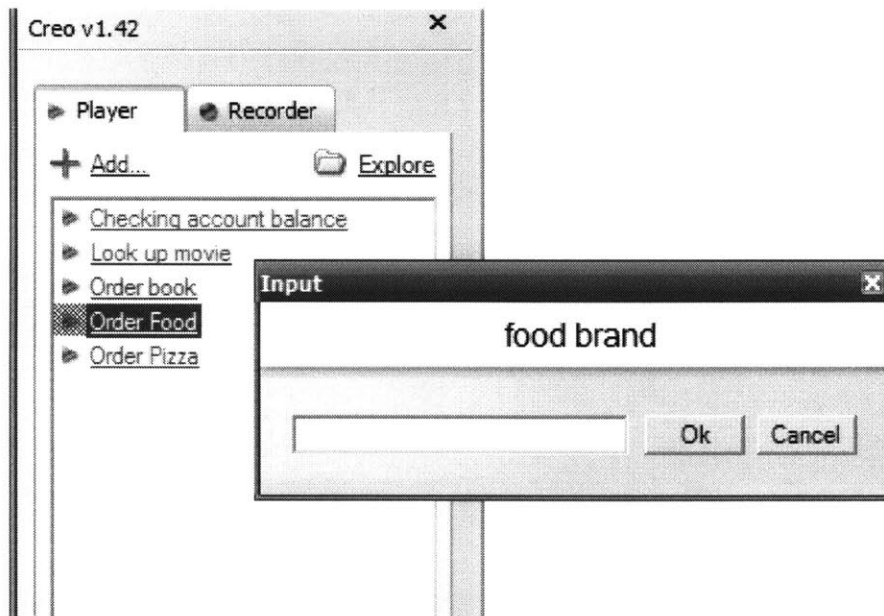


Figure 2-16 Creo prompts the user for input when directly playing a recording

For this recording, performing a search on a generalized piece of information (any kind of food) was the first action in the recording. However, more complicated tasks may require generalized input later in the recording. For instance, creating a recording to set up a stock purchase might involve (1) navigating to E*Trade, (2) logging into the site, (3) navigating to the "Trading & Portfolios" tab, (4) selecting "stocks," (5) selecting the order type (buy, sell, sell short, etc.), (6) entering the price type (market, limit, stop etc.), and then, (7) entering the ticker symbol, which can be generalized. Creo can be used for short tasks, like preparing to buy food from a grocery store, or tasks that involve many more steps, like setting up a stock purchase.

2.3 Using Recordings

2.3.1 Using Miro to Match Recordings against the Semantic Context of a Web Page

When playing recordings from Creo's *Player* tab, users must provide required input because Creo doesn't have any context to assume what action they want to take.



Figure 2-17 Without any context, Creo must ask the user for input (not so grrreat)

The Miro toolbar can scan the Web page the user is viewing for concepts that match the user's recordings. If it finds a match, it can replace the appropriate concepts on the page with hyperlinks that execute the appropriate recording. To instruct Miro to look for matches between the concepts on the page and the current set of recordings, the user hits the *Scan Page* button.

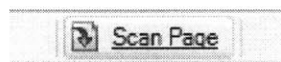


Figure 2-18 Using Miro to scan a page for concepts

Miro will then break the page apart into phrases and words, and look up each using MIT's ConceptNet and Stanford's TAP. The semantic context of each word is then matched against the list of generalizations for each piece of input for each of the user's recordings. If Miro's learning features are turned

on (described in Section 2.4), Miro will also attempt to perform natural language processing on the page, looking for new knowledge.

After the user hits the *Scan Page* button, Miro will display a list of detected recordings, represented as buttons in the toolbar. For instance, if the user was looking at a recipe, Miro would recognize that there are many foods listed on the page and display the *Order Food* recording, which was created in the previous section.



Figure 2-19 Miro suggests the *Order Food* recording

When the button is pressed in, all of the foods in the page will turn into hyperlinks to the *Order Food* recording. The color of the hyperlinks matches the color of the text in the pushed in button. This color is randomly generated when the recording is created from a pallet of colors used in the Windows XP Luna theme, avoiding any shades of blue. This color will remain consistent throughout the life of the recording so that users can build up a mental model of which colors match to which recordings, reducing mode errors.



Figure 2-20 The user clicks the *Order Food* button

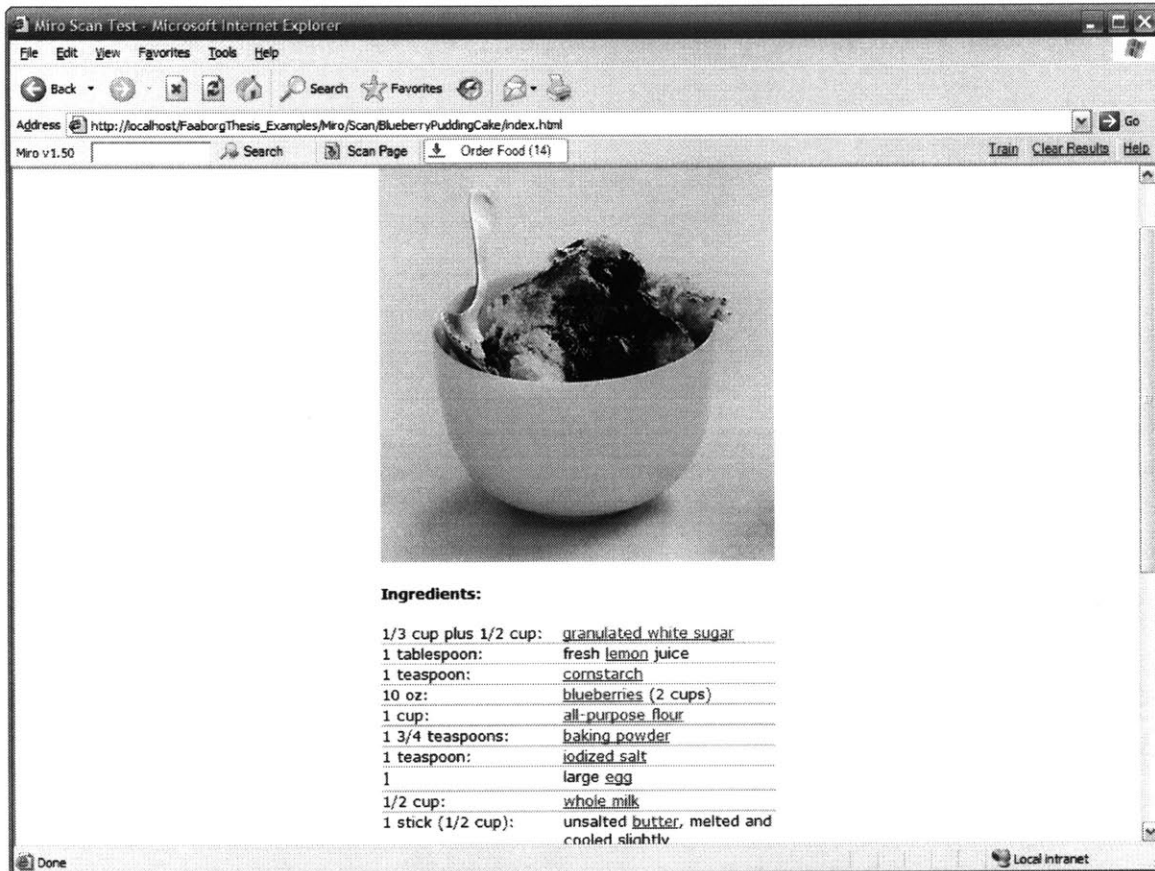


Figure 2-21 All of the foods in the recipe turn into hyperlinks to the Order Food recording

If more than one recording is matched against the semantic context of the page, then the user can select which recording to use. Only one recording can be selected at a time, and the activated concepts on the page, and their color, will change depending on the selected recording.

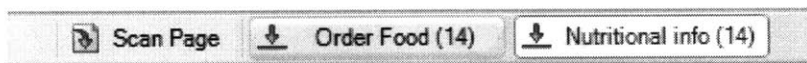


Figure 2-22 Two recordings match the semantic context of the page

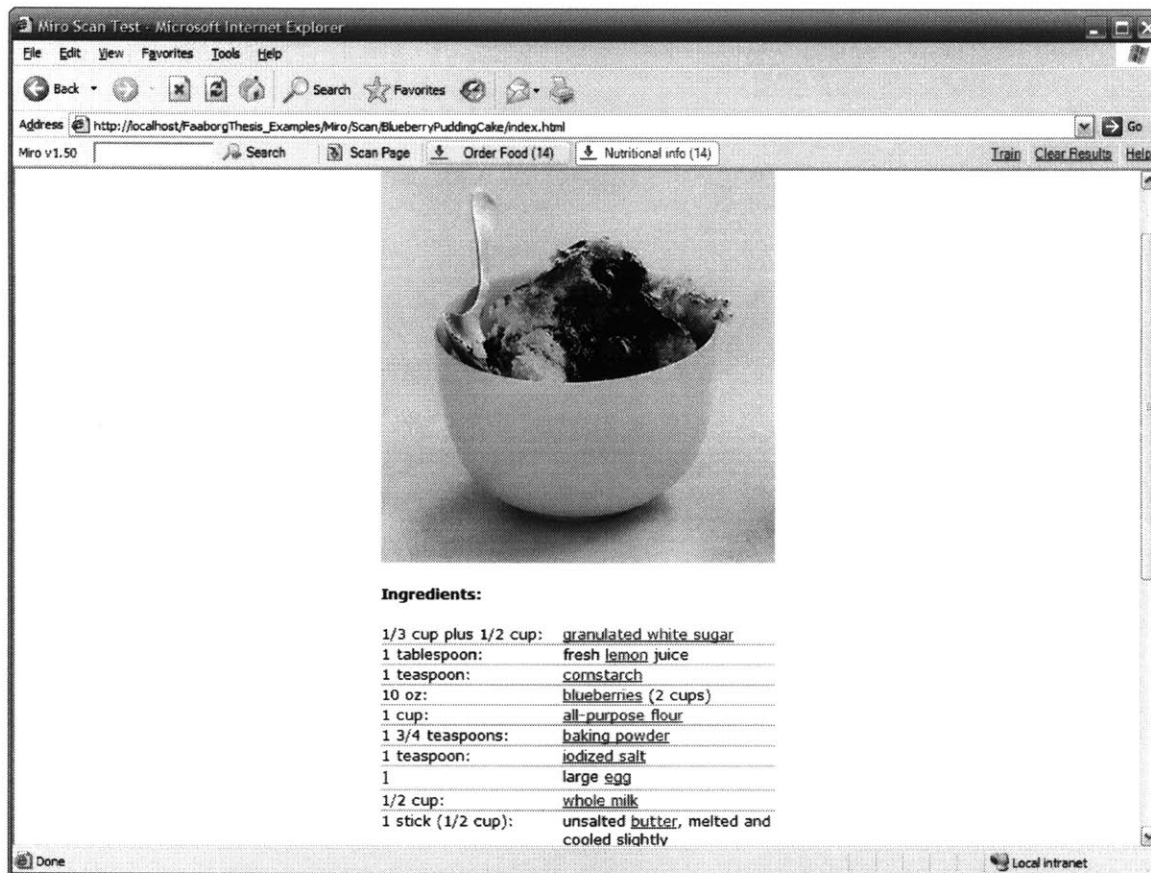


Figure 2-23 The user selects the Nutritional Info recording instead of the Order Food recoding

A Web without Semantics

It is important to note that there are no semantics on the Web page itself. All of the semantic information is coming from ConceptNet and TAP. The vision of the Semantic Web often takes a different approach, assuming that authors will provide semantic markup when they are authoring documents and services. For instance, the original Semantic Web article in Scientific American [1] describes such a scenario:

"These semantics were encoded into the Web page when the clinic's office manager (who never took Comp Sci 101) massaged it into shape using off-the-shelf software for writing Semantic Web pages" [1].

Ideally, semantics would be added both proactively by the author, and retroactively by a user's Web browser. However, the Web in its present form currently contains no semantic information (aside from a few exceptions, like James Hendler's home page [27]). Additionally, the recent popularity of blogs, wikis and message boards has resulted in large portions of the Web being created without rich client software. In the interim between the Web and the Semantic Web, large knowledge bases of commonsense facts like ConceptNet and TAP can provide semantics to a Web that has none.

2.3.2 Putting Users in Control of their Data and Services

Miro's ability to turn text into hyperlinks is similar in function to Microsoft's Smart Tags, and Google's AutoLink feature, as well as a number of Data Detectors discussed in Chapter 4.

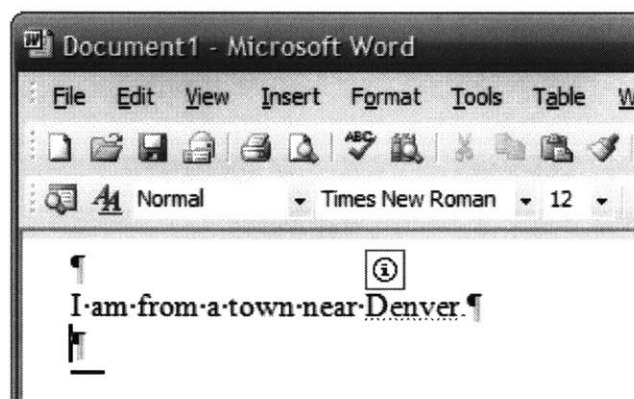


Figure 2-24 An example of Smart Tags in Microsoft Word

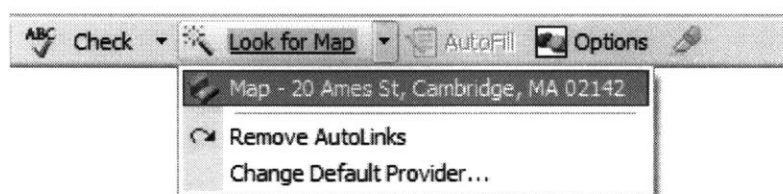


Figure 2-25 Google's AutoLink feature

Limitations of Smart Tags and AutoLink

Unlike Miro however, both Microsoft Smart Tags, and Google AutoLink have a very brief list of types of concepts that they can detect. In the case of Smart Tags, the user is limited to eight different types of concepts, listed below.

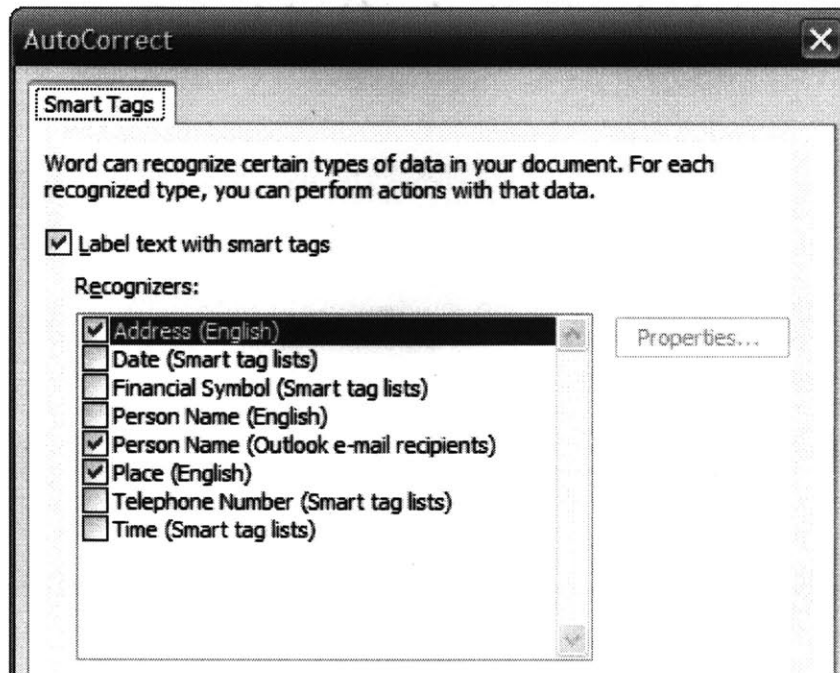


Figure 2-26 Types of information detected by Smart Tags

It is possible for software developers to create new Smart Tags, and then either distribute them inside their company, or sell them. However, end users are not in control of defining what types of data are detected, or defining the services that their data connects to.

Both limitations are also true for Google's AutoLink feature. The current Google Toolbar only detects addresses, ISBNs, and VINs. For each type of information, the user can select from a predetermined list of two to four service providers.

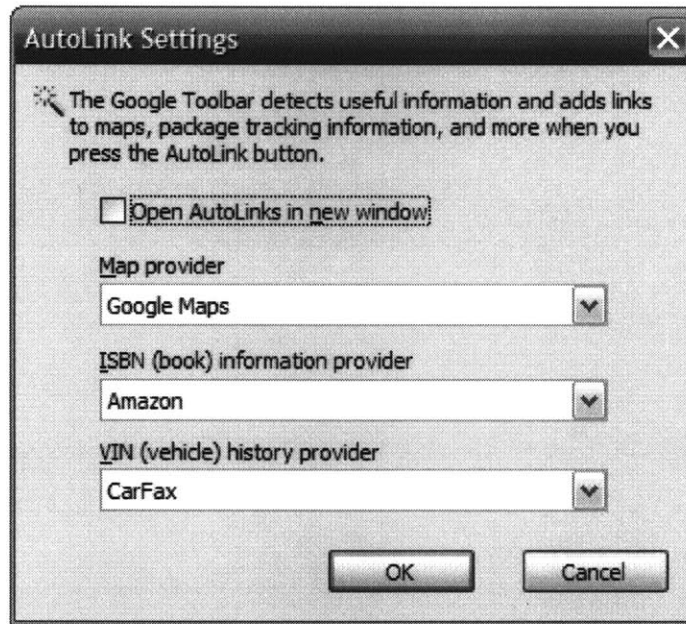


Figure 2-27 Types of information detected by AutoLink

Criticisms of Smart Tags and AutoLink

The reason that users are not in control of specifying which services (beyond those listed) they would like to associate with various types of data is because doing so would require some type of end user programmer environment (like Creo). Additionally, both Google and Microsoft have a business incentive to direct users to their particular services.

Both companies have taken a considerable amount of criticism for controlling the set of available services. While Microsoft has released Smart Tags in its office products, the feature was pulled from Internet Explorer 6 directly before the release of Windows XP due to a strong public outcry. Critics believe that it would be inappropriate for Microsoft to leverage its position in the operating system market to redirect users to its MSN Web properties [28]. In an article in the Wall Street Journal, columnist Walter Mossberg wrote, "Using the browser to plant unwanted and unplanned content on these pages -- especially links to Microsoft's own sites -- is the equivalent of a printing company adding its own editorial and advertising messages to the margins of a book it has been hired to print. It is like a television-set maker

adding its own images and ads to any show the set is receiving.... Microsoft executives have done the right thing this week in killing the feature” [29].

Google received a similar level of criticism for its AutoLink feature, and many critics related the feature to Microsoft’s actions four years earlier. Leslie Walker of the Washington Post wrote, “There must be limits to Google’s power to steer us around the Internet -- otherwise Google might morph into a meaner monopolist than Microsoft.” [30]. Technology journalist Dan Gillmor wrote, “What Google isn’t taking into account is that its market power, and the tendency of users to accept the default -- to eat what’s on the plate someone puts in front of them -- will tend to create Google’s version of the Web, not the users’ version.” [31].

Legality of Smart Tags and AutoLink

The legality of features like SmartTags and AutoLink is also questionable. Robin Gross of the Electronic Frontier Foundation points out that changing a Web page creates a derivative work of the original content, and therefore violates US copyright law [32]. However, Cory Doctorow (also of the Electronic Frontier Foundation) states that as long as the content is not being redistributed, users are free to change it [33]. What is abundantly clear is that US copyright law was not written with a Web browser’s cache in mind.

The Need for Broad Knowledge Bases of Instance Data

While control over available services has business implications, control over the type of data that can be recognized is due to technical limitations. The reason that users cannot control the types of data that are detected by Smart Tags or AutoLink is due to the fact that neither company has integrated their application with a broad knowledge base of instance data like ConceptNet or TAP.

Putting Users in Control

Creo and Miro empower users to define their own services. Because users are in control, Creo and Miro avoid the ethical and legal arguments surrounding similar interfaces like Smart Tags and AutoLink. But more importantly, Creo and Miro can be used to create a broader range of services than those available by Microsoft and Google. Users are free to record interactions with any site on the Web. And these recordings can be associated with any type of data that ConceptNet and TAP know about (movies, musicians, foods, athletes, universities, sports teams, books, etc.).

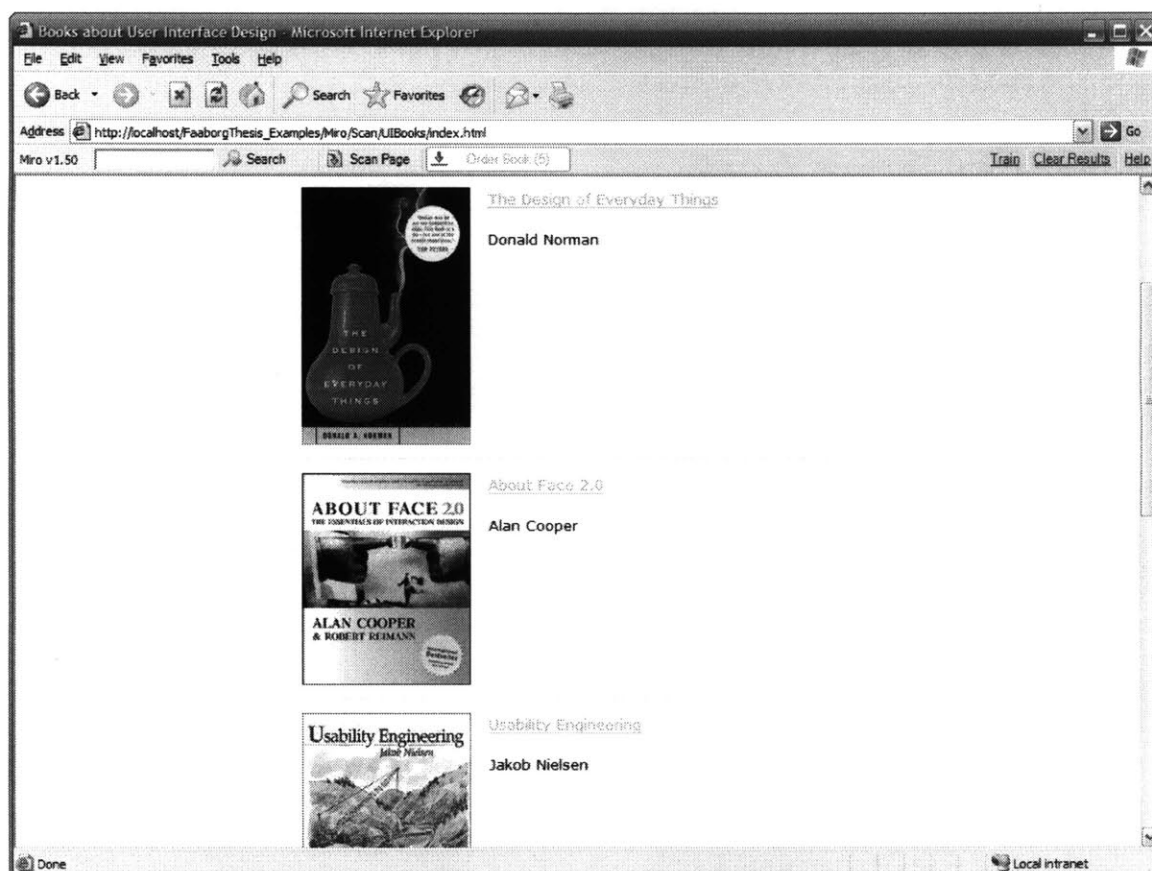


Figure 2-28 Using Miro's Scan feature to link the titles of books to the user's favorite book store

Creo and Miro use the ConceptNet and TAP knowledge bases to handle the details of associating particular types of data with recordings created by the user. Because of this, users can create a general-purpose recording by providing a single example.

2.3.3 Creating a General-Purpose Recording with a Single Example

In Section 2.2, the user created a general-purpose recording for ordering food with the single example of "diet coke." The user was able to do this because Creo automatically generalized the input.

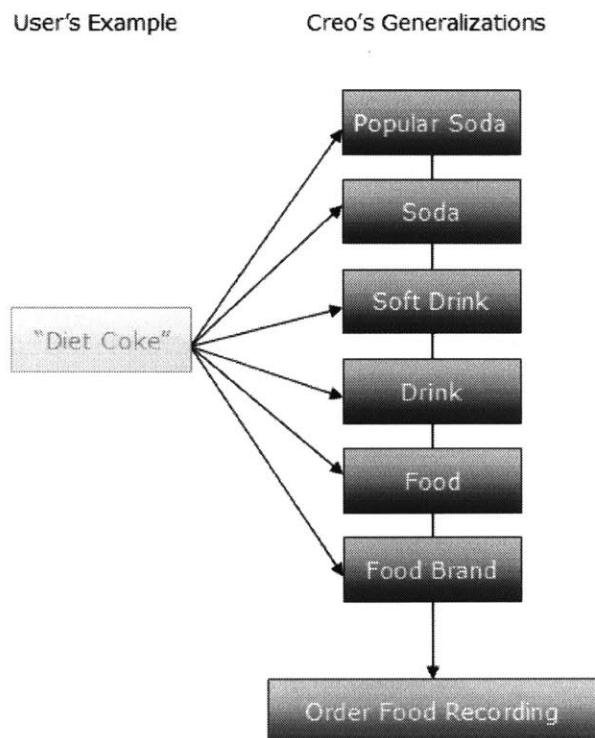


Figure 2-29 Creo automatically generalizes the user's input

When users click the *Scan Page* button in Miro, it performs a similar operation on each word and phrase on the page.

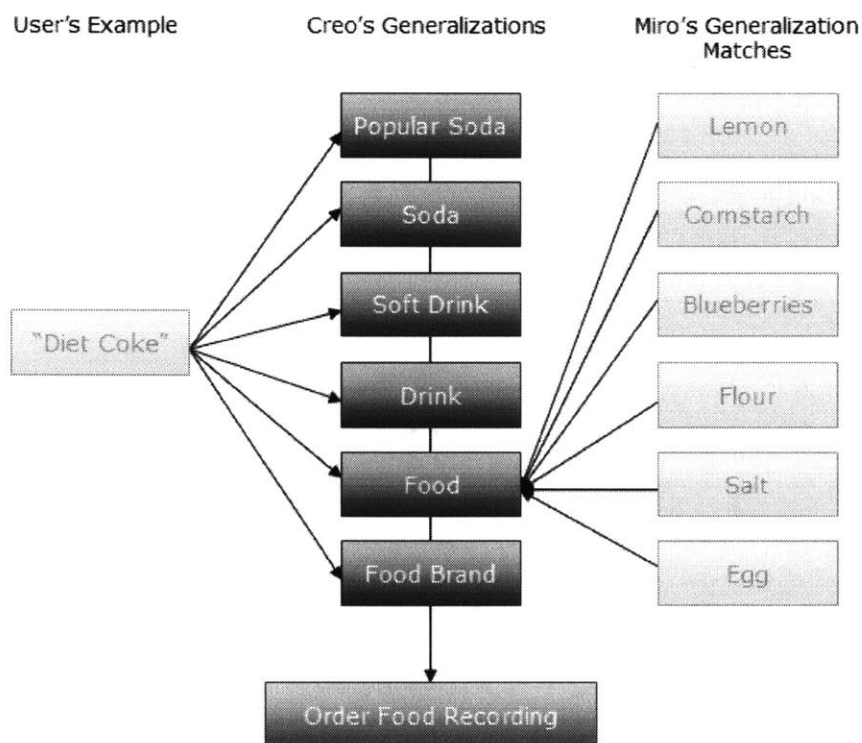


Figure 2-30 Miro matches concepts on a Web page against the current set of recordings

In this case, a large number of concepts matched the generalization of "Food." Because both Creo and Miro have access to ConceptNet and TAP, users can create general-purpose recordings in a very small amount of time. While the recording was only trained on the concept of "diet coke," the recording will now activate for any type of food that ConceptNet and TAP know about. Even simple recordings like the "Order Food" example can save users a considerable amount of time. An evaluation of Creo and Miro based on this same scenario is described in detail in Chapter 5.

Semantics with Messy Knowledge

The knowledge in ConceptNet and TAP (to a much lesser extent) is imprecise, however perfect data is not required for successfully matching the semantic context of a term to the generalizations of a recording's input. This is because a recording's list of generalizations attempts to be as long as possible, without being inaccurate. In the previous example, a list of ingredients in a recipe all generalized to the same concept, "Food." In other scenarios, the matching may be more chaotic. For instance, let's imagine the user is scanning a Web page about music, and their "Buy Music" recording was trained on the infamous search example of "Britney Spears."

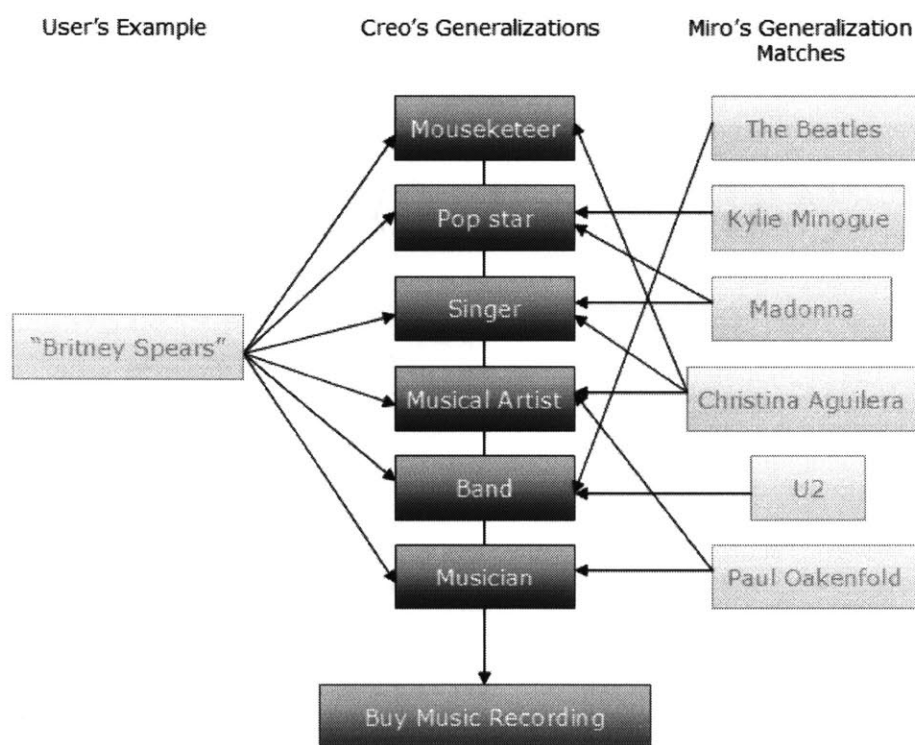


Figure 2-31 Miro generalizes musicians using a messy knowledge base

Because of the way the knowledge in ConceptNet was collected, some areas of knowledge are incomplete, and sometimes similar terms are used for the same concept. Anyone who designs ontologies or taxonomies for a living

would be very displeased with the organization of ConceptNet. They would likely react, "What is the difference between 'musician' and 'musical artist'? Which is correct? Why aren't all 'musical artists' also 'musicians'? Who built this thing?" There is no central authority or architect to the knowledge in ConceptNet: it has grown organically. Because of this, when Creo is generalizing a concept, its objective is not to find the "correct" generalization, but to create the longest list of generalizations it can (that are also accurate). Creo is trying to build the largest target possible, so that Miro has the best chance of correctly matching concepts to the recording. When creating a list of generalizations for a recording, Creo can make simple decisions about which generalizations are more useful than others by analyzing how connected a node is in ConceptNet's semantic network. For instance, generalizing "Britney Spears" to "mouseketeer" is less useful than the generalization of "musician," because there are thousands of musicians in the knowledge base, and considerably fewer mouseketeers.

Because the nodes in ConceptNet are expressed in natural language, Creo and Miro must also deal with ambiguous concepts.

Semantics without URIs

The Semantic Web relies on URIs to define specific entities. URIs nullify the ambiguity of natural language. For instance, if someone says "apple" (without any additional context) it is not clear if they are referring to the tasty fruit, or the stylish computer company.

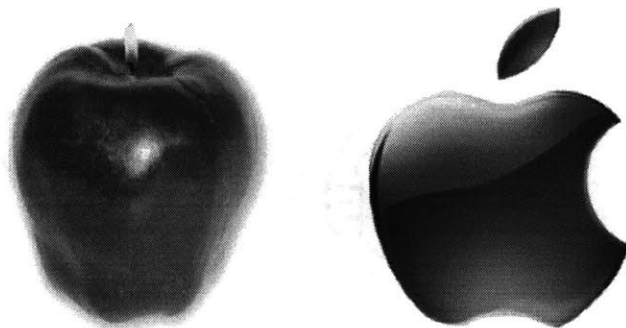


Figure 2-32 I want to buy an apple

The simple solution to this dilemma is to use URIs to describe each concept:

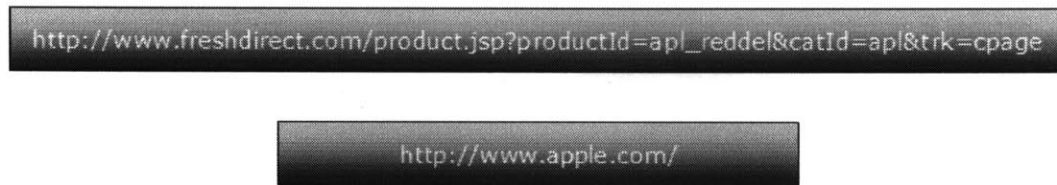


Figure 2-33 Apple URIs

However, there are several challenges with URIs. The first is that an intelligent human knowledge engineer is required for the ontology alignment of concepts. What is the correct URI for apple (the fruit)? It would appear the answer is to make up your own, and then retroactively associate it with others URIs once you agree that you are talking about the same concept. If humans used this same method for communication, the resulting Rosetta Stone (etched in RDF) would grow to be larger than the size of the earth, destabilizing the earth's orbit, and completely messing up all of our seasons. Of course, data takes up considerably less physical space, but that still only solves part of the problem.

The second challenge with URIs is that they are not usable by average humans, like the "clinic's office manager (who never took Comp Sci 101)" [1]. Any user interface that involves tasks like annotation, knowledge acquisition or semantic searching, must still be based (at least on an interface level) on natural language.

The Open Mind knowledge base was able to grow very quickly due to the fact that it collected knowledge from volunteers over the Web, in natural language.

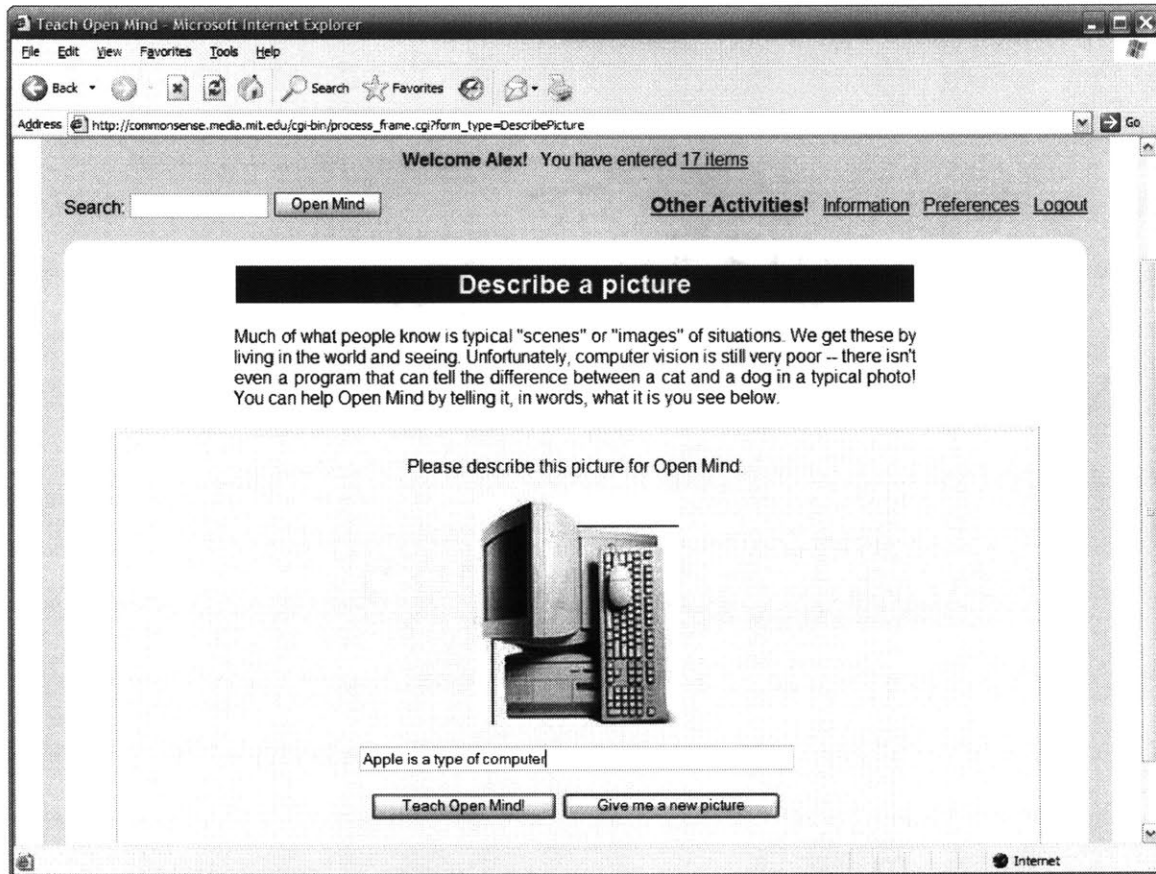


Figure 2-34 Telling Open Mind that Apple is a type of a computer

Because of this, ConceptNet suffers all of the ambiguity inherent in natural language. For instance the following two statements are currently in ConceptNet:

```
(IsA "apple" "computer")  
(IsA "apple" "fruit")
```

Without URIs, Creo and Miro must disambiguate concepts based on patterns and context. This is similar to the way that humans are able to effectively disambiguate concepts. Consider the following two lists.

Strawberry	Dell
Blueberry	Hewlett Packard
Apple	Apple
Raspberry	Gateway
Banana	Toshiba

Table 2-2 Using patterns and context to disambiguate concepts

Now, without the need for URIs, correctly parsing the meaning of “apple” becomes considerably easier (assuming you can look up the semantic context of the surrounding terms). Miro will re-rank the semantic context of a concept based on the semantic context of the surrounding concepts. Miro also bases its suggestions on thresholds that can be modified by the user.

Determining the semantic context of a term based on patterns and thresholds is important because misinterpreting terms can become very annoying to the user. Paul Thurrott of WinInfo ran into this problem with Microsoft’s Smart Tags when he was writing his review of Office XP:

Mr. Thurrott typed the word “nice.” Up popped a smart tag offering to book a flight to Nice, France using Microsoft’s Expedia website. When he typed the word “long,” up popped a smart tag from ESPN offering more information on Oakland Athletics centerfielder Terrence Long. As Thurrott put it, “Folks, this is lame.” [32]

Google’s AutoLink feature does not have to deal with ambiguity because the three types of data it recognizes, full address, ISBNs and VINs are all unique identifiers to begin with.

Determining semantic context based on patterns and thresholds is not a perfect solution. For instance, if someone wrote a blog about how they spilled apple juice all over their brand new apple G5, Miro will have difficulty

understanding the apples. However, because Creo and Miro are applications designed for the Web as it exists today, they cannot rely on URIs for disambiguation. Creo and Miro must be able to make intelligent assumptions about natural language, whether it is the text of a Web page, or the text of a search query.

2.3.4 Personalized Semantic Searches

In addition to the *Scan Page* feature, Miro also provides users with a search box.



Figure 2-35 Miro's search box

When the user enters text and clicks *Search*, Miro will follow the same steps that it takes when analyzing concepts on a Web page. Miro will look up the semantic context of the text using ConceptNet and TAP, and then attempt to find a match against the current set of recordings. For instance, returning again to the "Order Food" example found in the previous sections, Miro will activate that recording for any foods that are typed into the search box.

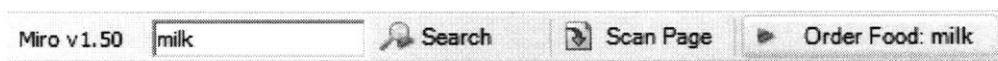


Figure 2-36 Using Miro to search for "milk"

Pushing in the result button that appears will play the "Order Food" recording, with "milk" as input. The search box is more efficient than Creo's *Player* tab, because users can start the interaction with their data. Using the *Player* tab, users must first select their recording, and then provide the necessary input. Miro uses the semantic context of the input itself to automate the selection of the recording, which will save users time if they have a very long list of recordings to choose from.

Weighted Semantic Searches

Like "apple," many concepts have multiple generalizations. For instance, "the lord of the rings" is both a series of movies, and a series of books. Because of this, entering "the lord of the rings" into Miro's search box will cause it to activate both the "Look up Movie" and the "Buy Book" recordings.



Figure 2-37 Searching Miro for "the lord of the rings"

Miro will use the context of what the user is currently viewing to weight the results of its searches. For instance, if the user is viewing information about books, the "Order Book" recording will appear before the "Look up Movie" recording.

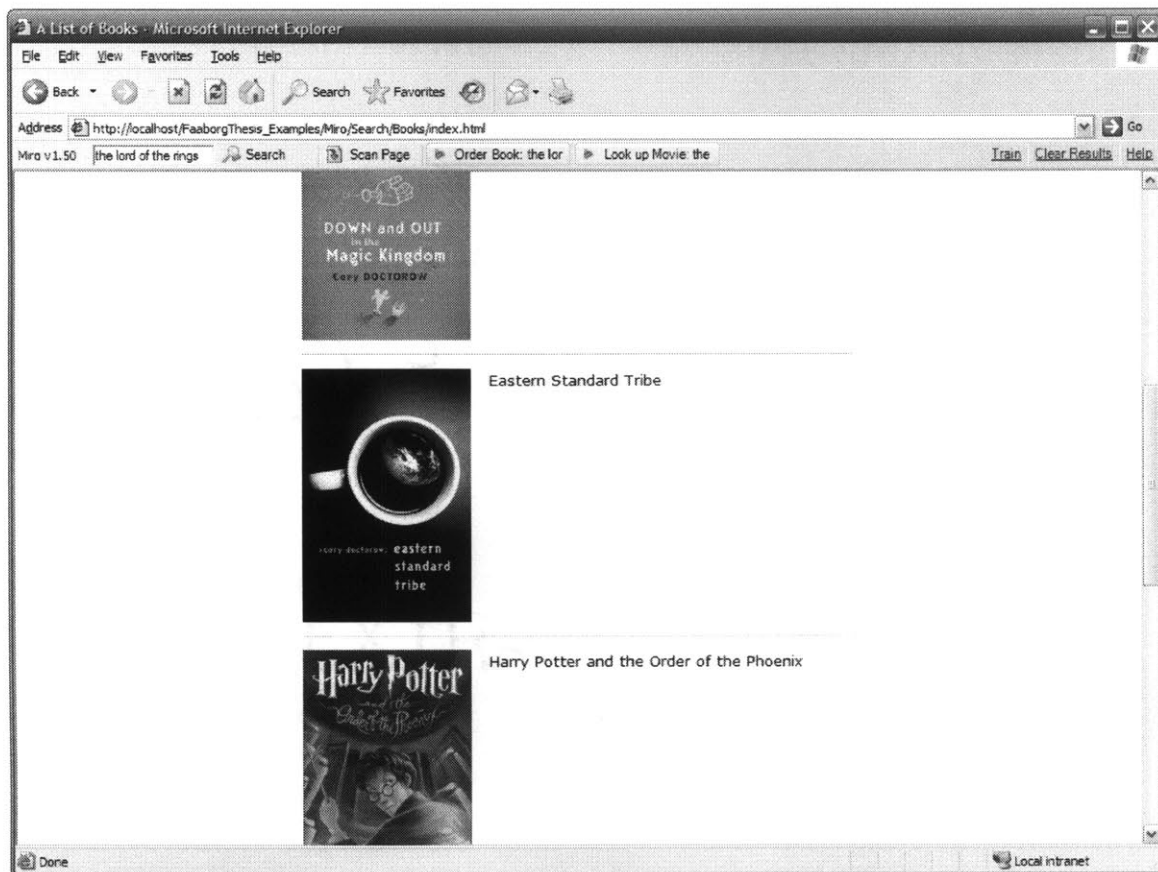


Figure 2-38 The results of Miro's search are weighted against the semantic context of the page (books)

Similarly, the recording "Look up Movie" will appear first in the results, if the user is viewing a Web page about movies.

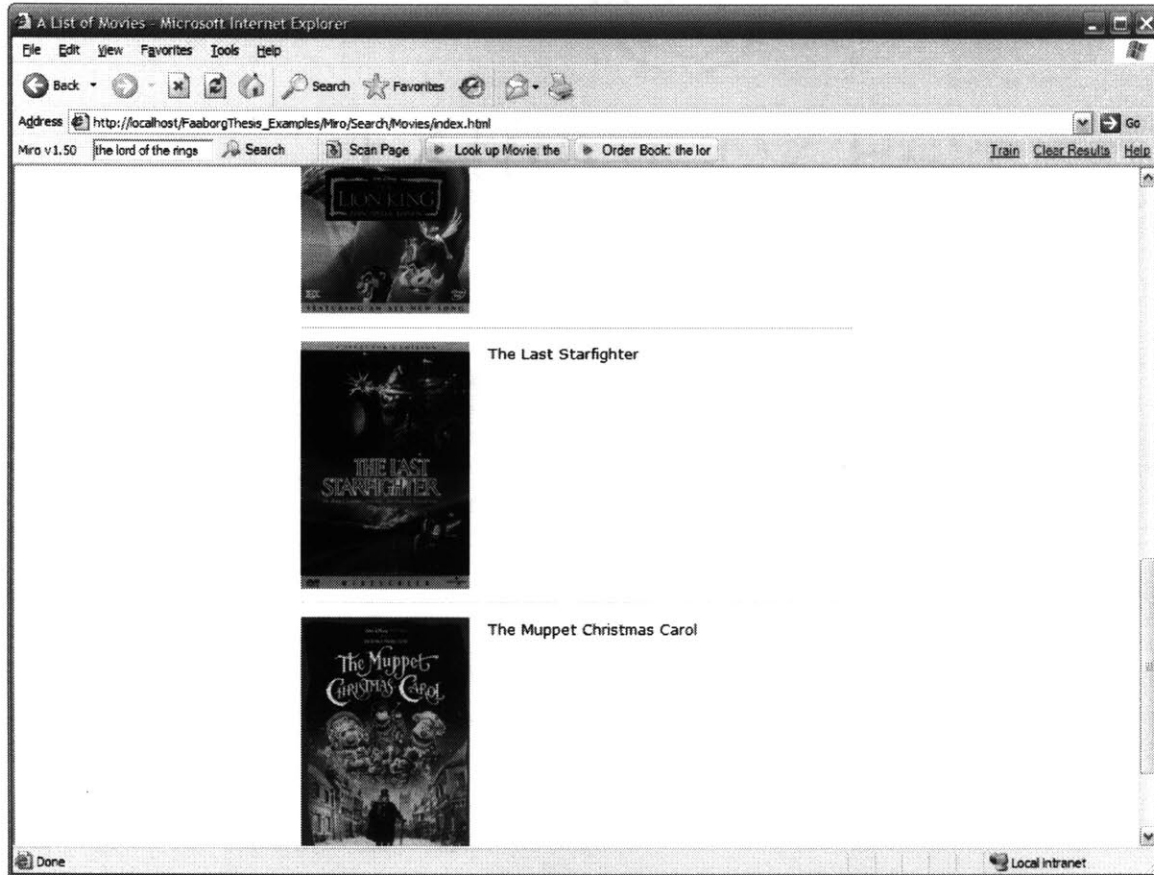


Figure 2-39 The results of Miro's search are weighted against the semantic context of the page (movies)

Personal Search

One difference between Miro and traditional Web searches is that the results returned from searching with Miro are personalized to the user. Because Miro is searching the set of recordings created by the user, the results that Miro returns will always be based on personal preferences: the user's favorite site to look up movies or buy books, the nearest grocery store, or the local pizza place. Users already perform searches at specific sites that they trust or prefer, however it is always a two-step process: (1) navigate to the specific site, and (2) enter the search. For instance, navigating to IMDB and then searching for a movie. The Web browser Firefox has streamlined this

common practice with a feature called Smart Keywords [34]. Smart Keywords allows users to associate keywords with some search fields. Using this feature, users can enter searches that target a specific information source by typing phrases like "imdb star wars," "etrade GOOG," or "freshdirect apple." Miro's ability to understand the semantic context of concepts allows users to enter targeted searches, while only providing the information they wish to search. Additionally, the Smart Keyword feature of Firefox only works on actions that include a single search. Miro can include searches in a sequence of actions, like logging into an account at E*Trade, navigating to the correct part of the site, setting several options, and then performing a search.

Comparing Miro's Search Feature to Traditional Web Searches

Searches with Miro are very different from performing a search using Google, and the two types of searches are complementary. Google has (at the time of this writing) indexed over 8 billion Web pages. Depending on the number of recordings a user has, Miro is likely to only search the generalizations of input for 10-100 recordings. However, the real difference is not in scale, but in the user's intention. Google is useful for information retrieval, while Miro is useful for completing specific tasks. A user would search Google when they have a question, or need to find a piece of information. A user would search Miro when they want to complete an action. Miro and Google lie on opposite sides of the spectrum of possible searches. Just as searching Google for "milk" will not accomplish very much, searching Miro for the "French Revolution" is not likely to have any utility.

2.3.5 Adeo

Adeo adapts the semantic search features of Miro, and Creo's *Player* tab for use on a Smartphone.

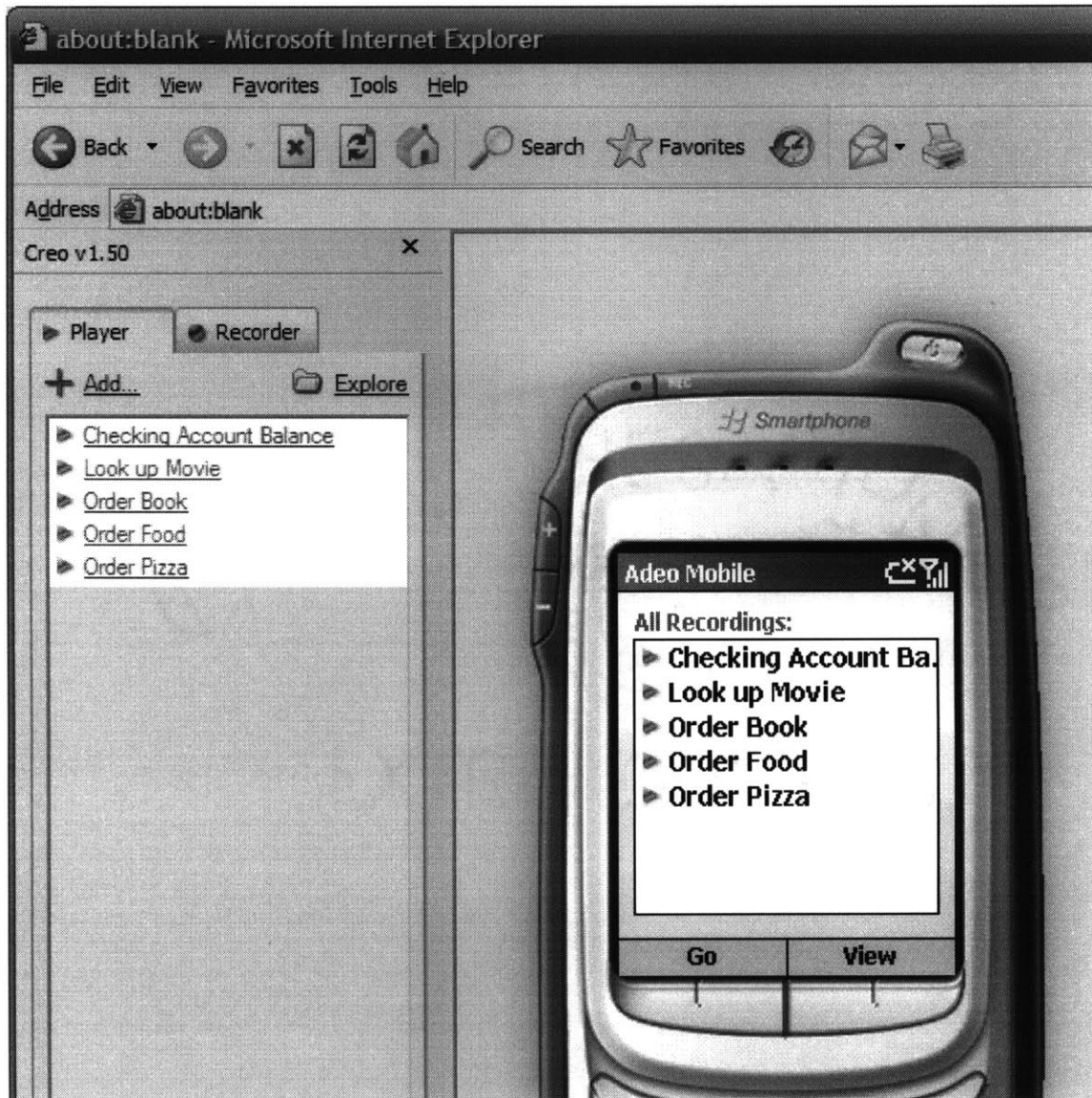


Figure 2-40 Adeo brings the functionality of Creo's *Player* tab to a mobile device

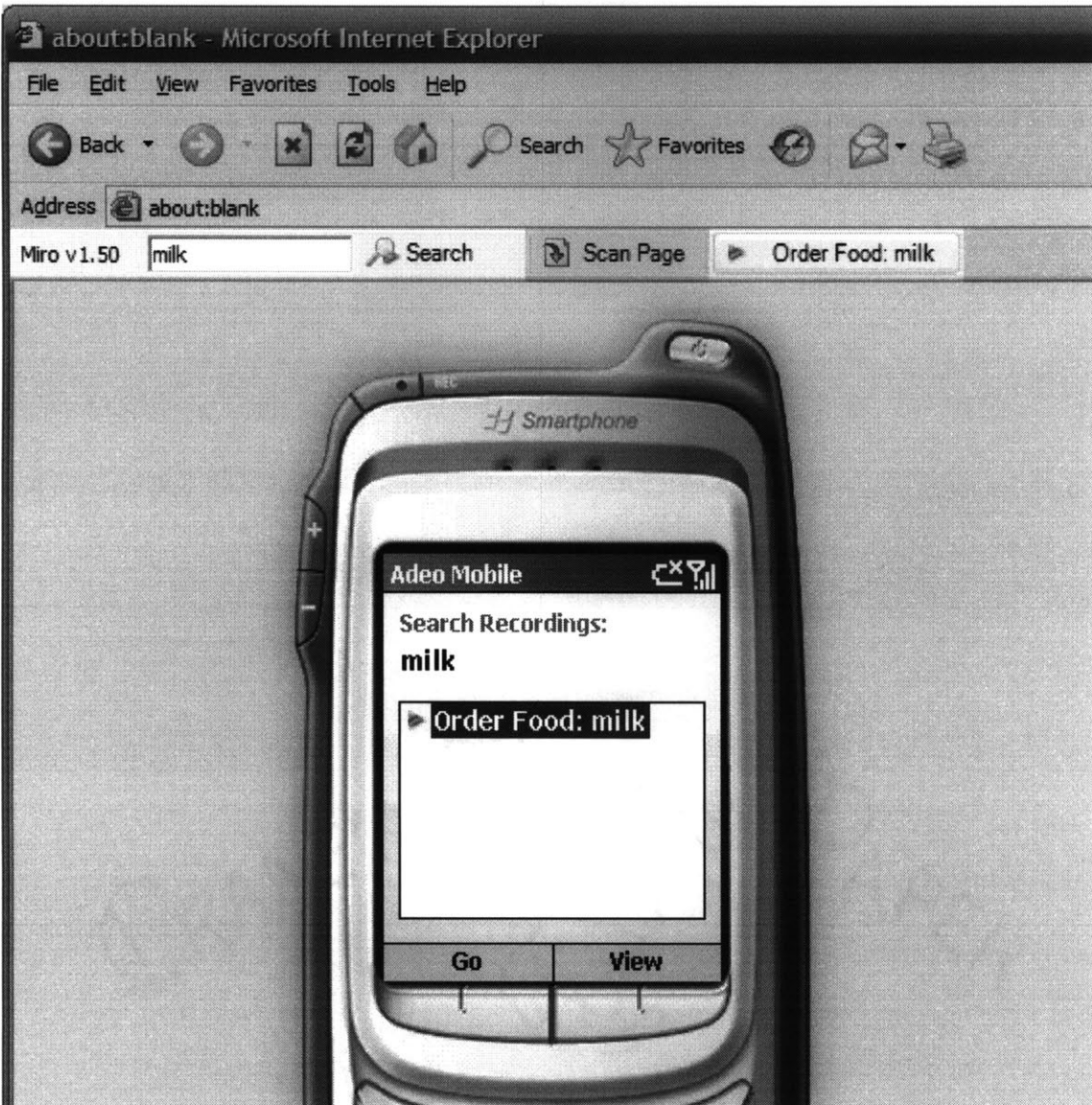


Figure 2-41 Adeo brings the functionality of Miro's semantic search to a mobile device

The Failure of WAP

In the early days of the Internet before the Web, consumers received information from proprietary services like GENie, CompuServe and Prodigy. Mobile devices went through a similar era of proprietary platforms including HDML by Unwired Planet, ITTP by Ericsson, and TTML by Nokia [35]. The wireless industry eventually decided that the industry as a whole would benefit by the creation of a standardized application platform, and in the late 1990s they created WAP, the Wireless Application Protocol.

However, unlike the Web, WAP is widely regarded as a colossal flop. In Europe and North America, WAP was a failure for a number of reasons. These included technical problems like idiosyncratic differences between WML and HTML, and a lack of easy to use development tools. The WAP standard included many optional features, which resulted in compliant devices not being able to interoperate [35]. Additionally, each device's implementation of WAP varied, and this inconsistency coupled with the physical constraints of mobile devices led to usability problems. Jakob Nielsen notes:

WAP has miserable usability for many reasons: ridiculously small screens, slow bandwidth, and the need to place a new call every time the device needs to connect. The digits-only keypad is a laughable input device, leading to the guideline to use numeric PIN codes instead of full passwords any time authentication is required. Also, the actual telephones vary in their design and sometimes have poor human factors that don't deliver as good a user experience as would be possible under the given constraints [36].

While WAP was not a proprietary format, the difficulty of entering a full URL with a phone keypad led to the vast majority of users never browsing beyond the home page of their service provider. While this was a user interface problem, and not a technical problem, this difficulty effectively made each carrier's WAP portal a walled garden. In Europe, several carriers launched WAP as a closed service.

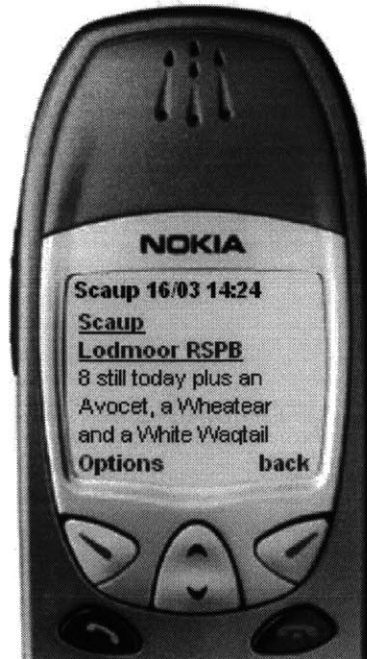


Figure 2-42 A WAP site for rare bird watchers (wonderfully ironic)

Successes of Mobile Browsing

WAP was more of a success in Japan and North Korea. Similar systems like NTT DoCoMo's i-mode have also been very successful in Japan [37]. Of the 87 million mobile subscribers in Japan, 34 million subscribers are currently signed up for 3G services [38]. The success of services like i-mode can be attributed to the same factors involved in every successful application platform: creating an open system, and then reaching a critical mass of content and services.

A Long Road to the Mobile Web

In Europe and North America, mobile content is still very far from reaching such a critical mass. While companies like Google and Yahoo have released mobile versions of their search engines, the amount of content specifically formatted for mobile devices remains miniscule compared to the size of the Web. In many cases, the Web itself is just now reaching a critical mass to attract companies since small and local businesses are very slow at adopting new technologies. For instance, a pizza place near a college campus may finally decide in the year 2005 that it is time to offer online ordering over the

Web, but it could take them another decade before they decide to offer online ordering in an interface formatted for the constraints of mobile devices.

While it is currently possible to view Web pages designed for personal computers using a Web browser on a mobile device, the constrained input, output, and bandwidth of mobile devices make doing so a slow and incredibly tedious process. For instance, going through the process of ordering a pizza through a mobile Web browser could take over 10 minutes if the page is formatted for viewing on a PC, and the user must enter their address and billing information into Web forms using a numeric phone keypad.



Figure 2-43 Interacting with a Web page designed for viewing on a PC can be a slow and frustrating process on a mobile device

In fact, instead of trying to interact with the Web page using a mobile browser, it would be considerably easier for the user to simply call the pizza place (they are, after all, holding a telephone). To be accepted by users, it must be easier to complete tasks using the Web than by placing a call.

Invoking Automated Interactions with the Web using a Mobile Device

Adeo is a software application designed for Smartphones that allows users to play the recordings they have created using Creo. This approach solves both of the major problems facing mobile browsing: (1) the lack of content on the Web designed specifically for mobile devices, and (2) the constrained user interface making completing complicated actions difficult. Users can play a lengthy recording using Adeo, with a single click.

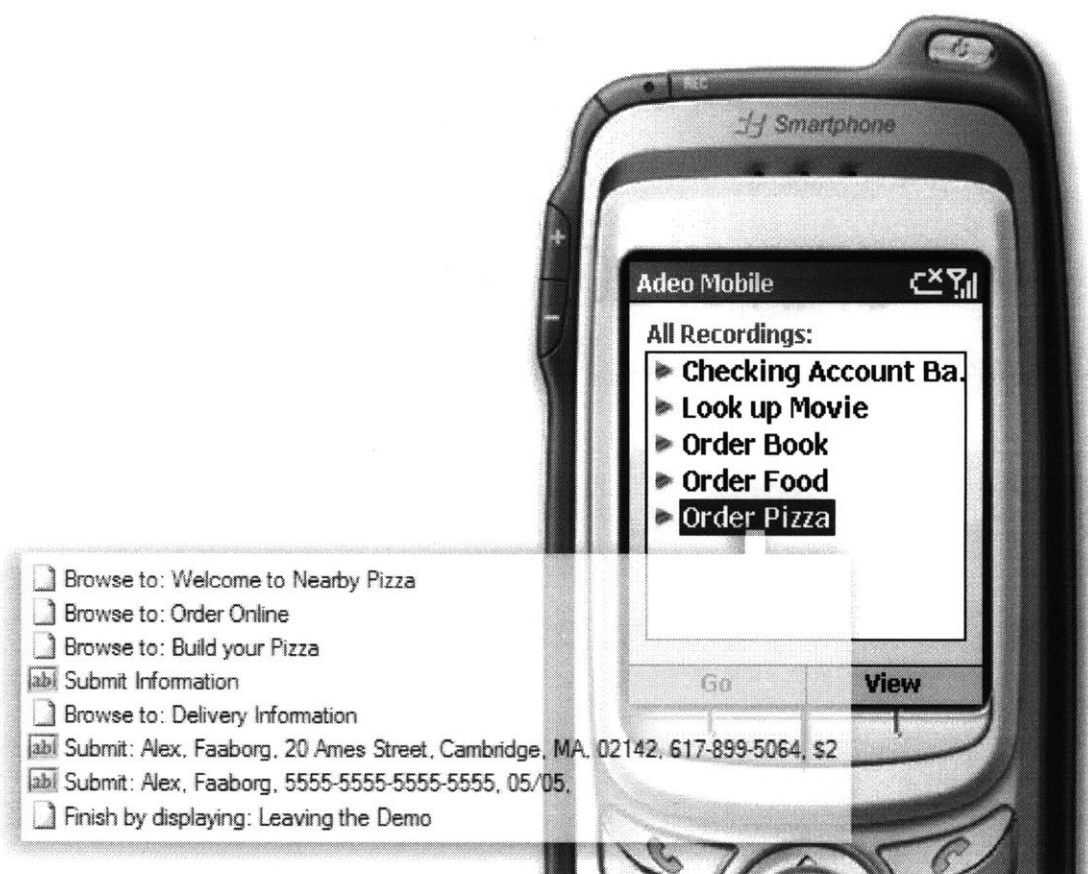


Figure 2-44 The user invokes a multi-step procedure with a single click

The user may have requested to provide some pieces of information when playing a recording, like the type of pizza or the amount to tip. If any

variable information is missing, Adeo will prompt the user to provide it. Adeo then plays the recording and displays the result.

Playing recordings with Adeo is a three step process: (1) selecting the recording and providing any missing input, (2) waiting while the recording is played, and (3) viewing the result. For instance, viewing the balance of a bank account:



Figure 2-45 Using Adeo to check the balance of a "hypothetical" bank account

Reducing a Recording to Minimal Input and Output

One of the design principles of Adeo is the idea of reducing a procedure to the least amount of input and output. For instance, for the action of checking the balance of a bank account, no input is required (unless the user would prefer to enter their password every time) and the most minimal amount of output is the balance of the account. By reducing the input and output of procedures to only the most critical pieces of information, the user experience is streamlined to the simplest possible interaction. Creo automatically detects which pieces of input it believes should remain static when users are creating recordings, however users can also directly control the input they would like to provide each time they play a recording. This

interface design is based on the idea that users do not want to “browse” on mobile devices, they want to complete specific tasks in the least number of steps possible.

Semantic Searching on Mobile Devices

An additional way Adeo can streamline the user experience is by allowing users to begin an interaction by providing their input. Adeo’s ability to perform semantic searches allows the interface to proactively suggest which recording matches any given information. For instance, if the user enters “snow crash,” the only recording matching that concept is “order book.”

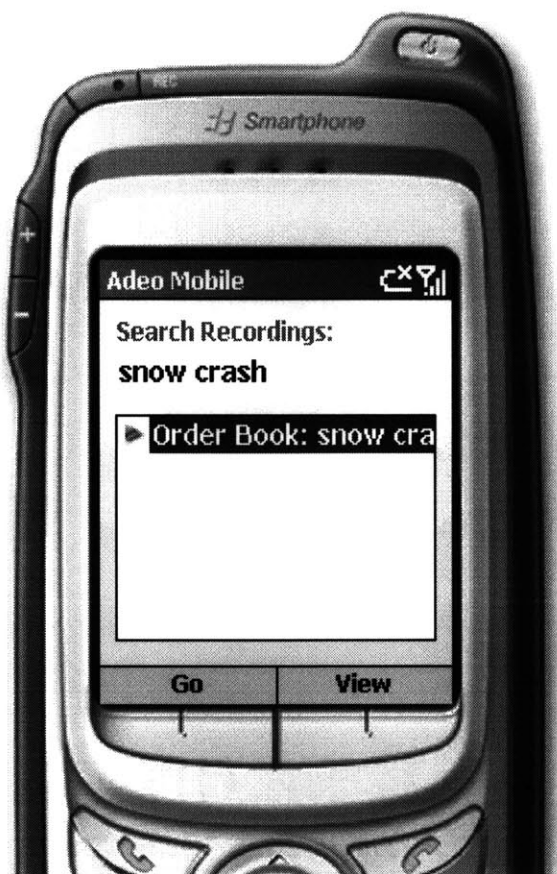


Figure 2-46 Adeo performs a semantic search, reducing the number of steps to complete a task

Other types of input may match multiple recordings. For instance, an email address might result in multiple relevant recordings: sending the person a message, sending the person PayPal, or sending the person an Evite for an

upcoming party. However, even in cases where multiple recordings are returned as possibilities, Adeo is able to significantly reduce the number of potential actions by performing a semantic search on the input.

Adeo enables users to efficiently complete actions using mobile devices. Users can invoke actions on the Web itself, without having to wait for each of their favorite service providers to create a user interface specifically designed for constrained input and output of mobile devices. By acting as a User Interface Agent, and mediating communication between the user and the Web, Adeo solves the two major problems facing the emergence of mobile browsing.

2.4 Learning

2.4.1 Introduction

This section discusses the various forms of learning performed by Creo and Miro.

Learning Procedures

Creo is able to learn how to complete procedures on the Web by watching the user. Section 2.4.2 discusses the difficulty of completing actions without the user's guidance, and the specific reasons that Creo must watch the user to learn how to complete tasks on the Web.

Learning Knowledge

Throughout the examples shown in the previous section, from Miro's ability to recognize the semantic context of a page, to the semantic searching features of Miro and Adeo, it has been assumed that knowledge about what the user is viewing or entering exists in ConceptNet or TAP. This, of course, is not always the case. Regardless of the size of the knowledge base used, information will always be missing. Section 2.4.3 discusses Miro's training interface. Miro's ability to learn from the Web using natural language processing is discussed in Section 2.4.4.

2.4.2 Learning in Creo

While Creo is able to automatically generalize input provided by the user, it is not able to infer how to interact with Web sites. Creo is only able to follow a specific set of steps, and must be trained by watching the user complete a task. There are several challenges facing Creo's ability to complete tasks on its own.

Challenge 1: Collecting Knowledge about Procedures

The first reason that Creo cannot complete tasks on its own this is that Creo does not have access to the knowledge needed to know how to perform even simple procedures. Before viewing a Web site’s interface, most users have a preconceived mental model of how the site will function. For instance, most users know ahead of time that ordering food from a grocery store involves locating the food, and adding it their shopping cart. Viewing the balance of a bank account involves logging in, selecting the account, and finding the balance. Creo does not currently have access to this commonsense knowledge about how to complete basic tasks on the Web. The Media Lab has been attempting to collect this type of information in a commonsense knowledge base called LifeNet [7, 39]. However, this knowledge base contains information on everyday life, and not specifically everyday interactions with the Web.



Figure 2-47 A sample of the LifeNet network

The format of temporal knowledge found in the LifeNet network could be used to reason about procedures on the Web. Unfortunately, the knowledge base is not yet broad enough to cover the majority of common Web interactions.

Challenge 2: Parsing the User Interface

Even if Creo had access to a perfect knowledge base of how people complete actions on the Web, this alone is not enough to enable it to perform actions by itself. The second challenge facing Creo's ability to complete tasks without training is correctly parsing the user interface. This is far from a trivial task. First, Creo would need to render each page it was viewing, since the descriptions next to form elements are often located in tables, and may not be nearby in the HTML. Additionally, many of the important phrases and concepts for completing a task are embedded in images. For instance, for Creo to automate the process of ordering food on its own, it would need to render the page, and then perform optical character recognition to locate the search field.

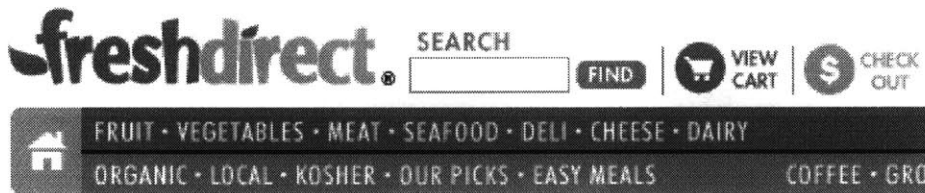


Figure 2-48 Many critical pieces of information are embedded in images

Simply looking at the names of HTML form elements would be an inconsistent solution, since many form element names are default values (like "T1" for a new text box in FrontPage), or are internal developer terms, like "sParam."

Challenge 3: Missing Information and Confusing UI Design

Due to poor user interface design, in many cases even humans (with their superior cognitive and perceptual abilities) have difficult completing a task for the first time on the Web.

For instance, in the following figure, currently only a human would be able to parse the difference between ElectriPaySM and ElectriCheckSM.

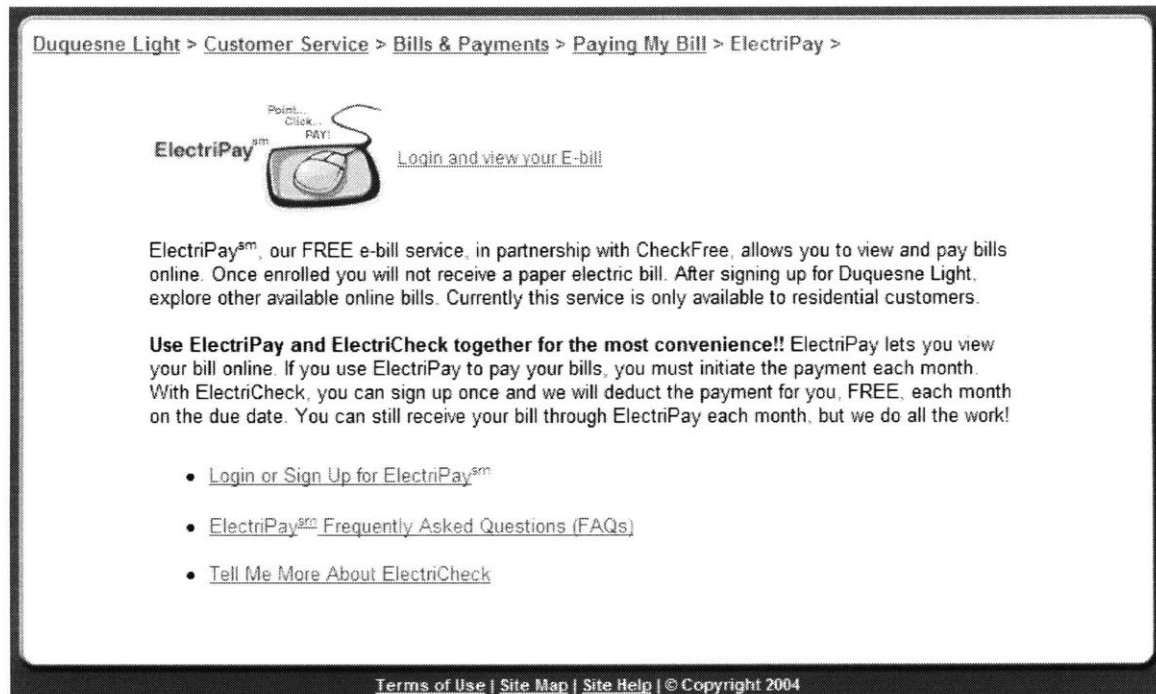


Figure 2-49 Attempting to set up automatic billing with an electric company, the user is confronted with a reverse Turing Test

Additionally, for this specific task, Creo cannot be expected to know all of the required input on its own, since even a human would need to look up their ABA routing number in order to initiate an ElectriPaySM.

Understanding the difference between ElectriPaySM and ElectriCheckSM was an unintentional test of the user's intelligence. However, it is also not uncommon to encounter a test intentionally designed to assess the user's perceptual abilities.

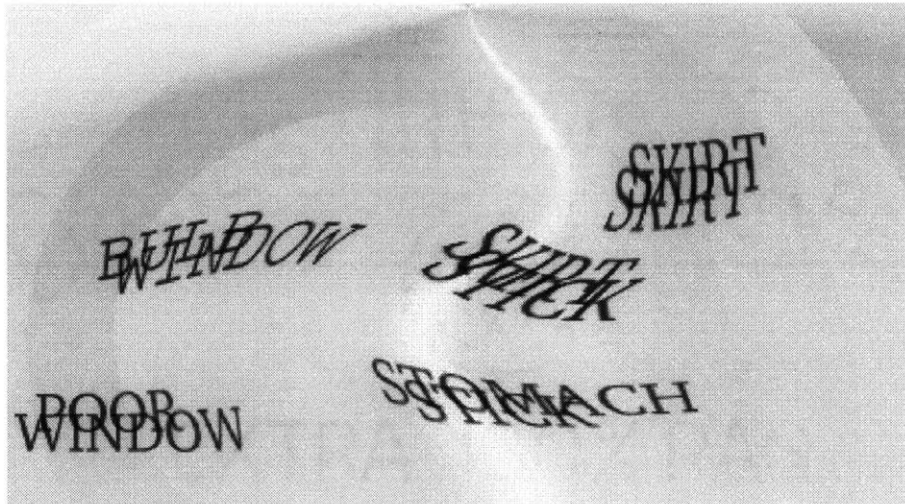


Figure 2-50 A captcha test, also described as a reverse Turing Test

If companies wished to block User Interface Agents from accessing their services (like sending PayPal to someone using a mobile device), they could require that the user complete a captcha test as part of the task. Currently, captcha tests [40] are only commonly used when creating accounts, conducting online polls, or submitting publicly viewable information, like to a search engine, directory, auction site, or message board.

Learning by Example

Creating a version of Creo that completes actions on the Web without training is not an impossible task, but it is an extremely difficult one. Because of these challenges, Creo is designed to be programmed by example, leveraging the user's far superior cognitive and perceptual abilities. This means that Creo is essentially a macro recorder, with an ability to generalize input. However, it is considerably easier to impersonate a user's specific actions than it is to impersonate a user's mind.

As the user completes an action on the Web, Creo records the URLs, HTML form element names, and the semantic context of the input the user provided to the procedure. For instance, here is how the start of the "Order Food" recording looks from the user's and Creo's perspectives.

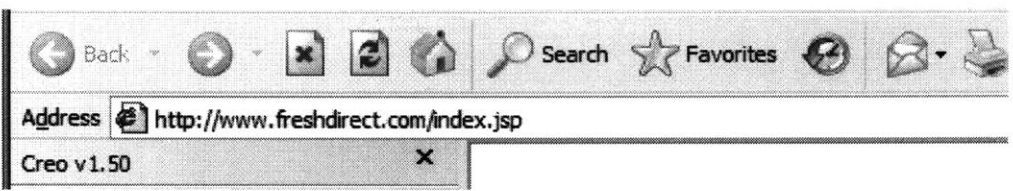

<p>User:</p>	
<p>Creo:</p>	<pre> <Action> <description>Browse to: Welcome to FreshDirect</description> <type>navigate</type> <title>Welcome to FreshDirect</title> <url>http://www.freshdirect.com/index.jsp</url> </Action> </pre>
<p>User:</p>	
<p>Creo:</p>	<pre> <Action> <description>Submit: Ask - food brand</description> <type>formSubmit</type> <title /> <url>http://www.freshdirect.com/index.jsp</url> <FormElement> <type>text</type> <name>searchParams</name> <val>diet coke</val> <ask>true</ask> <askPrompt>food brand</askPrompt> <personalInfo>false</personalInfo> <Generalization> <name>food brand</name> <use>true</use> </Generalization> <Generalization> <name>drink</name> <use>true</use> </Generalization> <Generalization> <name>soft drink</name> <use>true</use> </Generalization> <Generalization> <name>soda</name> <use>true</use> </Generalization> <Generalization> <name>popular soda</name> <use>true</use> </Generalization> </FormElement> <formSubmit_generalizedFormElements>true</formSubmit_generalizedFormElements> <formSubmit_submitNumber>1</formSubmit_submitNumber> </Action> </pre>

Table 2-3 An example of how Creo encodes the user’s actions with a Web site

Training Creo by Importing Other User's Recordings

In addition to training Creo by recording an example of interacting with a Web site, users can also train Creo to complete procedures by dropping .xml files in its recordings folder. This is discussed in detail in Section 2.5.3, The Viral Spread of Recordings.

Debugging Recordings in Creo

If a Web site significantly changes its user interface, Creo will not be able to play a recording since the recoding is trained for the site's previous interface. Users can debug their recordings by clicking the *Debug Recording* hyperlink on the *Player* tab of Creo, and selecting the recording they wish to debug.

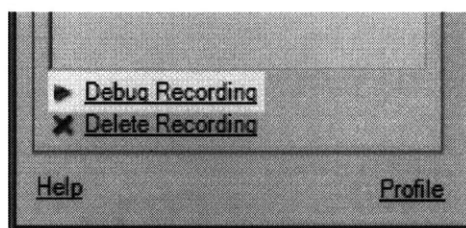


Figure 2-51 The Debug Recording hyperlink

Creo's debug interface allows users to step through actions, one action at a time.

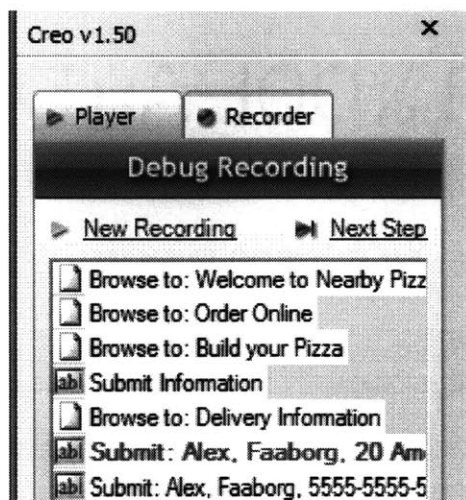


Figure 2-52 The user steps through actions in a recording

By stepping through a recording, and trying various types of input, users are able to determine why a recording is not currently playing correctly.

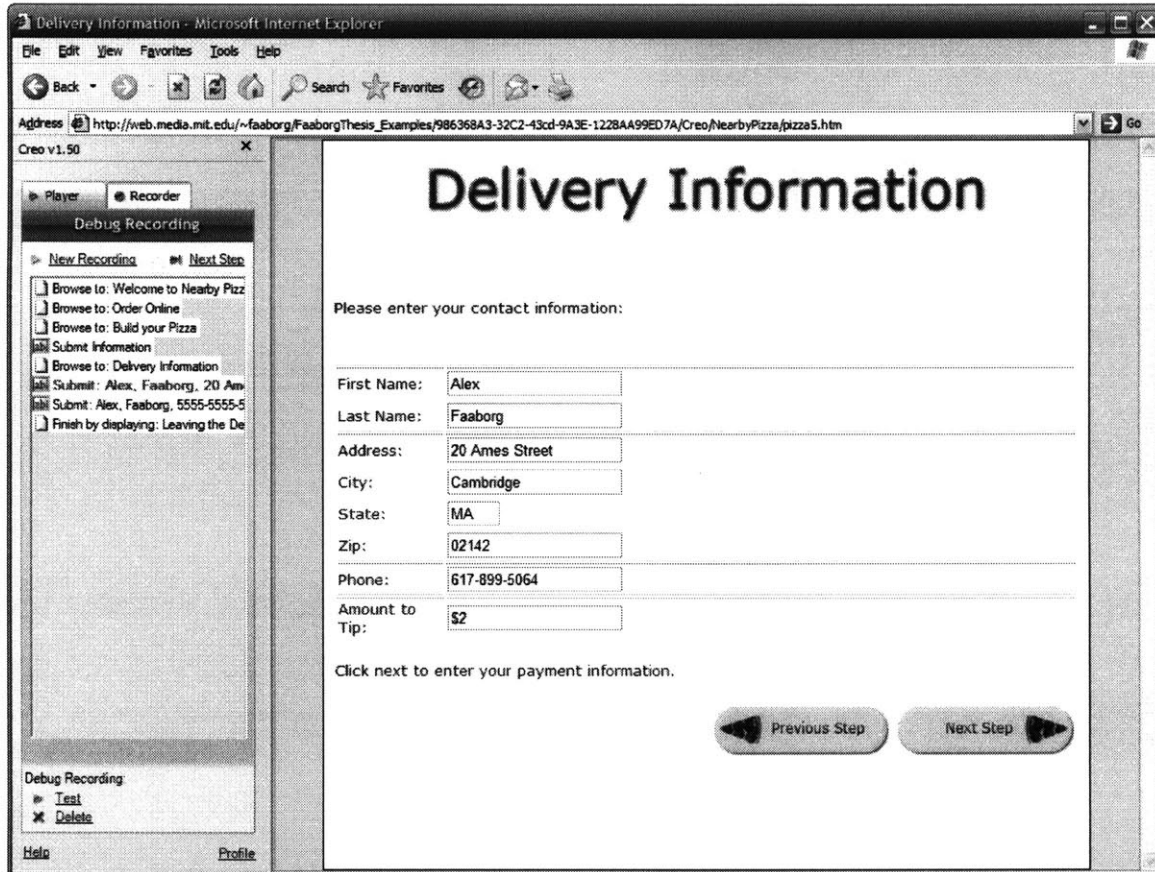


Figure 2-53 Creo's debug interface

2.4.3 Learning in Miro - Learning From the User

Miro's ability to understand the context of a Web page and perform semantic searches is directly based on the knowledge found in ConceptNet and TAP. When these knowledge bases are missing a piece of information, Miro is unable to match that information to the user's set of recordings. Because of this, Miro includes a training interface so that the user can help teach it new knowledge.

If the user performs a semantic search, and no results are returned, the *Train* hyperlink appears. For instance, if Miro does not know what a "guava" is, the *Train* hyperlink will appear.

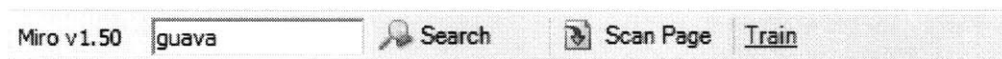


Figure 2-54 The *Train* hyperlink appears instead of results

To be fair, Miro actually knew what a guava was, but that piece of knowledge was deleted for this example. (A guava is a lemon-sized fruit that tastes like a strawberry-pineapple-banana, with a papaya-like texture.)

If results are returned, the *Train* hyperlink also always appears on the right side of the toolbar in case the results returned by the search or scan are not entirely correct.



Figure 2-55 The *Train* hyperlink

Regardless of the type of mistake Miro makes, the training wizard consistently follows the same three-step process. For all semantic searches the first step is omitted.

	Scan:	Semantic Search:
Step 1: Identify Concept		<p>Text of the Semantic Search:</p>
Step 2: Identify Generalization		
Step 3: Match Recording		

Table 2-4 Miro’s three-step training wizard

The training wizard can be used to correct three different kinds of mistakes: (1) Miro does not know the concept, (2) Miro knows the concept, but does not know the correct generalization, and (3) one of the recordings created by Creo does not have the correct set of generalizations to match the concept.

Miro Does Not Know the Concept

Let's assume Miro did not know the concept of "guava" when the user was attempting to perform a semantic search.

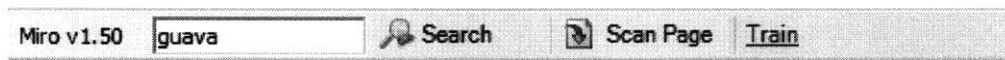


Figure 2-56 The user performs a semantic search on an unknown concept

Clicking *Train* will launch Miro's training wizard.

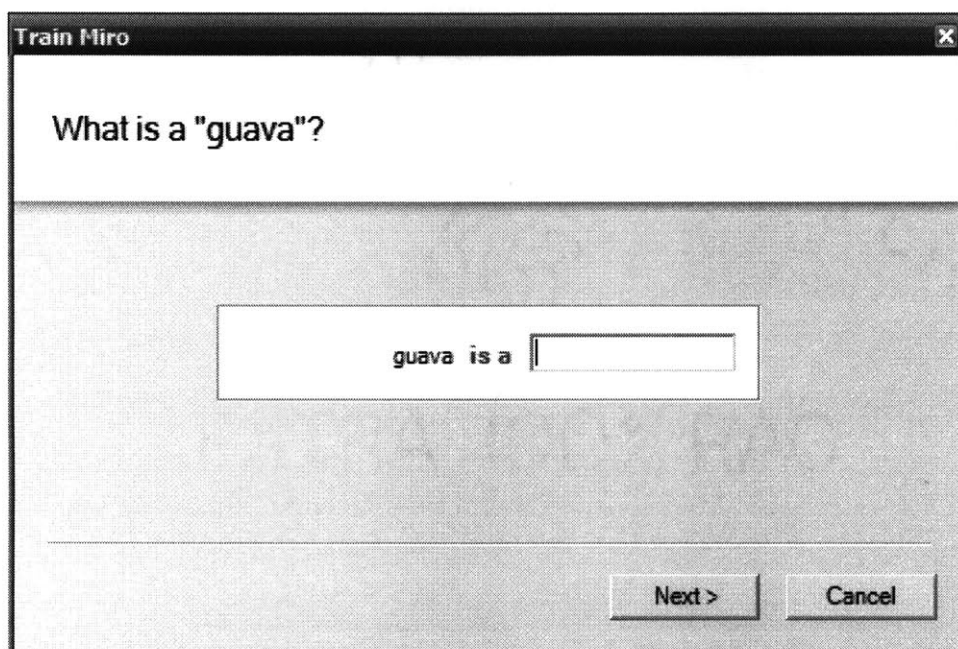


Figure 2-57 Miro asks the user to generalize the unknown concept

If the user was performing a scan on the Web page, the training wizard begins by asking them what concept should have been detected. For instance, perhaps Miro does not know about Cory Doctorow's new book, *Eastern Standard Tribe*, because it was recently released.

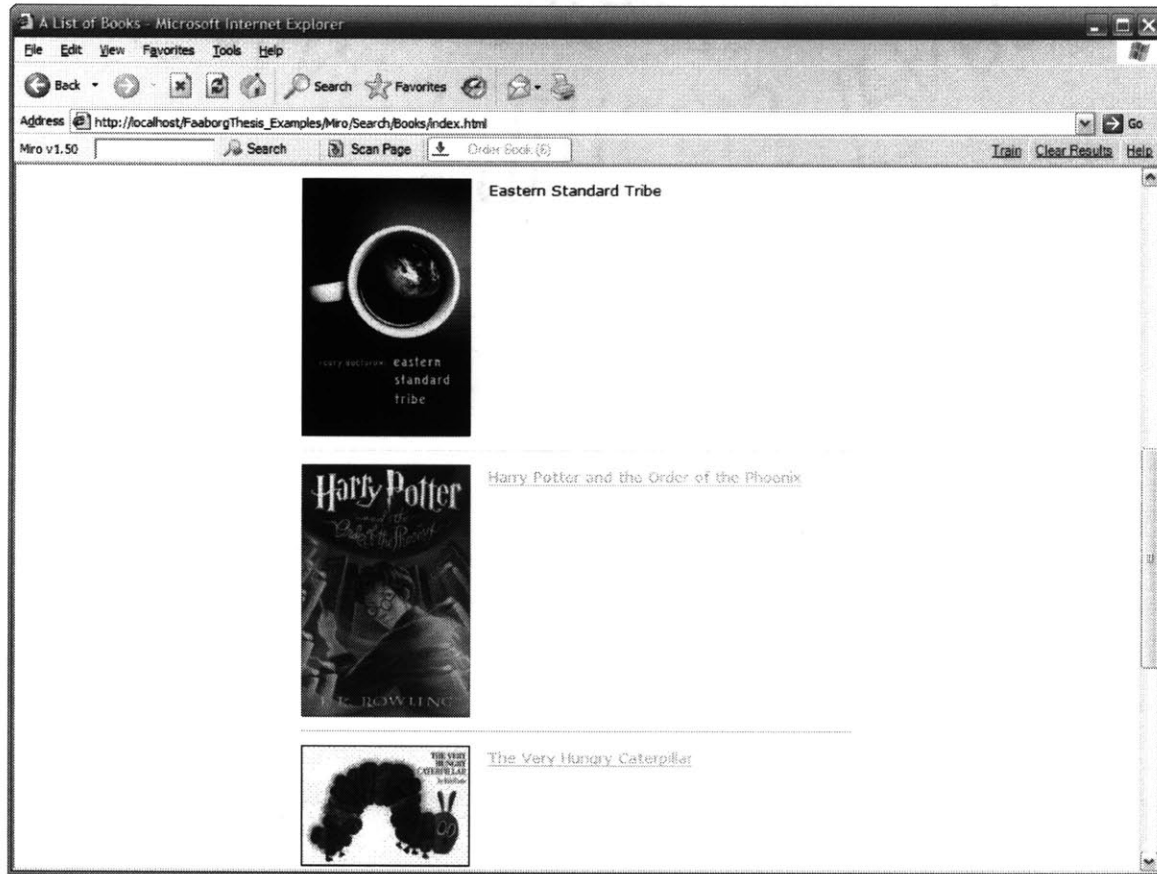


Figure 2-58 Miro does not know about one of the books because it is new. When the user clicks *Train*, the wizard asks which concept should have been detected, performing word completion on the text they enter.

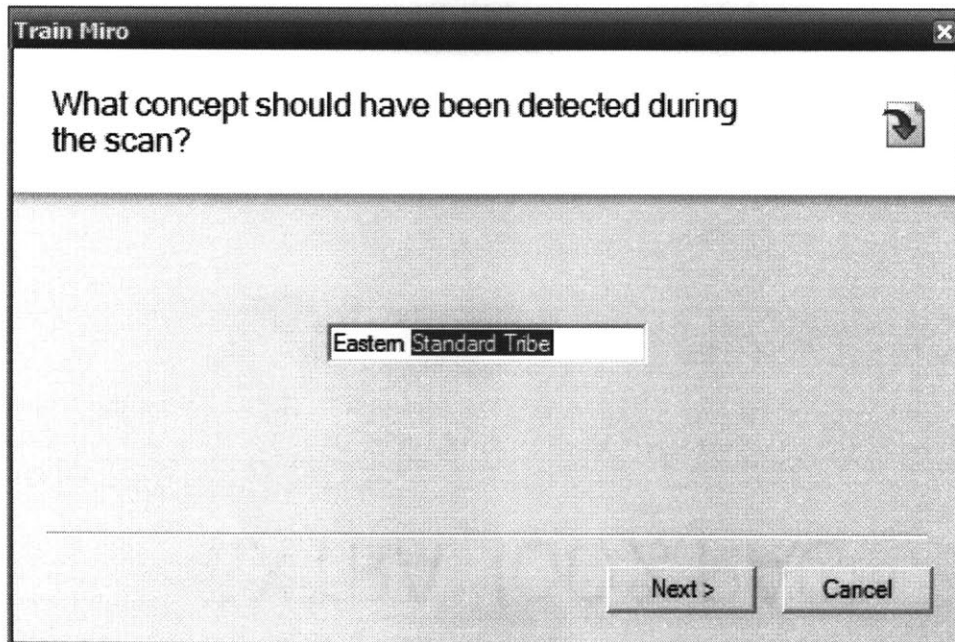


Figure 2-59 The user tells Miro which concept should have been detected

After the user clicks *Next*, they are asked to enter what the concept is. Users are taken directly to this step when launching Miro's training wizard after performing a semantic search.

Again, Miro performs word completion on what the user types. However, this time the words Miro completes for the user are weighted against how general they are. For instance, "book" is the most general term starting with b, where generality is defined by the number of concepts in ConceptNet and TAP that have that generalization.

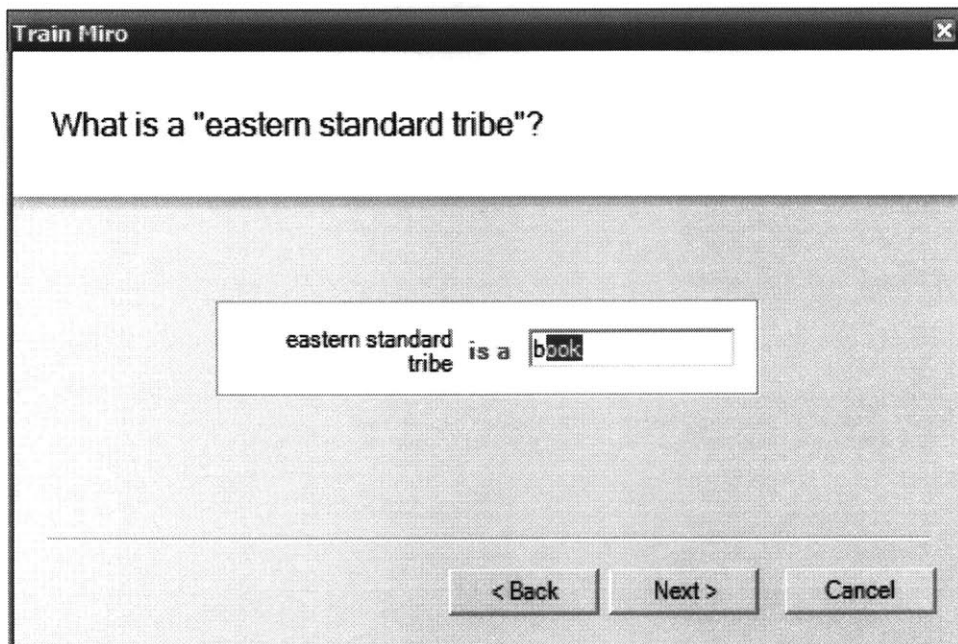


Figure 2-60 Miro performs word completion for the user, weighted against the generality of concepts

By using word completion, users are subtly coerced into providing more general descriptions of concepts. This is important, because a more general description of a concept will have a better chance of matching the generalizations of the recordings created by Creo.

For instance, returning to the semantic search example, if the user was about to type that a "guava" is a "fruit," after they type "f," the word "food"

appears. This interface is built on the assumption that if the user is presented with a correct generalization (even if it is not the one they were going for), they will accept it.

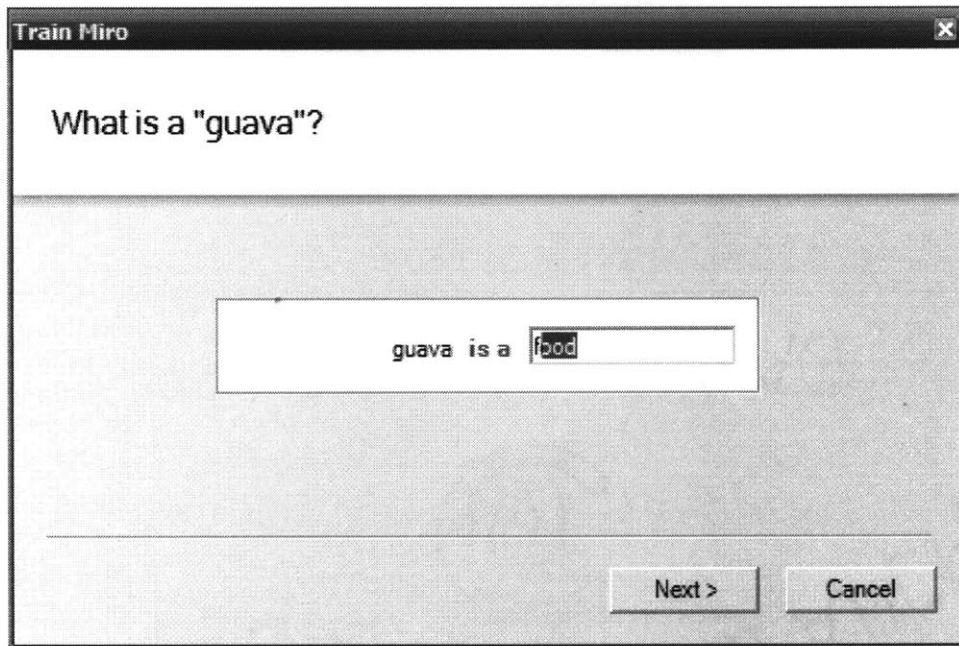


Figure 2-61 The user was going for "fruit" but "food" is also true

The generalization of "fruit," while correct, is not general enough to match any of the current recordings. This is because the recording "Order Food" was trained on the concept of "diet coke," and while it contains the generalization of "food" it does not contain the generalization of "fruit."

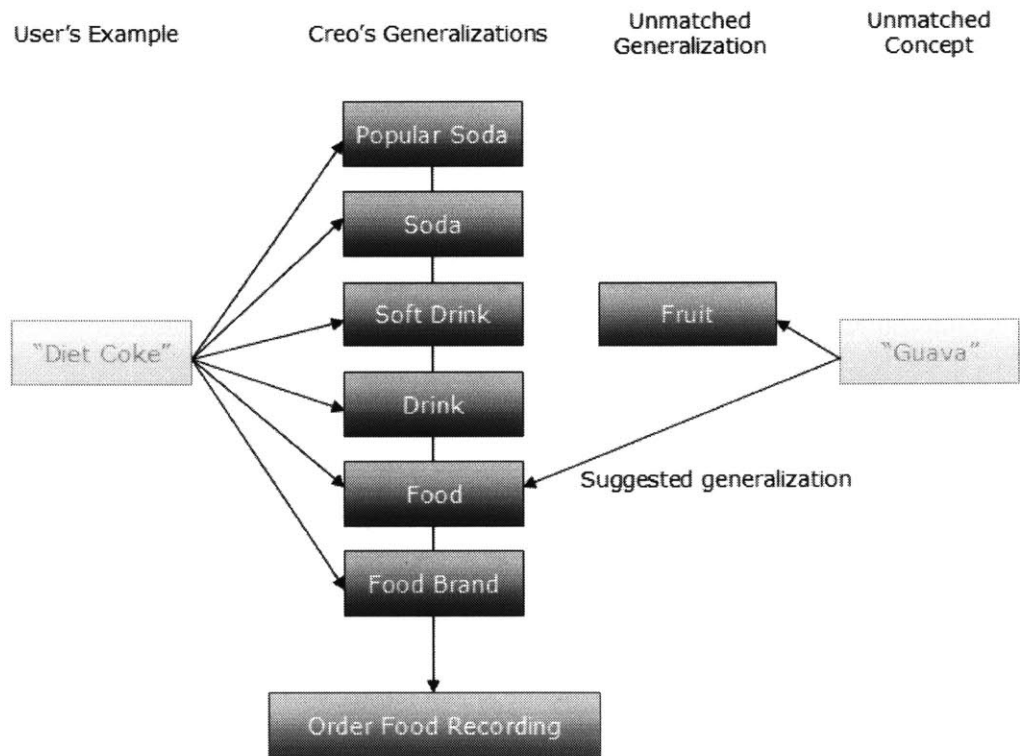


Figure 2-62 More general generalizations have a better chance of matching the recordings created with Creo

It should be noted that there are many user interfaces, beyond word completion, that could be implemented to direct the user into providing better generalizations of concepts. For instance, the interface could allow the user to enter a generalization, like "fruit." It could then generalize that generalization (resulting in concepts like "food"), and ask the user "which of these are also true." In the future, it is worth exploring alternative types of user interfaces for completing this step in the training wizard.

The final step of the training interface is to select which recordings should have been activated. Based on the generalization the user provided, some recordings may already be checked.

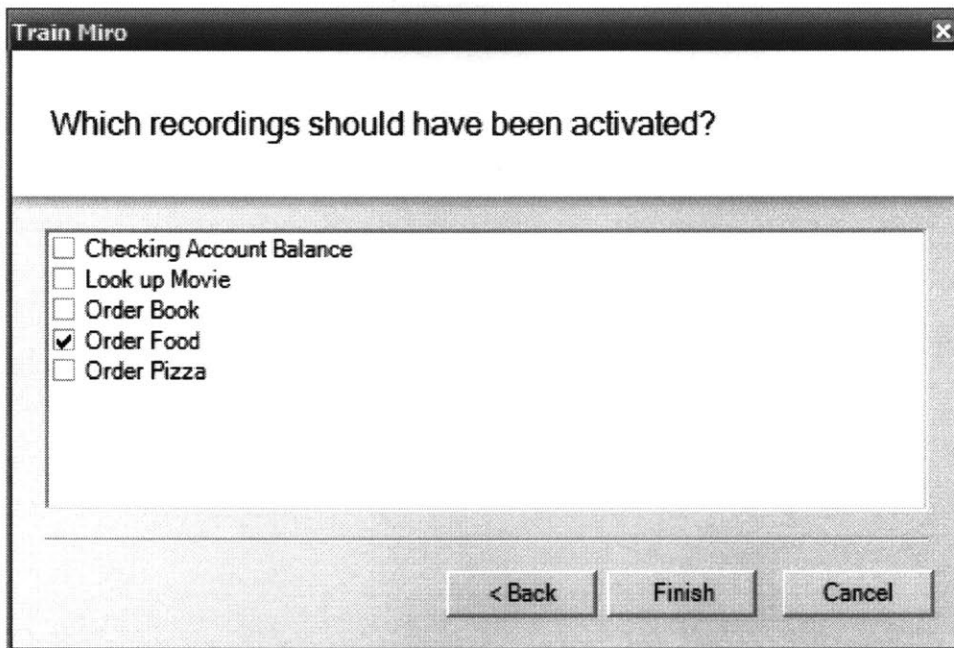


Figure 2-63 The final step of the training wizard

After the user clicks *Finish*, the training wizard writes the new knowledge the user has provided to the file *miro_learned.txt* for future reference, and adds it to the knowledge it currently has stored in memory.

```
(IsA "guava" "food")
```

Miro Does Not Know the Correct Generalization

Another type of error that Miro can make is that it knows the concept the user entered, but it does not know all of the correct generalizations of that concept. For instance, the concept "hitchhiker's guide to the galaxy" has been adapted to so many different forms of media that it is hard for Miro to keep up. If a user was performing a semantic search on "hitchhiker's guide to the galaxy" in the spring of 2005, they may have been referring to the movie, and not the other adaptations.

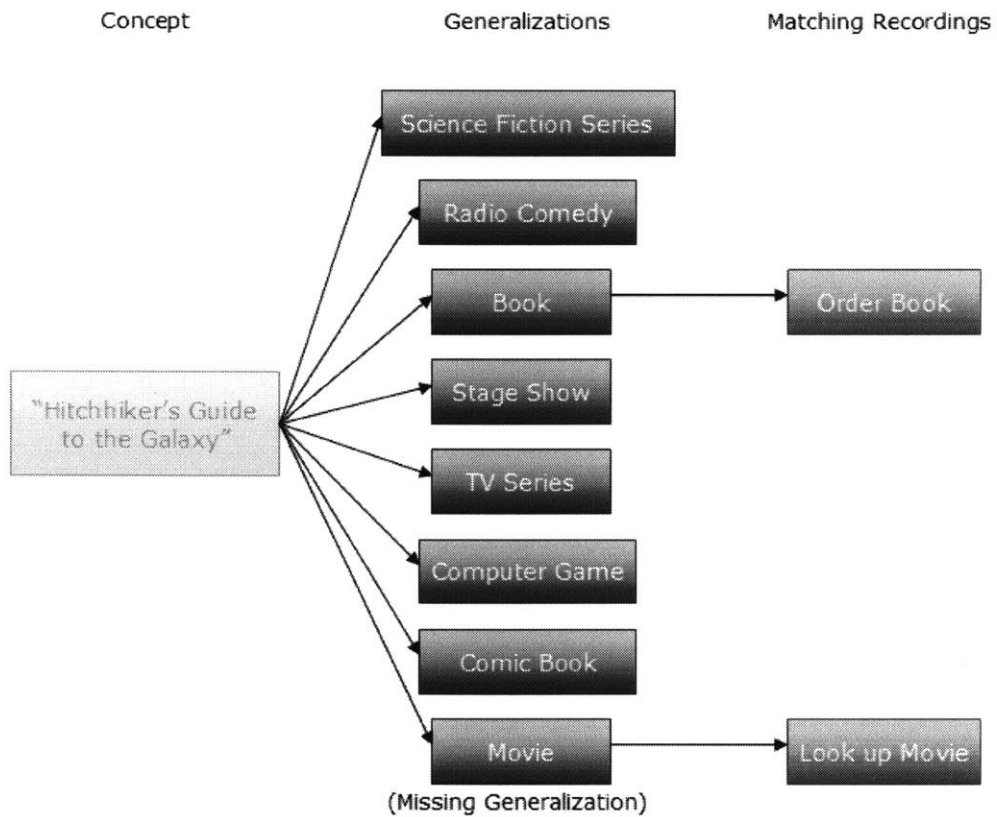


Figure 2-64 The Vogons would be very displeased with missing "Hitchhiker's Guide to the Galaxy" generalizations

In the second step of the training wizard, Miro displays the information that it currently knows about "hitchhiker's guide to the galaxy."

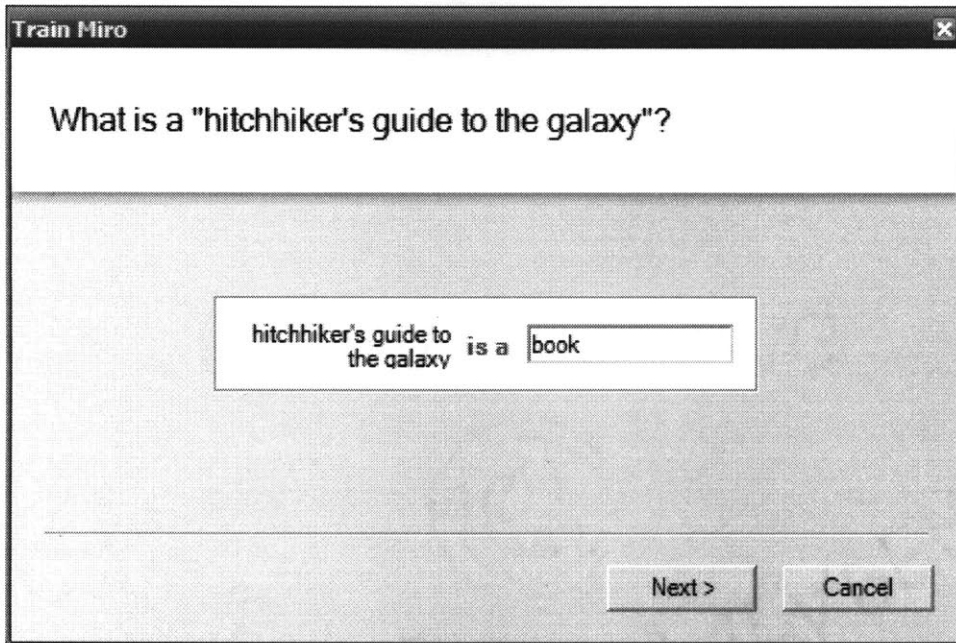


Figure 2-65 Miro thinks "Hitchhiker's Guide to the Galaxy" is a book

The user can then edit this field to instruct Miro that "hitchhiker's guide to the galaxy" is also a "movie." The generalization of "movie" completes after typing "mo."

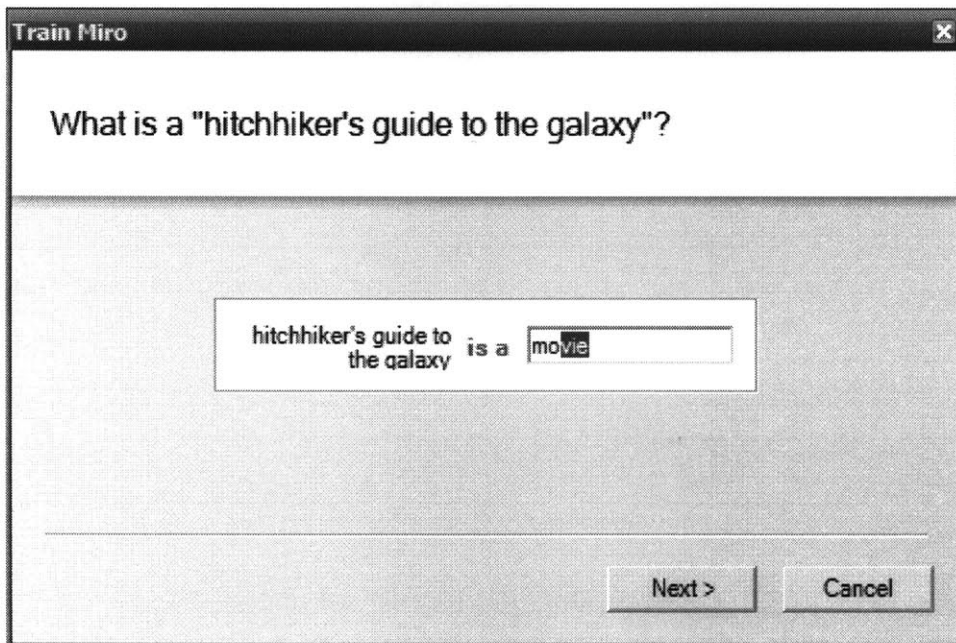
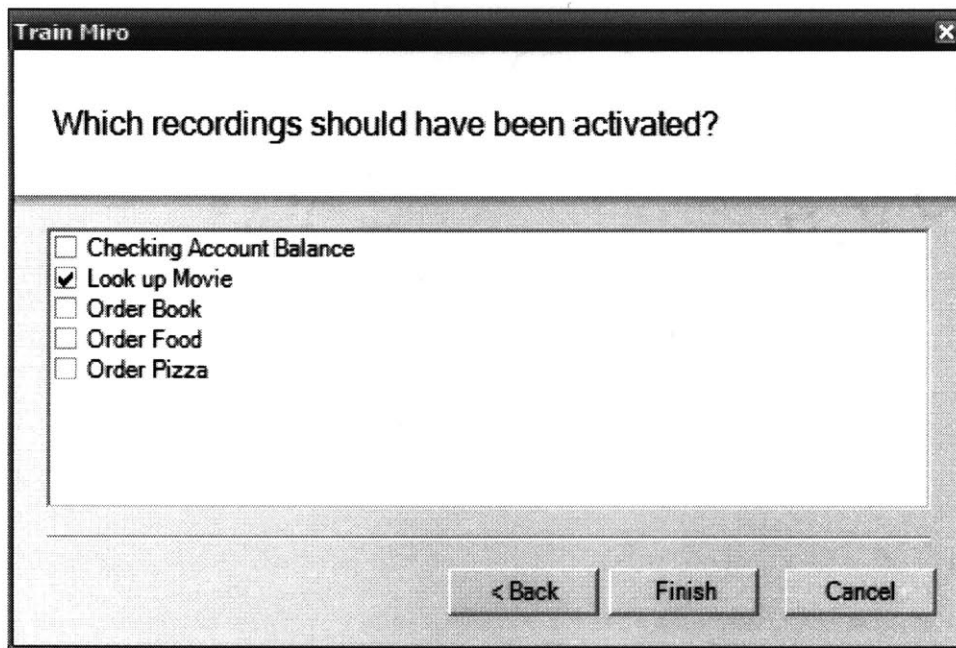


Figure 2-66 The user instructs Miro that "Hitchhiker's Guide to the Galaxy" is also a movie

In the final step, Miro automatically checks the recording "Look up Movie."



When the user clicks *Finish*, Miro writes the new piece of knowledge to *miro_learned.txt*, and adds it to the knowledge stored in memory.

(IsA "hitchhiker's guide to the galaxy" "movie")

One of the Current Recordings Does Not Have the Correct Set of Generalizations

The final mistake that can be corrected using the three-step training wizard is when one of the current recordings does not have the correct set of generalizations. For instance, let's imagine that the user has just spent several hours watching Hitchhiker's Guide to the Galaxy while gorging on guavas, and now has a headache. The user performs a semantic search on "tylenol," but no results are returned (again).

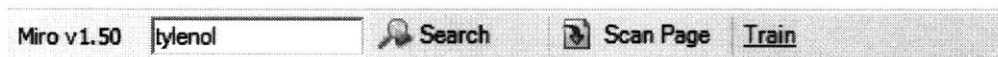


Figure 2-67 Miro does not know which recording should match the concept of "tylenol"

The user has created a recording called "Order Food" for the online grocery store FreshDirect. While FreshDirect does sell Tylenol, and other grocery store products, Miro does not realize this because the recording was originally trained on the single example of "diet coke."

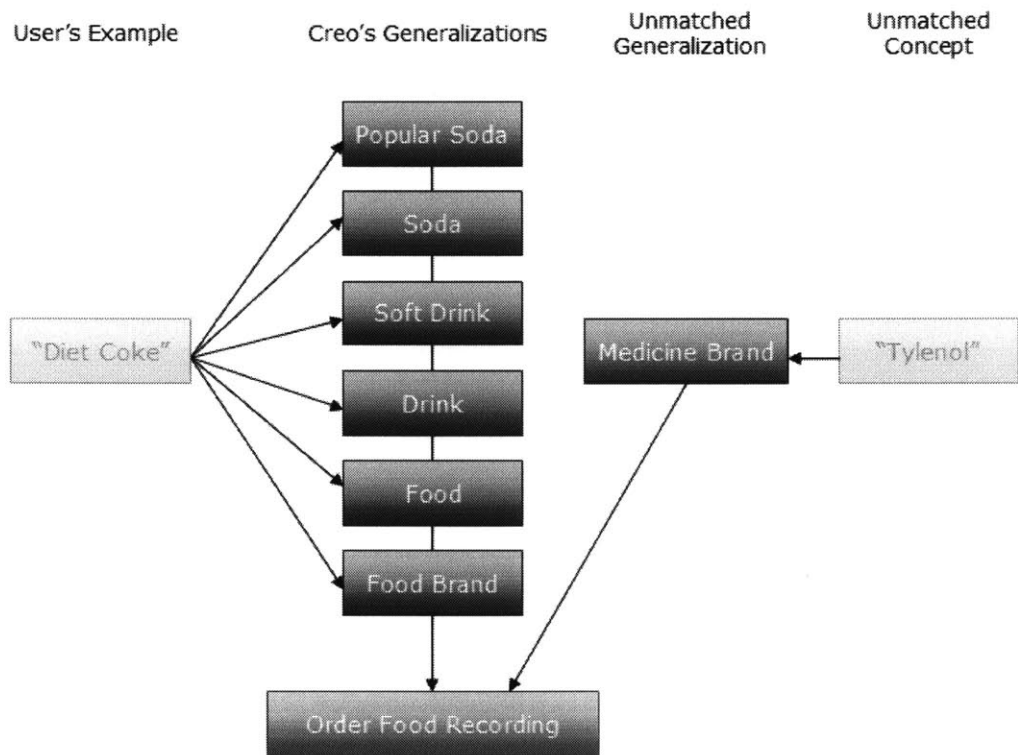


Figure 2-68 Tylenol is not a food, but you can order it from FreshDirect

To correct this mistake, the user hits the train button and Miro displays that it already knows what Tylenol is.

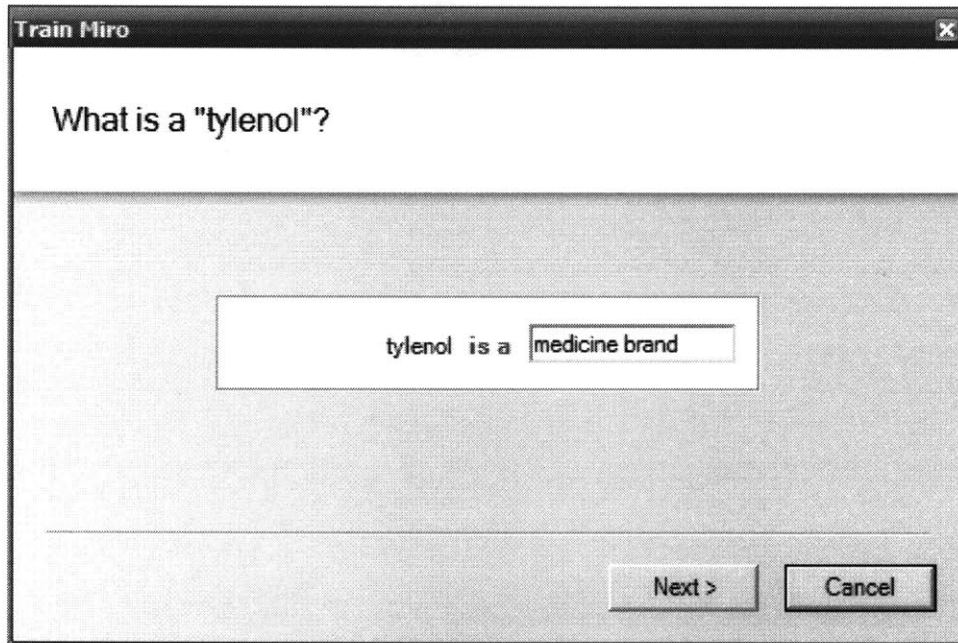


Figure 2-69 Miro already knows what Tylenol is

However, on the third step of the training wizard, Miro does not know what recording should have been activated, so no recordings are checked.

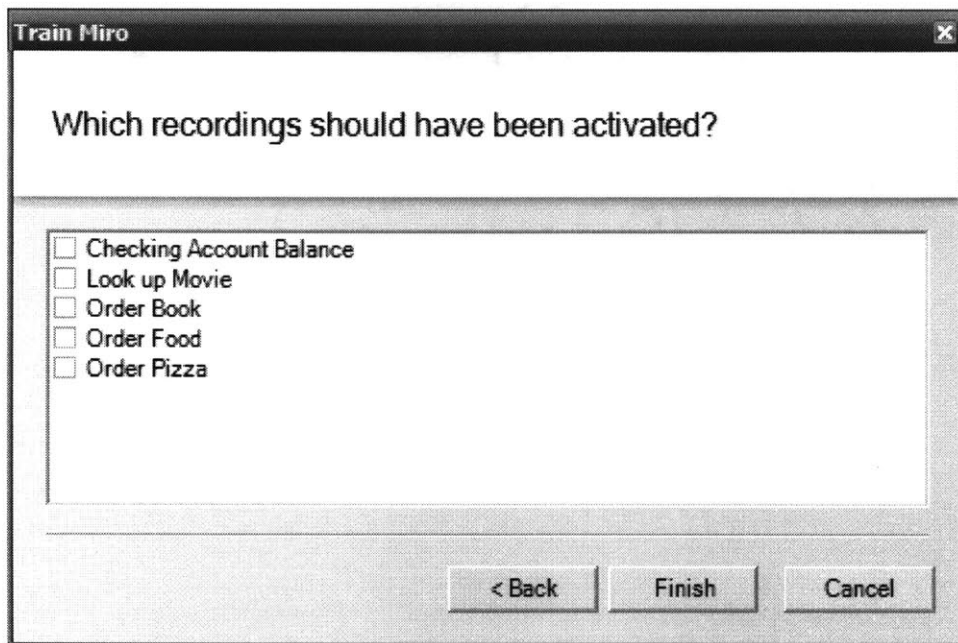


Figure 2-70 Miro does not know which recording should have been activated

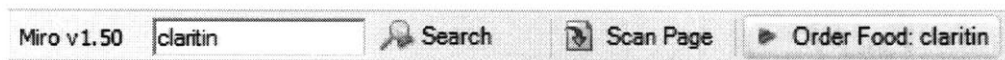


Figure 2-71 Miro now associates all types of medicines with the user's online grocery store

Miro Has Incorrect Knowledge

The fourth type of mistake that Miro can make is based on knowledge in ConceptNet or TAP being inaccurate. For instance, in the current version of Miro, a search for "Elton John" may return an unexpected response.

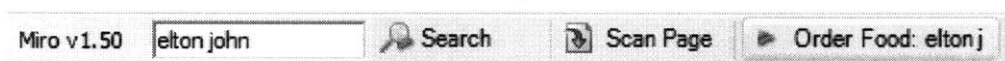


Figure 2-72 Why does Miro think "Elton John" is a food?

It turns out that this error is caused by line 172,049 in the ConceptNet file *predicates_nonconcise_nonkline.txt*, which contains the offensive statement:

```
(IsA "elton john" "fruit")
```

Currently, Miro has no user interface for deleting knowledge, since this type of error is usually pretty rare. However, a user interface for removing inaccurate (and immature) statements from Miro's knowledge bases should be included in future versions.

Design Principles for the Training Wizard

One of the design principles found throughout each step of the training wizard is the use of intelligent defaults. Miro leverages its knowledge bases to perform word completion on what the user is typing, like completing "mo" to "movie." Miro also leverages its knowledge bases to automatically fill out information for the user, like knowing that "Tylenol is a medicine brand." And finally in Step 3 of the training wizard, Miro automatically selects which recordings should have been activated based on the generalization provided in Step 2. If Miro doesn't know the concept in question, doesn't know the appropriate generalization, and doesn't know how that generalization relates

to the current set of recordings, then it will not be able to provide any intelligent defaults. However, for most errors, Miro should be able to provide intelligent defaults for at least one or two of the steps in the training wizard.

2.4.4 Learning in Miro - Learning From the Web

In addition to learning from the user through the training wizard, Miro has also been designed to learn new knowledge by reading Web pages that the user is looking at. For instance, by reading the text of this blog entry, Miro is able to learn that “The Killers” is a “band,” and activates the “Buy Music” recording for the user.



Figure 2-73 Miro learns a new piece of knowledge by reading the Web

From the user’s perspective, there is no difference in the user interface between when Miro matches a recording based on a piece of knowledge in

ConceptNet or TAP, and when it matches a recording based on a piece of knowledge it has just learned by reading a Web page.

When Miro’s learning feature is turned on, it passes each Web page loaded in the browser through a four-step process: (1) sentence boundary detection, (2) isolation of nouns and words used for pattern matching, (3) lemmatization, and (4) pattern matching against a set of regular expressions to extract IsA relationships. Each of these steps is discussed in the following sections.

Step 1: Sentence Boundary Detection

The first step Miro takes is to break the Web page apart into sentences. Miro’s sentence boundary detection method takes into account a set of common abbreviations like A.M., Dr. and Apt.

Input:	Output:
Everyone was sore from snowboarding the day before, so we decided to spend the day at the cabin. We rented several movies including Star Wars, The Empire Strikes Back and Return of the Jedi. Around 8 P.M. we ate at a small grill on Cooper Ave. near the mountain.	Everyone was sore from snowboarding the day before, so we decided to spend the day at the cabin.
	We rented several movies including Star Wars, The Empire Strikes Back and Return of the Jedi.
	Around 8 P.M. we ate at a small grill on Cooper Ave. near the mountain.

Table 2-5 Step 1 - Sentence boundary detection

Step 2: Isolation of Nouns and Words Used for Pattern Matching

The second step Miro performs on each sentence is removing every word that is not a noun, and is also not used for pattern matching in Step 4. The following non-noun words are kept for pattern matching: “are, is, a, an, some, kind, type, sort, instance, example, of, one, the, such, as, and, like, or, other, similar, including, especially, than.”

Input:	Output:
We rented several movies including Star Wars, The Empire Strikes Back and Return of the Jedi.	movies including Star Wars, The Empire Strikes Back and Return of the Jedi

Table 2-6 Step 2 - Isolation of nouns and words used for pattern matching

Terms that are not nouns but are capitalized are also kept. For instance, the movie "Rocky," or the TV show "Lost."

Step 3: Lemmatization

The remaining nouns are lemmatized, like "countries" → "country" or "books" → "book." Miro's lemmatization method also takes into account around 2,000 exceptions like "dwarves" → "dwarf" and "movies" → "movie." Title case words are not lemmatized.

Input:	Output:
movies including Star Wars, The Empire Strikes Back and Return of the Jedi	movie including Star Wars, The Empire Strikes Back and Return of the Jedi

Table 2-7 Step 3 - Lemmatization

Step 4: Pattern Matching Against a Set of Regular Expressions to Extract IsA Relationships

The final step is to perform pattern matching on the text to look for any type of IsA relationship. Miro currently looks for 11 different patterns. The sentence used throughout this example matches Pattern 8:

(O) including (S) [, (S)] [(and|or) (S)]

Input:	Output (using Pattern 8):
movie including Star Wars, The Empire Strikes Back and Return of the Jedi	(IsA "star wars" "movie") (IsA "empire strikes back" "movie") (IsA "return of the jedi" "movie")

Table 2-8 Step 4 - Using regular expressions to extract IsA relationships

This sentence generated three pieces of knowledge, although in this case the knowledge isn't new since Miro is already familiar with the Star Wars trilogy.

If the user had any recordings that took a generalization of "movie," Miro would then activate these terms.

Examples

The following examples demonstrate the 11 different patterns that Miro looks for to extract IsA relationships. The *After Parsing* row contains the sentence after the isolation and lemmatization of nouns (Steps 2 and 3). Red terms in the *After Parsing* row were kept because they are used when pattern matching. These patterns are based on the six lexio-syntactic patterns for hyponymy presented by Marti Hearst in his paper *Automatic Acquisition of Hyponyms from Large Text Corpora* [41], and the suggestions of Rada Mihalcea, who was a visiting professor at the Media Lab in June and July 2005.

Original Sentence	PowerBlast is a new kind of sports drink.
After Parsing	powerblast is a kind of sport drink
Matched Pattern	(S) (are is) (a an some) (kind type sort instance example) of (O)
Potential Semantic Knowledge	(IsA "powerblast" "sport drink")

Table 2-9 Example of Pattern 1

Original Sentence	Star Wars Episode III is a captivating movie
After Parsing	Star Wars Episode III is a movie
Matched Pattern	(S) is (a an) (O)
Potential Semantic Knowledge	(IsA "Star Wars Episode III" "movie")

Table 2-10 Example of Pattern 2

Original Sentence	Henry Lieberman is one of the research scientists here.
After Parsing	Henry Lieberman is one of the research scientist
Matched Pattern	(S) is one of (the)? (O)
Potential Semantic Knowledge	(IsA "Henry Lieberman" "research scientist")

Table 2-11 Example of Pattern 3

Original Sentence	Some animals such as lions, tigers and bears seem very scary.
After Parsing	animal such as lion, tiger and bear
Matched Pattern	(O) such as (S) [, (S)] [and (S)]

Potential Semantic Knowledge	(IsA "lion" "animal") (IsA "tiger" "animal") (IsA "bear" "animal")
------------------------------	--

Table 2-12 Example of Pattern 4

Original Sentence	He follows Baseball teams like the Red Sox, Yankees and Rockies.
After Parsing	Baseball team like the Red Sox, Yankees and Rockies
Matched Pattern	(O) like the (S) [, (S)] [and (S)]
Potential Semantic Knowledge	(IsA "red sox" "baseball team") (IsA "yankees" "baseball team") (IsA "rockies" "baseball team")

Table 2-13 Example of Pattern 5

Original Sentence	Remember to bring hamburgers, popsicles and other picnic foods.
After Parsing	hamburger, popsicle and other picnic food
Matched Pattern	(S) [, (S)] (and or) other (O)
Potential Semantic Knowledge	(IsA "hamburger" "picnic food") (IsA "popsicle" "picnic food")

Table 2-14 Example of Pattern 6

Original Sentence	She should get the Audiovox SMT5600 or a similar Smartphone.
After Parsing	the Audiovox SMT5600 or a similar Smartphone
Matched Pattern	(The the) (S) or a similar (O)
Potential Semantic Knowledge	(IsA "audiovox smt5600" "smartphone")

Table 2-15 Example of Pattern 7

Original Sentence	We rented several movies including Star Wars, The Empire Strikes Back and Return of the Jedi.
After Parsing	movie including Star Wars, The Empire Strikes Back and Return of the Jedi
Matched Pattern	(O) including (S) [, (S)] [(and or) (S)]
Potential Semantic Knowledge	(IsA "star wars" "movie") (IsA "the empire strikes back" "movie") (IsA "return of the jedi" "movie")

Table 2-16 Example of Pattern 8

Original Sentence	Do not forget important camping supplies especially a rain tarp, insect repellent and a compass.
After Parsing	camping supply especially a rain tarp, insect repellent and a compass
Matched Pattern	(O) especially (a the) (S) [, (S)] [(and or) (a the) (S)]
Potential Semantic Knowledge	(IsA "rain tarp" "camping supply")

Knowledge	(IsA "insect repellent" "camping supply") (IsA "compass" "camping supply")
-----------	---

Table 2-17 Example of Pattern 9

Original Sentence	Any airport other than LaGuardia would be better.
After Parsing	airport other than LaGuardia
Matched Pattern	(O) other than (S)
Potential Semantic Knowledge	(IsA "laguardia" "airport")

Table 2-18 Example of Pattern 10

Original Sentence	Sports: swimming, tennis and golf
After Parsing	Sports: swimming, tennis and golf
Matched Pattern	(O): (S) [, (S)] [(and or) (S)]
Potential Semantic Knowledge	(IsA "swimming" "sport") (IsA "tennis" "sport") (IsA "golf" "sport")

Table 2-19 Example of Pattern 11

Limitations

Returning to the original example, Miro was able to detect that "The Killers" is a "band" by reading the Web page.

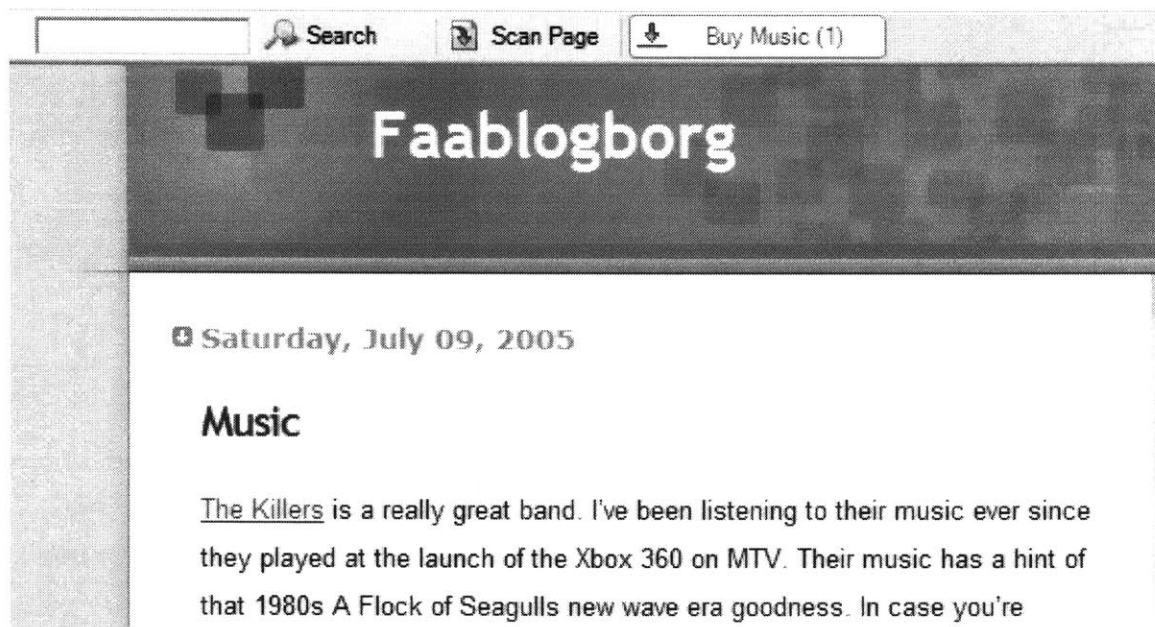


Figure 2-74 1980s new wave era goodness

In the third sentence: "Their music has a hint of that 1980s A Flock of Seagulls new wave era goodness," the author relates the band "The Killers" to the band "A Flock of Seagulls." A human reading this sentence would be able to infer that A Flock of Seagulls is also a band, even though it is not explicitly stated, and even if the human had never heard of "A Flock of Seagulls." Miro however, is not nearly advanced enough to figure out that A Flock of Seagulls is also a band. Miro can only infer knowledge from sentences that match one of its 11 patterns. And even when Miro finds a pattern, the knowledge is not guaranteed to be correct. For instance, Miro would mistakenly parse this example:

The following sentence is not true: A tennis ball is a tasty fruit.

Because Miro will occasionally make mistakes parsing knowledge from Web pages, it works cooperatively with the user to infer which knowledge is useful.

Working Cooperatively with the User to Learn New Knowledge

After Miro has located a piece of potentially useful knowledge, Miro monitors the actions that the user takes to infer if the knowledge is useful or not. The first step that Miro takes after locating a new piece of knowledge is to display a result button in the toolbar. At this point the new knowledge is held in memory as potentially useful. If the user clicks the button to highlight the information on the page that relates to the selected recording, Miro notices this action and records the new piece of knowledge to the file *LearnLog.xml*.

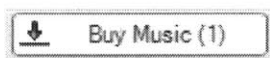


Figure 2-75 The user selects the activated recording, indicating that the new knowledge may be useful

The reason Miro writes the new piece of knowledge to *LearnLog.xml* is because if the user clicks through on "The Killers," the recording will play in a

new instance of Internet Explorer, and the only way for the new instance of Miro to know that the information being used in this recording was inferred from the text of a Web page is by keeping a record of potentially useful knowledge in the file *LearnLog.xml*. After the user has clicked through on "The Killers," the new instance of Miro playing the recording notices that this knowledge was recently learned, and that the user decided to click through on it. Miro then records the new knowledge (ISA "The killers" "band") to the file *miro_learned.txt*. In retrospect, tracking if users click through on selected concepts on the page is not by itself enough to conclude that the new piece of information is true. Users may decide to click through on a piece of information that is incorrect, if they are curious why the information was turned into a hyperlink. After noticing that the user showed an interest in a new piece of knowledge, Miro should ask the user the direct question to confirm the new knowledge (like "Is 'The Killers' a band?") before it permanently records this information.

Challenges Facing Unassisted Knowledge Capture through Crawling the Web

Since Miro is able to learn new pieces of knowledge by reading the Web, an obvious idea is: "Why not crawl 8 billion pages and build a significantly larger version of Open Mind?" Unfortunately, doing so would not necessarily result in a very good knowledge base. The reason for this is that commonsense knowledge, the semantic information between the lines of a Web page, is usually inferred by the reader. Since the author assumes that the reader already knows certain pieces of common information, a lot of the useful semantic knowledge is never directly stated on the Web.

There are, of course, exceptions. For instance:

"Henry Lieberman is one of the research scientists here."

Or:

"The Killers is a really great band."

In these sentences, the author is telling the reader a piece of information that the reader may not know. Additionally, there are many sentences where semantic knowledge can be extracted, even though the author assumes the reader already knows the semantic information. For instance:

"Do not forget important camping supplies, especially a rain tarp, insect repellent and a compass."

In this sentence, the author likely assumes that the reader already knows that a "rain tarp," "insect repellent" and a "compass" are all "camping supplies," but the author states that they are anyway.

Exceptions like these make it possible to extract semantic information from the Web. However, there is no relationship between how often a piece of knowledge is stated on the Web and how useful that piece of knowledge is. Unfortunately, the inverse might even be true. It would appear that the most general and obvious pieces of information (and therefore the most useful for Miro to learn) also happen to be the least likely to be directly stated, because in most cases, the author assumes the reader already knows this information.

Let's consider an example of this. This diagram shows useful generalizations of the famously popular search query "Britney Spears."

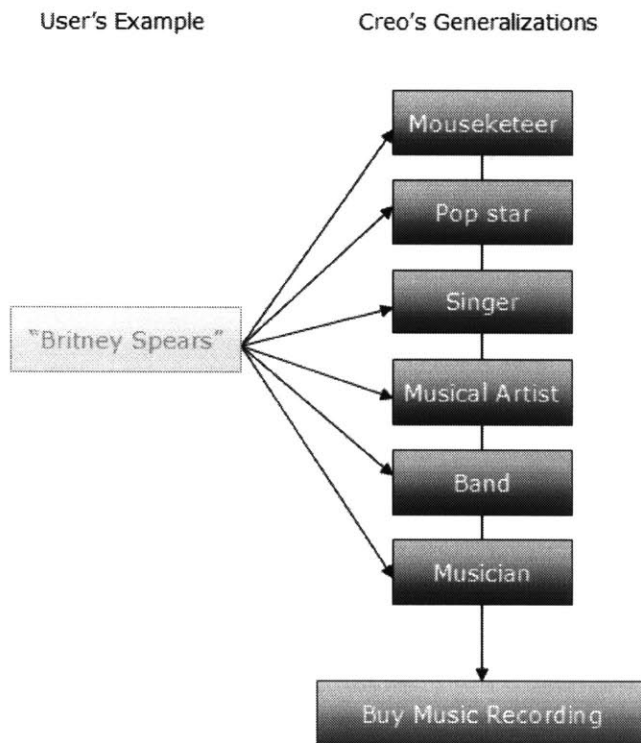


Figure 2-76 Generalizations of Britney Spears

But how easy is it to extract the same type of generalizations from the Web? Using Google's cache of the Web, and Miro's Pattern 1 "(Britney Spears) is (a|an) (O)," produces a very noisy (and rather colorful) set of results. Duplicate results have been removed, and some "generalizations" have been censored.

Google Search for "Britney Spears is a", first 100 unique results:
Britney Spears is a three-headed alien
Britney Spears is a pop idol
Britney Spears is a bigamist
Britney Spears is a dangerous idiot
Britney Spears is a cultural phenomenon
Britney Spears is a slut
Britney Spears is a musical genius
Britney Spears is a popular American singer

Britney Spears is a **female man**
Britney Spears is a **new elvis presley**
Britney Spears is a **silly c_nt**
Britney Spears is a **lightweight pop princess**
Britney Spears is a **superstar**
Britney Spears is a **good role model**
Britney Spears is a **losing battle**
Britney Spears is a **b___h**
Britney Spears is a **teenager**
Britney Spears is a **pop culture icon**
Britney Spears is a **skank**
Britney Spears is a **single woman**
Britney Spears is a **bad role model**
Britney Spears is a **public health crisis**
Britney Spears is a **secret coke head**
Britney Spears is a **respectable artistic musician**
Britney Spears is a **product**
Britney Spears is a **hoochie**
Britney Spears is a **whore**
Britney Spears is a **fraud**
Britney Spears is a **very good singer**
Britney Spears is a **bit serious**
Britney Spears is a **candidate for mother of the year**
Britney Spears is a **very talented singer**
Britney Spears is a **mousekeeter**
Britney Spears is a **wholesome southern girl**
Britney Spears is a **step mom**
Britney Spears is a **big f___ing dork**
Britney Spears is a **tevisual retard**
Britney Spears is a **man**
Britney Spears is a **skanky ho bag**
Britney Spears is a **mega babe**
Britney Spears is a **tramp**
Britney Spears is a **pop star**
Britney Spears is a **doomed attempt**
Britney Spears is a **dream within**
Britney Spears is a **paragon**
Britney Spears is a **upper white class piece of trash**
Britney Spears is a **revved up redhead**
Britney Spears is a **talented singer**
Britney Spears is a **disaster**
Britney Spears is a **women**
Britney Spears is a **part of this youth culture**
Britney Spears is a **good singer**
Britney Spears is a **virgin**
Britney Spears is a **boy band**
Britney Spears is a **bad influence**
Britney Spears is a **million dollar stripper**
Britney Spears is a **very pleasant kisser**
Britney Spears is a **big user of bath and body works**
Britney Spears is a **long shot**
Britney Spears is a **slave**
Britney Spears is a **laser physicist**

Britney Spears is a success
Britney Spears is a joke
Britney Spears is a pretty easy target
Britney Spears is a threat to decency
Britney Spears is a super hot chick
Britney Spears is a tremendous spokesperson
Britney Spears is a lame way to exercise your venom
Britney Spears is a master of the pop
Britney Spears is a fan of president bush
Britney Spears is a tart
Britney Spears is a bad influence
Britney Spears is a pre-pubescent boy
Britney Spears is a horrid b___h
Britney Spears is a hypocrite
Britney Spears is a has been
Britney Spears is a dropkick
Britney Spears is a democrat
Britney Spears is a brat
Britney Spears is a a_s h__e
Britney Spears is a daughter of olivia
Britney Spears is a really good person
Britney Spears is a sagittarius
Britney Spears is a national monument
Britney Spears is a common scrubber
Britney Spears is a born-again christian
Britney Spears is a pig
Britney Spears is a liar
Britney Spears is a drag queen
Britney Spears is a soft and sensual fragrance
Britney Spears is a world-renowned semiconductor physicist
Britney Spears is a bit of a punk
Britney Spears is a moron
Britney Spears is a tough task
Britney Spears is a fat cow
Britney Spears is a walking ad for pepsi
Britney Spears is a major cause for global conflict
Britney Spears is a very popular airhead
Britney Spears is a talented artist
Britney Spears is a kewl woman

Table 2-20 Britney Spears is a three-headed alien

While some useful generalizations of Britney Spears do appear in the results (like “musician” and “singer”), the results favor statements about Britney Spears that the reader may not already know. For instance, “Britney Spears is a three-headed alien,” or “Britney Spears is a public health crisis.” Additionally, some of the results are sarcastic, like “Britney Spears is a world-renowned semiconductor physicist.” Filtering this list by the generality of each description does not improve the results.

Britney Spears is too common of a concept to reliably learn about by reading the Web. The concept of guava however, is less common. This leads to a much better set of generalizations when reading the Web.

Google Search for "guava is a", first 30 unique results:
guava is a package that implements coding theory algorithms
guava is a very serious, habitat-altering pest
guava is a good source of lycopene
guava is a relatively slow-growing shrub
guava is a small tree
guava is a prolific fruiter
guava is a home fruit tree
guava is a registered trademark
guava is a leader in the area of cellular analysis
guava is a common shade-tree
guava is a very nutritious fruit
guava is a fruit
guava is a tree fruit
guava is a very nutritious fruit
guava is a prime host of the mediterranean
guava is a preferred host for fruit flies
guava is a favorite fruit
guava is a form of strawberry guava
guava is a hundred times more loaded than the body
guava is a great fruit
guava is a small-sized tree
guava is a scrubby tree
guava is a shallow-rooted, many branched shrub
guava is a popular flavor for jams and juices
guava is a bit overpowering
guava is a round yellowish-green fruit
guava is a lime-green, egg-shaped fruit
guava is a just in time compiler
guava is a source of vitamin A
guava is a zone 8 and 9 plant

Table 2-21 Guava is a tree/fruit

Learning from the Web

Learning commonsense knowledge and semantic information by reading the Web is not an impossible task. However, any algorithm that attempts to do so must be able to tell the difference between Britney Spears and a guava. More specifically, the algorithm must be able to tell the difference between noise, and a consistent pattern of accurate generalizations. It would appear

that learning commonsense knowledge about uncommon concepts (like a guava) is rather easy, while learning commonsense knowledge about common concepts (like Britney Spears) is difficult. Telling the two apart is left to future research.

Leveraging Volunteers on the Web for Assisted Web Crawling

One potential solution would be a hybrid approach: using a Web crawler to find consistent patterns of generalizations, and then relying on volunteers on the Web to approve the knowledge before it is recorded in Open Mind. The advantage of this approach is that it is faster to click *Yes* or *No*, than it is to enter raw knowledge. So, instead of entering knowledge through the current set of Open Mind activities, volunteers using the Open Mind site could quickly confirm or deny pieces of knowledge presented to them. Open Mind would ask the volunteer questions like:

Is Britney Spears a three-headed alien?

Is a guava a fruit?

It would also be interesting to record the time it took users to confirm or deny a piece of knowledge, and to potentially integrate this information into Open Mind as well.

Conclusion

Because Miro only reads pages the user is looking at, it does not have to face many of the challenges that come with unassisted knowledge capture. Additionally, Miro is only looking for generalizations that match the user's current set of recordings. So, if Miro encounters the sentence "Britney Spears is a three-headed alien," it will parse the sentence using Pattern 1. However, since none of the user's recordings were looking for the generalization of "three-headed alien," the knowledge will be subsequently thrown out. Miro's ability to learn by reading the Web is not a flawless solution, and it has many limitations. However, by working cooperatively with the user, the captured knowledge is of a much higher quality than would be obtained through unassisted Web crawling.

2.4.5 Conclusion - Knowledge Capture Through the Use of Applications

Miro's ability to learn by reading the Web allows it to passively increase the size of its knowledge base over time. Miro's training interface allows users to actively teach Miro new knowledge, to make it a more useful application.

Many of the applications developed by the Software Agents Group over the last two years that leverage the Open Mind knowledge base [42] have treated Open Mind as a static resource. Some, like ARIA [43-46], capture knowledge about the specific user, creating a knowledge base of personal common sense. However, very few applications have closed the loop entirely, by adding knowledge directly to Open Mind throughout their use, in addition to leveraging the knowledge already there.

The impact of linking knowledge capture to the use of an application could have significant implications for the artificial intelligence community if the application became immensely popular. For instance, if Miro was used by as many people as the Google toolbar, its ability to learn by reading the Web could potentially result in the creation of a very large knowledge base, in a very short amount of time.

Just as Open Mind has benefited from the intelligence of thousands of volunteers on the Web, interactive applications that utilize the knowledge in Open Mind should leverage the intelligence of their users to continue the knowledge capture process.

2.5 Security and the Viral Spread of Recordings

2.5.1 Introduction

This section discusses the security implications of creating and sharing recordings.

Security

Recordings created with Creo contain a significant amount of personal information about the user, including their address and credit card number, and many of their passwords. Additionally, Adeo allows users to invoke recordings remotely, resulting in additional privacy and security concerns. The way Creo manages the user's personal information is discussed in Section 2.5.2. The security precautions taken by Adeo are discussed in Section 2.5.4.

Miro is not discussed in this section because Miro relies on Creo for the tasks of opening and playing recordings.

The Viral Spread of Recordings

Due to the way Creo manages the storage of users' personal information, users are free to give recordings they create to their friends, sharing the functionality of the recording, without sharing any of their specific personal information. The implications of this are discussed in Section 2.5.3.

2.5.2 Creo's Detection and Storage of Personal Information

As users create recordings with Creo, Creo needs to decide which pieces of input should remain static, and which pieces of input will change each time the recording is played. Creo makes this decision based on three observations:

1. If the text is a piece of personal information about the user, like their name or email address.
2. The number of generalizations of the text. For instance, "diet coke" generalizes to five concepts, while the text afaaborg (a username) does not generalize to any concepts.
3. If the name of the text field matches a predetermined list of fields that should remain static. This list includes field names like "user," "username," and "userNumber."

These three observations lead Creo to decide if the information should remain static or not when the recording is played. For instance, in the figure below, Creo defaults to always entering the user's address, instead of prompting the user to enter an address every time the recording is played.

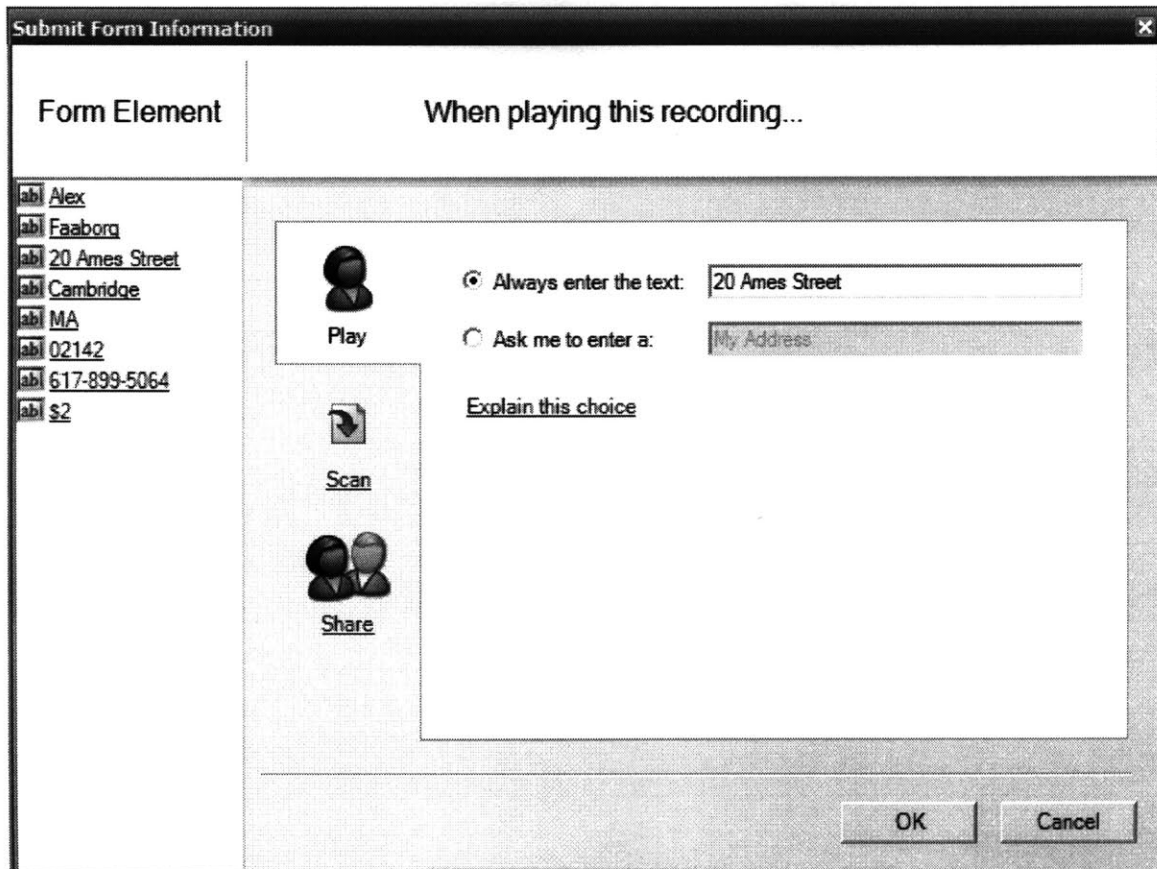


Figure 2-77 Creo recognizes the user's address

The User's Profile

Creo was able to recognize the user's address by matching the text of the form element against a profile that the user has filled out.

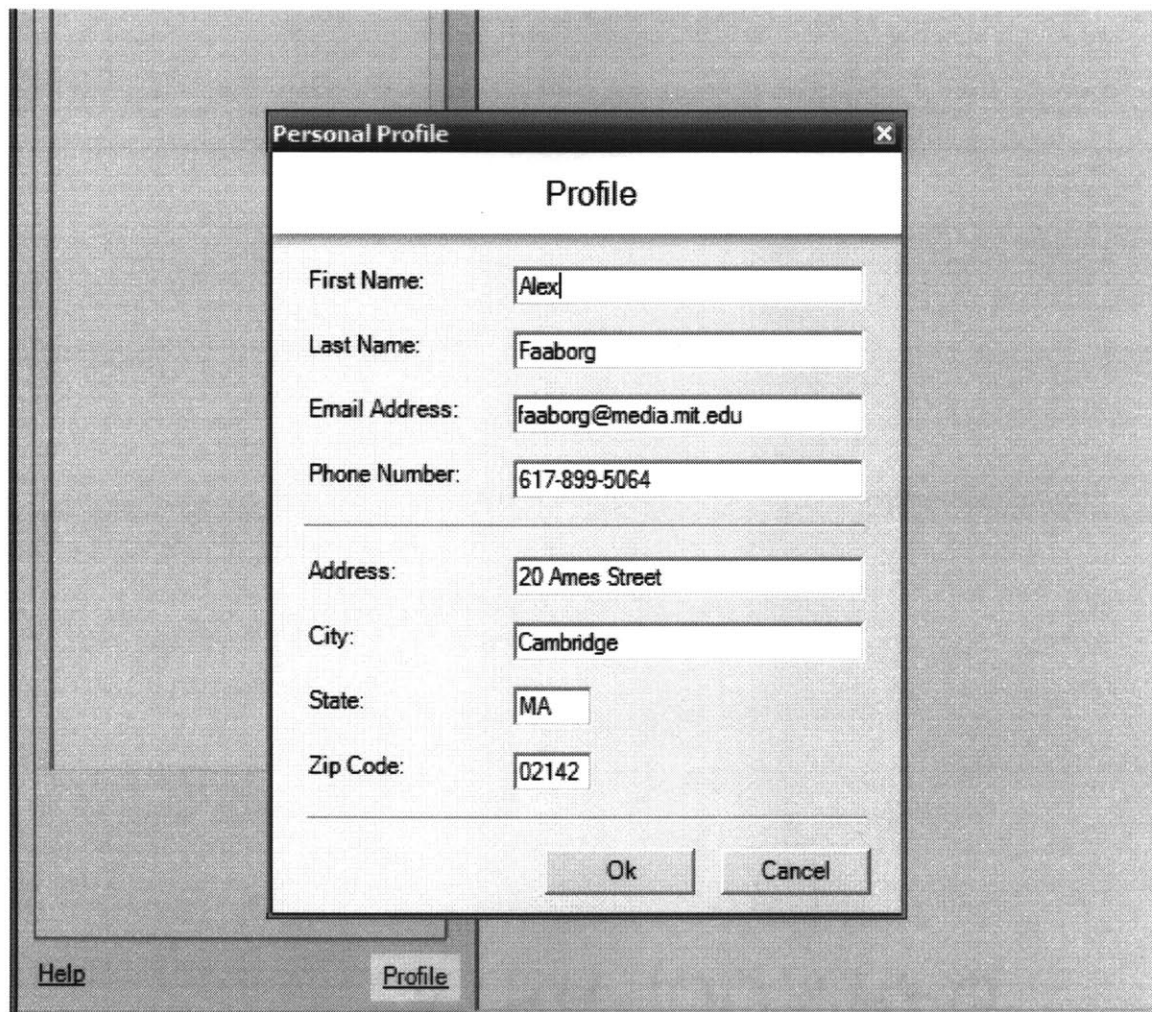


Figure 2-78 The user's profile

Currently, the *Profile* window is overly simplistic. Future iterations of the *Profile* window should include user defined fields, and the ability to enter multiple values for each field. For instance, most users would want to enter their home address and their work address. Similarly, many users will use multiple email addresses, depending on the Web site they are interacting with. The code underlying Creo's user profile supports this type of expandability. However, the user interface itself currently does not. It is also worth exploring in future versions ways to build up the profile over time

through analyzing data in the user’s recordings, instead of having the user provide this information explicitly. The profile should also be used to drive an auto fill feature, similar to the Google toolbar.

When Creo finds a match between the text the user has entered into a form element and the user’s profile, Creo marks this input as personal information. Users can also directly control which input is marked as personal information by checking a box on the *Share* tab of the form window.

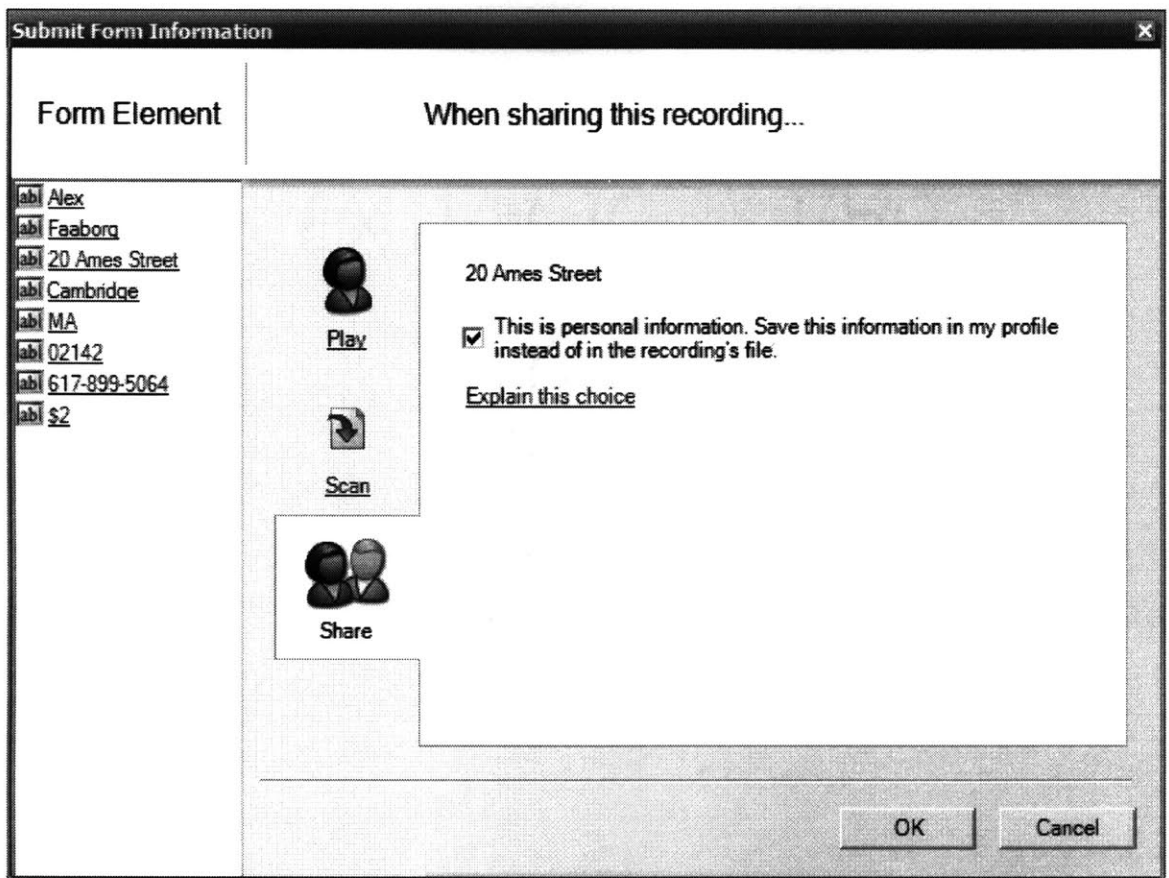


Figure 2-79 Users can control which pieces of input are personal information from the *Share* tab

By checking “This is personal information. Save this information in my profile instead of in the recording’s file,” the user is instructing Creo where to store this information. In the case of the user’s street address, shown above, this information is already stored in the user’s profile. One advantage of this

design is the ability to update personal information in a single location. For instance, if the user moves, they can update the address in the *Profile* window, and all of their recordings will automatically switch to using the new address.

Personal Information Specific to a Particular Recording

In addition to general personal information like a user's phone number, or email address, some recordings contain specific personal information, like a bank account number, or password.

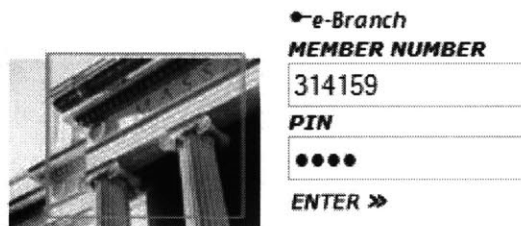


Figure 2-80 Step 1: The user types their account number into a form

Due to the form element field names, Creo automatically tags the account number and password as personal information.

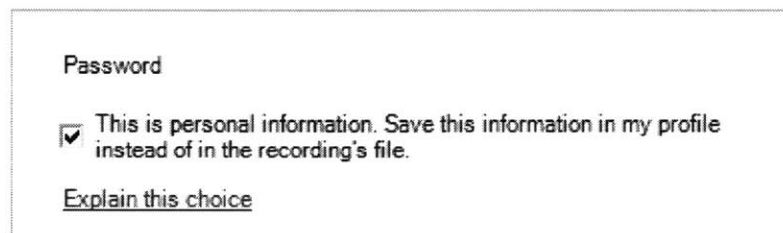


Figure 2-81 Step 2: Creo automatically identifies the input as personal information

When the recording is saved, Creo writes the user's bank account number and password to their *profile.xml* file:

```
<PersonalInformation>
  <recordingGuid>2bca8bbc-47f5-4d74-b615-c6b4caa61b96</recordingGuid>
  <valGuid>f1c2a234-09ac-433c-91cd-d56a752be23c</valGuid>
  <actualVal>314159</actualVal>
</PersonalInformation>
<PersonalInformation>
  <recordingGuid>2bca8bbc-47f5-4d74-b615-c6b4caa61b96</recordingGuid>
  <valGuid>8279ac98-89c3-4da8-b3eb-5176189703cc</valGuid>
  <actualVal>2653</actualVal>
</PersonalInformation>
```

Both pieces of personal information are assigned GUIDs, or globally unique identifiers. These GUIDs are used to reference the information in the recording's xml file, *Checking_Account_Balance.xml*:

```
<FormElement>
  <type>text</type>
  <name>userNumber</name>
  <val>f1c2a234-09ac-433c-91cd-d56a752be23c</val>
  <ask>>false</ask>
  <askPrompt>userNumber</askPrompt>
  <personalInfo>>true</personalInfo>
</FormElement>
<FormElement>
  <type>password</type>
  <name>password</name>
  <val>8279ac98-89c3-4da8-b3eb-5176189703cc</val>
  <ask>>false</ask>
  <askPrompt>password</askPrompt>
  <personalInfo>>true</personalInfo>
</FormElement>
```

Because the recording's xml file contains a GUID referencing the user's personal information, and not the information itself, users are free to publicly share this file with their friends.

When Creo or Miro open the recording file stored on the hard drive, they resolve any personal information GUIDs with the actual values stored in the user's *profile.xml* file.

2.5.3 The Viral Spread of Recordings

Creo's ability to detect personal information allows users to share the functionality of a recording, without sharing any of the specific data. For information already stored in the user's profile, this transition should be seamless. For instance, a user creates a recording to order a pizza from a local pizza place down the street, charging it to their credit card and delivering it to their apartment. Then they give this recording to their friend. When the user's friend plays this recording, the pizza will be automatically charged to their credit card, and delivered to their apartment.

Exporting Recordings

To share a recording, a user simply needs to click the *Explore* hyperlink on the *Player* tab of Creo, and then transfer the recording's file to the person they wish to share the recording with.

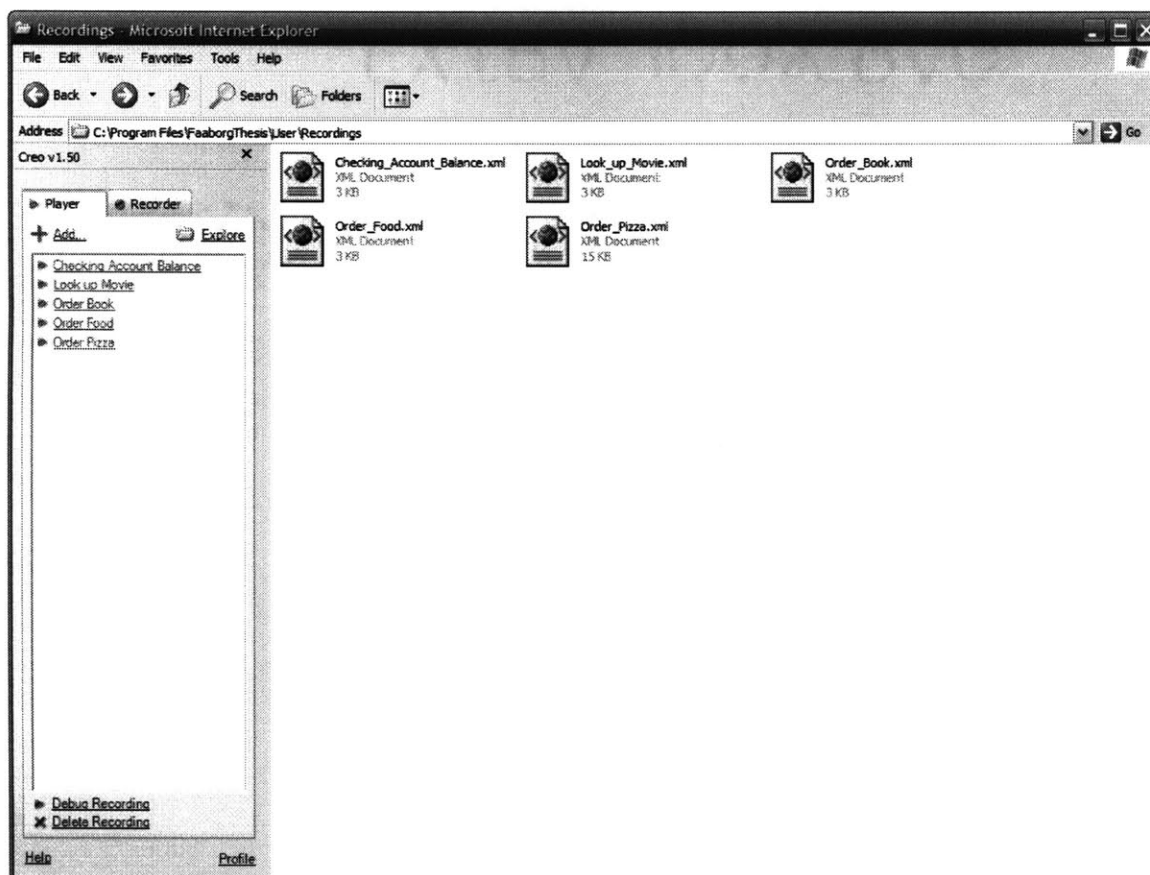


Figure 2-82 The user clicks *Explore* and is taken to their recordings folder

Importing Recordings

When a user adds a new recording file to their recordings folder, the new recording will be automatically loaded into memory. If any specific personal information is missing, the user is prompted to provide it. For instance, if a user adds a new recording to check their account balance at the MIT Federal Credit Union, the recording's XML file will contain unresolved personal information GUIDs for the user number and password. Immediately after adding the recording, the user will be asked to enter their user number and password, since their *profile.xml* file does not currently contain this information.

Leveraging Social Networks for Trust

Because these recordings deal with sensitive information, like the user's credit card number or bank account, it is important that users receive recordings from a trusted source. By creating a system where recordings spread virally through a social network, each user knows who they can hold accountable if a recording acts maliciously. Recordings that do not work correctly, or engage in phishing, are not likely to spread very far. This is because creating and sharing such a recording with your immediate friends is socially unacceptable. By relying on the implicit trust of social networks, many of the security challenges involved in creating a public global repository of recordings can be avoided.

Two Classes of Users – Producers and Consumers

This model of sharing recordings through a social network also has usability implications. By enabling users to add new recordings from their friends, some users will be able to take advantage of the features of Miro and Adeo without ever having to use Creo to create a recording. Even though Creo allows users to create recordings by example, and does not require users to know how to program, it is still too complicated a process for a class of novice users. In user testing (discussed in detail in Chapter 5), some users didn't understand why they personally had to create recordings. One subject

stated, “Why do I even have to create the recording? Shouldn’t the site do it, or someone else, and I can just download it?” Another subject felt that Creo should be smart enough to complete actions without training, stating, “I don’t understand why you have the recording interface. I shouldn’t have to record actions for it to know what to do.” By allowing users to share recordings with each other, those that do not understand how to use Creo, or who disagree with Creo’s existence, can avoid having to use it.

Additionally, even users who are familiar with using Creo to create recordings will likely rely on importing recordings from their friends, since doing so is more efficient. For instance, if a user’s office mate has already created a recording to order a pizza from a pizza place down the street, or check an account balance at a local bank, it is easier to copy the files than it is to re-record the procedures. The sharing model also makes it easier for users to quickly transition to new recordings when a Web site updates its user interface, effectively breaking Creo’s ability to interact with the site.

Future Work to Improve Security When Sharing Recordings

Due to the security and privacy concerns of sharing recordings, it is very important that the user is aware of what specific actions Creo is taking when exporting a recording, or playing a newly imported recording. Future versions of Creo should contain a user interface for closely inspecting recordings before they are exported or imported.

When exporting recordings, there is currently no user interface for reviewing which pieces of information are treated as personal information. This is obviously very important, since Creo’s methods for automatically detecting personal information are not perfect, and the user might not be thinking about sharing a recording when they first create it.

Similarly, when importing recordings, there is currently no user interface for inspecting what specific actions the recording will take. For instance, while

there is a confirmation interface for playing recordings, this window currently only displays page titles, instead of URLs. This interface was designed so that users could remember what a particular recording did before playing it, and details like a page's URL were omitted to improve readability. To prevent phishing attacks, a more detailed confirmation interface should be developed so that users can thoroughly inspect a new recording without having to look directly at its XML representation.

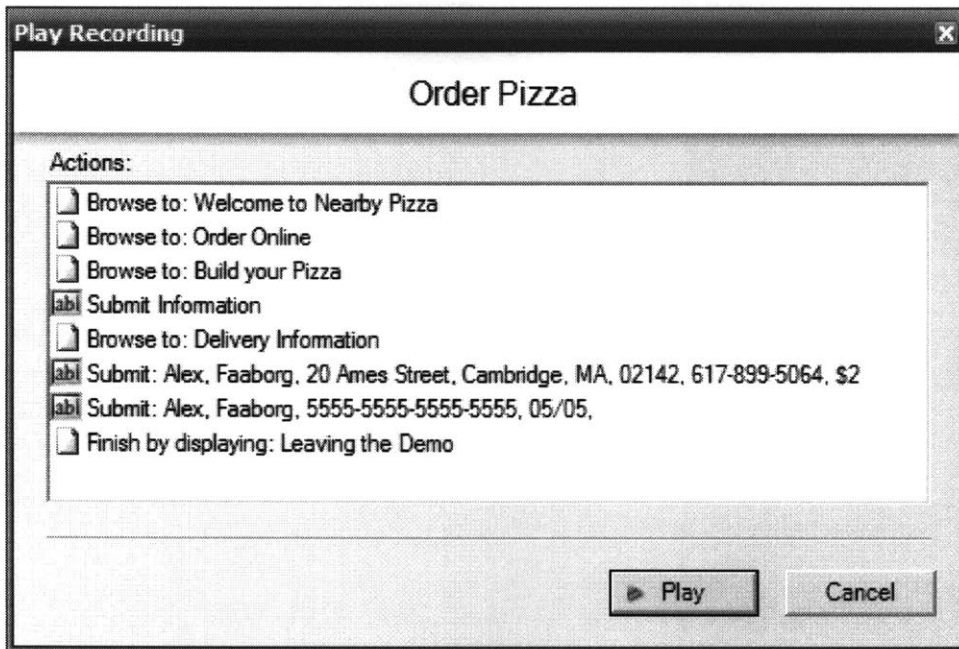


Figure 2-83 Creo's confirmation interface

The confirmation window also currently contains several bugs. The fourth action shown in the figure above, "Submit Information," should resolve to the type of pizza the user is ordering.

Additionally, the ability to step through a recording's actions (discussed in Section 2.4.2) should be added to the user interface for importing a recording.

Beyond user interface changes, the file *profile.xml* should be encrypted, since this file contains a wealth of personal information about the user. In user

testing, several users stated concern over storing a credit card or bank account number on their hard drive, since it is not uncommon for computers to be compromised or stolen.

2.5.4 Security in Adeo

Reducing a Recording to the Minimal Input and Output Helps Mitigate the Threat of Packet Sniffing

Adeo allows users to complete tasks from mobile devices, providing the most minimal amount of input and output. For instance, the minimal input for checking the balance of a bank account is clicking *Go* and the minimal output is viewing the number returned. Beyond streamlining the user interface, this approach has the additional benefit of reducing the amount of bandwidth used by the mobile device, and makes it impossible to packet sniff personal information like the user's account number. This is because this information is never transmitted to the mobile device. The user's *profile.xml* file remains on the personal computer or server responsible for playing recordings, and at no point in the process of playing a recording is the information found in the user's *profile.xml* file transferred to the user's mobile device.

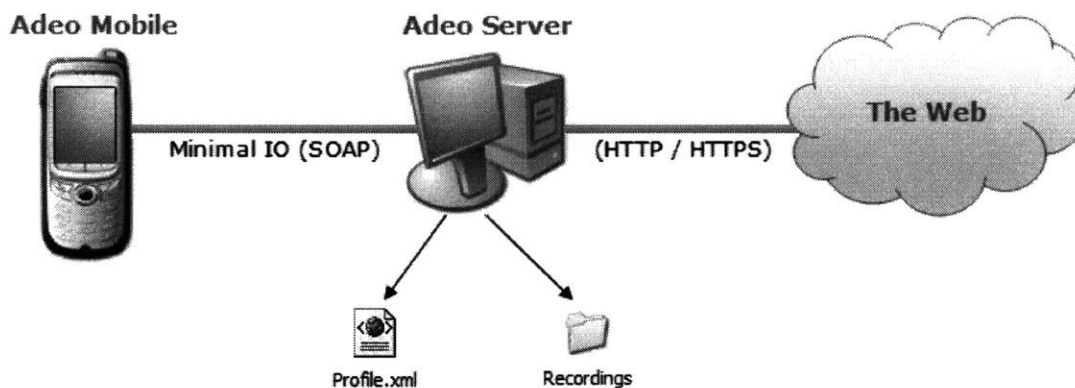


Figure 2-84 Adeo's thin client design

In fact, in the case of a user checking the balance of their bank account, it wouldn't even be possible to determine which bank they use by analyzing the packets sent between their Smartphone and the server.

Future Work to Improve Security in Adeo

Even though a minimal amount of information is sent between Adeo and the server, it is still worth encrypting this information, since the information may still be sensitive. For instance, the balance of a user's bank account could be embarrassingly low [47].

Also, in the case that the user's Smartphone is stolen, anyone with the phone will have the ability to play the user's recordings. Future versions of Adeo should require users to authenticate themselves when opening the application on their Smartphone.

2.5.5 Conclusion

It is impossible to automate procedures on the Web without storing the user's personal information. However, Creo and Adeo have been designed to be careful when both storing and transmitting this information. Because Creo is able to understand the semantics of a user's personal information, users are able to share their recording's functionality, without sharing any of their data.

2.6 Conclusion

The remaining slice of pizza had gone from cold to mummified, The Killer's play list had ended hours ago, and Pete, Ian and Sara were finally done studying.

Closing his textbook, Pete looked up. "There is a big surprise birthday party for my older sister Lucy this Saturday, free food... you guys want to go?" "Sure," Sarah said smiling. Ian nodded while yawning. Using his Smartphone, Pete instructed his agent to send an Evite to each of them. Still yawning, Ian mumbled that he was off to bed. "I'll walk you home," Pete said to Sara.

About two thirds of the way to her apartment, Pete had finally collected the courage to ask. "After the party on Saturday, do you want to get dinner?" "That sounds great," Sarah replied. "Cool, I'll see if I can get us a reservation somewhere," Pete said excitedly, launching Adeo on his Smartphone. He keyed for his reservation recording nervously; it had been months since he trained his agent to interact with the OpenTable Web site, and on a student's budget, he had never even had an opportunity to test the recording. Luckily everything worked, and a moment later they had their reservation.

On the walk back Pete sent his sister Lucy a text message: "know a good florist?" Her single character response ("y?") arrived a minute later, along with a recording to train his agent how to order flowers from a florist near campus. "thx," Pete wrote back smiling.

Chapter 3

Design and Implementation

3.1 Introduction

The design of Creo and Miro is discussed in Section 3.2: Designing a User Interface Agent for the Web. The implementation of Creo and Miro is discussed in Section 3.3: Implementing a User Interface Agent for the Web. Adeo's design and implementation is discussed in Section 3.4: Invoking Procedures on the Web from a Mobile Device. The code for Creo, Miro and Adeo totals over 20,000 lines, so their implementation sections in this chapter cover a high level view of their architecture, and specifically explain only the most fundamental features. Limitations of the current implementations are discussed in Section 3.5.

3.2 Designing a User Interface Agent for the Web

3.2.1 Software Agents vs. Direct Manipulation Interfaces

The majority of current applications have a direct manipulation user interface, where changes to the application's state have a one-to-one relationship with commands explicitly invoked by the user. There are many benefits to a direct manipulation interface, like providing the user with a sense of consistency and control. However, one significant disadvantage of

Direction Manipulation Interfaces is that as their functionality increases, they do not scale particularly well. For instance, Word 2003 currently has 1,500 commands [48]. Here is what Word 2003 looks like with all of its toolbars turned on:

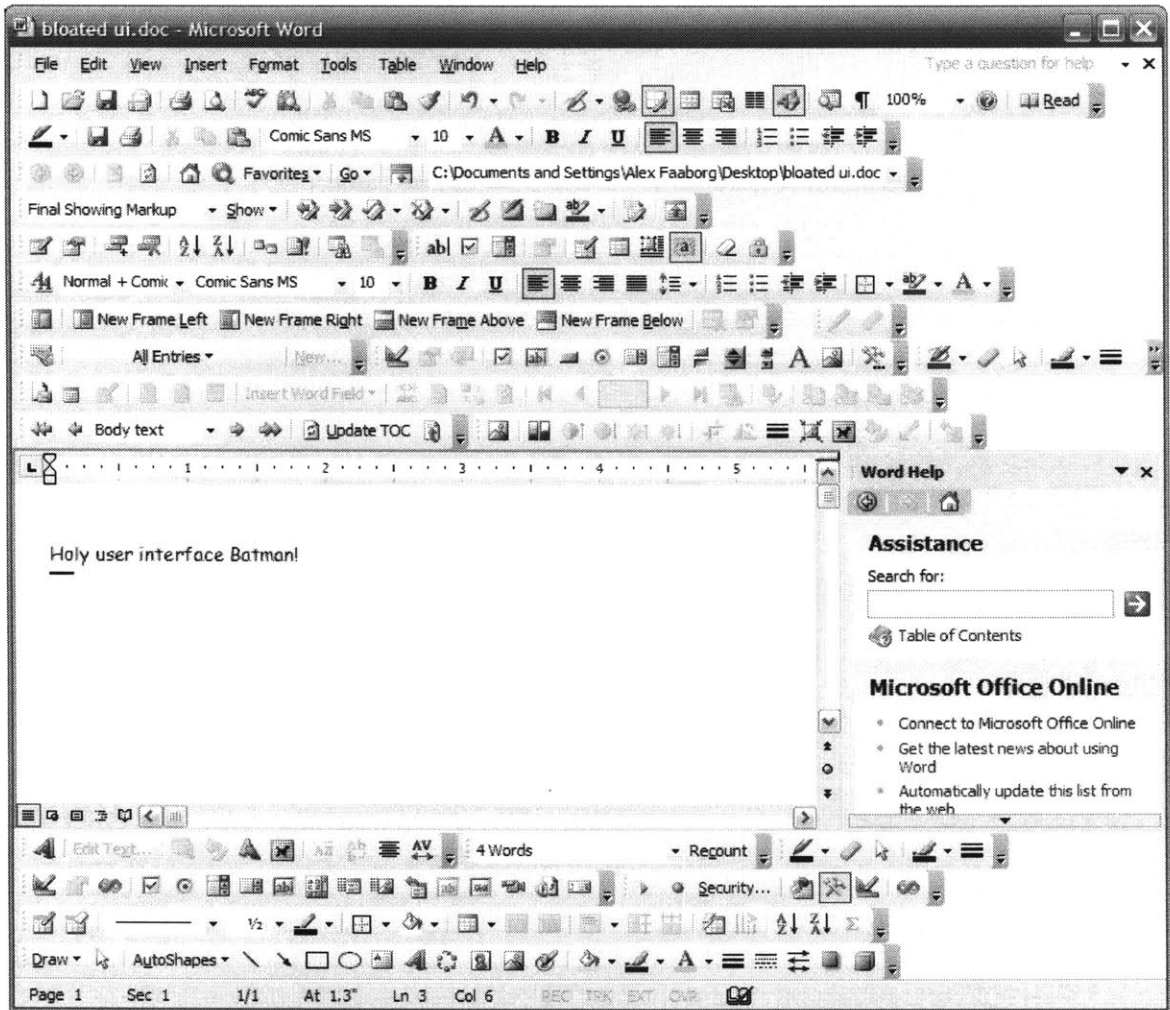


Figure 3-1 Direct manipulation interfaces do not scale well as functionality increases

Software Agents differ from Direct Manipulation Interfaces in that the agent can invoke commands on the user's behalf, and the agent can proactively anticipate and present commands that the user is likely to invoke, based on their current context. Building third party software agents on top of rich client software applications poses a number of implementation challenges

related to monitoring the user's actions and controlling the application through its API.

A New Era of Direct Manipulation Interfaces

Based on windows, menus and icons, Direct Manipulation Interfaces have been the predominant form of user interface since the 1980s, and thrived during the desktop publishing era of computing. Today, computing has transitioned from an era of rich client desktop applications to an era of applications built for the Web. This is largely because applications designed for the Web are easier to develop (compared to the Win32 API), are system independent, can be updated and improved without the user having to take any action, and do not require any form of installation. From a user interface perspective, this shift means that the traditional Direct Manipulation elements of windows, menus and icons have been replaced with the hyperlink, and a small number of form elements. While the interfaces themselves are usually less complex, complexity comes from the vast number of sites on the Web that an average user regularly interacts with. The Web's Direct Manipulation Interface often results in users being forced to manually complete repetitive actions, like copying and pasting a list of information between sites, or filling out the same lengthy sequence of forms every time the user wishes to complete an action.

Software Agents for the Web

Creo and Miro are both User Interface Agents [49], software that actively assists a user in operating an interactive interface. Miro can also be considered an Autonomous Agent, software that takes action without user intervention and operates concurrently with the user.

Letizia, by Henry Lieberman [50], is also an example of a Software Agent for the Web, autonomously locating information the user may find interesting. However, unlike Creo and Miro, Letizia treats the Web as a static series of documents, as opposed to a mixture of static documents and interactive

applications. In fact, if Letizia began performing reconnaissance for the user and explored every available hyperlink while the user was interacting with a Web-based application, it could potentially wreak quite a bit of havoc, as recently demonstrated by Google [51].

In the previous era of computing, desktop publishing, Software Agents failed to present a compelling alternative to Direct Manipulation Interfaces. This is mostly due to the fact that the majority of users found anthropomorphic interfaces like the Microsoft Office Assistant interruptive, condescending, and socially inept.

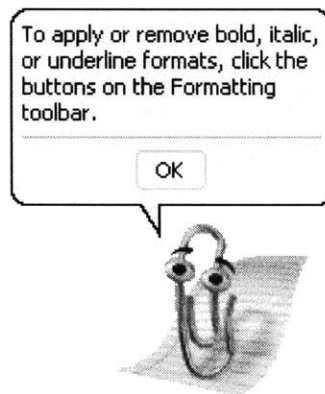


Figure 3-2 Is this better than a Direct Manipulation Interface?

In an effort to provide a better user experience, and to demonstrate the value of Software Agents in comparison to Direct Manipulation Interfaces, Creo and Miro follow a set of design principles for Software Agents established by Henry Lieberman [49, 52, 53].

3.2.2 Design Principles for Software Agents

Initiative

Most conventional interfaces simply sit still unless the user is actively commanding them. Interface agents, on the other hand, can be proactive, actively working while the user is thinking, or performing other actions. This saves the user time, since the system is making use of time that would otherwise be wasted [53].

Miro takes the initiative to parse Web pages as the user browses. However, unlike proactive agents that expose that they are acting independently or choose to interrupt the user, the initiative that Miro takes is not visible in the user interface.

Suggest Rather Than Act

Many people are afraid of granting autonomy to interface agents because of the fear that the agent will make a bad decision without their consent. This fear is not without justification [49].

After the user clicks *Scan Page*, or *Search*, the results that Miro provides are only suggestions. Miro will never play a recording on its own. Instead, users must decide to directly invoke a recording. Additionally, users can request that certain recordings display a confirmation window before they are played.

The types of suggestions an agent provides can be “knowledge-based” or “behavior-based” [49]; Miro uses both sources of information. Miro’s suggestions are based on both the semantic knowledge in ConceptNet and TAP, and the behavior-based recordings created by the user.

Take Advantage of Information the User Gives the Agent "For Free"

The actions taken by the user in the user interface constitute information that the system can use to infer the goals and interests of the user "for free" [49].

Creo takes advantage of monitoring the user’s actions when creating recordings, and Miro takes advantage of monitoring the user’s actions when watching if the user clicks on a piece of potential knowledge that Miro learned by reading the current Web page.

Take Advantage of the User's Think Time

Running the agent autonomously while the user is thinking takes advantage of compute time that would otherwise be wasted. This is especially important in search or exploration tasks when the idle time

can be used to provide independent exploration of the search space [49].

Miro follows this design guideline by loading and parsing Web pages every time the user browses to a new page. However, this initiative is entirely transparent to the user. On an interface level, there is no way to know that Miro is processing information in the background while they browse.

Autonomous Interface Agents May Have a Different Tradeoff between Deliberation and Action

Running as an autonomous interface agent gives [it] the opportunity to make a "quick and dirty" guess first, then spend more time, if it is given the opportunity, to make a more refined judgment [49].

While Miro is an Autonomous Interface Agent, it does not currently produce a varying quality of suggestions based on available time. However, if it was accessing knowledge from several online knowledge bases, or was programmed to perform considerably more advanced parsing of a Web page's content, it could take advantage of this design principle.

The User's Attention May be Time-Shared

The user may find output happening on the screen at a time that they may not expect it [49].

Miro was not designed to follow this principle. The reason for this involves the following principle:

An Autonomous Interface May not Fit the Cognitive Styles of All Users

Some users may be uncomfortable with nonlinear interfaces where more than one thing is going on at once [49].

From the user's perspective, Miro is still a linear interface. This is because they must click *Scan Page* before Miro displays any suggestions of recordings that match the Web page's content.



Figure 3-3 The Scan Page button is a debatable design decision

3.5 Limitations of the Current Implementation

This section discusses several specific technical limitations of the current implementations of Creo, Miro and Adeo.

3.5.1 Limitations of Creo

Detecting the User's Actions in Flash and Java Applets

One of the biggest limitations of Creo is that it can only automate interactions with HTML. This limits the number of recordings that a user can make. For instance, the user cannot link music artists to MTV.com because MTV has a ridiculously fancy search field, implemented using Flash:

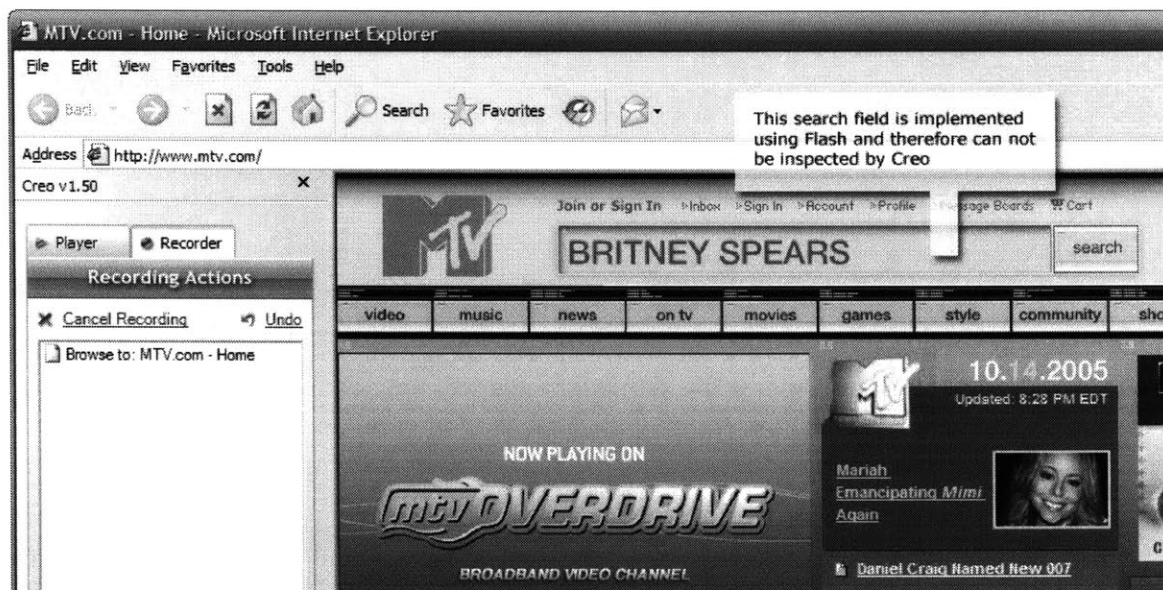


Figure 3-14 Creo cannot parse bling-bling search fields implemented in Flash

The same is true for Java Applets and other non-HTML elements of Web pages. This limitation is similar to the challenges facing the development of third-party Software Agents for client-side applications. To support recording, the agent must be able to capture the user's actions.

Generalizing Navigation Events

Creo is able to automatically generalize the input to Web forms, but this represents only part of the challenge in creating a general-purpose procedure. To create truly general-purpose procedures, Creo must also be able to generalize navigation events. For instance, the "Order Food" recording shown throughout this thesis ends on the results page at FreshDirect.com, it doesn't actually order the food (kind of a deceiving name). This is because Creo has no way of automating the action of clicking on the first hyperlink in the list of results. It is also probably not the case that the user would want Creo to make this decision. At this point they would likely want to take over.

The ability to generalize navigation events was never added to Creo because (1) it does not directly involve using ConceptNet and TAP, so it is outside the scope of this (already rather broad) thesis, and (2) many of the Programming by Example systems for the Web discussed in Chapter 4 have already addressed this issue.

Detecting the Difference between Form Submissions and Navigation Events

Another technical limitation of Creo that impacts its overall usability is that Creo is currently unable to tell the difference between navigation events and form submissions. It is likely that this issue can be solved by better understanding the information passed to the action listeners associated with IE's navigation events. However until this problem is solved, users must click the *Record Form and Submit* hyperlink, which many users find very confusing.

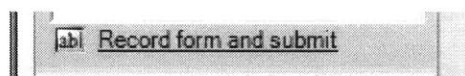


Figure 3-15 This hyperlink shouldn't exist

3.5.2 Limitations of Miro

Internet Explorer's Document Object Model is Read Only

A technical limitation currently affecting Miro involves errors caused by modifying, and then reloading a Web page. While Miro can access the Document Object Model of the current Web page through Internet Explorer, all of the objects in the DOM are read only. This means that Miro cannot turn plain text into hyperlinks by directly manipulating the page. Instead, Miro loads the page into a new DOM, modifies that, and then passes the new DOM back to Internet Explorer:

```
IHTMLDocument2 doc = (IHTMLDocument2)Explorer.Document;  
doc.write(new object[] {html});  
Explorer.Refresh();
```

A number of errors are introduced when Miro calls `Explorer.Refresh()` on a Web page without getting a new copy of the page from the server, including the (rather annoying) duplication of banner advertisements:

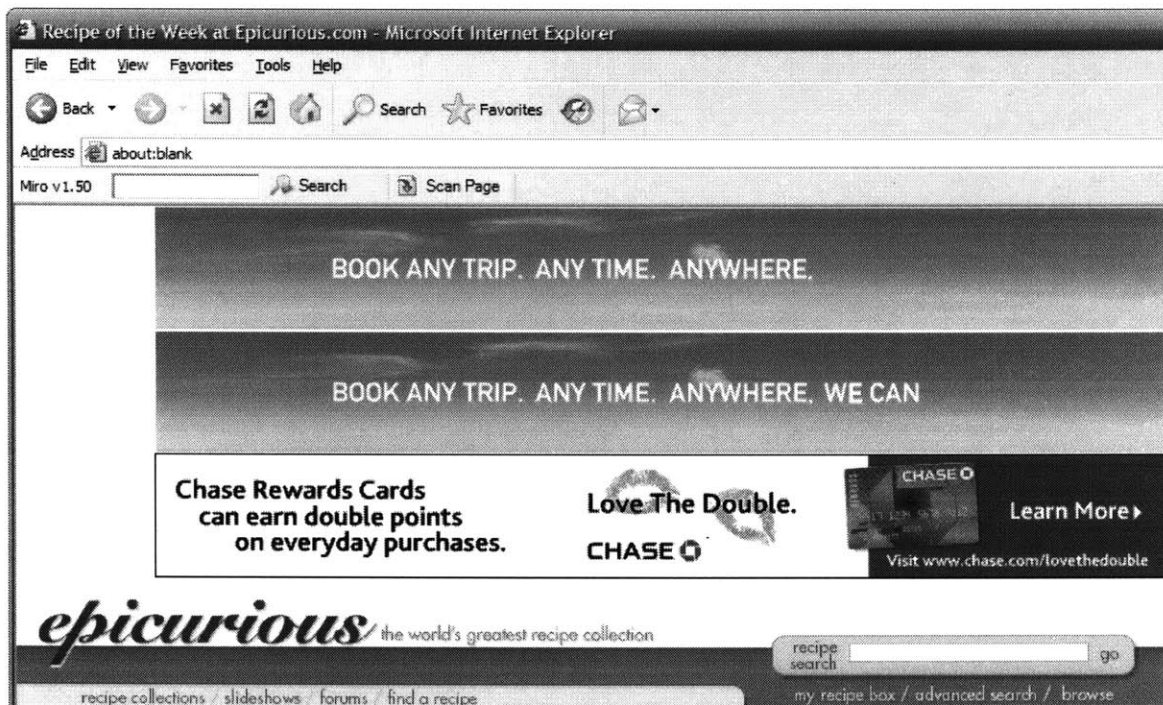


Figure 3-16 Miro's inability to directly modify the page breaks many Web sites

These problems can be resolved through a better understanding of how to directly modify `Explorer.Document`, without reloading it.

Detecting Recordings that Take Multiple Variables

An additional limitation of Miro is its inability to aggregate data on a page into a single recording. While the recordings detected by Miro can take multiple pieces of input, the user must provide additional input through pop-up windows. For instance, a recording that calculates the distance between two addresses currently requires the user to click on one of the addresses, and then enter the second. This limitation will be addressed in future versions of Miro.

3.5.3 Limitations of Adeo

Most Users Do Not Have Externally Contactable Computers

Adeo currently accesses and plays recordings by directly contacting the user's desktop computer. For this to actually work, a number of unlikely assumptions must be made, including (1) the user has a desktop computer, (2) the user's desktop computer is turned on, and (3) the user's desktop computer is externally contactable. The third assumption is probably the biggest stretch. The prevalence of Network Address Translation devices and port blocking means that the Internet, at least as it exists today, is one-way. While Universal Plug and Play (UPnP) allows software applications to automatically forward a port on the user's router, most of the routers currently on the market have extremely inconsistent and bugging implementations of UPnP.

Due to increasing concerns over security, the one-way nature of the Internet is not likely to change very soon. Because of this, Adeo's system architecture is really more of a conceptual prototype than a proposed design.

For Adeo to be released on a wide scale, a central server would need to be created to manage the storage and invocation of every user's recordings. This would introduce a number of technical problems not addressed by this thesis.

Most Procedures Created with Creo Require Some Amount of User Intervention

The second major limitation of Adeo's current design is the fact that many of the procedures created with Creo require some amount of user intervention. For instance, the procedure of ordering food cannot be fully carried out with a single click. Because of this, future designs of Adeo will need to provide the user with an interface built on top of the Web that is considerably more interactive than the current implementation of Adeo, while still streamlining the number of decisions the user has to make.

For simple information retrieval tasks like checking the balance of a bank account, viewing a movie review, or completing actions that do not contain too many variables, like ordering a pizza, sending PayPal and Evites, the current implementation of Adeo works fine. However, procedures with a more complicated set of dependencies and options, like making the arrangements for a trip, or selecting a doctor, and then scheduling an appointment [1] are not possible with Adeo's current implementation.

Chapter 4

Background and Related Work

4.1 Introduction

This chapter discusses related research from three angles: (1) how the work in this thesis relates to recent commonsense reasoning research at the MIT Media Lab, (2) how previous research relates to the specific tasks accomplished by Creo, Miro and Adeo, and (3) how this research fits into the broader vision of the Semantic Web.

Creo, Miro and Adeo are able to leverage the knowledge stored in Open Mind to understand the semantics of text. Recently, many other interactive applications have also leveraged the Open Mind knowledge base. A brief review of these applications is included in Section 4.2.

Work related to Creo is discussed in Section 4.3. This section covers previous Programming by Example systems, and macro recorders for the Web. Work related to Miro is discussed in Section 4.4, including a historical look at Data Detectors, and recent research in the field of semantic search. Research to adapt content on the Web for mobile devices is discussed in Section 4.5, along with other work related to Adeo.

The chapter concludes with Section 4.6, relating this research to the broader field of the Semantic Web.

4.2 Beating Some Common Sense into Interactive Applications

Following the creation of the Open Mind knowledge base in 2000, the Software Agents Group at the MIT Media Lab [42], along with many other researchers [58] have applied this commonsense knowledge to a variety of software applications. In addition to the descriptions provided below, a good review of applications that leverage the Open Mind knowledge can be found in Henry Lieberman's AI Magazine article, *Beating Some Commonsense into Interactive Applications* [54].

4.2.1 Common Sense and the Web

Woodstein

Woodstein (named after the nickname for Watergate reporters Woodward and Bernstein) is an agent developed by Earl Wagner and Henry Lieberman that works with a user's Web browser to help the user diagnose and explain Web processes, such as purchases [59-62]. Woodstein watches the way a user interacts with a Web site (similar to Creo), and allows users to inspect specific pieces of data on Web pages (similar to Miro).

Woodstein bases its observations about the user on a specific library of predefined process models. While these process models do not technically originate from the Open Mind knowledge base, they can be considered a form of common sense. For instance, Woodstein can recognize that the user is involved in purchasing an item by noticing specific steps, like the user clicking an "add to shopping cart" hyperlink. Woodstein uses a plan recognizer to match the user's actions against processes that it knows about.

While Woodstein and Creo both monitor the user in a similar fashion, their overall objective is different. Woodstein is designed to match user's actions against a predefined set of process models. Creo is designed to create new process models. Additionally, Woodstein uses process models to provide users with different visualizations of actions they have already completed. Creo uses process models to automate the completion of actions in the future.

Woodstein also allows users to activate data in Web sites, similar to Miro's *Scan* feature. When the user clicks on Woodstein's logo in the lower right corner, Woodstein is put into inspection mode, and any text on the page that it recognizes is turned into a button. For instance, in the figure below, Woodstein has recognized the specific "Transaction ID" for an online purchase the user has previous completed. The user can click *Why was this created* to view the actions they took during the process of ordering the item.

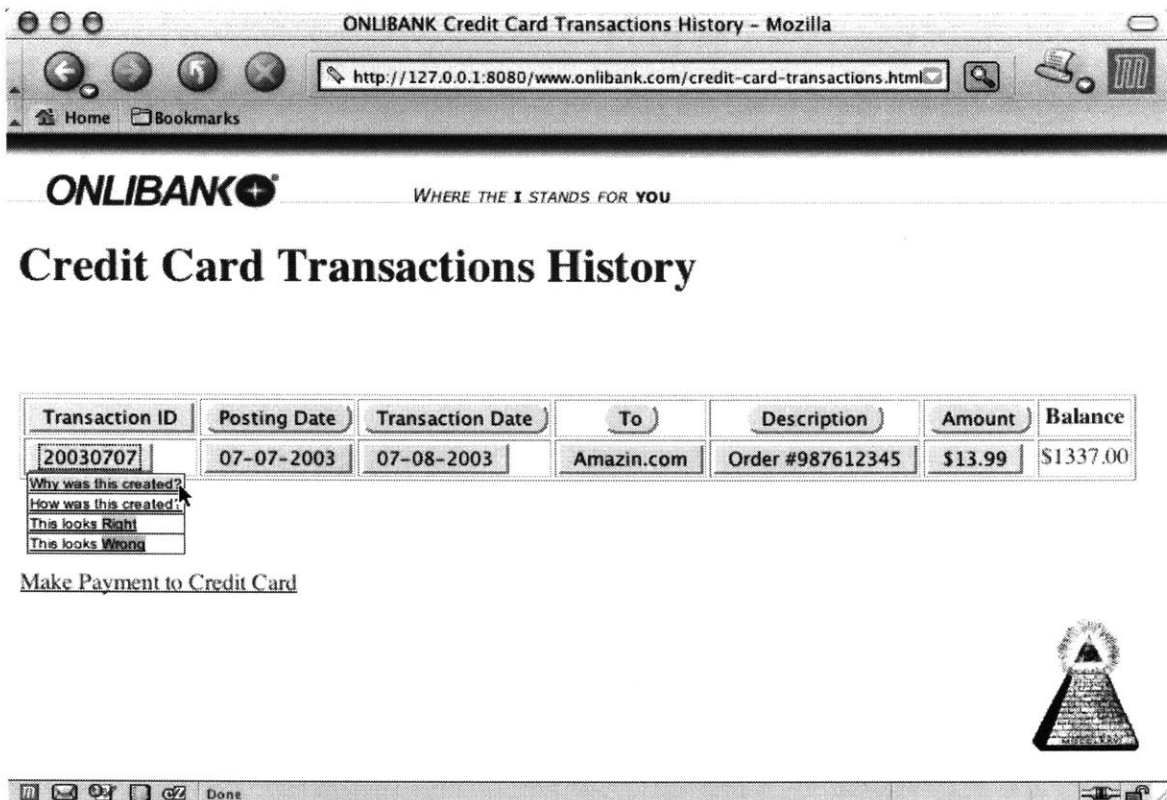


Figure 4-1 Woodstein, by Earl Wagner

Woodstein is based around the idea of debugging processes that have already taken place, while Creo and Miro are based on the idea of automating actions on the user's behalf in the future. Put in terms of common Web browser functions: Woodstein uses process models and inspection to provide the user with advanced *History* functionality. Creo and Miro use process models and inspection to provide the user with advanced *Favorites* functionality.

GOOSE

GOOSE (Goal-Oriented Search Engine with commonsense), was created by Hugo Liu, Henry Lieberman, and Ted Selker [63, 64]. GOOSE uses natural language processing and commonsense knowledge to translate a novice user's search into an effective query. For instance, GOOSE translates the novice search "help me get rid of the mice in my kitchen" to the more effective query, "pest control" in Cambridge, MA.

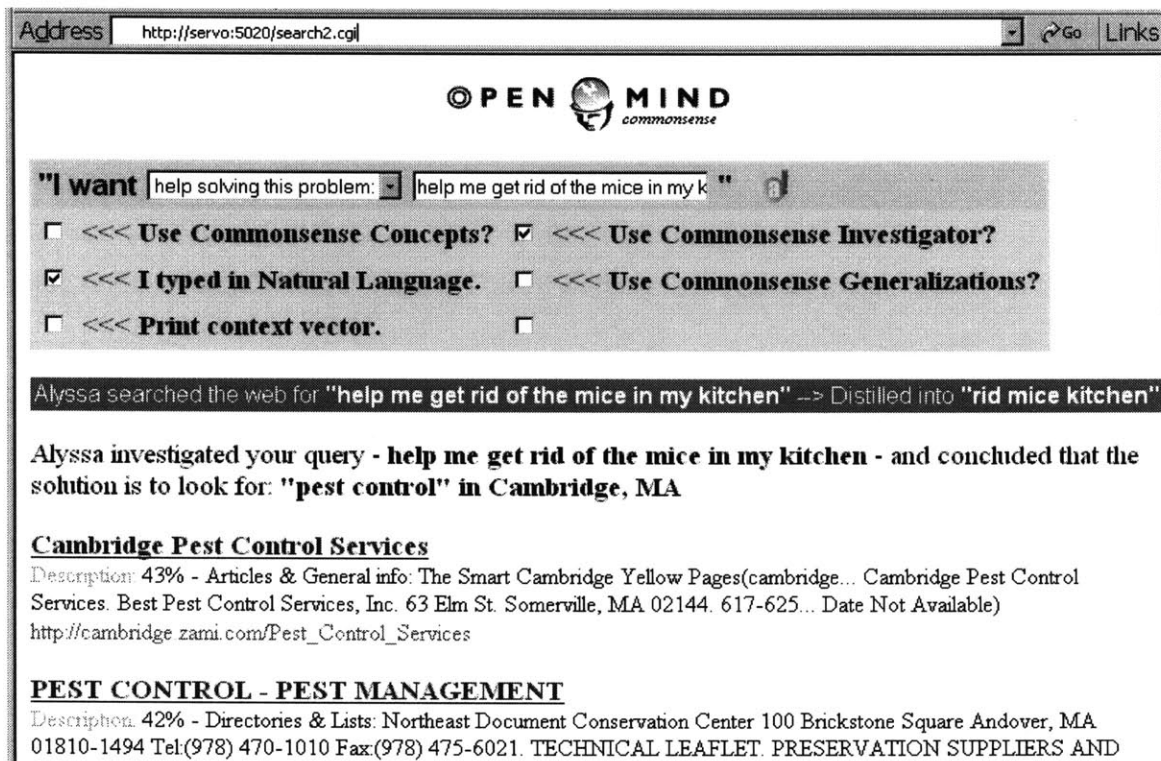


Figure 4-2 GOOSE, by Hugo Liu

In addition to typing in their search query, GOOSE requires the user to select a search goal using a drop down list box. The five available search goals are:

1. "I want help solving this problem:"
2. "I want to research..."
3. "I want to find websites about..."
4. "I want to find other people who..."
5. "I want specific information about the product/service..."

The reason GOOSE asks users to provide this information is so that it can parse the search query into the correct semantic frame. This differs from Miro's approach of automatically matching the query against the set of procedures stored in memory.

However, by asking the user to specify their search goal (and thereby its semantic frame), GOOSE is able to parse more complicated statements. Using the first semantic frame, GOOSE can parse the sentence "My cat is sick and wheezing" and determine that the problem object is "cat" and the problem attribute is "sick." By comparison, Miro does not perform any natural language processing on the user's search queries. This is because asking users to create a semantic frame would be too much of a burden, and this information is not easily inferred through examples. A search of Miro for "my cat is sick" will still lead to the veterinarian, but it very well might also lead to a pet store.

Miro attempts to automatically infer the user's search goal from their query using only the query's semantic context. For instance, possible search goals for "milk" might include buying it from the local grocery store, or looking up its nutritional information, because "milk" is a "food." Unlike GOOSE, users are not asked to specify their goal, and the actions Miro returns as results are personalized to the specific user.

4.2.2 Using Commonsense Knowledge to Understand the Context of Text

This section contains a brief summary of applications that like Creo, Miro and Adeo, leverage the Open Mind knowledge base to understand the semantic context of text.

Predictive Text Entry and Speech Recognition

The knowledge found in Open Mind and ConceptNet can also be used to improve the quality of predictive text entry, as demonstrated by Alexander Faaborg, Tom Stocky, and Henry Lieberman [65-67]. By understanding the semantic context of a text message, mobile devices can do a better job of anticipating which words the user is likely to type in the future.

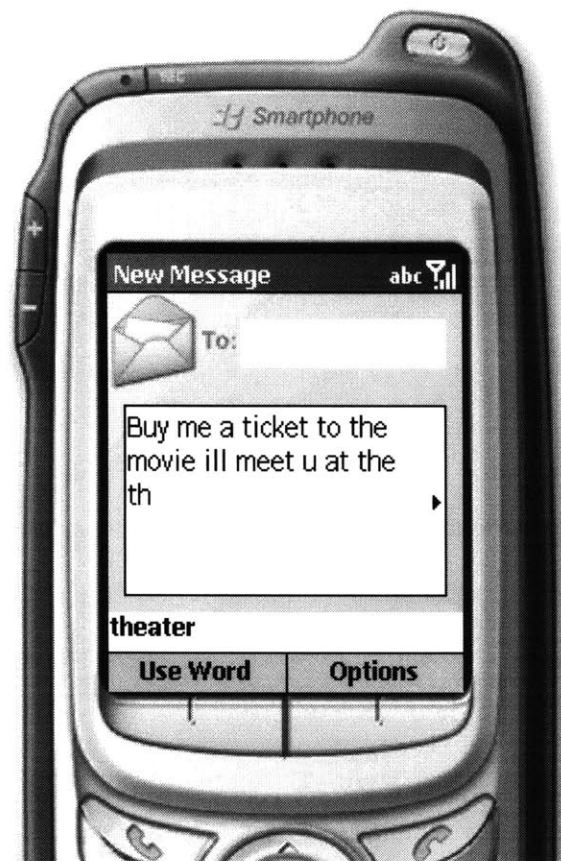


Figure 4-3 Predictive Text Entry, by Alexander Faaborg

The same algorithms used to improve predictive text entry can also be applied to speech recognition, demonstrated by Henry Lieberman, Alexander Faaborg, Waseem Daher, and José Espinosa [68, 69]. Filtering the list of acoustical hypotheses in a speech recognition engine against a semantic context generated using ConceptNet helps streamline error correction interfaces and improves the overall throughput of dictation systems.

ARIA

ARIA (Annotation and Retrieval Integration Agent) is a software agent created by Hugo Liu and Henry Lieberman that suggests relevant photos based on the semantic context of an email message [43-46].

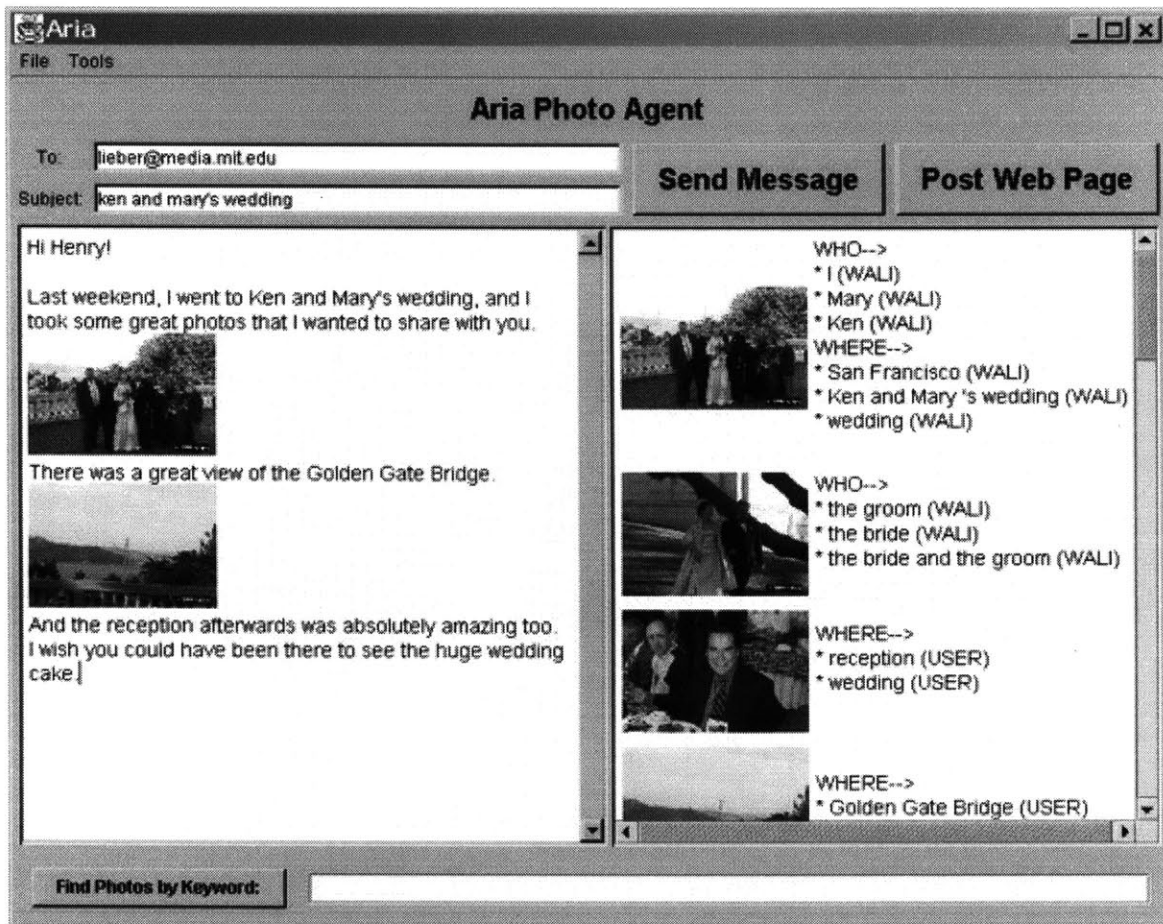


Figure 4-4 ARIA, by Hugo Liu

ARIA represents a form of implicit semantic search: as the user types, images are retrieved based on the semantic context of the message. For instance, if the user types the word “wedding,” an image annotated “bride” will still appear because the term is semantically related.

Detecting the Affective Quality of Text

Open Mind and ConceptNet have also been used to detect the emotional value of text [70-72]. While the sentence “My wife left me; she took the kids and the dog” does not contain any specific mood words, Hugo Liu, Henry Lieberman, and Ted Selker have shown that Open Mind can be used to determine that this is a negative sentence. Interfaces based on understanding the affective quality of text include an email agent that notes the emotion in an email message [70], and the ability to navigate a document based on its affective structure [72].

Detecting the Semantic Context of Conversations

Nathan Eagle, Push Singh and Alex Pentland have shown that ConceptNet and LifeNet can be used to detect the gist of conversations, even when spontaneous speech recognition rates fall below 35% [73, 74]. This enables mobile phones and PDAs to understand their user’s context by overhearing conversations.

4.2.3 Additional Applications of Commonsense Knowledge

Access to the Open Mind knowledge base and ConceptNet has also enabled a wide variety of other applications. These include: predicting user’s goals when they search their social network [66, 75], predicting a user’s goals when they make appointments in their calendar [76], creating a dynamic phrasebook for tourists [66, 77], detecting cultural differences in an email message [78], dynamically generating stories [79, 80], dynamically generating 3D environments [81], mapping a novice’s knowledge to an

expert system [82-85], and reducing the complexity of consumer electronic devices [3, 86].

4.3 Work Related to Creo

This section contains a discussion of previous Programming by Example systems, end-user programming systems for the Web, and research on extracting specific pieces of information from a Web page.

4.3.1 Programming by Example

Traditional interfaces leave the user with the cognitive burden of having to figure out what sequence of actions available to them will accomplish their goals. Even when they succeed in doing this for one example, the next time the same or a similar goal arises, they are obliged to manually repeat the sequence of interface operations. Since over time, goals tend to re-occur, the user is faced with having to tediously repeat procedures over and over. A solution to this dilemma is Programming by Example [87]. A learning system records a sequence of operations in the user interface, which can be associated with a user's high-level goal. It can then be replayed in a new situation when the goal arises again. However, no two situations are exactly alike. Unlike simple macro recordings, Programming by Example systems generalize the procedure. They replace constants in the recording by variables that usually accept a particular kind of data. This section describes several Programming by Example systems similar to Creo.

SMARTedit

SMARTedit (Simple MACro Recognition Tool), by Tessa Lau, Steven Wolfman Pedro Domingos and Daniel Weld, is a Programming by Example system that automates repetitive text-entry tasks [87-89]. While SMARTedit and Creo

are in very different application domains (text editing vs. Web browsing), there are a number of similarities in how they function as Programming by Example systems. SMARTedit, like Creo, is built on the idea of the user acting as a teacher, and the application as a proactive yet considerate student. The user shows the application demonstrations. The application asks questions, and proposes examples and solutions. Unlike simple macro recorders, both SMARTedit and Creo do not simply record the user's actions; they generalize the examples to create procedures that work in different contexts.

SMARTedit uses five of eight different interaction mode classes to learn from the user, resulting in a mixed-initiative interface [89]. Creo and Miro currently support three of these interaction mode classes. A discussion of each interaction mode class appears below:

1. User Solution: *In our simplest class, the standard for machine learning, the learning system observes an example and solution that the user provides and uses the result as input to the learning algorithm [89].*

The *User Solution* interaction mode is represented by Creo's *Recorder* tab. As the user interacts with the Web site, Creo observes and records each action the user takes. Input provided by users into HTML forms is passively parsed and generalized by Creo.



Figure 4-5 The User Solution interaction mode in Creo

2. Collaborative Solution: *The system executes a learned hypothesis on the next example and presents a solution to the user. The user either confirms the solution - in which case the example becomes training data for the system - or she asks for an alternate solution. The system can then propose an alternate solution to the user for acceptance or rejection, and so the process continues [89].*

This interaction mode class is represented by Creo's interface for modifying the generalizations of input. Creo uses several different methods to determine if a piece of input should be generalized, and if it is, the user is given the option of controlling which generalizations are correct using a set of check boxes. In the figure below, Creo has automatically generalized the input of "diet coke" to six generalizations. Creo and the user can now work together collaboratively to determine which of these generalizations best describes the concept for this particular recording.

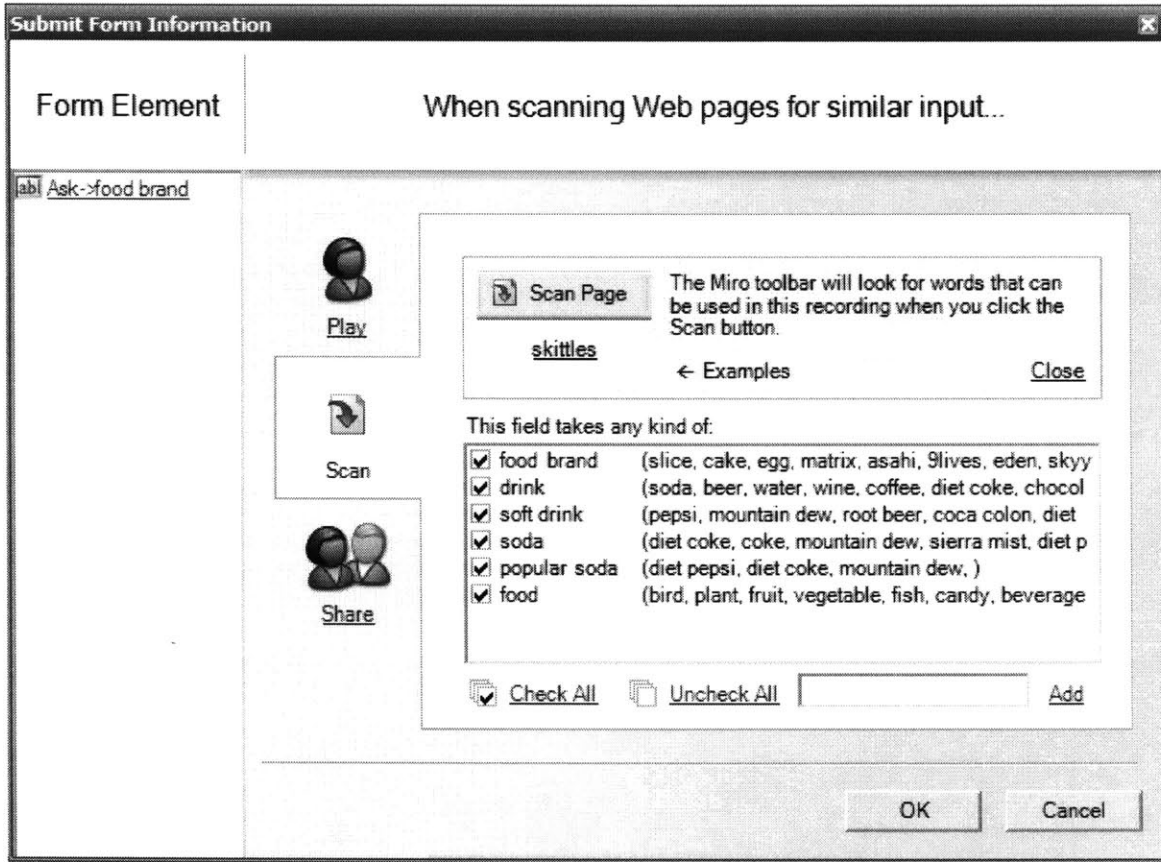


Figure 4-6 The Collaborative Solution interaction mode in Creo

Another example of the *Collaborative Solution* interaction mode is the training wizard in Miro. Miro will attempt to provide the user with intelligent defaults throughout each step of the process. For instance, if the user does a search on "Tylenol," this concept will not activate the "Order Food" procedure (which was trained on "diet coke") because "Tylenol" is not a "food brand." Together, Miro and the user are able to reach a *Collaborative Solution*. Miro provides the intelligent default that "Tylenol" is a "medicine brand," and the user instructs Miro that the "Order Food" recording should have been activated. For many errors corrected by the training wizard, the application and the user will each provide 50% of the solution.

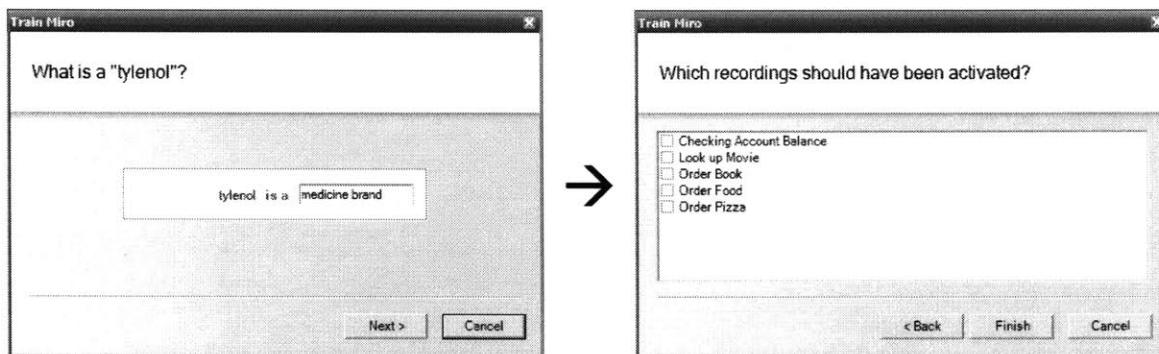


Figure 4-7 The Collaborative Solution interaction mode in Miro

3. System Solution: *In this class of interactions, the learning system assumes slightly more control, executing its learned concept on the next example (or a part of it), demonstrating the solution to the user, and giving her a chance to reject that solution [89].*

An example of the *System Solution* interaction mode is Miro's ability to learn new knowledge by performing natural language processing on a Web page, and then match this knowledge against the user's current set of recordings. For instance, Miro learns that "The Killers" is a "band" and then activates the "Buy Music" recording. While the user has no direct way of rejecting the solution (in fact, the user doesn't even know that Miro is trying to bootstrap its knowledge), failing to click on "The Killers" is treated by Miro as an implicit rejection. The way Miro tracks the user's actions once it presents a piece of potential new knowledge is also an example of the *Collaborative Solution* interaction mode, although the user is not cognizant of the collaboration.

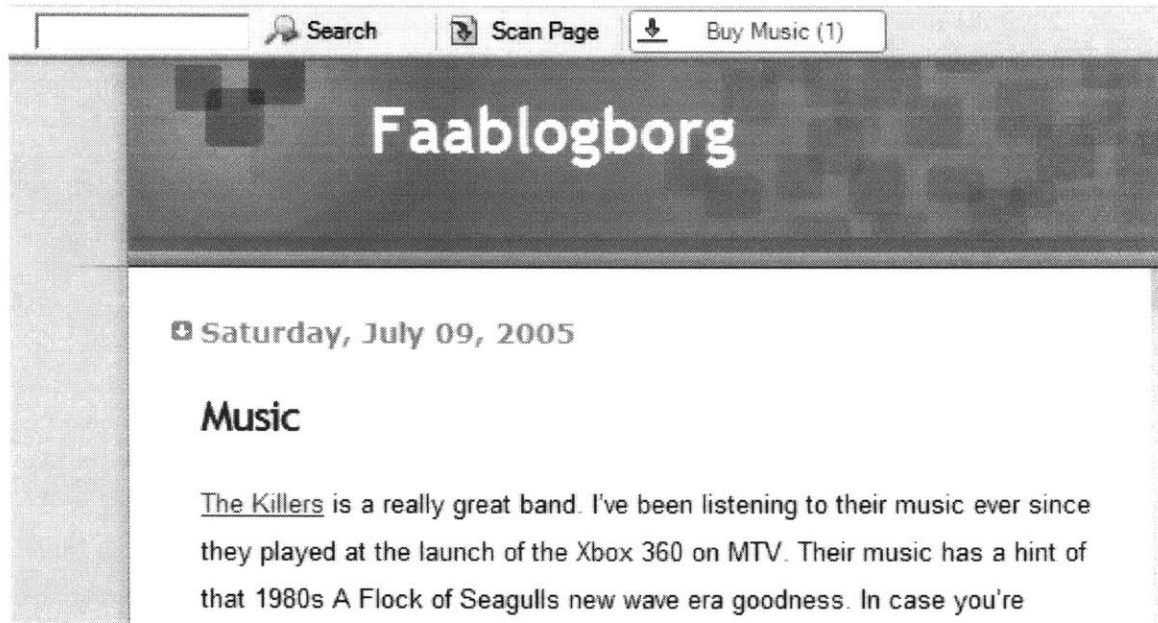


Figure 4-8 The System Solution interaction mode in Miro

The remaining interaction mode classes identified by the SMARTedit team are not currently supported by Creo and Miro.

4. Performance: *In a performance interaction mode, the learning system takes full control and executes its concept autonomously on new examples [89].*

Because many of the actions taken by Creo and Miro result in changes in state of the real world (like a pizza arriving at your door, and your bank account being depleted), Creo and Miro do not currently support the *Performance* interaction mode. It is theoretically possible to try to identify recordings that do not result in real-world effects, so that Creo and Miro could attempt to practice different generalizations on their own. However, doing so represents a significant lack of user control, and violates one of Jakob Nielsen's usability heuristics [90]. Since SMARTedit is in the application domain of text editing, it can safely implement the *Performance* interaction mode. Unfortunately for Creo, most local pizza places do not support the notion of *Undo*.

5. System Example Selection: *The system proposes to the user that she shift her attention to a particular example, rather than a user-selected example, a randomly-chosen example, or the next one in order. The system-driven example selection can often provide greater learning benefit because the system can collect data on the most interesting examples first [89].*

This interaction mode is not currently supported in Creo or Miro. However, it is an interesting idea. For instance, if the user was recording a procedure at FreshDirect.com and entered the example of “apple,” Creo could pose the direct question to the user: “Do you mean the fruit or the computer?” These targeted questions could be based on detecting situations where certain generalizations of a concept did not match the semantic context of the page (“the most interesting example”). Currently, the user must check or uncheck generalizations, while viewing them all at the same time.

6. User Example Selection: *The system requests that the user select or provide a promising new example to investigate. Ideally, this request is accompanied with a high-level description of the kind of example that would be most valuable to operate on next [89].*

Similar to the *Collaborative Solution* interaction mode, Creo could prompt users to provide additional examples if it determined it did not have enough information. SMARTedit does not implement this interaction mode.

7. System Example Generation: *In this class, the system generates a new example for the user to solve. In learning applications with relatively unstructured and unrestricted input, the system may be able to generate arbitrary examples. However, in more structured environments, modifying existing examples may be more effective than generating entirely novel examples [89].*

As discussed in Section 2.4.2, attempting to generate recordings without user assistance is incredibly difficult because from an agent's perspective, the Web is essentially one big captcha. From poorly named form elements to information imbedded in images, there are many cases where Creo relies on its user's cognitive and perceptual abilities to learn how to complete a task.

While this interaction mode is more feasible in the domain of text editing, it is not implemented by SMARTedit.

8. User Example Generation: *Here, the system asks the user to create a new example which can then be solved. This type of interaction is valuable if the system can specify to the user some quality of the new example which would help to discriminate among contending hypotheses [89].*

Because of the breadth of possible recordings users can create with Creo, this interaction mode is not currently supported. However, this interaction mode could provide an interesting way of introducing new users to the system. Creo requires that users actively turn its recording mode on and off, instead of passively monitoring all of the user's actions on the Web. This is due to privacy and control concerns. However, this also means that Creo is only useful if users take the initiative to train it. If Creo detected that users were doing an action repetitively, it could proactively ask, "Would you like to train me to complete this action for you?" This type of *User Example Selection* interaction would help users take full advantage of Creo's abilities. However, this type of unexpected and mildly invasive question is reminiscent of the infamous Office Assistant's statement: "It looks like you're writing a letter." This interaction mode is also not implemented by SMARTedit.

SMARTedit scores each of these interaction modes based on a decision-theoretic framework (a utility function). This allows the learner to balance an interaction's expected burden to the user against its estimated value to the task and learner. This is more advanced than Creo, as Creo simply has

different user interfaces that implement the various types of interaction modes. Since users have varying opinions on how active a learning system should be, SMARTedit also adapts its utility function for each individual user. This means that different users will spend varying amounts of time with each of the interaction modes. While users do have the option of turning Miro's *System Solution* (learning from the Web) interaction mode off, this is also an example of how SMARTedit is a more advanced Programming by Example system.

Both SMARTedit and Creo use a Version Space Algebra [88] to efficiently represent the set of functions that are consistent with the user's demonstrated examples. In SMARTedit's case, these consist of functions to modify text. In Creo's case, these consist of generalizations of a function's input. Both SMARTedit and Creo can learn general procedures based on a very small number of examples. Because SMARTedit and Creo keep a record of all generalizations of the user's actions, they can fail gracefully, falling back to their next best guess, rather than failing completely.

While SMARTedit is not in the same application domain as Creo, they have a number of similarities in how they interact with the user and the variety of interaction modes they employ. Both applications learn with a collaborative and mixed-initiative user interface.

The following Programming by Example system, Grammex, is in the domain of data detection, making it similar to Creo and Miro in utility.

Grammex

Grammex (Grammars by Example), created by Henry Lieberman, Bonnie Nardi and David Wright, is a user interface for allowing novice users to define grammars interactively [87, 91, 92].

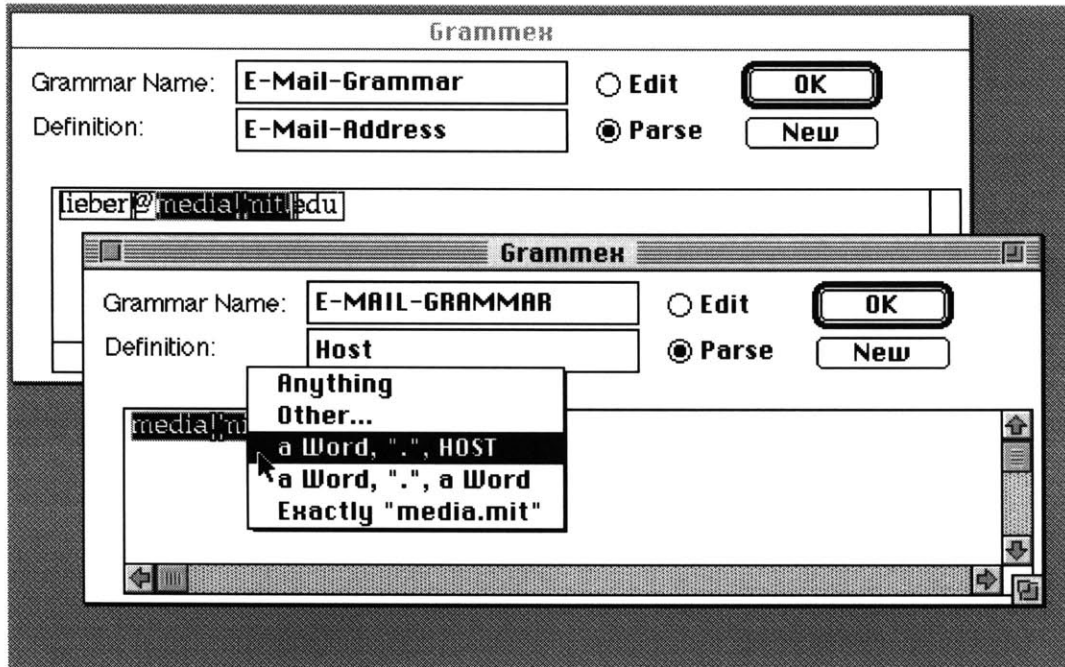


Figure 4-9 Grammex, by Henry Lieberman

Similar to Creo and Miro, Grammex allows users to train their computer to recognize patterns of data, and then take specific actions. For example, users can train Grammex to recognize the format of an email address with the example "lieber@media.mit.edu". Users can then enter into a recording mode and define an action for this piece of information by providing an example (like sending an email). However, the actions that users can define are limited by the number of Macintosh applications that are "recordable" (reporting user actions to the agent).

Grammex allows novice users to create context-free grammars to recognize patterns of text, like email addresses, URLs, and date formats. However, Grammex only understands the format of the data, and not the data itself. While Creo parses "Apple Computer" as a "corporation", Grammex parses "Apple Computer" as a "Word Word."

Grammex is a Programming by Example system for data detection, making it similar to both Creo and Miro. The following four Programming by Example

systems described in this section are similar to Creo in that they allow users to train their computer to interact with the Web by providing a demonstration of the procedure.

TrIAs

The TrIAs (Trainable Information Assistants) framework by Mathias Bauer, Dietmar Dengler, Gabriele Paul and Markus Meyer, allows users to automate information gathering on the Web [87, 93-96]. TrIAs allows users to create scripts for extracting information from Web sites using Programming by Example. This training stage is similar to the way users interact with Creo. TrIAs's information gathering framework also relies on user interaction in order to recover from failed information gathering actions. An example of using TrIAs is scheduling a trip: TrIAs's information broker can aggregate information from airline, hotel, weather and map sites on the Web. Another example would be using TrIAs to extract a music artist's name from an online concert listing. The Programming by Example and information extraction features of Creo are very similar in nature to TrIAs's features. One difference between the systems is that Creo automatically encodes semantic generalizations along with the steps of the procedure, while HyQL (TrIAs's SQL-like query language) focuses primarily on the selection of information in HTML documents. Additionally, while TrIAs's use of Programming by Example allows users to define HyQL wrappers without any programming knowledge, the user interface contains many instances of implementation-level terminology.

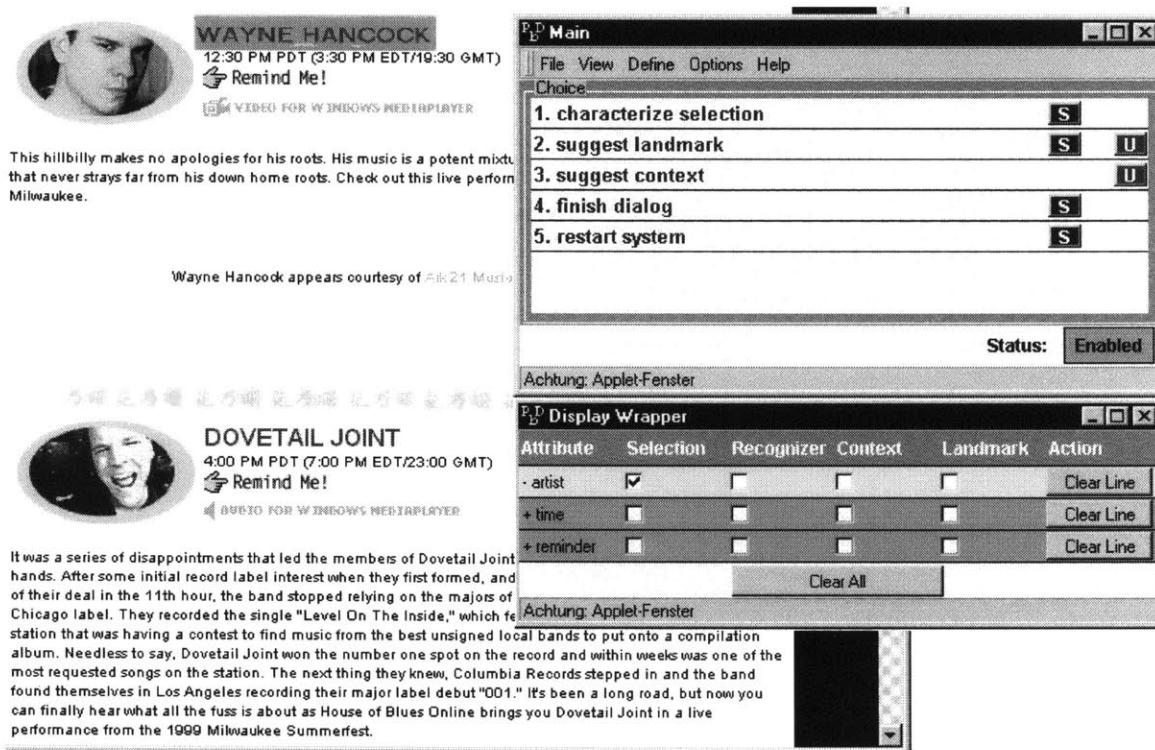


Figure 4-10 TrIAs, by Mathias Bauer

WebVCR

WebVCR, by Vinod Anupam, Juliana Freire, Bharat Kumar and Daniel Lieuwen, allows users to record their navigations on the Web, and create shortcuts to dynamically generated pages, referred to as “Smart Bookmarks” [97]. Using WebVCR, “a user can record smart bookmarks for any task he might need to perform multiple times, such as accessing local weather information, searching used car classifieds, checking for best airfares to a particular destination, or filling up an online shopping basket” [97]. The creators of WebVCR also note that the recordings can be used to improve Web browsing on mobile devices, similar to Adeo.

Turquoise

Turquoise, by Rob Miller and Brad Myers, is a Programming by Example system that allows non-technical users to create dynamic Web pages by demonstration [98]. Turquoise can be used to create composite pages from multiple information sources, reformat the layout of HTML pages, fill out

forms, and automate common tasks. For instance, users can create a custom newspaper by copying and pasting information, or automate the process of aggregating multiple lunch orders into the same order. Unlike many Programming by Example interfaces, Turquoise is actually designed for non-technical users: the interface refers to HTML elements using human readable names, like "link" and "paragraph." Turquoise differs from Creo in that it is designed for authoring Web pages. However, the overall theme is the same: enabling novice users to program interactions with the Web by example.

Internet Scrapbook

Similar to Turquoise, the Internet Scrapbook, by Atsushi Sugiura and Yoshiyuki Koseki, is a Programming by Example system that allows users with little programming skills to automate their daily browsing tasks [87, 99]. With the Internet Scrapbook, users can copy information from multiple pages onto a single personal page. Once this page is created, the system will automatically update it as the source pages change. This allows users to browse commonly accessed information on a single Web page.

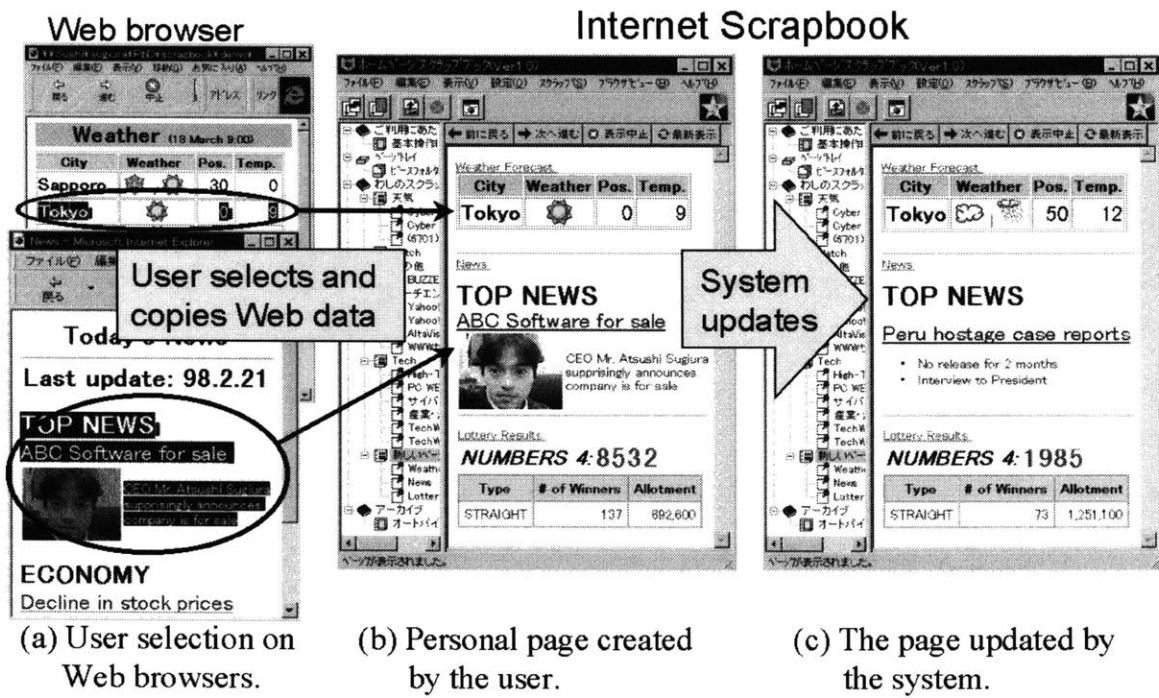


Figure 4-11 Internet Scrapbook, by Atsushi Sugiura

Web Macros

Web Macros, created by Alex Safonov, Joseph Konstan and John Carlis, allows users to interactively record and play scripts that produce pages that cannot be directly bookmarked [100-103]. Unlike Creo, which is integrated into the user's browser, Web Macros was created using a proxy server. Similar to Creo, Web Macros supports sharing recordings without disclosing personal information. However, personal information is not automatically detected, and must be flagged by the user.

LAPIS

LAPIS (Lightweight Architecture for Processing Information Structure), by Rob Miller is a Web browser and text editor that supports lightweight structure, a new approach to structure description [104]. Lightweight structure allows a system's basic structure concepts to be defined by a variety of mechanisms, including grammars, regular expressions, and any kind of pattern or parser. LAPIS then allows these structure concepts to be composed and reused, regardless of the mechanism that was used to define them. LAPIS shows how lightweight structure can be applied to a range of applications, including: pattern matching, multiple-selection editing, selection guessing, simultaneous editing, outlier finding, Unix-style text processing, and (similar to Creo), Web automation.

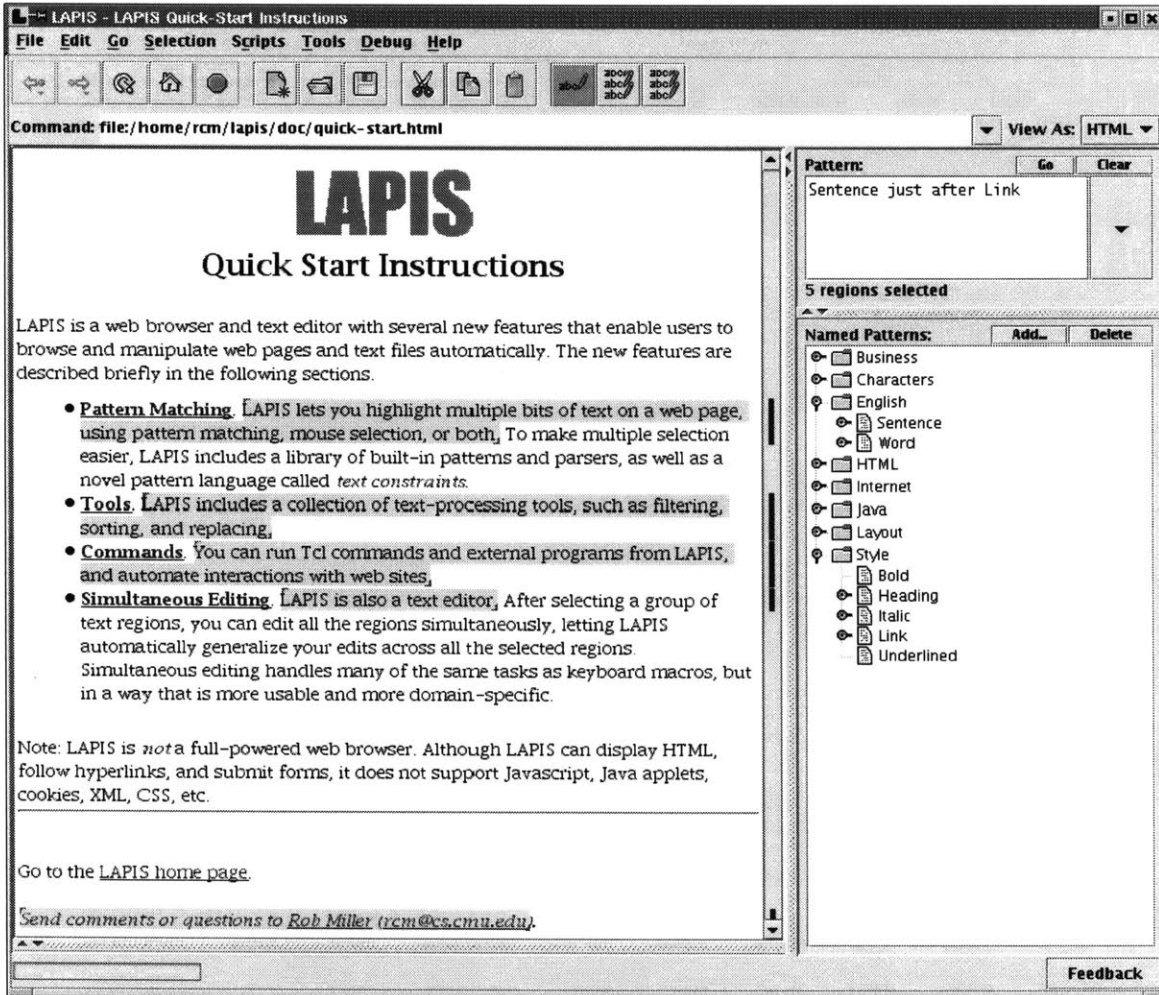


Figure 4-12 LAPIS, by Rob Miller

LAPIS can automate Web browsing by executing scripts that navigate through dynamic Web sites and online transactions. These scripts can be either entered by the user, or automatically generated by watching the user demonstrate the sequence of actions in LAPIS’s Web browser pane.

LAPIS displays the sequence of actions as events in the browsing history. Unlike the browsing history interface found in current Web browsers that only include navigation events, LAPIS integrates events like filling out a form into the browsing history.

The design decision to include a *Scan Page* button is debatable. Automatically displaying suggested recordings could annoy the user if they are not interested in carrying out the actions, even if Miro's suggestions are accurate and relevant. While the only real damage is that the user wasted a second performing a visual scan of the results, the Office Assistant's famous "It looks like you're writing a letter" interruption shows that these types of suggestions can still significantly degrade the overall user experience. However, this also means that users will never be pleasantly surprised with Miro's interruptions. The *Scan Page* button essentially makes Miro an Autonomous User Interface agent with a Direct Manipulation front end.

Personalization

Users of conventional applications expect to see exactly the same interface. But "one size fits all" is not appropriate for computer interfaces. Interface agents can learn the individual characteristics, idiosyncracies, unique needs and preferences of a user, and adapt, giving each user a personalized interface [53].

Creo and Miro are built around this design principle. Unlike Microsoft's Smart Tags and Google's AutoLink (and a large number of other Data Detectors described in Chapter 4), Creo and Miro allow users to be directly in control of their services, and the types of data associated with them.

User Modeling

The most sophisticated way to acquire user models are adaptive, which means they can create dynamic behavior without being pre-programmed. [53].

Creo follows this design principle by automatically generalizing information when creating recordings. From a single example, Creo can model other types of data that the user might want to use in the procedure, creating "dynamic behavior without being pre-programmed" [53].

Instructibility

When we interact with others, they learn from interactions and improve their interaction with us over time. The same should be true of interface agents. Learning is therefore a critical capability... Agents should record user actions and try to use them to improve future agent actions [53].

As discussed in Section 2.4, Creo learns procedures through Programming by Example, and Miro learns new pieces of knowledge through its training wizard, and by reading the Web.

Trust

For an agent to be useful, the user must have trust in the agent [53]. Creo and Miro generate trust by keeping the user informed of their actions through confirmations and feedback, which is the next design principle:

Feedback

Important to developing confidence in an agent is feedback. The agent should always be able to explain its actions, or have the results of its actions examined and critiqued by the user [53].

Users can request that Miro and Creo display a confirmation window indicating every action that will be taken before playing a recording. Users can also watch as these actions are carried out in their Web browser. Creo allows users to debug a recording, stepping through each action and seeing the result. Users can also critique and improve Miro's knowledge through the training wizard.

Anthropomorphism

The central issue regarding anthropomorphism is whether we are attracting a person's attention to the things they are actually trying to do or distracting them. If the character is entertaining or supportive, then anthropomorphism can be helpful. If on the other hand it takes a person's attention away from the things that they are focusing on, then it's not [53].

In the case of tutoring or entertainment, anthropomorphic agents are an effective way of interacting with the user. However User Interface Agents that integrate themselves with Web browsers (or office applications) should not be anthropomorphic. Miro's interface is designed to match the rest of the browser, consisting of text boxes and buttons.

Fail-Soft Design

Any agent based on a knowledge base of commonsense facts is likely to make mistakes. Because of this, most of the agents that leverage ConceptNet have followed the rule of *fail-soft design*. The idea behind *fail-soft design* is if the agent makes an incorrect suggestion or mistake, the user should be no worse off than if they didn't have the agent in the first place. [54].

Because Miro only provides suggestions and never directly invokes a recording on its own, its design is inherently fail-soft. When Miro makes a mistake or fails to correctly anticipate the user's goal, the user is free to carry out the task as they normally would, or instruct Miro how to perform better in the future. Unfortunately, the current implementation of Adeo is not fail-soft.

When playing a recording, if a Web site changes and Miro is no longer able to automate the procedure, or a piece of data causes unexpected results, Miro will stop taking actions and the user can begin to directly interact with the Web site from the point where Miro became confused. This design allows users to seamlessly transition to completing a task as they would with a conventional Web browser.

Take Advantage of Interfaces that are Under-Constrained

System implementers often fail to realize how under constrained many user interface situations are. In many cases, systems either do nothing, or perform actions that are essentially arbitrary. These applications show that there exists the potential to use common sense

knowledge to do something that at least might make sense as far as the user is concerned [54].

Web browsers are a good example of an under-constrained interface. Current Web browsers create a very thin interface between the user and the Web, and provide only simplistic functions like *Refresh*, *Stop*, *Back* and *Forward*, that have nothing to do with the user's higher level goals. The suggestions that Miro makes are not always correct, but they certainly help the user more than a Web browser that sits back and does nothing.

3.2.3 Designing the User Interface

The user interface of Creo and Miro was designed through an iterative process, described in detail in Chapter 5. This section discusses several notable attributes of the final design. Three design decisions that play a fundamental role in Creo and Miro's user interface are: (1) the record and play metaphor, (2) dynamically generated contextual help, and (3) intelligent defaults.

The Record and Play Metaphor

The record and play metaphor is used throughout the user interfaces in Creo and Miro. This metaphor manifests itself in both icons and terminology.



Figure 3-4 The record and play metaphor

Technically this metaphor is not really accurate, because unlike a simple macro recorder, Creo extrapolates a general-purpose procedure from a single demonstration. However, this metaphor was selected for two reasons: (1) it grounds the interface in a familiar mental model of *creation* and *use*, and (2) it establishes a consistent terminology that is used throughout all of the various user interfaces. Before the user interface was designed around this central metaphor, very early designs used a variety of terms like "functions,"

“procedures,” “services” and “methods.” However, because Creo is targeted at users who do not necessarily have a programming background, the term “recordings” prevailed.

Dynamically Generated Contextual Help

When the user is creating a recording, they can click on a form submission action in the *Recorder* tab’s action list to view the *Submit Form Information* window. In this window the user can make changes to how Creo treats particular pieces of information. For instance, was “diet coke” an example, or should the recording always submit this information?

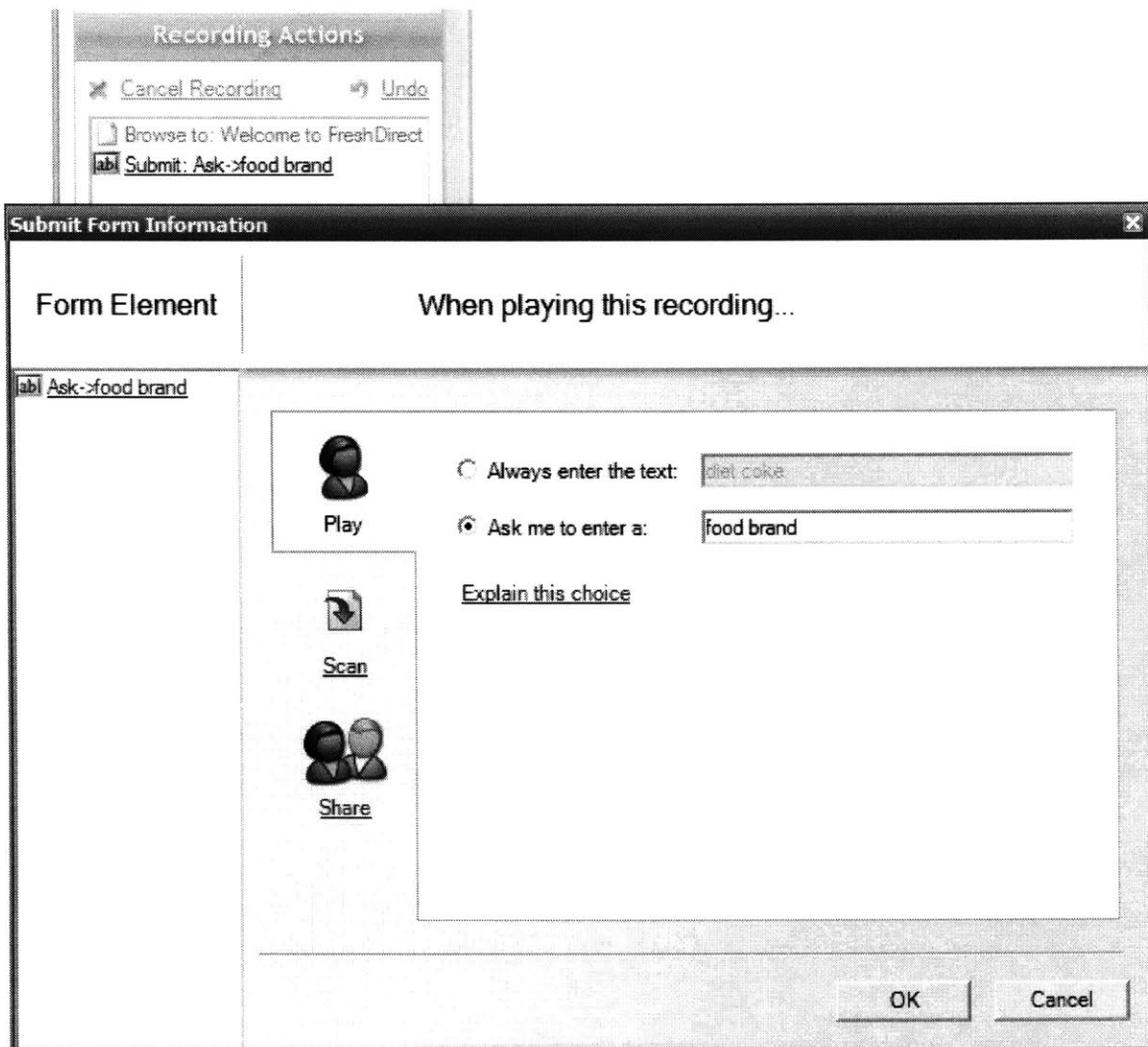


Figure 3-5 The Submit Form Information window

The two options on the *Play* tab allow the user to directly manipulate if Creo should generalize this information. The user can specify either *Always enter the text "diet coke,"* or *Ask me to enter a "food brand."* It is possible that the user may find this choice confusing, especially if they have just started using Creo. If confused, the user can click the *Explain this Choice* hyperlink to view more information.

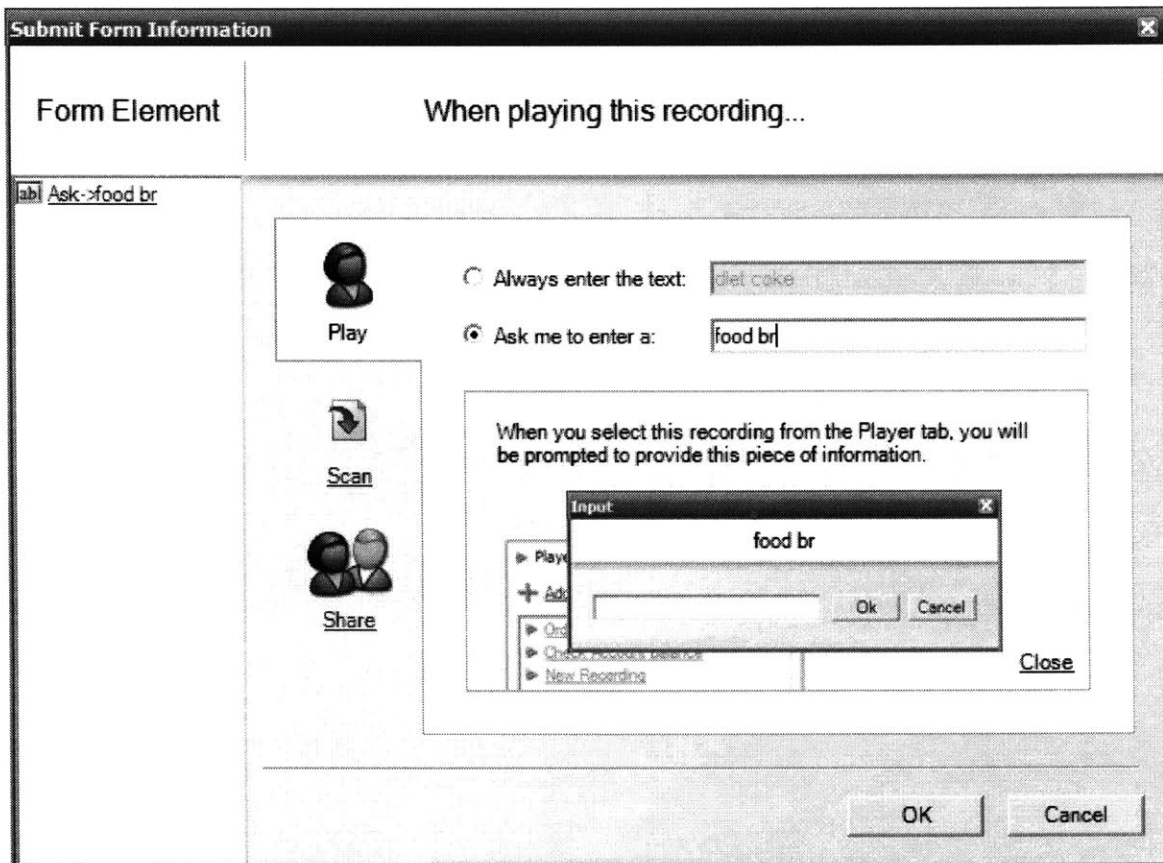


Figure 3-6 Example of dynamically generated contextual help

The contextual help that appears when the user clicks *Explain this Choice* is dynamically generated based on what the user selects and types. In the figure above, the user has selected *Ask me to enter a:*, and in the contextual help area, a pop-up window appears, representing what will happen when they play this recording. Additionally, the help reads *When you select this recording from the Player tab, you will be prompted to provide this piece of*

information. As the user types "food brand," the text appears in real-time as the label of the pop-up window in the diagram.

If the user selects *Always enter the text: "diet coke,"* both the image and the text in the contextual help area dynamically update to reflect the change:

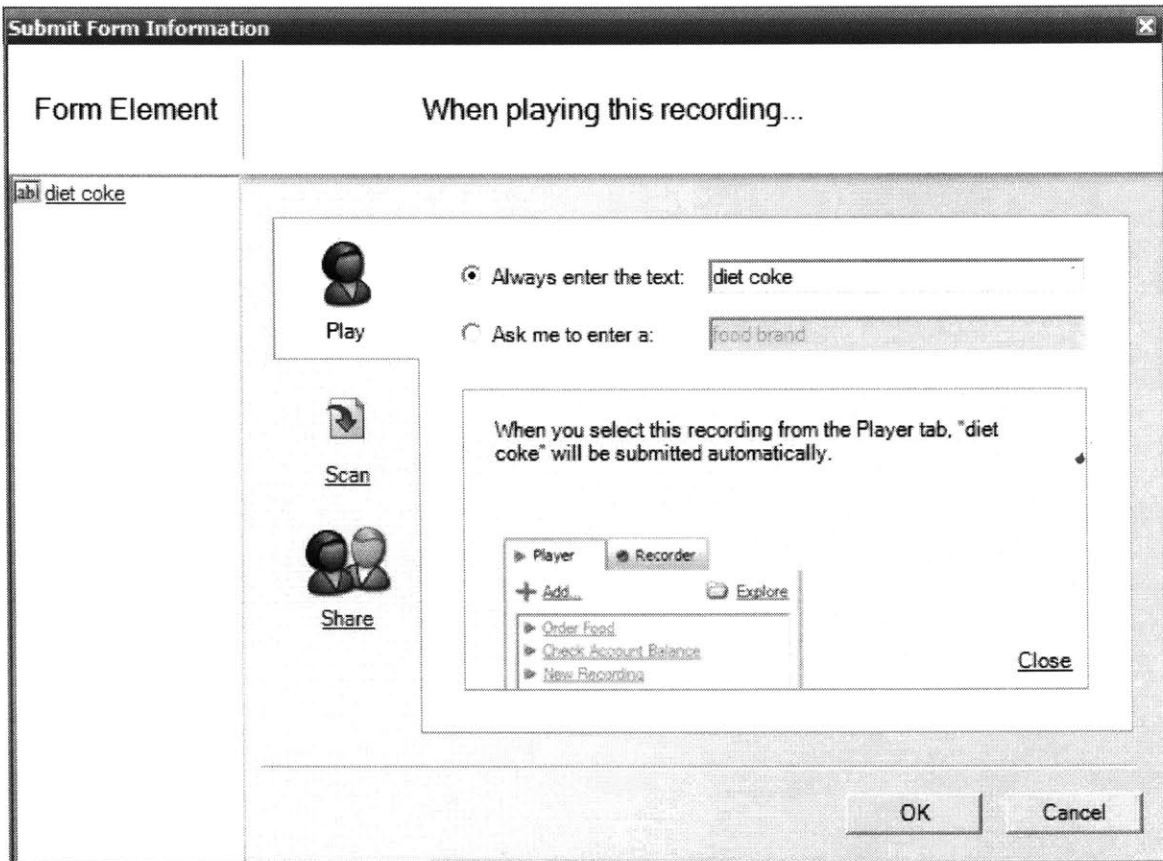


Figure 3-7 Example of dynamically generated contextual help

Another instance of dynamically generated contextual help is the examples shown in the *Scan* tab. As the user checks and un-checks the generalizations, the types of terms appearing will change. Creo displays a new example term every 500 milliseconds by querying ConceptNet and TAP.

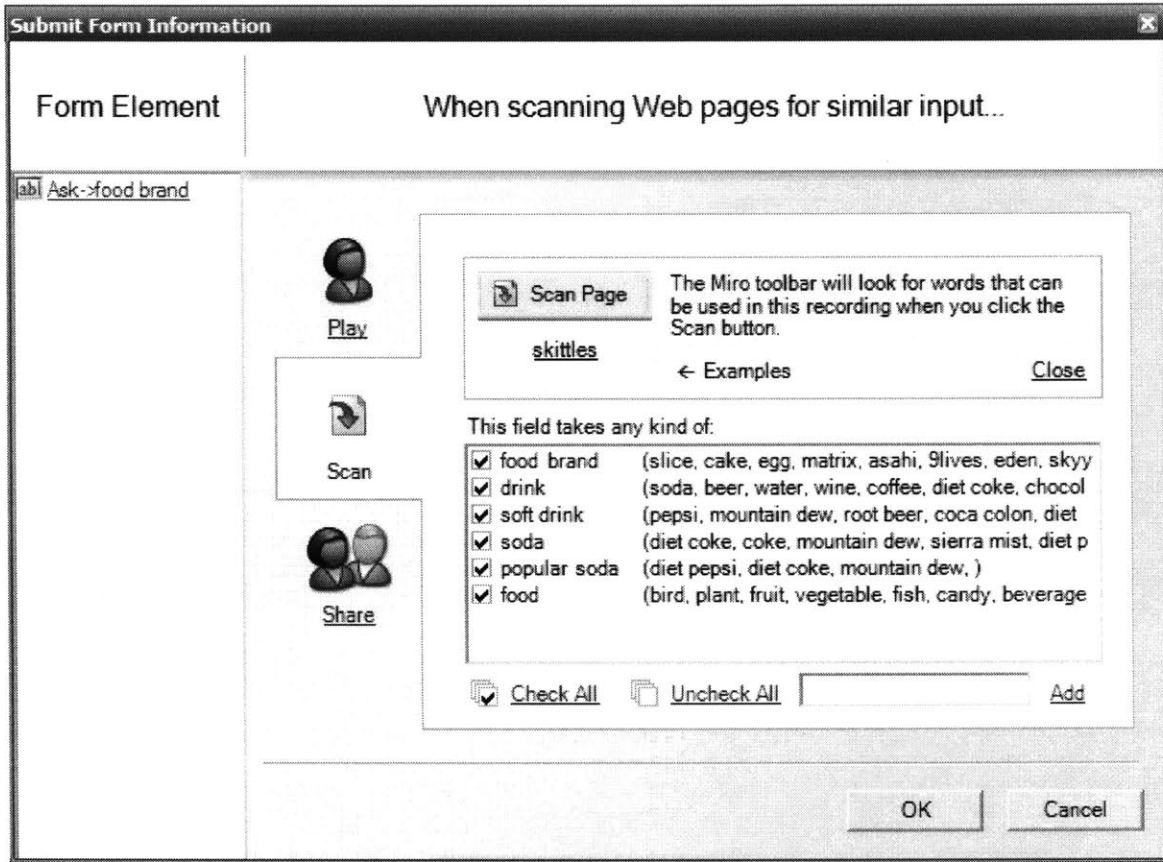


Figure 3-8 Skittles, taste the rainbow™

The breadth and variety of examples shown in the contextual help area of the *Scan* tab help the user understand that they are creating a general-purpose procedure, and not a simple macro recording. For instance, the user provides the example of “diet coke,” but they are seeing the word “skittles” appear in the contextual help area, as an example of a term that will be linked by Miro into this recording.

The purpose of dynamically generated contextual help is to aid the user in establishing a mental model of what effect various options will have in the future.

Intelligent Defaults

The third notable aspect of the user interface design is the prevalent use of intelligent defaults. While users can still directly manipulate all of the

options, like the ones shown in the previous figures (and the dynamically generated contextual help aids them in understanding what these options mean), Creo should set all of these options correctly by default. In the *Submit Form Information* window, Creo provides intelligent defaults for all of the available options:

1. *Play* Tab: If a piece of information should remain static, or if it should be generalized
2. *Scan* Tab: The set of generalizations for the information, which determines the types of terms that will be picked up by the *Scan Page* button in Miro
3. *Share* Tab: If the information is personal information or not, which determines where the information is stored

Because Creo provides intelligent defaults for all of these options, the user should be able to create a recording without ever having to open the *Submit Form Information* window. The user can still drill down in the interface and inspect Creo's decisions, but they don't have to.

Miro also provides intelligent defaults in every step of its training wizard, which streamlines the process of teaching it new information. This is described in detail in Section 2.4.3.

3.3 Implementing a User Interface Agent for the Web

Creo consists of 8,795 lines of code, and Miro consists of 9,215 lines of code. This section provides a high level view of their implementation, and briefly describes the sections of code that represent these applications' most fundamental abilities.

3.3.1 Language and Platform

Creo and Miro are implemented in C# using the .NET Framework, Version 1.1.4322, and integrate with Internet Explorer 6, which ships with Windows XP. Creo and Miro have not been tested with any other versions of the .NET Framework, Internet Explorer, or Windows.

Creo and Miro's ability to integrate with and control Internet Explorer using COM is based on the example code of two projects at the Web site codeproject.com, Pavel Zolnikov's *Extending Explorer with Band Objects Using .NET and Windows Forms* [55], and vadaa's *IE Advanced Toolbar for Favorite Site Navigation and Log-In* [56].

3.3.2 Monitoring and Impersonating the User

Creo and Miro's integration with Internet Explorer provides two core functions: (1) *Monitoring*, the ability to directly capture the user's actions, and what the user is currently looking at, and (2) *Impersonation*, the ability to recreate actions inside the Web browser, and make it appear as if an actual user was completing them.

The majority of Programming by Example systems for the Web, (discussed in Chapter 4), are implemented away from the user's computer, through the use of a proxy server. By hooking directly into the user's Web browser, Creo

and Miro are able to avoid many of the challenges that face a proxy server architecture. For instance, while HTTP is a stateless protocol, cookies allow servers to keep track of the client's state between HTTP requests. A proxy server implementation must deal with the complicated process of capturing and storing cookies while recording the user's actions. Further problems arise when the user is later playing the recording: should the system rely on cookies already stored on the server, or the cookies currently on the user's hard drive? Creo and Miro are able to avoid these types of complex implementation issues by integrating between the user and the Web browser, instead of between the Web browser and the Web.

From a Web site's perspective, there is no difference between the user completing actions by controlling their Web browser, and Creo and Miro completing actions by controlling the Web browser. Aside from the fact that Creo and Miro are faster (which actually caused problems with some Web sites, so they were subsequently slowed down) Creo and Miro do a perfect job of impersonating the user's actions. This, of course, does not include captcha tests [40], or completing any other type of higher level perceptual or cognitive challenges.

3.3.3 Basic Abilities of a Web Browser Agent

User Interface Agents for the Web like Creo and Miro must be able to monitor and impersonate a basic set of abilities. This section discusses each of these abilities, along with how they are achieved by controlling Internet Explorer. Miro relies on Creo for the impersonation abilities when playing a recording.

Capturing Navigation Events (Monitoring)

A User Interface agent for the Web must also be able to monitor navigation events. This is achieved by associating an action listener with the event:

```
Explorer.DocumentComplete +=new  
SHDocVw.DWebBrowserEvents2_DocumentCompleteEventHandler(Explorer_Document  
Complete);
```

Navigating (Impersonation)

Perhaps the most basic ability of a User Interface Agent for the Web is navigating the browser to a particular URL. Creo does this in the `ability_navigate` method:

```
public void ability_navigate(string url)
{
    Object o = null;
    Explorer.Navigate(url, ref o, ref o, ref o, ref o);
}
```

Scraping a Form (Monitoring)

Creo is able to retrieve the information currently in a form with the method `ability_formScrape`. This code fragment shows Creo retrieving the name and value of each element:

```
if(o is IHTMLInputElement){
    HTMLInputElementClass input = (HTMLInputElementClass)o;
    try
    {
        if(input.type != null){
            string type = input.type.ToLower();
            if(type == "submit" || type == "image" || type ==
"button" || type == "radio" || type == "checkbox" || type == "text" ||
type == "password" ){
                string name = input.name;
                string val = input.value;
```

Filling out a Form (Impersonation)

Creo is able to fill out a form with the method `ability_formFill`. This code fragment shows Creo getting the current page, and filling out a text field:

```
HTMLDocumentClass doc = (HTMLDocumentClass)Explorer.Document;

//for each item in UserList fill the values on the active page.
foreach(FormElement it in items)
{
    if(it.type == "text" || it.type == "password" )
    {
        try
        {
            //get the item by id.
            HTMLInputElementClass ele =
(HTMLInputElementClass)doc.getElementById(it.name);
            ele.value = it.val;
        }
    }
}
```

Training Creo to Scrape Text from a Web Page (Monitoring)

Creo learns how to scrape a specific piece of information from a Web page with the method `ability_textScrapeTrain`.

Scraping Text from a Web Page (Impersonation)

Creo is able to retrieve a specific piece of information from a Web page with the method `ability_textScrapeRetrieve`.

Turning Plain Text into a Hyperlink (Miro only)

When the user clicks on a result button in Miro, Miro turns all of the terms on the page that matched that recording into hyperlinks into the recording. This is achieved with the method `ability_miro_textLink2`, and the implementation relies on the *.NET HTML Agility Pack* by Simon Mourier [57].

3.3.4 Advanced Abilities of a Web Browser Agent

This section discusses the implementation of some of the more advanced features of Creo and Miro.

Creating an Abstract Representation of a Procedure on the Web in XML (Creo)

The overall purpose of Creo is to turn the user's example of interacting with a Web site into a general-purpose and machine-readable procedure. Creo is Latin for "to create." Here is an example of what Creo literally creates:

```
<?xml version="1.0" encoding="utf-8"?>
<Recording xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <guid>a0f94716-ba03-4432-9530-21472cc09a82</guid>
  <name>Order Food</name>
  <color_r>0</color_r>
  <color_g>153</color_g>
  <color_b>0</color_b>
  <goal>Order Food</goal>
  <confirm>>false</confirm>
  <computerUse>>true</computerUse>
  <Action>
    <description>Browse to: Welcome to FreshDirect</description>
    <type>navigate</type>
```

```

<title>Welcome to FreshDirect</title>
<url>http://www.freshdirect.com/index.jsp</url>

<formSubmit_generalizedFormElements>>false</formSubmit_generalizedFormElements>
  <formSubmit_submitNumber>0</formSubmit_submitNumber>
  <returnValue_trainedTerm />
  <returnValue_startTerm />
  <returnValue_endTerm />
</Action>
<Action>
  <description>Submit: Ask-&gt;food brand</description>
  <type>formSubmit</type>
  <title />
  <url>http://www.freshdirect.com/index.jsp</url>
  <FormElement>
    <type>text</type>
    <name>searchParams</name>
    <val>diet coke</val>
    <ask>>true</ask>
    <askPrompt>food brand</askPrompt>
    <personalInfo>>false</personalInfo>
    <Generalization>
      <name>food brand</name>
      <use>>true</use>
    </Generalization>
    <Generalization>
      <name>drink</name>
      <use>>true</use>
    </Generalization>
    <Generalization>
      <name>soft drink</name>
      <use>>true</use>
    </Generalization>
    <Generalization>
      <name>soda</name>
      <use>>true</use>
    </Generalization>
    <Generalization>
      <name>popular soda</name>
      <use>>true</use>
    </Generalization>
  </FormElement>
  <FormElement>
    <type>image</type>
    <name />
    <val />
    <ask>>false</ask>
    <askPrompt />
    <personalInfo>>false</personalInfo>
  </FormElement>

<formSubmit_generalizedFormElements>>true</formSubmit_generalizedFormElements>
  <formSubmit_submitNumber>1</formSubmit_submitNumber>
  <returnValue_trainedTerm />
  <returnValue_startTerm />
  <returnValue_endTerm />
</Action>
<Action>
  <description>Finish by displaying: FreshDirect - Search</description>
  <type>returnPage</type>
  <title>FreshDirect - Search</title>

```

```
<url>http://www.freshdirect.com/search.jsp?searchParams=diet+coke&amp;x=0&amp;y=0</url>

<formSubmit_generalizedFormElements>>false</formSubmit_generalizedFormElements>
  <formSubmit_submitNumber>0</formSubmit_submitNumber>
  <returnValue_trainedTerm />
  <returnValue_startTerm />
  <returnValue_endTerm />
</Action>
<fileName>Order_Food.xml</fileName>
</Recording>
```

To simplify the implementation, the XML files created by Creo are serializations of the C# objects that are used internally to manage recordings. Ideally, this information should be expressed using RDF. However, since Miro and Adeo are currently the only applications that play these recordings, abstracting the procedure to RDF instead of XML is a low priority feature.

The beginning of the file contains information about the recording, like its name, GUID, the hyperlink color that will be used in Miro, and a Boolean value that determines if a confirmation window will be displayed to the user before the recording is played.

The rest of the XML file consists of a series of actions. Each action has a type, such as `navigate`, `formSubmit`, `returnValue`, and `returnPage`. If the action is a `formSubmit` it will also contain a list of form elements, each containing a specific value or a list of generalizations. How these generalizations are produced is discussed in the following section.

Rapidly Accessing Semantic Knowledge with IsaNet (Creo and Miro)

All of the semantic information used by Creo and Miro is stored locally on the hard drive, and loaded into memory after Internet Explorer is opened. This information is loaded into memory because it must be accessed extremely rapidly for applications like Miro to function. Creo and Miro access the knowledge in ConceptNet and TAP through the class `ability_commonsenseIsaNet`. The combination of these knowledge bases is

referred to throughout the code as IsaNet because the combined knowledge base contains only the ISA relations found in ConceptNet and TAP, to reduce the amount of required memory, and speed up the time it takes to load this information from the hard drive.

Earlier versions of Creo and Miro accessed the ConceptNet API through XML RPC. The knowledge in TAP was also available through the ConceptNet's XML RPC API because the TAP knowledge base was reformatted to match ConceptNet's formatting. Reliance on ConceptNet's API through XML RPC was eventually abandoned for several reasons: (1) requiring the user to launch a separate server application before using Creo and Miro is annoying, (2) setting up the server creates more complicated install instructions, (3) the ConceptNet API does not contain a feature to look up instances, like finding all of the concepts X where $(ISA\ X\ Y)$ (which makes features like the dynamically generated contextual help in Creo's *Scan* tab impossible), and most importantly, (4) processing a quick succession of thousands of requests to the ConceptNet API through XML RPC is so slow that the user's Web browsing experience is significantly degraded when Miro is running.

For all of these reasons, access to ConceptNet and TAP was re-implemented in C# in a knowledge base called IsaNet, adding several missing features to the API, ignoring many unneeded features, and considerably increasing the speed at which concepts can be generalized.

The class `ability_commonsenseIsaNet` contains methods like `isaNet_getGeneralizations`, which returns all of the generalizations of a concept, and `isaNet_getInstances`, which returns all of the instances of a concept. Miro's training wizard uses the methods `isaNet_getGeneralizationWordPredictions`, which returns word predictions for a string, weighted by generality, and `isaNet_addNewKnowledge`, which adds a new piece of information to the knowledge base, both in memory and on the hard drive.

Scanning a Page (Miro)

When Miro scans a Web page, it first breaks the text on the page into phrases and terms, and then looks up the generalizations of each using IsaNet. This results in a very quick succession of thousands of requests to IsaNet for generalizations, every time the user loads a Web page. Each of the generalizations returned for a particular piece of information on the page is compared to the list of generalizations for the user's recordings. During the scan Miro also performs natural language processing on each piece of text on the Web page, to see if it can potentially learn any new information. This process of breaking a Web page apart to look for generalization matches and perform natural language processing is completed by this code fragment:

```
//load isaNet before performing the scan
this.miro_loadIsaNet();

//retrieve all of the text of the page
ArrayList pageText = this.ability_textAll();

foreach(string textNode in pageText)
{
    //check the entire text node for genMatches
    this.miro_scan_checkText(textNode);
    //attempt to extract knowledge from the textNode
    if(this.miro_feature_learnFromWeb)
    {
        //do Sentence Boundary Detection before sending it
        ArrayList sentences = this.nlp.nlp_getSentences(textNode);
        foreach(string sentence in sentences)
        {
            //debug.writeLine(sentence);
            //debug.writeLine("-----");
            this.miro_scan_learnKnowledge(sentence);
        }
    }

    //should be chunking here instead of splitting into words
    string[] terms = textNode.Split(' ');
    foreach(string term in terms)
    {
        //check the term for genMatches
        this.miro_scan_checkText(term);
    }
}
```

If a generalization match is found, Miro creates a `GenMatch` object and adds it to an `ArrayList` of potential results. Miro disambiguates terms based on patterns and thresholds after the entire scan has been completed.

Miro's *Semantic Search* feature is implemented by simply performing a scan on the text of the search box, minus the natural language processing steps.

Associating the Playback of Recording with a URL (Miro)

When the user selects one of the activated recordings after clicking *Scan Page*, all of the plain text terms that matched that recording are converted into hyperlinks into the recording, with each of the activated terms replacing one of the recording's variables.



Blueberry Pudding Cake

Figure 3-9 Miro converts plain text into hyperlinks that invoke a recording

To create hyperlinks that link to a sequence of actions (instead of a static document), Miro generates and watches for URIs that follow a specific pattern. For instance, if the word "Blueberry" is activated for a recording that performs a search at the user's grocery store, Miro replaces the word with the HTML:

```
<a style="color: rgb(0,153,0)" target="_blank"
href="http://localhost/miro/miro.htm?a0f94716-ba03-4432-9530-
21472cc09a82&food_brand=blueberry">Blueberry</a>
```

The format of the URI is:

`http://localhost/miro/miro.htm?RecordingGUID&GeneralizationMatch=SpecificValue`. Note that this is just a URI, the user does not need to be running a server, and the file `miro.htm` does not need to actually exist. Also

note how the GUID and hyperlink color match the information in the XML representation of the "Order Food" recording shown earlier.

When the user clicks the hyperlink, it will open a new Internet Explorer window. Miro will catch the navigation event, notice the URI, and then execute the recording in the Web browser, instead of IE loading the Web page.

Natural Language Processing (Miro)

Miro's ability to perform natural language processing tasks is enabled through the class `ability_nlpTools`. This class contains methods like `nlp_getSentences`, which performs sentence boundary detection, `nlp_checkNoun`, which returns if a term is a noun or not, and `nlp_lemmatize`, which returns the singular version of any term passed to it, accounting for thousands of exceptions like "dwarves" → "dwarf." These methods were programmed based on the valuable advice of Rada Mihalcea, who was a visiting professor at the Media Lab in June and July 2005, and is from the Language and Information Technologies research group at the University of North Texas.

Learning from the Web (Miro)

Miro's ability to learn new information by reading the Web takes place in the methods `miro_scan_learnKnowledge`, which calls the method `miro_scan_learnKnowledge_checkText`. Section 2.4.4 shows the 11 patterns that Miro uses to attempt to extract IsA relationships from text. Here is a code fragment that shows how Pattern 3 is implemented. This pattern would extract `{IsA "Henry Lieberman" "research scientist"}` from the sentence "Henry Lieberman is one of the research scientists here."

```
string regex_rad1 = @"(?<subjectTerm>^(.+)) is one of (the)?  
(?<objectTerm>.+)" ;  
Match m_rad1 = Regex.Match(text, regex_rad1);  
if (m_rad1.Success)  
{
```

```
        foundMatch = true;
        string objectTerm =
m_radal.Groups["objectTerm"].ToString().ToLower().Trim();
        debug.WriteLine("    Found match using Radal:");
        debug.WriteLine("    subjectTerm=" +
m_radal.Groups["subjectTerm"]);
        debug.WriteLine("    objectTerm=" + m_radal.Groups["objectTerm"]);

        if(objectTerm.Equals(generalization))
        {
            debug.WriteLine("Object Term matches: " + recording.goal);

            //create the genMatch
            string subjectTerm =
radal.Groups["subjectTerm"].ToString().Trim().ToLower();
            string subjectTermPageText =
m_radal.Groups["subjectTerm"].ToString().Trim();

            if(subjectTerm.Length > 0)
            {
                GenMatch gm = new
GenMatch(objectTerm,subjectTerm,subjectTermPageText);
                gm.learned = true;
                this.miro_scan_addGenMatch(recording, gm);
            }
        }
    }
}
```

After Miro locates a potential new piece of knowledge, it writes it to the hard drive so that future instances of Miro can track if the user clicked through on the knowledge or not. This is handled with the methods `miro_learn_openLearnLog` and `miro_learn_saveLearnLog`.

Miro also contains a number of methods to enable the functionality of Adeo, these are discussed in Section 3.4.

3.3.5 Source Code and Documentation

The source code of Creo and Miro, along with further documentation, can be found in Appendix C. If Appendix C is not included with this document, or you are reading a physical copy of this document, try Googling "DDFB2B94-8F51-4b34-8BA7-41A7FA03E4F0." If you are not sure what Google is, it was a very popular search engine at the turn of the century (what happened to them?) Any form of search on whatever type of fancy newfangled Web you apparently have access to should be able to locate the document.

3.4 Invoking Procedures on the Web from a Mobile Device

This section discusses the design and implementation of Adeo.

3.4.1 Design

User Interface Design

Adeo's user interface was designed with the specific objectives of simplicity, consistency, and feedback. Once Adeo is opened, the user can invoke a recording with a single click, following the design principle of simplicity. Adeo's list of recordings matches, and is automatically synchronized with, Creo's *Player* tab. This follows the design principle of consistency.

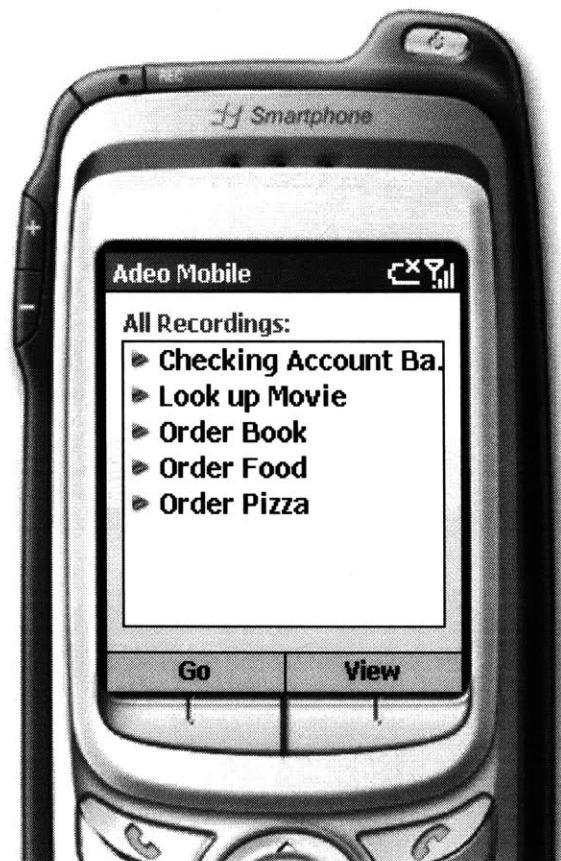


Figure 3-10 Adeo's list of recordings is automatically synchronized with the user's computer

The third design objective, feedback, involves the user experience when a recording is being played. When the user is playing a recording, it takes several seconds for the server to receive the request, process it, and return the result to the user. While waiting for the server's response, the user is presented with a (rather cool and mildly difficult to get working) pulsating green bar that represents the connection between their device and the computer:



Figure 3-11 Adeo provides the user with feedback

Architecture

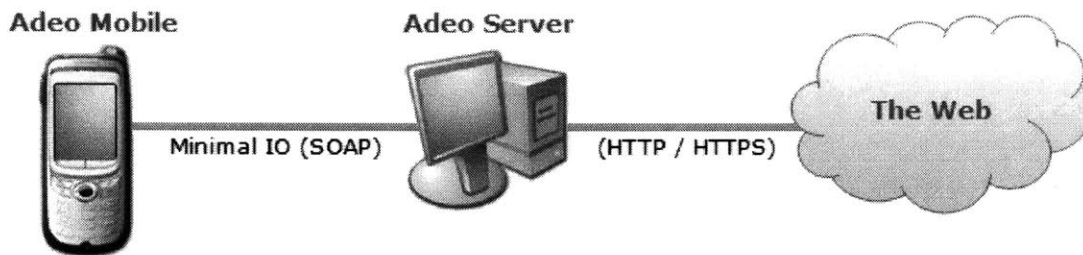


Figure 3-12 Adeo's client/server system architecture

Adeo consists of three components: (1) an application on the user's Smartphone called Adeo_m, (2) a server application called Adeo_s, and (3) a

collection of methods in Miro that receive and carry out commands sent from Adeo_s.

3.4.2 Implementation

Implementation of Adeo_m

Adeo_m consists of 1,238 lines of code and was programmed in C# using the .NET Compact Framework. Adeo_m has been developed for Smartphones, but it could be easily adapted to run on Pocket PCs as well. Adeo_m has been tested using Windows Mobile 2003.

Adeo_m is a thin client application. The first thing that Adeo_m does after launching is download an updated list of the user's recordings from their computer. Adeo_m receives this information by querying Adeo_s. Just as Adeo streamlines completing procedures down to the minimal amount of necessary input and output, Adeo also streamlines its bandwidth usage. Adeo_s sends Adeo_m a minimal amount of information about the recordings, consisting of only their names and GUIDs.

Because Adeo_m must deal with the limited storage space and processing power of a mobile device, none of the semantic information found in ConceptNet and TAP is stored directly on the device. Previous mobile applications that use commonsense knowledge have loaded scaled down versions of ConceptNet into memory. However, since this semantic information is used by Adeo to understanding the meaning of a very broad range of searches, reducing the size of the knowledge base would have limited the application's functionality. Adeo_m must query Adeo_s for generalization matches when the user is performing a semantic search. Assuming there is no significant amount of network lag, from the user's perspective the device is performing these searches on its own.

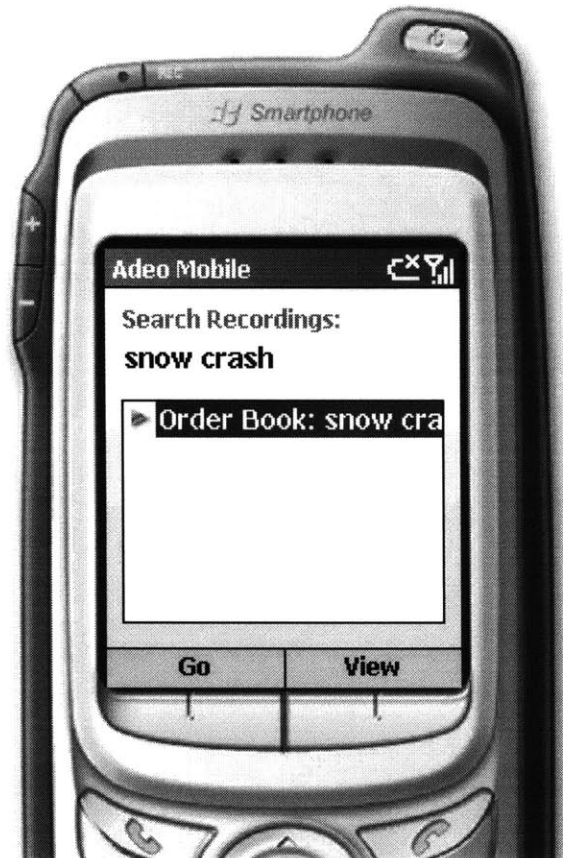


Figure 3-13 Adeo_m performs semantic searches by querying Adeo_s for generalizations

Implementation of Adeo_s

Adeo_s consists of 1,466 lines of code and was programmed in C# using ASP.NET. The purpose of Adeo_s is to expose the user's recordings as Web Services, which can be queried using SOAP. As graphically represented in Figure 1-2 in Chapter 1, Adeo_s is able to literally expose actions on the Web as Web Services.

The main function of Adeo_s's is to facilitate communication between Adeo_m and Miro. These communications include (1) sending a list of the user's current recordings from Miro to Adeo_m, (2) receiving a request to play a recording from Adeo_m and sending it to Miro, and (3) retrieving the result of a completed recording from Miro and sending it to Adeo_m.

Implementation of Adeo code in Miro

The Adeo code in Miro is used for facilitating communications with Adeo_s.

This code can be instructed to play a recording and then return the recording's result. This is achieved through the methods

`adeo_monitorProcessLog`, `adeo_checkProcessLog`, `adeo_openProcessLog`
and `adeo_saveProcessLog`.

3.4.3 Source Code and Documentation

The source code and further documentation for Adeo_m and Adeo_s can also be found in Appendix C.

Adeo's thin client architecture has several limitations. These are discussed in detail in the following section.

LAPIS is different from Creo in that the user has a choice of either writing a script themselves, or generating the script through demonstration. With Creo, users must demonstrate every action in a recording. Another difference between LAPIS and Creo is that Creo automatically generalizes scripts, while with LAPIS “an experienced user can generalize the demonstration on the fly by typing commands at crucial points instead of pointing-and-clicking” [104]. Because LAPIS requires users to directly edit scripts in order to generalize them, novice users will not be able to create the same types of general-purpose procedures with LAPIS that they can create using Creo.

However, there are some advantages to the scripting environment that LAPIS provides: “since LAPIS embeds a full scripting language, the resulting scripts can be significantly more expressive than recorded macros, without taking much more time to develop” [104]. Additionally, technically minded users may prefer the scripting environment of LAPIS to the Programming by Example environment of Creo.

This type of end-user programming for the Web is discussed in greater detail in the following section.

4.3.2 End-User Programming for the Web

Chickenfoot, by Michael Bolin and Rob Miller is an end-user programming environment that enables users to automate, customize, and integrate Web applications without examining a page’s source code [105].

Chickenfoot allows users to script against the *rendered model* of a Web page, as opposed to the *Document Object Model* or the *string (HTML) model*. For instance, to script the action of clicking on the button “I’m Feeling Lucky” on the Google homepage, a user would write: `click("I’m Feeling Lucky")`

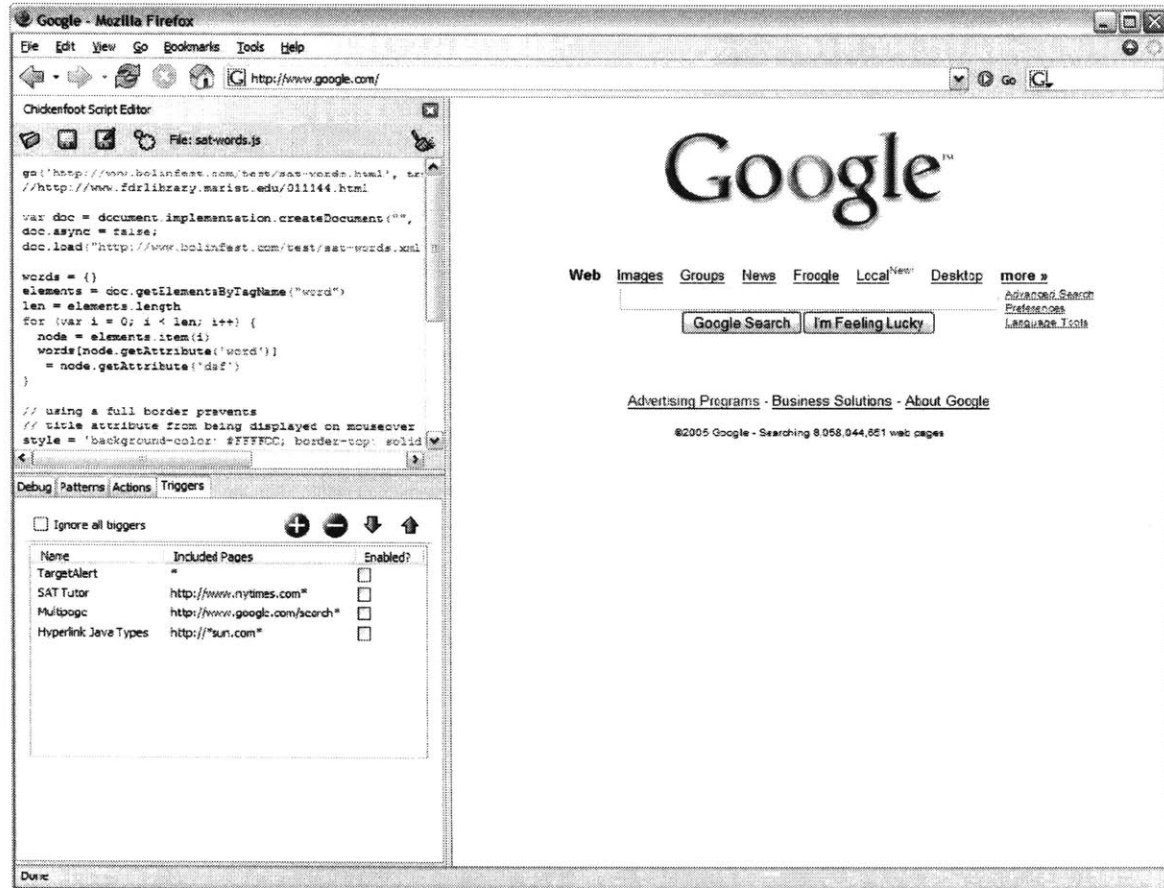


Figure 4-13 Chickenfoot, by Michael Bolin

Even though Chickenfoot requires that users write scripts, as opposed to programming the system by example, the scripting environment still matches the user’s mental model of the page. Additionally, Chickenfoot allows users to script behaviors that cannot be programmed through demonstrations. For instance, a major feature of Chickenfoot is the ability to reformat Web pages.

It would be interesting to integrate a scripting environment like Chickenfoot with a Programming by Example system like Creo. This would enable advanced users to switch between the two modes while recording. The interface would be similar to the way many development environments allow users to make modifications in both the *Design View* and the *Source View*.

4.3.3 Extracting Information from the Web

One of the features of Creo is to extract information from the Web. For instance: extracting the balance of a user's bank account. Many of the applications mentioned in this section provide this functionality, including TrIAs [87, 93-96], Turquoise [98], the Internet Scrapbook [87, 99], Web Macros [100-103], and Chickenfoot [105]. In fact, the majority of these systems' ability extract knowledge from the Web is more robust, accurate, and fault-tolerant than Creo's implementation. Creo does not focus heavily on this feature since a significant amount of research has already been directed at solving this problem. In particular, Nicholas Kushmerick's work on both Wrapper Induction [106] and Wrapper Verification [107] addresses this issue in much greater detail.

4.4 Work Related to Miro

This section describes research related to the three main features of Miro: data detection, semantic search, and learning knowledge from the Web.

4.4.1 A Brief History of Data Detection

1945: The Memex

As mentioned in the introduction, the origin of Data Detection can be attributed to Vannevar Bush. Data Detection is referred to as *Selection by Association*, or *Associative Indexing*:

Associative Indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the memex. The process of tying two items together is the important thing. [18]

Vannevar Bush's use of the phrase "select immediately and automatically" implies he was not just referring to static hyperlinks, but also the real-time detection and association of data.

The history of Data Detection continues in the late 1990s with a quick succession of products and research.

1995: IntelliSense

Word 95, part of Microsoft Office 7.0, contains a feature called IntelliSense that provides real-time spell checking. IntelliSense in Word 95 is the binary equivalent of data detection, in that it simply differentiates between words and unknown terms. IntelliSense checks each term the user types against a dictionary, and underlines terms it does not recognize with a red line,

indicating that the term may be misspelled. The closest matching word is available to the user in a contextual menu.

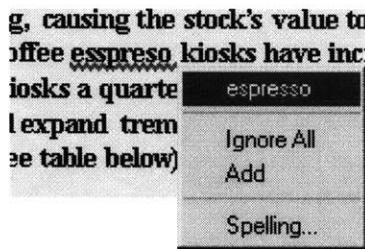


Figure 4-14 1995: IntelliSense³

The following applications go beyond simply detecting words, to detecting a larger set of types of data.

1997: The Intel Selection Recognition Agent

Presented at IUI 97, the Intel SRA (Selection Recognition Agent), created by Milind Pandit and Sameer Kalbag, recognizes meaningful words and phrases in text, and enables useful operations on them [108].

The SRA automatically turns plain text into a kind of hypertext, by quickly recognizing selected text, and then linking it to related information and applications. Thus, the SRA turns the entire desktop into a kind of hypertext document. [108]

The Intel SRA is able to detect six types of data: geographic names, dates, email addresses, phone numbers, Usenet news groups, and URLs. Similar to IntelliSense, actions appear in a context menu, a user interface convention introduced with Windows 95. While the Intel SRA recognizes only six different types of data, the source code was made available so that other developers could program their own recognizers. In user tests, they found

³ The term "IntelliSense" originated when a Microsoft employee said "intelligence" with a mouth full of beer.

that for every task they tested, users saved both time and effort, in some cases over 50%.

1998: Apple Data Detectors

Similar to the Intel SRA, Apple Data Detectors, created by Bonnie Nardi, James Miller and David Wright, detects information in documents and links this information to relevant actions [109]. The authors relate data detection to goal-based design:

Apple Data Detectors therefore has the ability to infer appropriate high-level goals from user actions and requests and take appropriate action to achieve these goals. When users invoke it on a region of text in a document, they are saying, in effect, "Find the important stuff in here and help me do reasonable things to it." [109]

Apple Data Detectors are able to detect phone numbers, fax numbers, street addresses, email addresses, email signatures, abstracts, tables of contents, lists of references, tables, figures, captions, meeting announcements, and URLs. This information is then linked to actions.



Figure 4-15 1998: Apple Data Detectors

Apple Data Detectors can also detect domain-specific information, like ISBN numbers and stock symbols, however this information must be provided by end-users as a list of strings. Creating an action that can be associated with data requires a programmer to write the appropriate action script.

Similar to Miro, multiple actions can be associated with a particular type of data. Additionally, Apple Data Detectors can be easily shared, although they make no reference to managing any type of personal information.

The authors describe Apple Data Detectors as “a first step toward extracting semantics from everyday documents without asking users to create documents in new ways” [109].

1998: CyberDesk

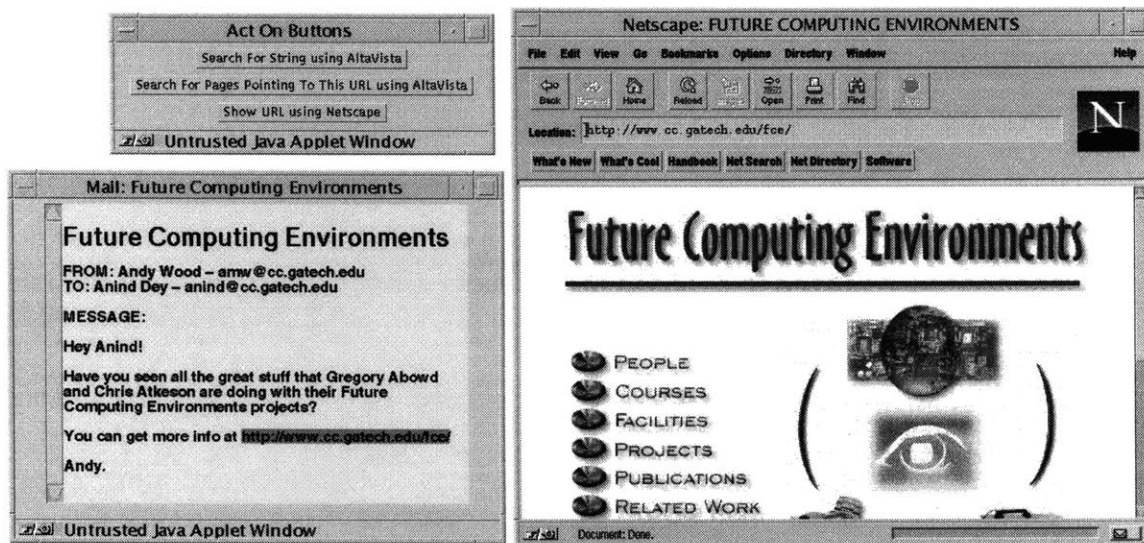


Figure 4-16 1998: CyberDesk

CyberDesk, created by Anind Dey, Gregory Abowd and Andrew Wood, provides an infrastructure for self integrating software, in which the integrations are caused by the user's actions [110]. They refer to this as “context aware integration.”

Context-aware integration changes the paradigm of interaction from a user seeking out functionality in software applications to the infrastructure seeking out the user at relevant times. [110]

CyberDesk detects eight kinds of data: dates, phone numbers, addresses, names, email addresses, GPS positions, and times. While this is less than the types supported by Apple Data Detectors, CyberDesk provides a more advanced framework for actions, including the ability to chain actions together, and to combine different pieces of data into the same action. CyberDesk also allows for data detection on mobile devices. For instance, CyberDesk provides the ability to associate a GPS position with the action of loading a URL. Like the Intel Selection Recognition Agent, and Apple Data Detectors, the only way to create new actions with CyberDesk is to program them.

1998: LiveDoc and DropZones

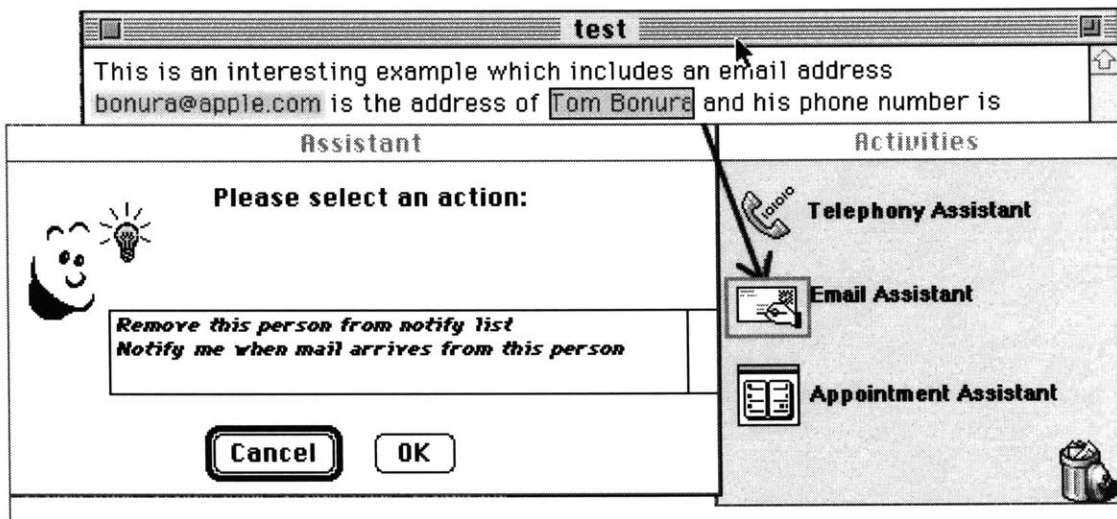


Figure 4-17 1998: LiveDoc and DropZones

LiveDoc and DropZones, created by Thomas Bonura and James Miller present a number of interface changes to Data Detection [111, 112]. Recognized objects are highlighted in place, and users can drag and drop objects onto actions. This allows actions to run on multiple objects. Like the Data Detectors preceding it, LiveDoc and DropZones requires that actions are created by a programmer.

4.4.2 Present-Day Data Detectors

The functionality of these Data Detectors has been integrated into several mainstream consumer products. Microsoft Office XP (released in 2001), provides data detection with a feature called Smart Tags, and the Google Toolbar 3.0 (released in 2005), adds data detection to Web browsing, with a feature called AutoLink. Microsoft's Smart Tags currently recognizes eight types of data, although a developer can program additional data types and actions. Google's AutoLink currently recognizes three types of data: addresses, ISBNs and Vehicle Identification Numbers. The actions associated with these types of data are controlled by Google.

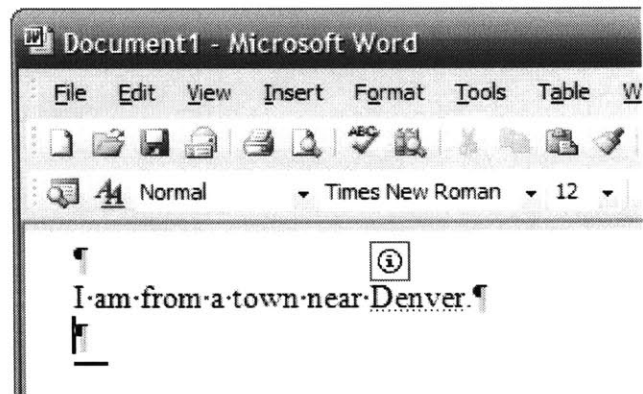


Figure 4-18 2001: Microsoft Smart Tags

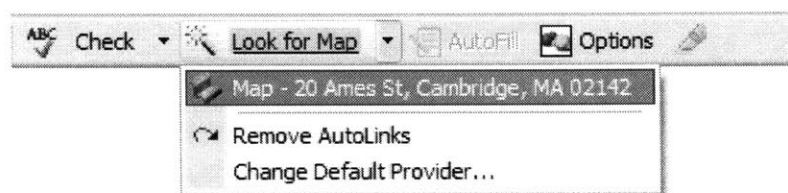


Figure 4-19 2005: Google AutoLink

4.4.3 Back to the Future

One similarity of all of the research on Data Detectors in the late 1990s is each paper's future work section.

Programming by Example and End-User Programming

First, all of the research mentioned the importance of Programming by Example and end-user programming. The creators of the Intel Selection Recognition Agent wrote, "We would like to enhance the Selection Recognition Agent along the lines of Eager [a Programming by Example system], allowing it to detect the repetition of action sequences in any application and automate these sequences" [108]. The creators of Apple Data Detectors wrote that a "goal is to complete a prototype of an end-user programming facility to enable end users to program detectors and actions, opening up the full Apple Data Detectors capability to all users" [109]. Finally, the creators of CyberDesk wrote that they were "investigating learning-by-example techniques to allow the CyberDesk system to dynamically create chained suggestions based on a user's repeated actions" [110].

Grammex (Grammars by Example) [92], previously mentioned in the section on work related to Creo, allowed users to create Data Detectors through Programming by Example. Like Creo, Grammex allowed users to define the actions to associate with data by providing demonstrations. However, Grammex was limited to the few Macintosh applications that were "recordable" (sending user action events to the agent) [92]. Similar to the Data Detectors preceding it, Grammex based its data detection on patterns of information. For instance, Grammex could learn how to detect email addresses if the user showed it several examples with the format *person@host*. Unfortunately, very few types of data outside of URLs, email addresses and phone numbers actually have a detectable structure, limiting the usefulness of such a system. This leads to the second "future work" topic mentioned by Data Detector researchers of the late 1990s: semantics.

Semantics

The creators of Apple Data Detectors noted that relying on pattern detection has many limitations:

It is easy to imagine a company might choose a syntax for its product order numbers—a three digit department code followed by a dash followed by a four-digit product code—that would overlap with U.S. telephone number syntax, thus leading Apple Data Detectors to offer both telephone number and part-ordering actions...We can do little about these overlapping syntaxes without performing a much deeper, semantic interpretation of the text in which the pattern appears [109].

The creators of CyberDesk also discussed the topic of semantic interpretation, writing that they were interested in “incorporating rich forms of context into CyberDesk, other than time, position, and meta-types” [110].

Miro expands the types of data that can be detected from the previous range of three types (Google’s AutoLink) and thirteen types (Apple Data Detectors), to the full breadth of knowledge found in ConceptNet and TAP.

4.4.4 Semantic Search

Miro provides the user with semantic search functionality in a Web browser, and Adeo provides the user with semantic search functionality on a mobile device. Both of the knowledge bases Miro uses to perform semantic searches, ConceptNet and TAP, have been used in the domain of search in the past. ConceptNet has been used by GOOSE (Goal-Oriented Search Engine with commonsense), as discussed in the earlier section “Beating Some Common Sense into Interactive Applications.” TAP has been used in a system called ABS (Activity Based Search).

ABS

The same group that created TAP, The Knowledge Systems Laboratory at Stanford, has applied the TAP knowledge base to improving search. ABS (Activity Based Search), was created by R. Guha, Rob McCool, Eric Miller, Richard Fikes, and Deborah McGuinness [11-15, 32]. The idea behind ABS is

that “even a very shallow understanding of the potential activities that make sense in the context of a given search can make a dramatic difference in the search experience” [13]. ABS looks up the search query in TAP, and then uses this knowledge to augment a traditional Google search with contextual information.

Given the search term "Yo-Yo Ma", we figure out that Yo-Yo Ma is a musician. The activities associated with a musician are (retrieving data about) his concert schedule, albums, posters, auctions... The system uses TAP to retrieve these pieces of data, which are then used to augment the search results. In this example, the search results are augmented with the data that Yo-Yo Ma is performing in Seattle on 5/12 (from TicketMaster), that there is an auction for one of his CDs on EBay which ends in 3 minutes, [and] that CDNow has his Appalachian Journey CD on sale for \$14.99 [13].

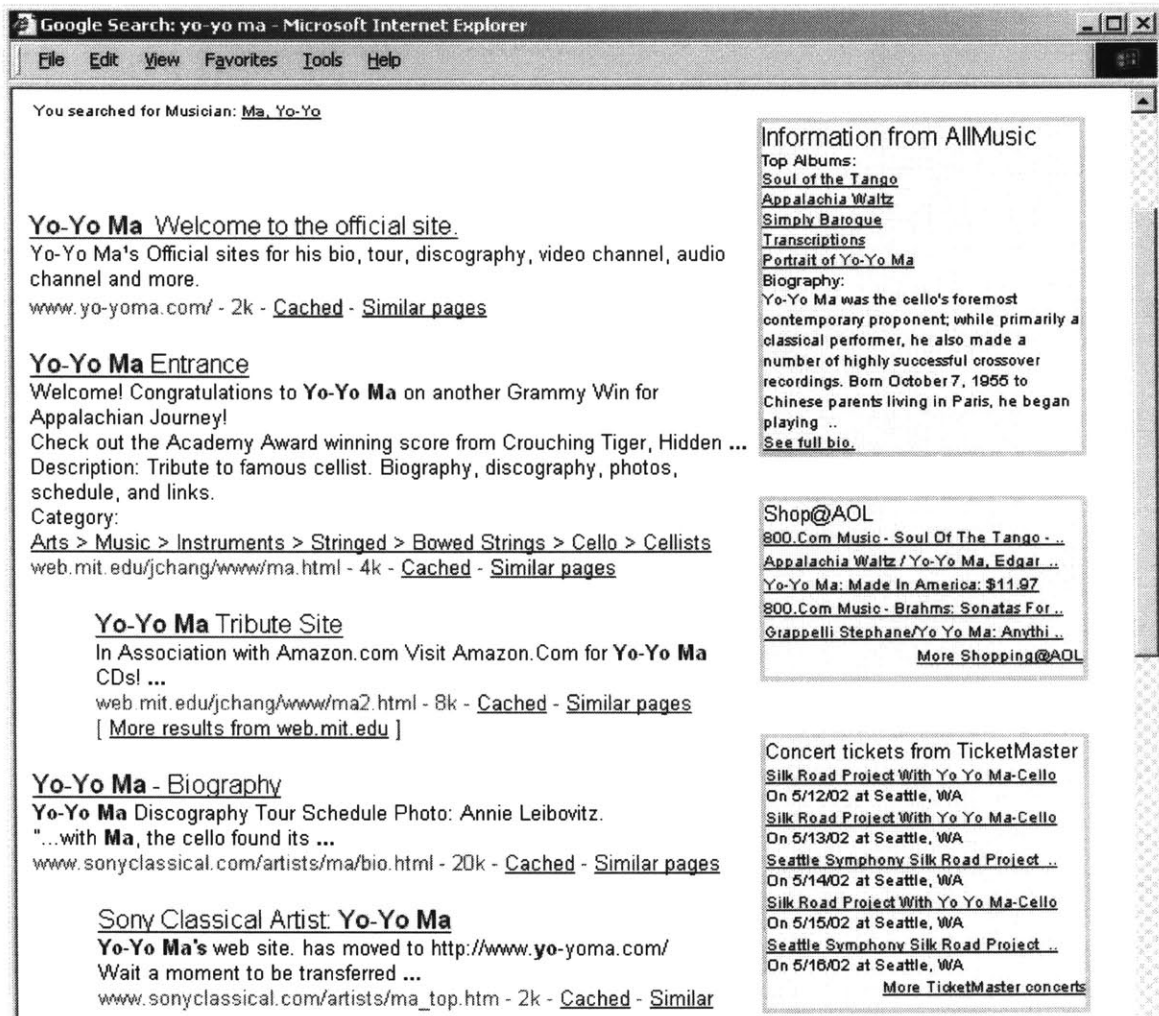


Figure 4-20 Activity Based Search by the Stanford Knowledge Systems Laboratory

The activities augmenting the traditional search are controlled by the search engine (as opposed to the user), creating a form of "real time advertising" [11].

Dealing with Ambiguity: What is the Matrix?

One of the challenges facing semantic search is ambiguity. The creators of TAP note:

The search "Matrix" yields 4,380,000 results and included in the first 20 are references to a movie, the mathematical concept, four different

companies, a data center, a monastery, a display device, a hair care product and an on-line community [14].

While using Creo, "Matrix" is also displayed as example input for a grocery store recording. This is caused by line 31237 in TAP, (IsA "matrix" "food brand").

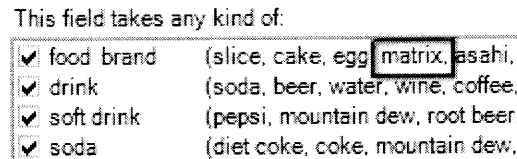


Figure 4-21 The Matrix?

Miro deals with the ambiguity of natural language by leveraging patterns and thresholds when scanning a Web page. For semantic searches, Miro uses the context of what the user is looking at to weigh the various meanings of the user's query.

The creators of ABS present three additional strategies to disambiguate the meaning of terms:

1. The popularity of the term as measured by its frequency of occurrence in a text corpus or the availability of data on the Semantic Web. *E.g., Paris, France is a preferred denotation for "Paris", compared to Paris, Texas, the music group Paris, etc. [14].*

2. The user profile may guide the selection of the denotation. *E.g., the phrase "Quark", as used by a Star Trek fan might be more likely to denote the Star Trek character than the subatomic particle [14].*

3. The search context can also help select the denotation. *E.g., if the user has been searching for information about musicians, the query "Pink" may more likely denote the musician than the color [14].*

All three of these strategies could improve Miro's ability to disambiguate terms. ABS does not take into account what the user is looking at because the search is accessed through a Web page, instead of a toolbar. Like Miro, ABS allows users to indicate which meaning of the term they meant, if the system makes an incorrect guess.

Using Context to Disambiguate Concepts: Reference by Description

When performing a semantic search or scanning a page, Miro uses the surrounding context to disambiguate terms. The Knowledge Systems Laboratory has created a brilliant approach to replacing URIs, based on providing enough description to uniquely identify concepts:

We believe that URIs as the sole mechanism for referring to nodes is not scalable from the perspective of coordination complexity. Getting everyone to agree on a set of names/URIs/URNs for every object they might want to exchange data about is not realistic...Straight forward approaches such as assigning every object a unique canonical name (a URI) involve getting millions of sites to agree on names for billions of objects leading to potentially insurmountable coordination complexities.

This problem is different from that of URLs, where every page is controlled by a single site and the creator of the object gives it a name when the object is created. In contrast, no one controls Beethoven (or Yo-Yo Ma) or any number of other objects. More importantly, a large number of sites have data about Beethoven and Yo-Yo Ma. Even if someone were to be designated as the URN assigner for Yo-Yo Ma (such as his publisher or even him), we would still have a problem mapping the identifiers/keys currently in use by different databases to this URI [13].

They solve this problem by providing just enough context to unique identify a concept, referred to as Reference by Description [13]. For instance, this RDF fragment uniquely identifies Yo-Yo Ma's Appalachian Journey:

```
<a:MusicAlbum>
  <dc:title>Appalachian Journey</dc:title>
  <a:hasAuthor>
    <a:Musician>
      <dc:title>Yo-Yo Ma</dc:title>
    </a:Musician>
  </a:hasAuthor>
</a:MusicAlbum>
```

Similar to how Miro uses surrounding context to disambiguate terms when performing a semantic search or scanning a page, the Stanford Knowledge Systems Laboratory has leveraged context to solve one of the biggest obstacles facing the emergence of the Semantic Web: agreement on URIs.

4.4.5 Learning from the Web

The eleven patterns used by Miro to extract IsA relationships from the Web come from two sources. Six of the patterns come from Marti Hearst's lexio-syntatic patterns for hyponymy presented in his paper *Automatic Acquisition of Hyponyms from Large Text Corpora* [41]. The remaining five patterns were suggested by Rada Mihalcea, who was a visiting professor at the Media Lab in June and July 2005, and is from the Language and Information Technologies research group at the University of North Texas.

The Knowledge Systems Laboratory at Stanford has also been researching ways to extract knowledge from the Web with a system called Vault [15, 16]. Unlike Miro, which works collaboratively with the user to learn new knowledge, by making suggestions and watching the user's actions, Vault attempts to extract knowledge from unstructured information on the Web by itself. Vault scans news articles at Yahoo News, learning facts like *Tony Blair is the Prime Minister of Britain*, and *Britain is a country* [15]. Since Vault performs knowledge extraction by itself, it will occasionally make mistakes

that a human would not. For instance, when scanning articles, Vault will learn that Josiah Bartlet (a character who plays the role of the President of the US in the television series "The West Wing") is the President of the United States [15, 16]. Miro would make the same mistake: if the user had a recording to look up presidents, Miro would read the sentence and activate the recording for Josiah Bartlet. However, the user would be able to catch the mistake. The Stanford Knowledge Systems Laboratory is attempting to solve the more difficult challenge of dealing with fictional information during unassisted knowledge capture through the use of a genre system. This enables Vault to automatically tell the difference between a news story and an entertainment story [15].

4.5 Work Related to Adeo

Adapting Web content for use on mobile devices is an extremely active area of research. One popular research direction is directly modifying the content for display on smaller screens. For instance, Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang have developed a way to browse Web pages on mobile devices by breaking a page into pieces, and allowing users to zoom into these pieces from a global thumbnail view [113]. Saikat Mukherjee and I.V. Ramakrishnan have created a way to segment a Web page based on structural analysis of its content, in an approach they call Semantic Bookmarks [114].

Another direction of research is predicting a user's navigation path and personalizing pages to each particular user. Proteus, by Corin Anderson, Pedro Domingos and Daniel Weld, is a system that automatically customizes and adapts Web content for each mobile user [115]. For instance, it can predict the user's navigation path and display it in bold face, or highlight content that it believes the user is likely to find interesting [115].

Adeo's approach of improving mobile browsing by allowing the user to playback pre-recorded procedures is most similar to WebViews [116] by Juliana Freire, Bharat Kumar and Daniel Lieuwen. WebViews is based on their earlier work on WebVCR [97], and allows users to record an interaction with a Web site, and then access these services through different types of interfaces, such as wireless devices or voice interfaces.

How Adeo differs from WebViews is its inclusion of semantic search. By generalizing the user's input and automatically selecting relevant recordings, Adeo is able to streamline the user experience even further.

4.6 Related Semantic Web Research

This section discusses how the research described in this thesis fits into the broader vision of the Semantic Web.

Recently, the majority of Semantic Web research has focused on defining standards like RDF (Resource Description Framework) and OWL (Web Ontology Language), along with focusing on topics like rules, query languages, logic, and trust. The end user experience of the Semantic Web is rarely discussed. This has left many people wondering what exactly a Semantic Web browser looks like. Research on the end-user experience of the Semantic Web can be classified into two categories: "*Semantic Web*" browsers and *Semantic "Web browsers"* [117].

4.6.1 "Semantic Web" Browsers

Haystack

Tim Berners-Lee describes the user interface for the Semantic Web saying: "The Semantic Web architecture does not involve HTML browsers as we know them. There is a new breed of generic Semantic Web browser, but they are more like unconstrained database viewing applications than hypertext browsers" [25].

The Haystack project at MIT [117, 118], is a universal information client designed to view Semantic Web information. Unlike HTML, which encompasses data with presentation, RDF contains only data, without any layer of presentation. Haystack allows users to view data encoded in RDF and directly control how it is presented. Like the Data Detectors discussed in the earlier section of work related to Miro, Haystack leverages annotations that declare which operations can be performed on particular types of data. Data Detection is ubiquitous throughout the Haystack interface: anything can be right clicked on to display its operations. Haystack follows the design

principles of storing a user's information all in one place, and allowing the user to focus on information, instead of which application they are using. Since the purpose of Haystack is to allow users to manage information in the ways that make the most sense to them. This makes Haystack's interface more reminiscent of Personal Information Management applications like Outlook, than a traditional Web browser.

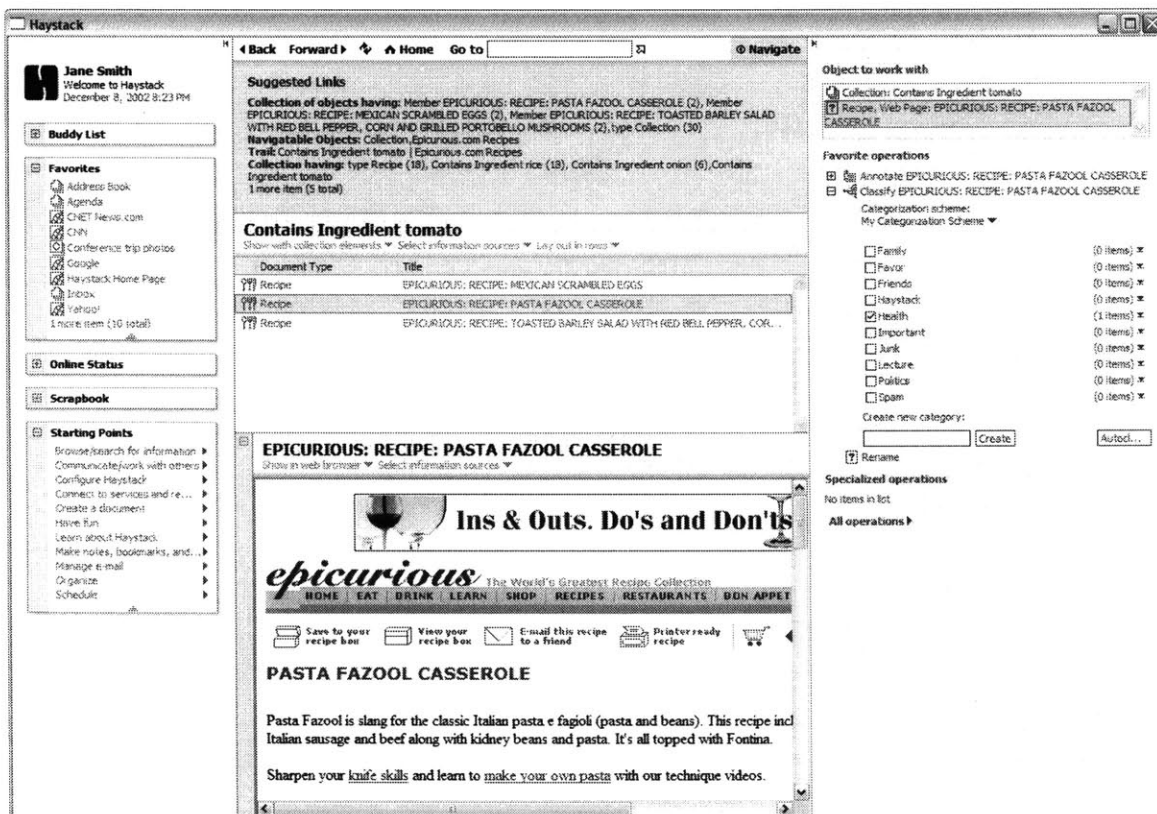


Figure 4-22 Using Haystack to categorize a recipe

4.6.2 Semantic "Web Browsers"

Tim Berners-Lee describes the evolution of current Web browsers saying: "A simple example of a browser that would be optimized for the Semantic Web would allow one to "view data" that might be associated with any of these services and provide the means for save, reuse and integrating this in a variety of other ways with other applications" [25].

While Miro does not rely on semantic markup in a Web page in order to locate data and associate it with other applications, in terms of functionality, Miro fits into this class of applications. Three additional examples of *Semantic "Web Browsers"* are Semantic Browsing, Piggy Bank, and Magpie.

Semantic Browsing

Semantic Browsing by Alexander Faaborg and Carl Lagoze allows users to author a layer of semantic metadata on top of the Web (with the Web Annotation Pane) and then leverage this semantic metadata to provide the user with relevant information and tasks (with the Web Task Pane) [19, 20].

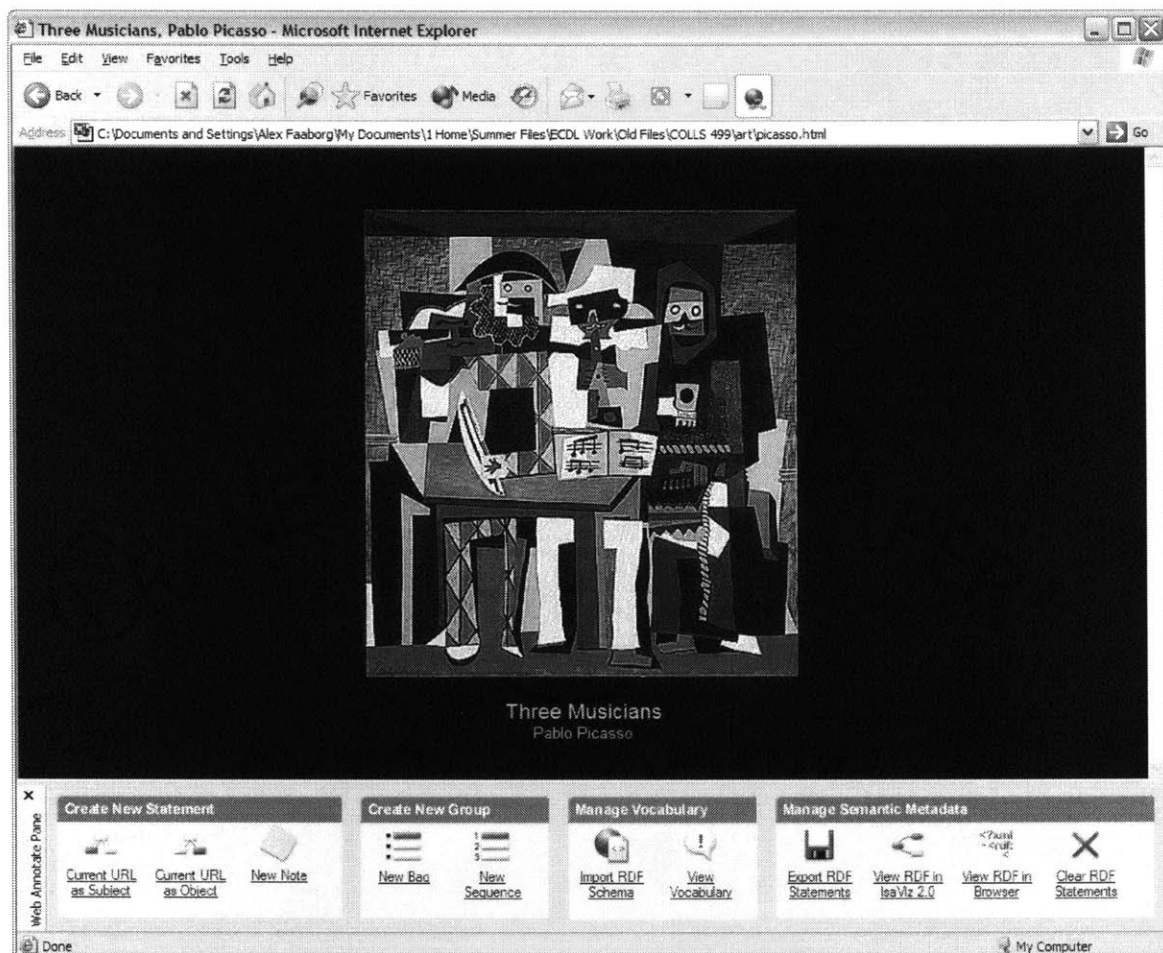


Figure 4-23 Semantic Browsing, the Web Annotation Pane

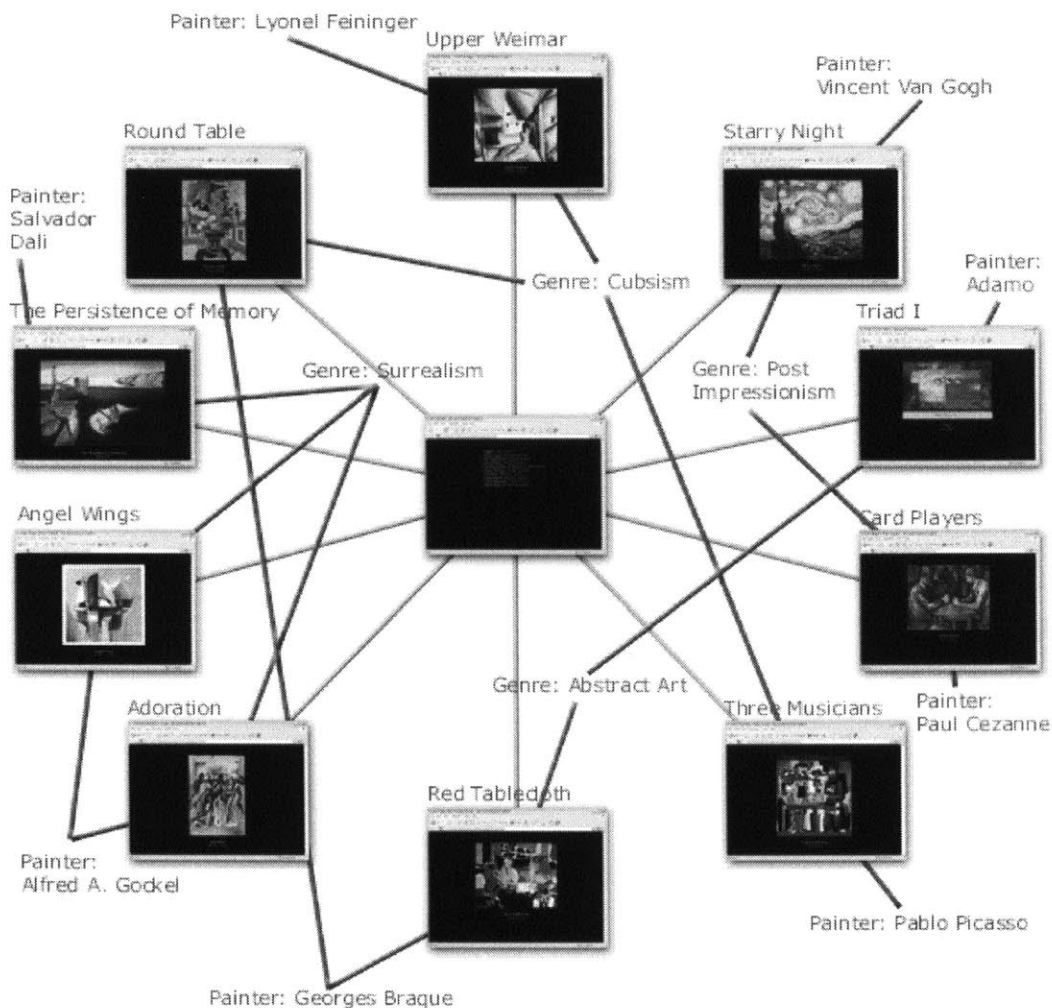


Figure 4-24 Defining a layer of semantic metadata on top of the existing Web

This system has a number of limitations. The Web Annotation Pane directly exposes users to RDF, making the tool only useful for technical users. While the related information presented by the Web Task Pane is useful, the actual *tasks* it presents the user with are all simply extensions of basic Web browser functionality that expand beyond the current page. These include various types of navigation beyond “Back” and “Forward,” printing a sequence of pages, and downloading a set of images. The tasks provided by Miro (like ordering food from the user’s personal grocery store) are both personalized, and based around the user’s high-level goals.

Piggy Bank

Piggy Bank, by David Huynh, Stefano Mazzocchi and David Karger, is a tool integrated into Firefox that lets Web users extract individual information items from within Web pages and save them in a Semantic Web format [119, 120]. The interface of Piggy Bank is similar to Tim Berners-Lee's idea of adding a "view data" button to Web browsers:

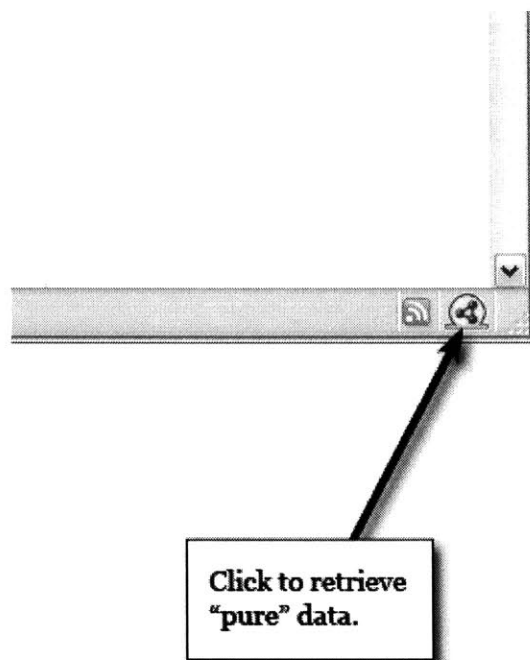


Figure 4-25 Piggy Bank's Data Coin

The *Data Coin* appears at the bottom of the Firefox browser, next to the *Live Bookmark* button when the site contains RDF metadata, or if Piggy Bank can construct RDF metadata using a custom screen scraper. Users can also annotate pieces of Web pages with RDF. For example, "one might tag a number of dessert recipes with 'durian,' then tag the 'durian' itself with 'fruit'" [120].

While Piggy Bank provides a framework for users to collect, organize and share information, detected data is not associated with specific actions, like

associating foods with a grocery store. The creators of Piggy Bank describe the system as “bookmarks on steroids” [121].

Another limitation of Piggy Bank is its reliance on semantic metadata or custom-built screen scrapers to understand information. The authors write:

Individuals wishing to share structured information through the Web must think in terms of a substantial publication process in which their information must be carefully organized and formatted for reading and browsing by others. While Web logs, or blogs, enable lightweight authoring and have become tremendously popular, they support only unstructured content. As an example of their limitation, one cannot blog a list of recipes and support rich browsing experience based on the contained ingredients [120].

Because Miro uses ConceptNet and TAP, separate knowledge bases of semantic information, to infer semantics in unstructured text, individuals writing blogs can continue their lightweight authoring. They do not need to worry about a “substantial publication process,” or making sure that their content is “carefully organized and formatted.” As demonstrated in Chapter 2, Miro is still able to provide a “rich browsing experience on the contained ingredients.”

Magpie

Magpie by Martin Dzbor, John Domingue and Enrico Motta, automatically associates an ontology based semantic layer to Web resources, allowing relevant services to be invoked within a standard Web browser [122, 123]. Like Miro, Magpie uses a semantic knowledge base to activate information in unstructured HTML documents. However, unlike Miro which relies on broad but shallow knowledge bases of common sense information, Magpie requires that users import hand crafted ontologies for specific domains. The authors write: “A pre-condition of a successful parsing within the Magpie browser

extension is the definition or download of an ontology-derived lexicon from an appropriate service provider” [123]. For instance, here the user is selecting which ontology they would like to use to parse Web pages, with the menu on the right:

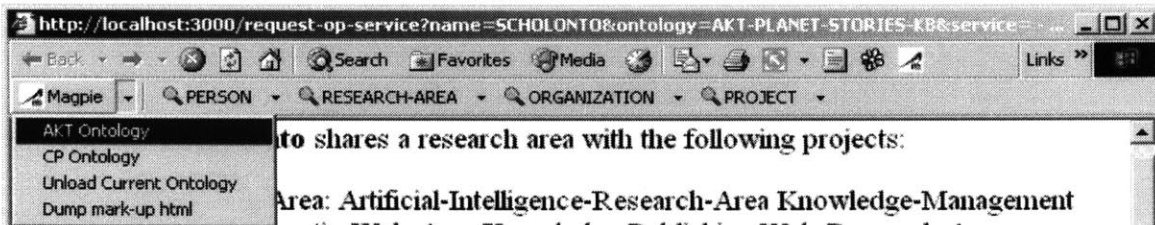


Figure 4-26 Selecting an ontology with Magpie

Like Miro, Magpie acts as a Data Detector. When the user right clicks on an activated concept, they receive a contextual menu of actions that can be performed on that object, called the Semantic Services menu:



Figure 4-27 Magpie’s Semantic Services menu

4.6.3 Usability of Semantic Web Browsers

One similarity between Magpie, Piggy Bank, the Web Annotation Pane, and Haystack is that in all of these systems, end users are exposed to implementation-level Semantic Web concepts. Magpie requires users to download, import, and then select domain specific ontologies. The Web Annotation Pane assumes the user is completely familiar with RDF. Depending on the type of Semantic Web information being displayed, Haystack may show the user URIs:

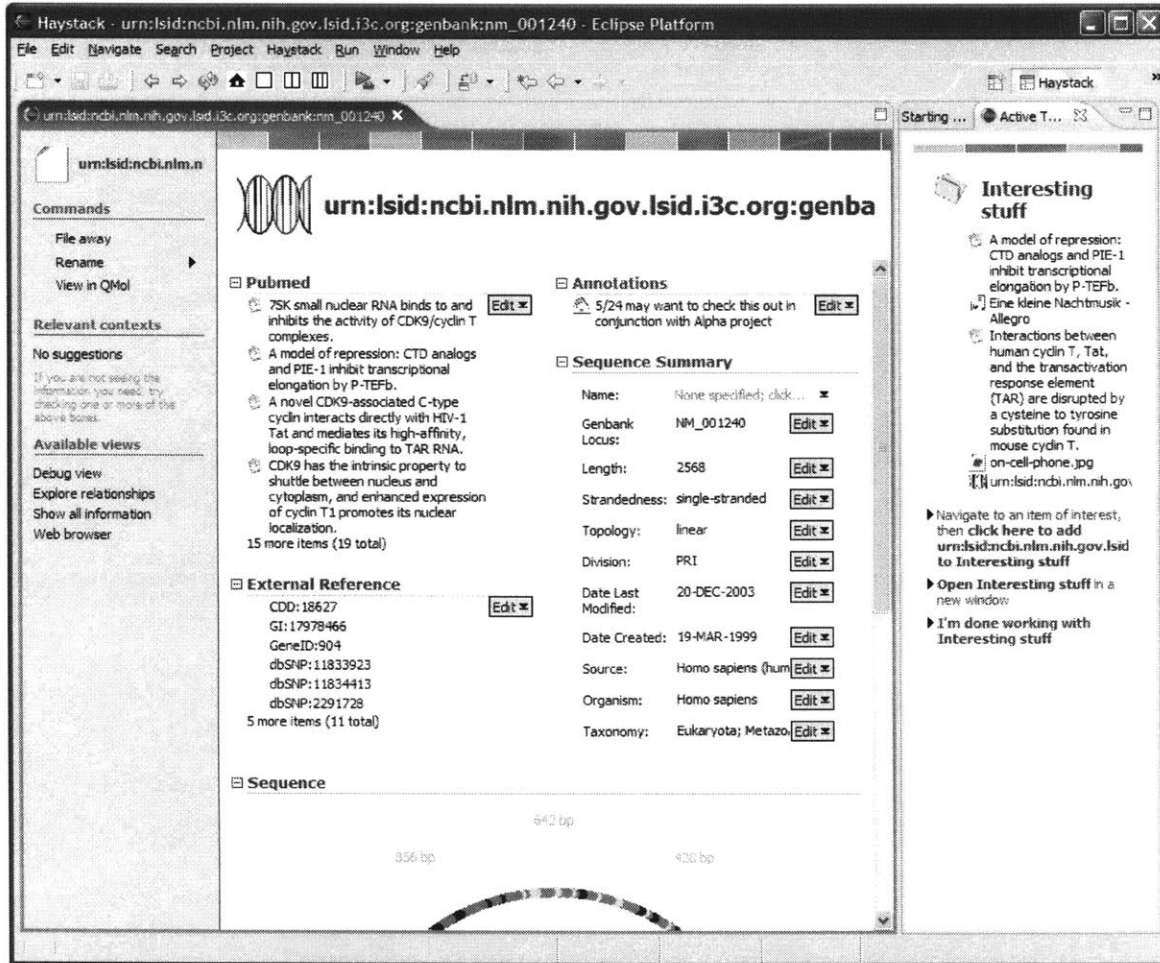


Figure 4-28 URIs in Haystack

Of all the Semantic Web browsers mentioned in this section, Piggy Bank has the best usability. Users are able to create RDF statements using a keyword tagging interface similar to Flickr [124] and del.icio.us [125], which they describe as the “first step toward full-fledged user-friendly RDF editing” [120]. Ironically, one of the main icons used in Piggy Bank is an image of an RDF triple:

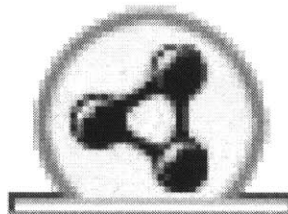


Figure 4-29 RDF triple icon in Piggy Bank

Creo, Miro and Adeo have been designed for novice end users who have no knowledge of the Semantic Web. The following chapter discusses a series of evaluations conducted to refine Creo's design, and a final evaluation to assess how the software improves an average user's effectiveness at completing a task.

Chapter 5

Evaluation

5.1 Introduction

Over the past year, a number of evaluations have been conducted to test both the usability and effectiveness of the software developed for this thesis. Section 5.2 describes four heuristic evaluations that were conducted during the design phase to analyze and improve the user interface of the software. Section 5.3 describes a final comprehensive study that was conducted to evaluate the software's overall effectiveness, using both quantitative and qualitative measures. This chapter concludes in with a summary of findings, and suggestions for further evaluations.

5.2 Evaluations of the User Interface

The software developed for this thesis has been designed to proactively match users' goals to the context of the information they are viewing, automate repetitive actions, and make users' digital lives easier. However, the software would not be able to meet these objectives if it was difficult to use.

In particular, the user interface design of Creo was considered critical to the success of the overall system. Programming by Example systems often extol the virtues of enabling novice users to train their computer, but then end up being simply GUIs for Computer Scientists ([92]). We wanted Creo to be

accessible to average end-users. Training an agent to interact with the Web can be a potentially complicated process, so early in the design phase, Creo was identified as the highest risk user interface. Because of this, Creo was designed using an iterative process, and each version of its user interface was evaluated against real users.

Over the past year, the user interface of Creo has been redesigned and evaluated four times. The first three evaluations were each conducted on three or four users, as part of the course 6.831 User Interface Design and Implementation [126]. The fourth evaluation was conducted independently on a set of 34 users. The details of each user interface evaluation are described in the following sections.

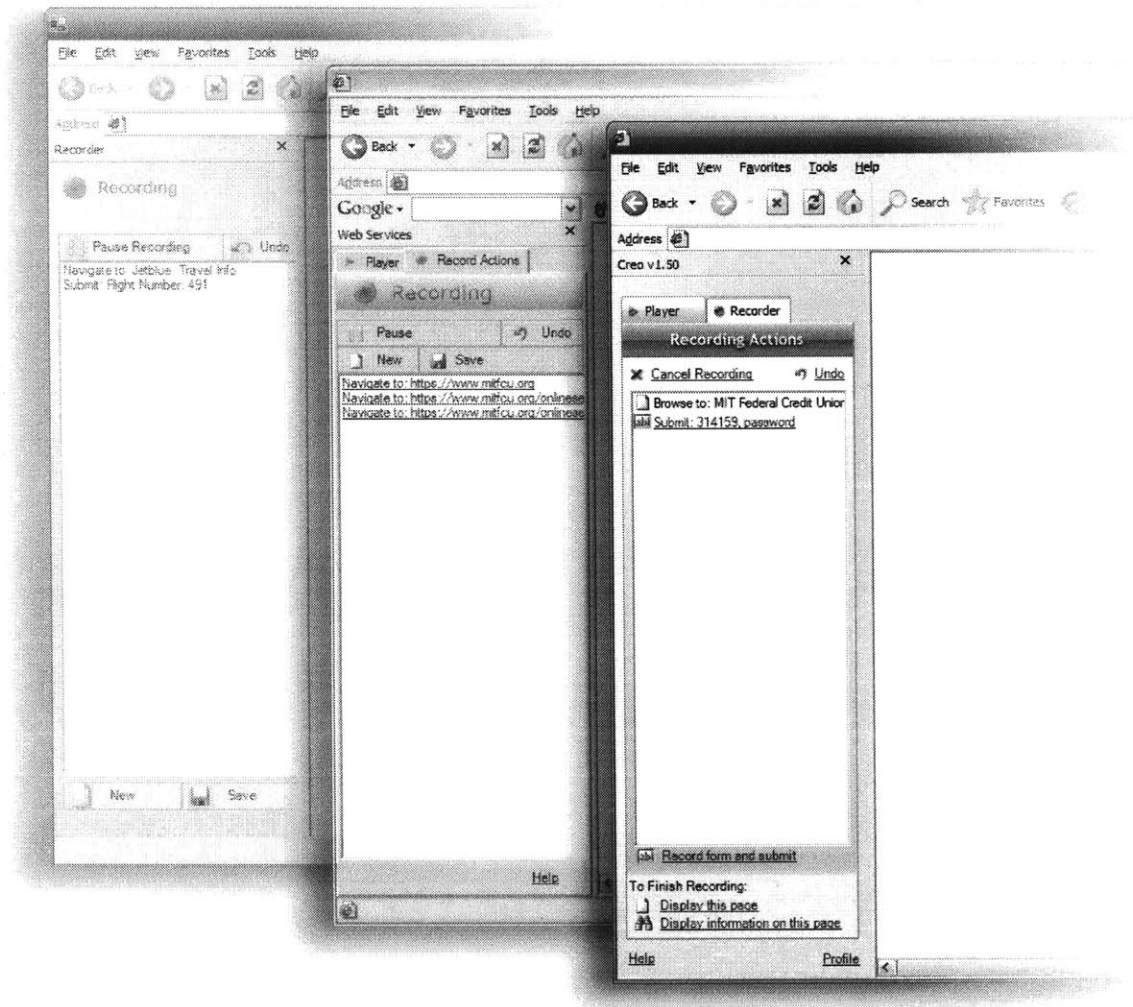


Figure 5-1 The evolution of Creo's design, Versions 2, 3 and 4

5.2.1 Evaluating Version 1

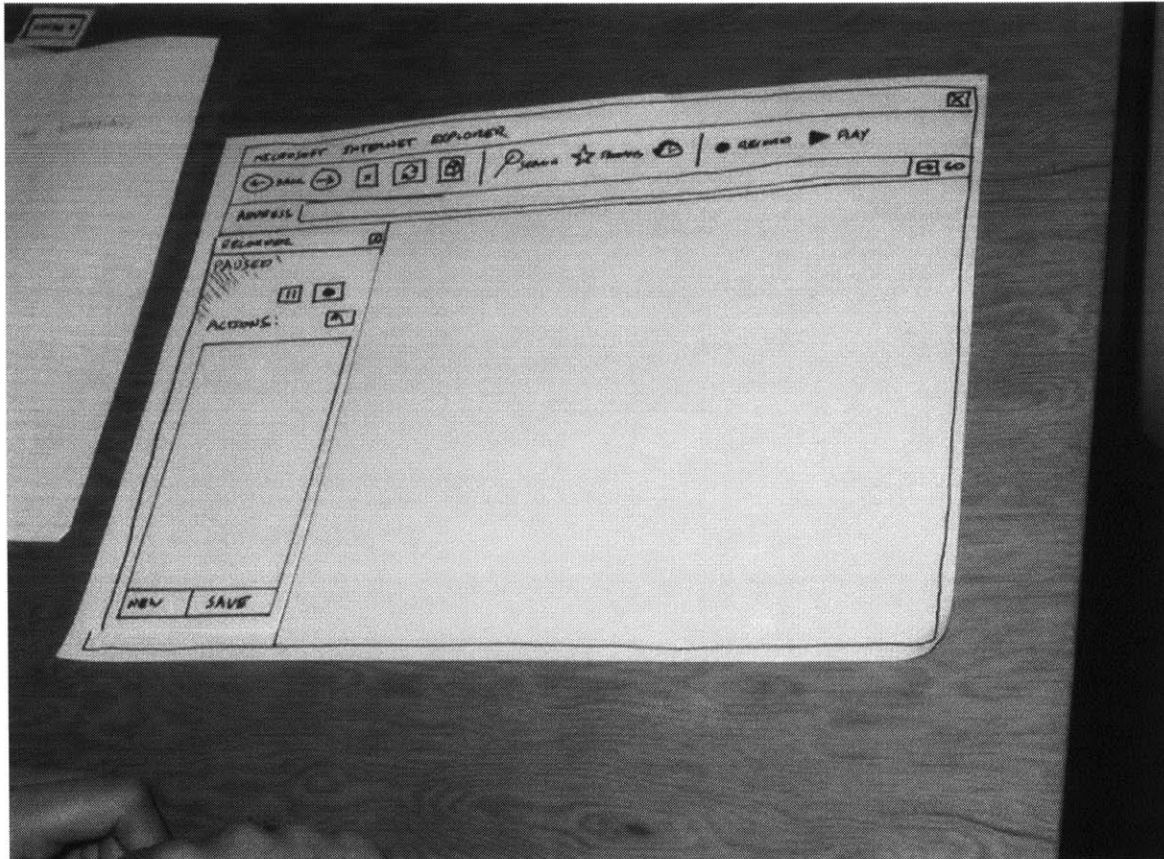


Figure 5-2 Creo Version 1: paper prototype

Users and Procedure

The first design of Creo was tested with three users in a paper prototyping session. Users were given a briefing of the purpose of the software application, and were then asked to interact with the application, thinking out loud. One experimenter made changes to the paper interface to mimic how the system would respond to actions, while another experimenter observed critical incidents and took notes on the user's statements. Each user was asked to complete three tasks:

Task 1:

- Create a script that automatically orders a pizza

Task 2:

- Play your script back to automate the process of ordering a pizza

Task 3:

- Create a script to order "orange juice"
- Generalize the script to order any kind of "grocery"

Changes to the Interface from the First Evaluation

A number of usability problems with the initial design quickly emerged during the paper prototyping session. Changes to the interface from user feedback included:

- Adding a confirmation option for playing recordings
- Adding feedback when playing back recordings, and when displaying that a recording has completed playing

In Task 3, users were asked to generalize the concept of "orange juice" to "grocery." To test how mistakes made during automatic generalization affect users' performance on the task, several errors were simulated. Users were presented with this dialog box:

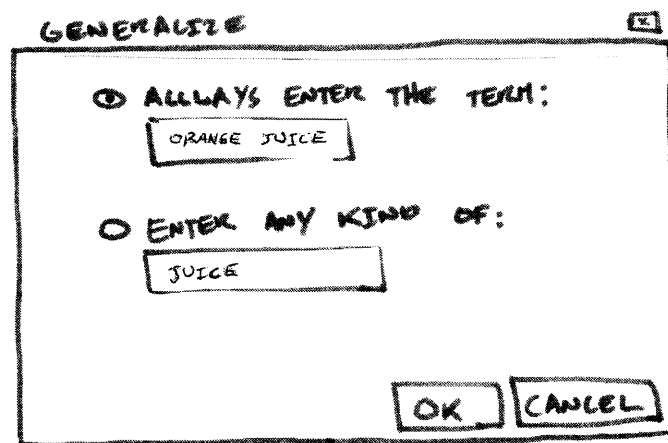


Figure 5-3 Early generalization interface

A number of users had difficulty completing the task when the interface produced a single incorrect generalization. This interface was later changed to display a list of check boxes showing all of the generalizations of the

concept. This allowed users to simply check and uncheck items, and reduced the level of abstract thinking.

5.2.2 Evaluating Version 2

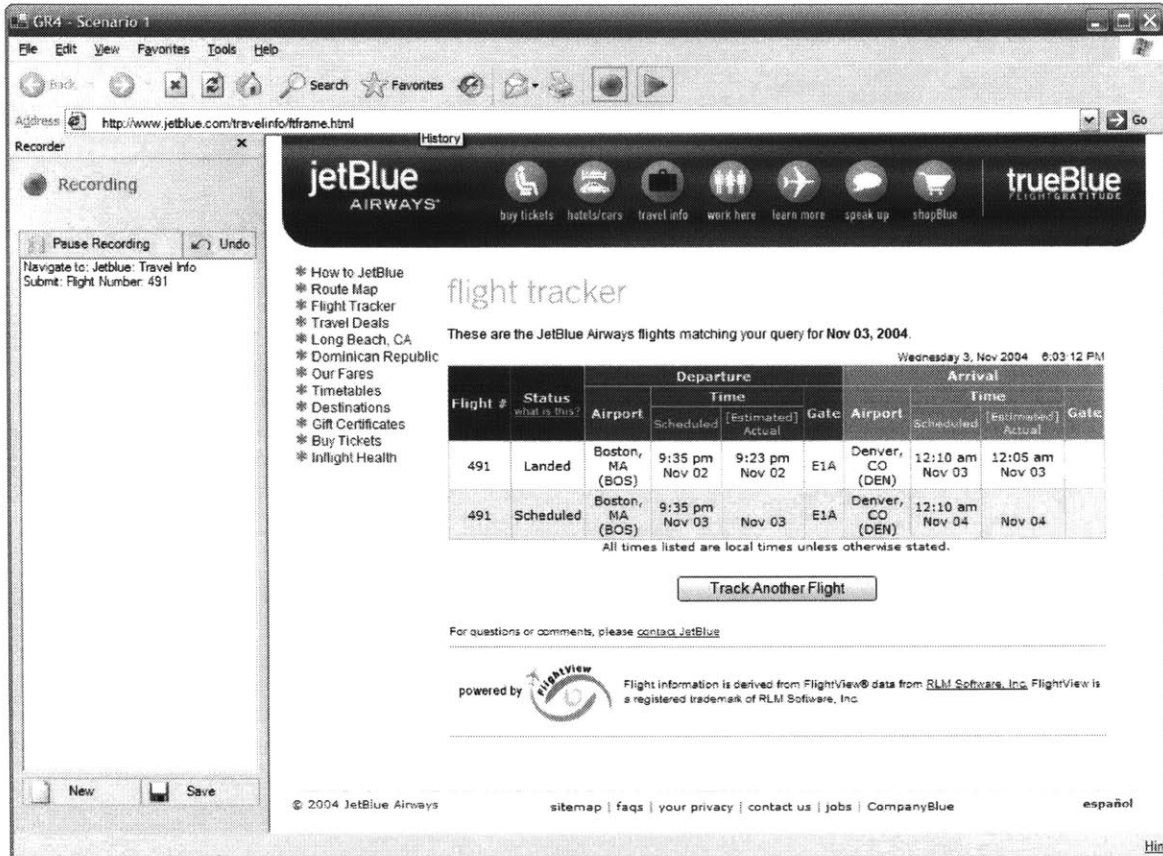


Figure 5-4 Creo Version 2: computer prototype

Users and Procedure

The second iteration of Creo’s design was tested against four users with a computer prototype. Only the front end of the software was implemented, and the scenarios were entirely simulated. Users were asked to complete the following three tasks:

Task 1:

- Create a process that tracks a flight
- Play the process back to see that it works

Task 2:

- Create a script that orders a pizza
- Play the process back to see that it works

Task 3:

- Create a process to order "orange juice"
- Generalize the process to order any kind of "grocery"
- Play the process back to see that it works

Each user independently conducted a heuristic evaluation of the software, and wrote a list of usability problems they found while interacting with the user interface. Their heuristic evaluation was based on Jacob Nielsen's ten usability heuristics [90]. Each user was asked to provide at least 20 useful comments, and to estimate each problem's severity (cosmetic, minor, major, or catastrophic).

Changes to the Interface from the Second Evaluation

These four reports were compiled together and resulted in several changes, including:

- Integrating the *Player* and *Recorder* into a tabbed interface in the same Explorer bar, to solve navigational problems
- Adding more detail to the confirmation window when playing a recording
- Moving the *New* and *Save* buttons to the top of the window so they are easier to find
- Visual changes to the way the application represents its current mode (paused or recording)

5.2.3 Evaluating Version 3

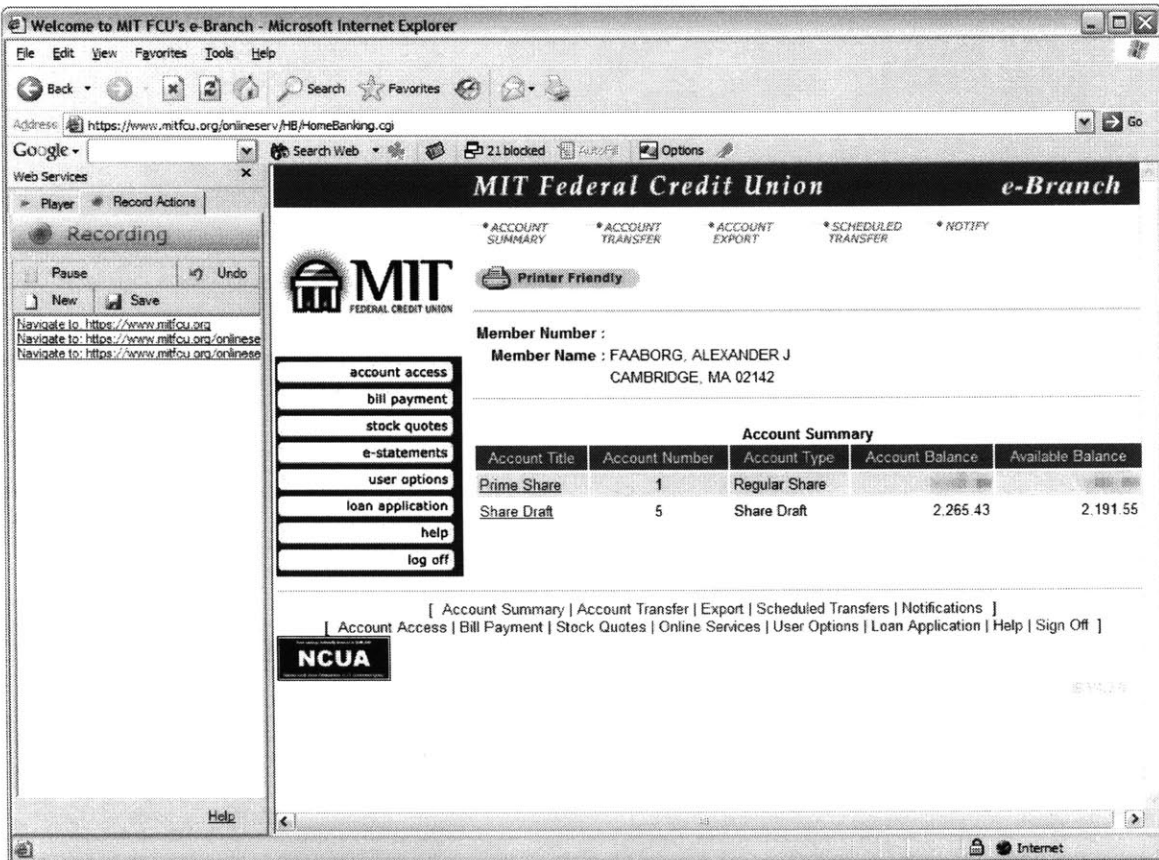


Figure 5-5 Creo Version 3: functional prototype

Users and Procedure

The third version of Creo's interface was evaluated in a usability test with three subjects. All of the users were informed that they would be helping the experimenter find problems with the interface, and that any mistakes they made were the fault of designer, and not their fault. Users were instructed to think out loud. They were also told they could end the evaluation at any time, that their personal information would remain private, and that the experimenter could answer any questions they had about the software at the end of the evaluation. After observing the users complete their tasks, the experimenter debriefed them, asking them to reflect on problems they had with understanding the interface. Despite earlier requests for the users to think out loud when they were completing the tasks, the experimenter received a large amount of new and useful information in the debriefing

when asking the question, “What specifically did you dislike with the user interface, and why did you find it confusing?”

All three users were familiar with the Web, and all three users reported that they interact with Web forms nearly every day. One of the three users had created a Web page, but none of the users had created a Web page with form elements. None of the users knew how to program, and only one user was remotely familiar with the macro recording features in office and graphics applications. All three users were familiar with Windows and Internet Explorer, but one was currently a Mac user. The users were not very different demographically: all three users were female, aged 20-25.

Users were asked to complete the following tasks:

1. Record the process of tracking Flight 490 from New York to Denver
2. Play the process
3. Modify the process so that it can track any flight, not just Flight 490
4. Play the new process to track a different flight

Changes to the Interface from the Third Evaluation

Changes to the interface from this iteration of usability testing include:

- Adding a confirmation when switching away from the *Recorder* tab while recording
- Removing the *Pause* button
- Removing the *New* button
- Replacing all buttons with hyperlinks, to make the interface less visually complex
- Changing the names of the modes from *Paused* and *Recording*, to *Ready*, *Recording Actions*, and *Recording Complete*
- Removing buttons that are not relevant given the current mode. For instance, the only button in the *Ready* mode is *Start Recording*
- Making changes in the application’s visual appearance more drastic when the user switches between modes

Many of these changes can be seen by comparing the top of the *Recorder* tab between Version 3 and Version 4 of the interface:

State	Version 3	Version 4
Start		
Recording		
Complete		

Table 5-1 Changes to the *Recorder* interface between Version 3 and Version

4

The generalization interface was also redesigned between Versions 3 and 4. In Version 4, form elements listed on the left side of the window are based on the text the user entered and not the form element's field name. Version 4 also contains additional options for scanning pages for related concepts and sharing the recording.

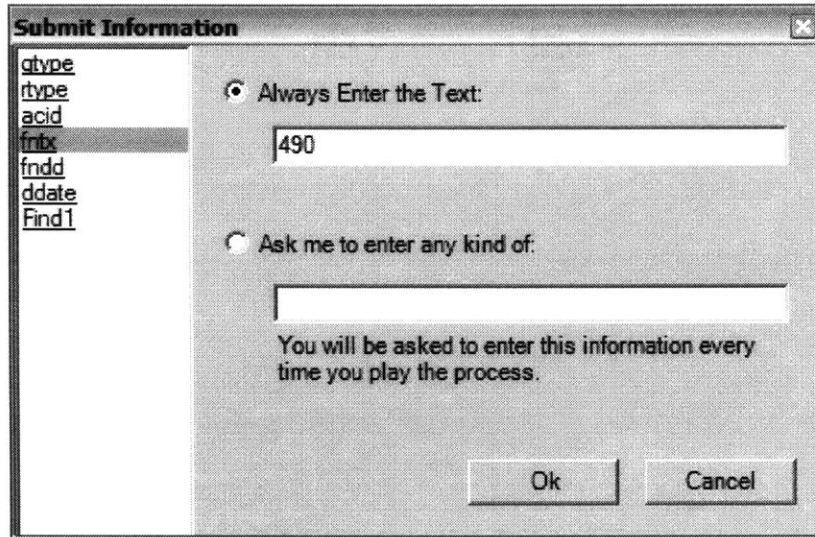


Figure 5-6 The generalization interface of Version 3

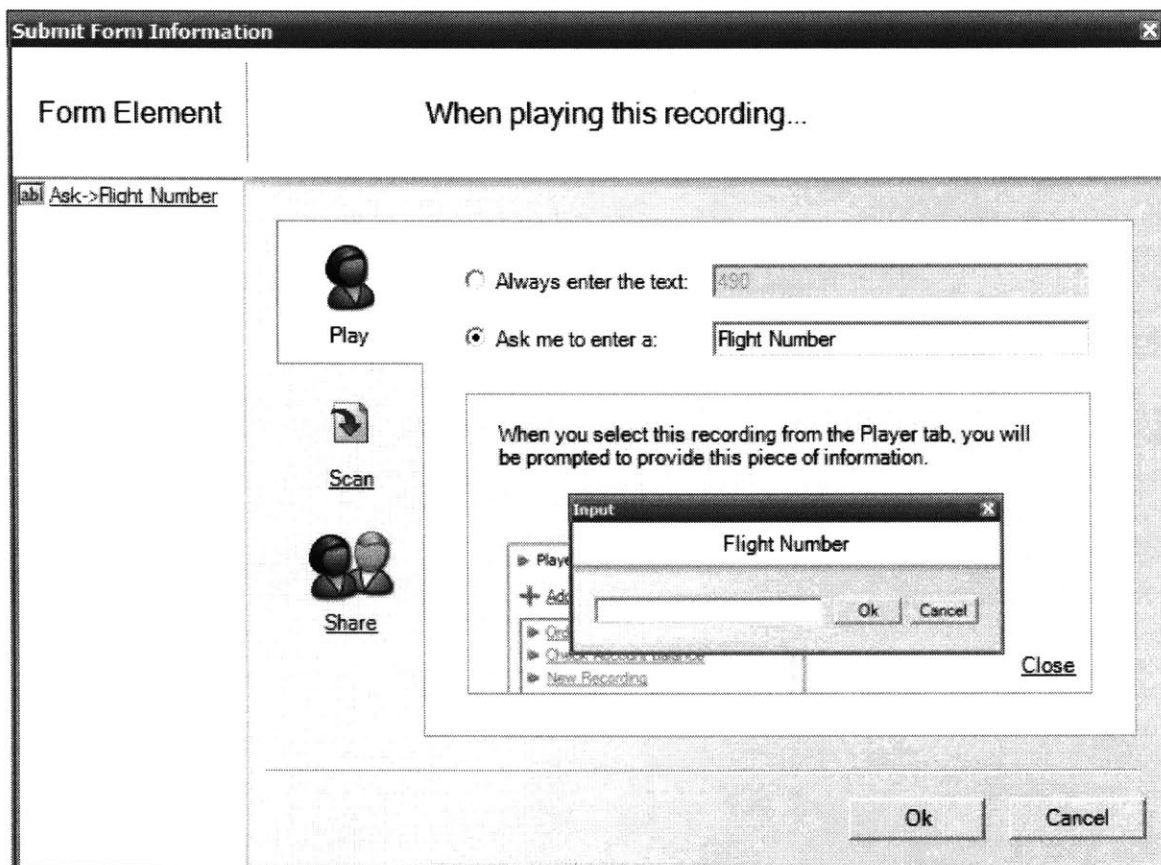


Figure 5-7 The generalization interface of Version 4, with contextual help

5.2.4 Evaluating Version 4

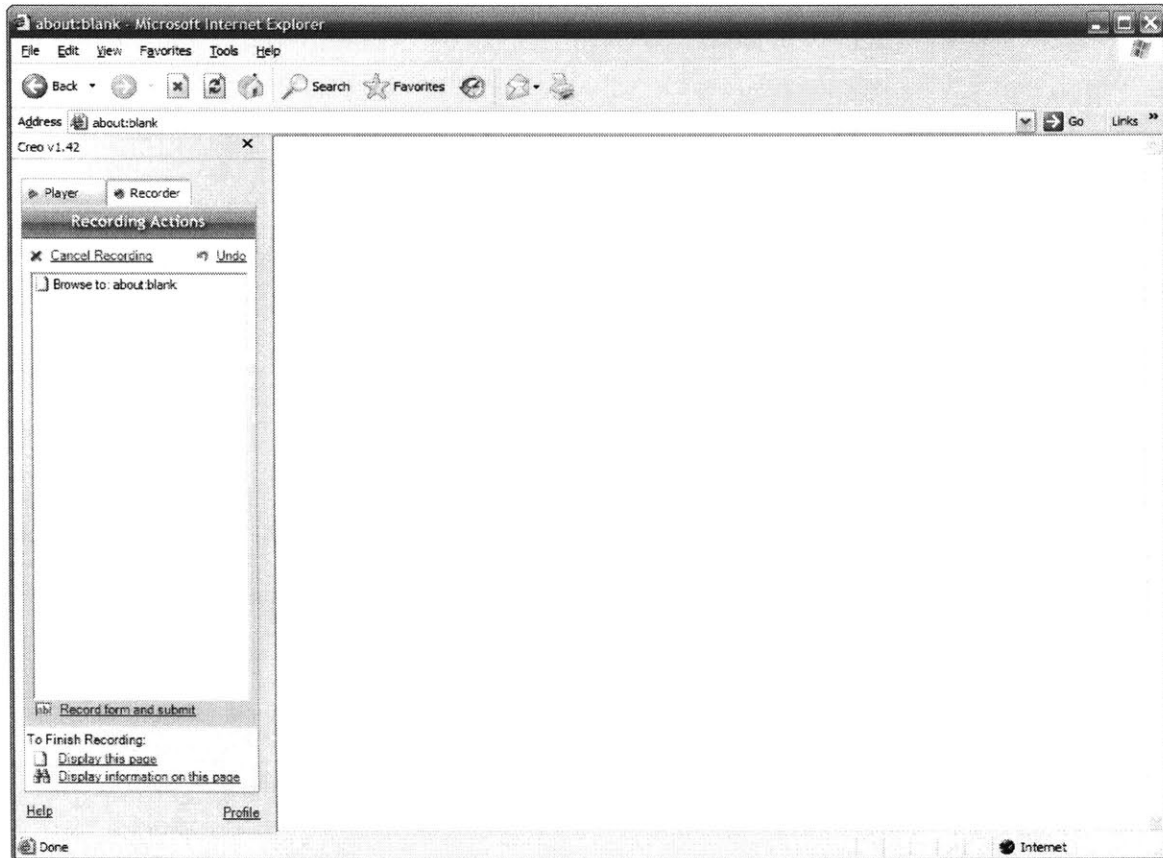


Figure 5-8 Creo Version 4: final interface

The user interface of the fourth version of Creo was tested as part of a comprehensive study that was conducted to evaluate the software's overall effectiveness, using both quantitative and qualitative measures. This study is described in detail in the following section.

5.3 Evaluating the Software's Overall Effectiveness

Each user interface evaluation of Creo identified usability problems with the current iteration of its design that were later solved in subsequent designs. The final evaluation took a different approach. Instead of attempting to

isolate specific problems with the user interface, the final evaluation was designed to determine the software's overall effectiveness, compared to using a traditional Web browser. To analyze the system as a whole, subjects completed a task using both Creo and Miro. While the experimenter did note specific usability problems, the purpose of the final evaluation was to (1) conclude if the overall system made users more effective at completing a task, and (2) to conclude if users understood the utility of the software, and if they would use software applications like Creo and Miro if they were included in their Web browser.

To analyze how Creo and Miro make users more effective compared to using a conventional Web browser, this evaluation focused on a single example of using Creo and Miro. We did not study the breadth tasks that Creo and Miro can perform for two reasons, (1) the ConceptNet and TAP knowledge bases are rapidly growing, and (2) the respective teams at MIT and Stanford responsible for the creation of these knowledge bases have already performed evaluations of their breadth [5-8, 12-16].

5.3.1 Experiment Hypothesis

To evaluate the overall effectiveness of Creo and Miro compared to a normal Web browser, we selected a user scenario that involved a lot of repetitive actions: ordering ingredients in a recipe.

Part 1: Evaluating the Effectiveness of Miro

In the first part of the experiment, subjects were asked to order 11 ingredients in a recipe for Blueberry Pudding Cake. The experimental group of subjects had access to the Miro toolbar, which could recognize common foods and automatically link them to the subject's grocery store. The control group of subjects completed the same task, but used Internet Explorer with Miro turned off. We hypothesized that the experimental group would be able to complete the task significantly faster than the control group.

Part 2: Evaluating the Effectiveness of Creo

In the second part of the experiment, all of the subjects were asked to create a recording with Creo that could order any type of food at a grocery store. Subjects completed this task after being shown an example of how Creo works. We hypothesized that subjects would be able to successfully complete this task in a trivial amount of time, and without having any significant problems understanding Creo’s user interface.

In addition to the quantitative measures recorded for these two tasks, subjects were also asked to provide qualitative feedback in a post-experiment questionnaire. We hypothesized that subjects would have a favorable opinion of Creo and Miro.

5.3.2 Experiment Design

Study Segment Times

At the beginning of each subject’s time slot, they were asked to fill out a consent form, and a questionnaire with demographic information. The subject then completed two tasks, which are described in the following sections. After completing these two tasks, the subject was asked to fill out a post experiment questionnaire, and was compensated \$10.

Study Segment	Time
Pre-experiment questionnaire and consent form	5 minutes
Part 1: Miro / normal browser	5 minutes
Part 2: Creo	5 minutes
Post-experiment questionnaire	5 minutes
Subject’s questions	0-10 minutes

Table 5-2 Study segment times

Study Procedure: Part 1

Each subject was asked to perform a search for each of the ingredients listed in the Blueberry Pudding Cake recipe at the online grocery store FreshDirect.com [127]. Subjects in the experimental group were asked to use the *Scan Page* feature of the Miro toolbar. Subjects in the control group were asked to complete the task however they naturally would (copying and pasting, retyping, using any type of keyboard shortcuts, etc.). While all of the subjects were aware that the time they took to complete the task was being recorded, each subject was specifically instructed to complete the task at a natural, normal pace, and not to treat the task like a race. Because we were interested in recording the time it took to transfer information between the recipe page and the grocery store site, subjects were asked to move onto the next ingredient after seeing the grocery store site had returned search results. Subjects were instructed not to read the entire list of results, and not to add any items to their shopping cart.

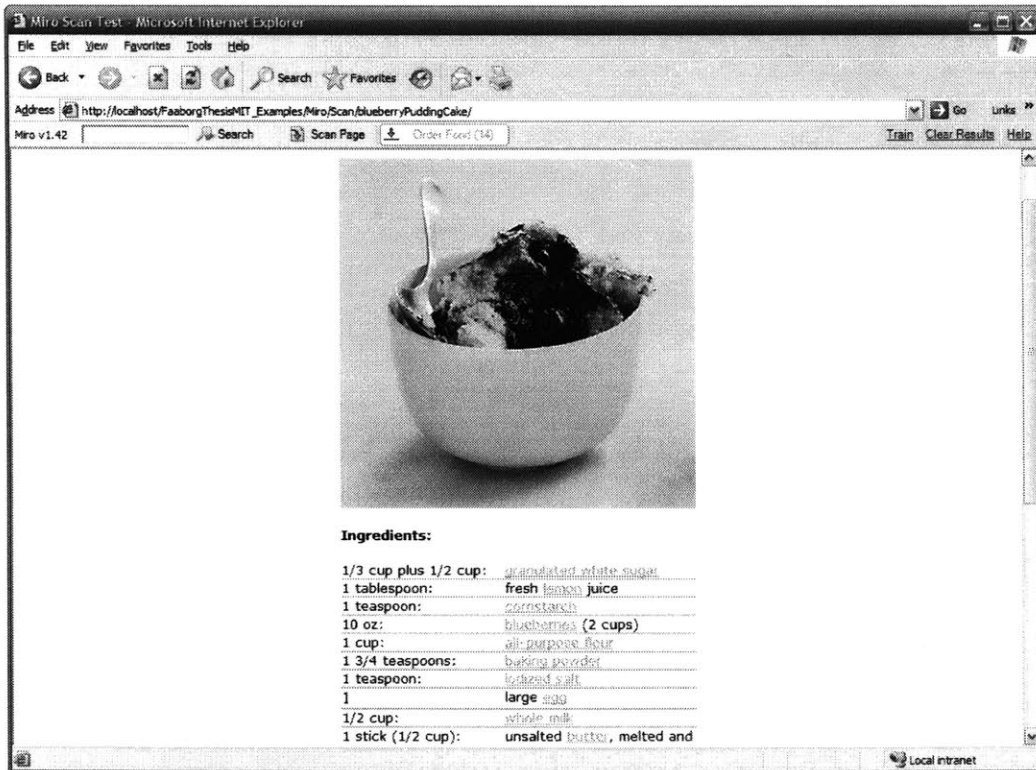


Figure 5-9 Experiment Part 1: experimental group, using the Miro toolbar

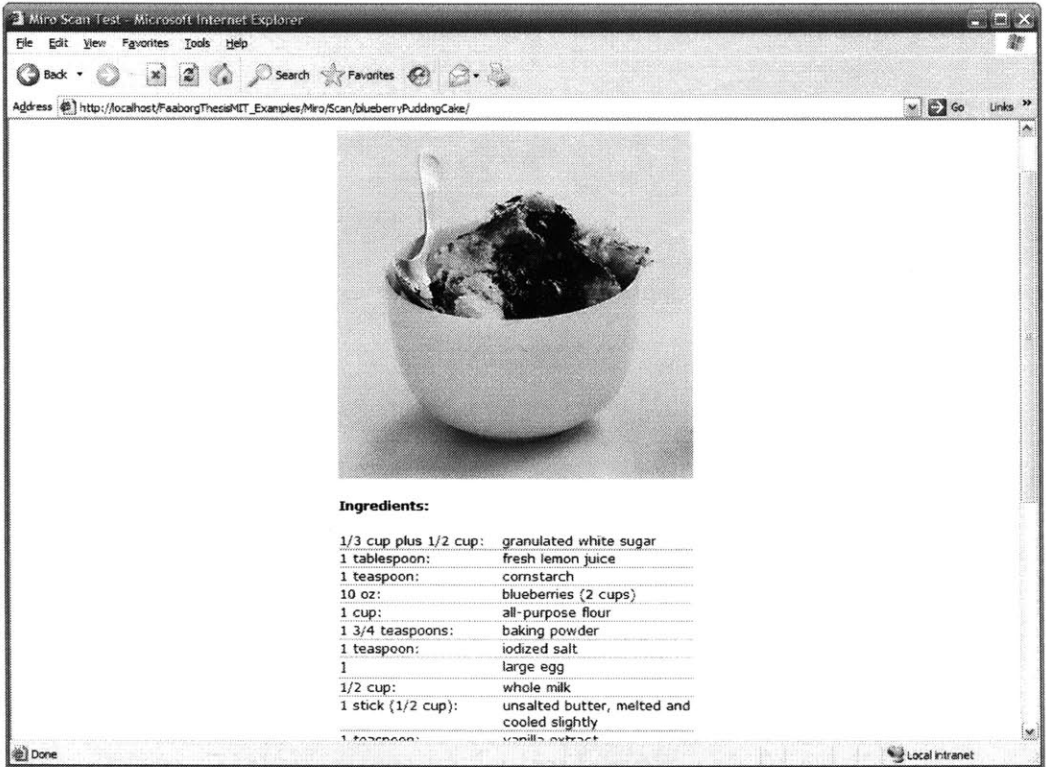


Figure 5-10 Experiment Part 1: control group, using a normal Web browser

Study Procedure: Part 2

In the second part of the experiment, all of the subjects were first shown how recordings created with Creo could be used by Miro. Subjects who were in the control group in Part 1 were shown the Miro toolbar after they completed Part 1. The experimenter showed the subjects how to use Creo with the example of creating a recording to look up any kind of movie at IMDB [128]. Subjects were shown an example of how to use Creo (as opposed to having to figure the interface out on their own, like the first three evaluations) because we were interested in capturing the time it would take an experienced user to complete the recording. Subjects were allowed to ask questions about Creo's interface and functionality before starting.

The steps involved in successfully completing Part 2 consisted of:

1. Starting a recording
2. Navigating to <http://www.freshdirect.com>

3. Entering an example food, either "diet coke" or any food the subject thought of
4. Ending the recording and giving it a name, such as "Order Food"

Similar to Part 1, the time it took subjects to complete the task was recorded. Subjects were asked to disregard the fact that they were being timed, and to complete the task at a normal pace.



Figure 5-11 Experiment Part 2: using Creo to associate foods with an online grocery store

Type and Number of Participants Involved

The evaluation was run with 34 subjects: 17 male and 17 female. In Part 1 of the evaluation, 17 people were in the experimental group, and 17 people were in the control group. The average age of the subjects was 29.3, with a

range of 19 to 58. Additional demographic information appears in Section 5.3.3.

Method of Recruitment

Subjects were recruited using a flier posted on bulletin boards on the MIT campus in the infinite corridor, student center and Stata Center. This flier appears in Appendix A.

Length of Participation Involved

The evaluation ran for 20 to 30 minutes. The time varied due to the number of questions the subject had about the software, after completing the evaluation. Some subjects had a large number of questions about how Creo and Miro worked, while other subjects were interested in receiving their compensation for participating in the study and leaving as quickly as possible.

Location of Research

The study took place in room 385 of the MIT Media Lab. Subjects used a 20" LCD monitor, standard keyboard, and optical mouse.

Procedures for Obtaining Informed Consent

Subjects were asked to sign a standard COUHES consent form for non-biomedical research. This form described the content of the study and provided the experimenter's contact information. A copy of the consent form can be found in Appendix B. In addition to the consent form, the experimenter also verbally instructed the subjects that they could leave the study at any time, and that they would still be compensated. The experimenter made sure the subjects understood this point before they signed the consent form.

Procedure to Ensure Confidentiality

Each subject was assigned a randomly generated 5-digit identification number. All recorded information in the study, including recorded times, questionnaires, and the experimenter's observations, reference the subject's identification number, and not the subject's personal information.

5.3.3 Experiment Results

In this section, the results of each segment of the study and the experimenter's observations are discussed.

Pre-Experiment Questionnaire

Demographic questions:

Question	Response
Age	Average: 29.3 Range: 19-58
Gender	Male: 17/34, 50% Female: 17/34, 50%

Table 5-3 Subject's demographic information

Web usage questions:

Question	Response
On average, how many hours do you spend using the Web each week?	Average: 23.2 Range: 5-60
On average, how many times per week do you interact with a form on the Web to complete an action? <i>[Note: This was a poorly worded question. The wide range of responses on this question is likely a result of subjects not knowing if search engines should count as a "form." It would appear that technical subjects counted any HTML form in their response, while non technical</i>	Average: 50.9 Range: 1-300

<i>subjects only include forms with many elements.]</i>	
Around how many different actions do you often repeat on the Web? Some examples of an "often repeated action" include: ordering groceries, checking the balance of your bank account, transferring money between two bank accounts, checking the current weather, sending money with PayPal, etc...	Average: 11.2 Range: 2-40

Table 5-4 Subject's Web usage

Which of the following things have you done on the Web?

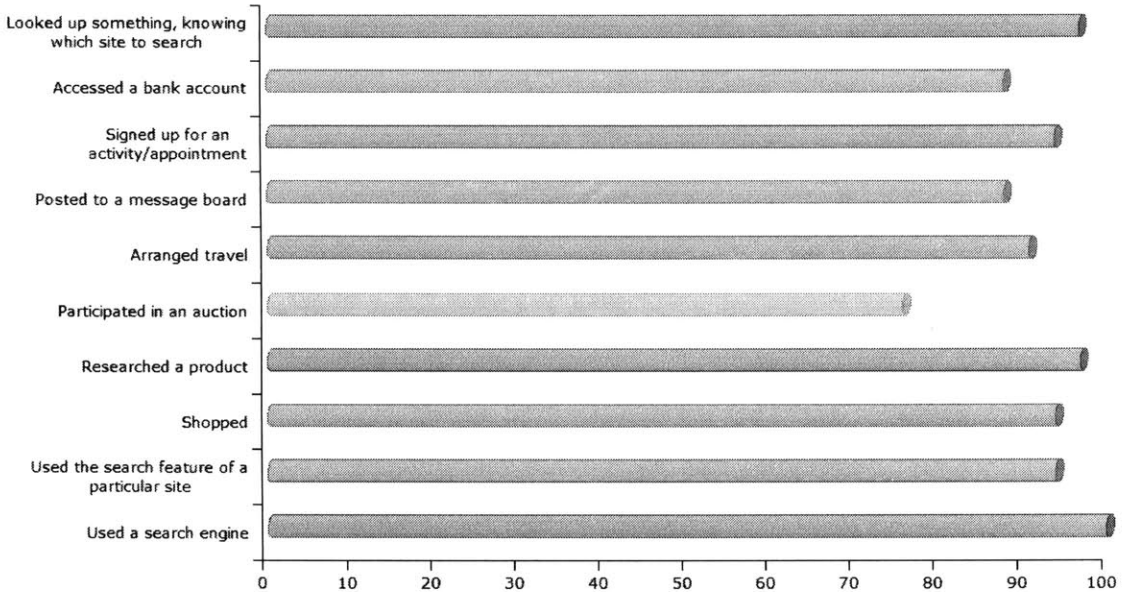


Figure 5-12 Subjects' Web usage

Technical experience questions:

Have you taken any classes involving programming? How many?

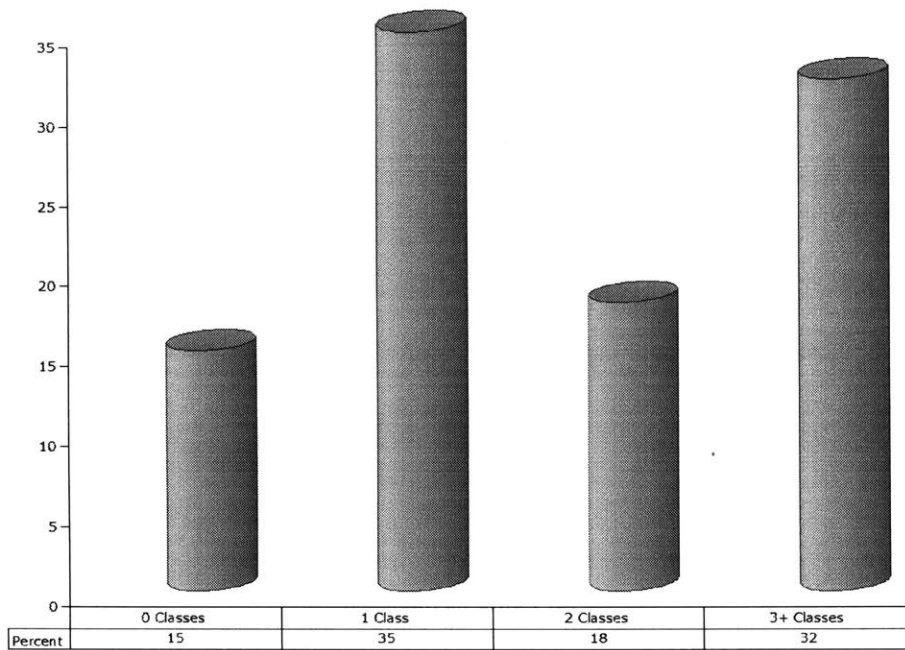


Figure 5-13 Subjects' programming background, classes

If you know how to program, how many years have you programmed?

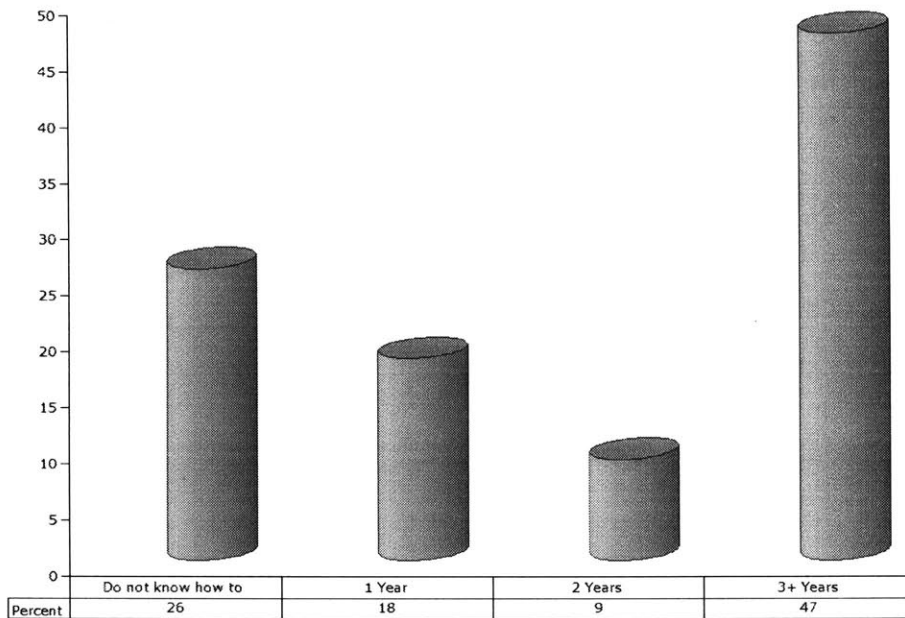


Figure 5-14 Subjects' programming background, years

Part 1: Time to Order Each Ingredient in the Recipe

In the first part of the study, subjects were asked to search for 11 foods in a recipe at a grocery store site. Half of the subjects were in the experimental group and used the Miro toolbar, and half of the subjects were the control group and used a normal Web browser.

Time to complete the task:

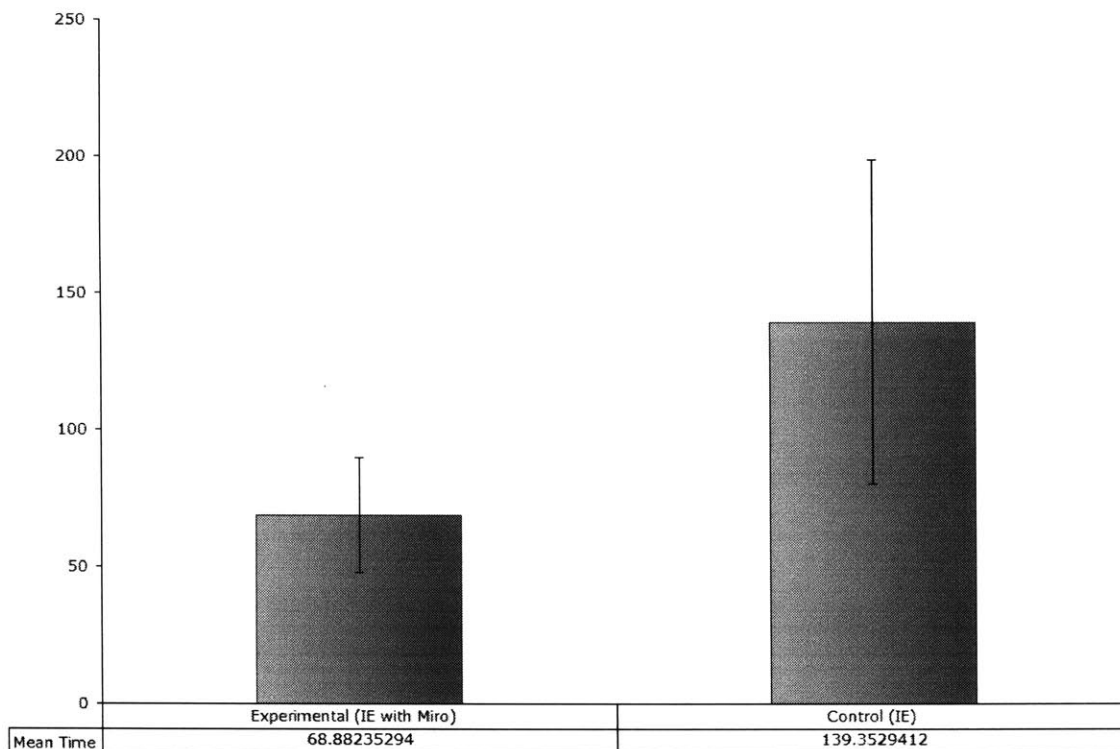


Figure 5-15 Mean time to complete Part 1

The experimental group completed the task with in average time of 68 seconds, with a standard deviation of 20 seconds. The control group completed the task in an average time of 139 seconds with a standard deviation of 58 seconds. These results are statistically significant ($p < .001$).

Part 2: Time to Create the "Order Food" Procedure

In the second part of the experiment, subjects were asked to use Creo to create a recording that could look up any type of food at a grocery store Web site. Subjects completed the task after they understood how to use Creo, because we were interested in capturing the time it took an experienced user to create a simple recording. The experimental and control groups from Part 1 completed the same task in Part 2.

Time to complete the task:

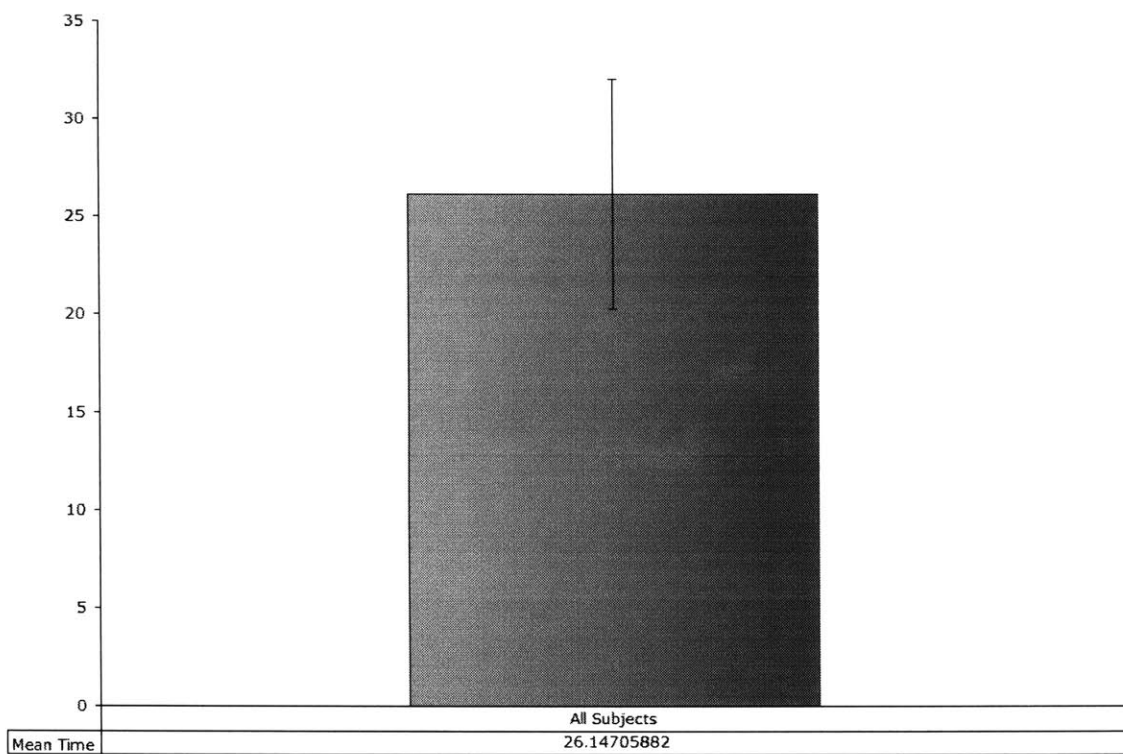


Figure 5-16 Mean time to complete Part 2

The subjects completed the task in 26 seconds, with a standard deviation of 5 seconds.

Post-Experiment Questionnaire

This section contains the results from the questionnaire that subjects were asked to fill out after completing Parts 1 and 2.

Part 1, Miro:

I felt that the user interface for selecting actions associated with the page I was viewing was easy to use:

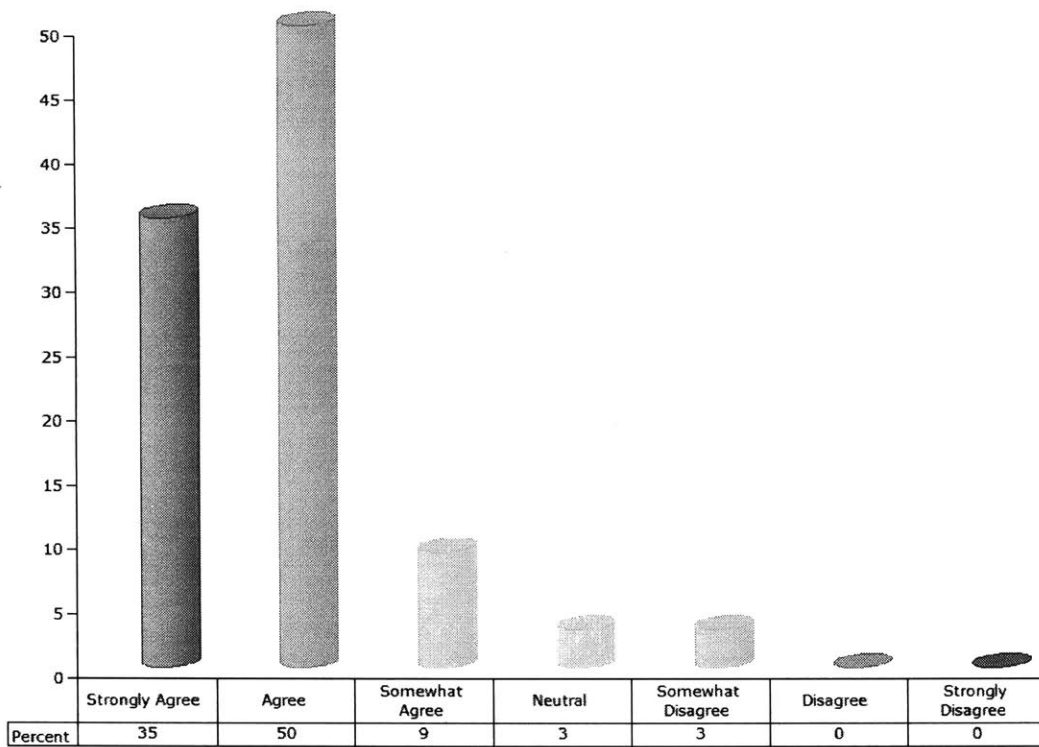


Figure 5-17 Did the subjects find Miro easy to use?

If my Web Browser included a feature to select actions associated with the page I was viewing, I would use it:

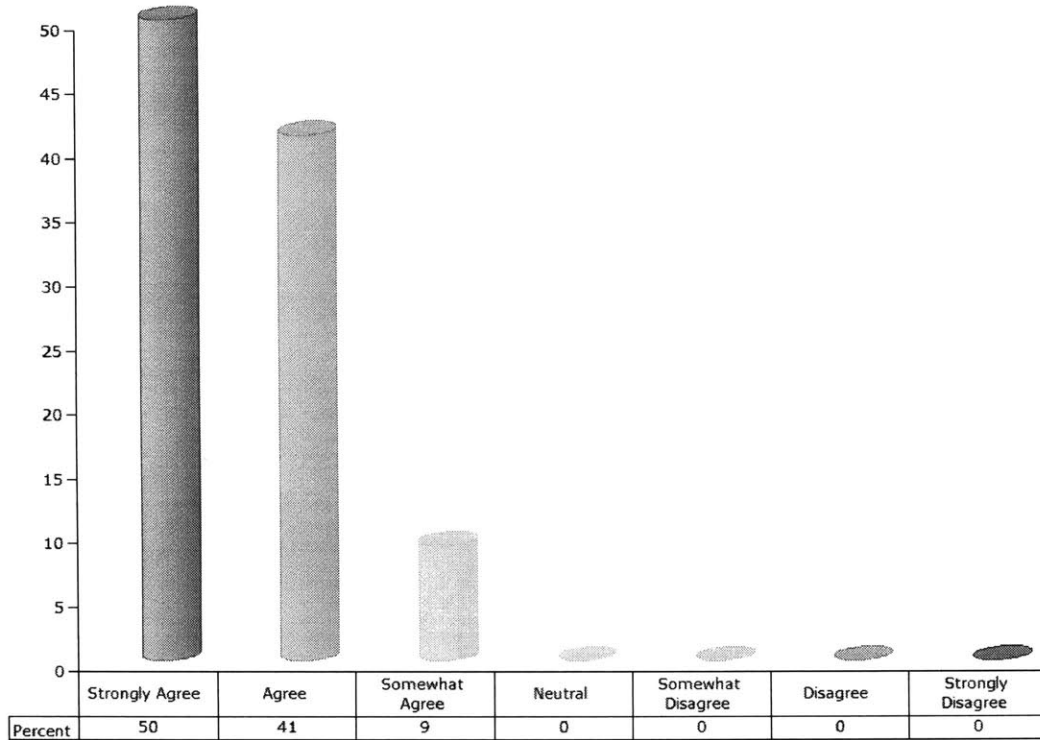


Figure 5-18 Would the subjects use Miro?

Part 2, Creo:

I felt that the user interface for recording actions on the Web was easy to use:

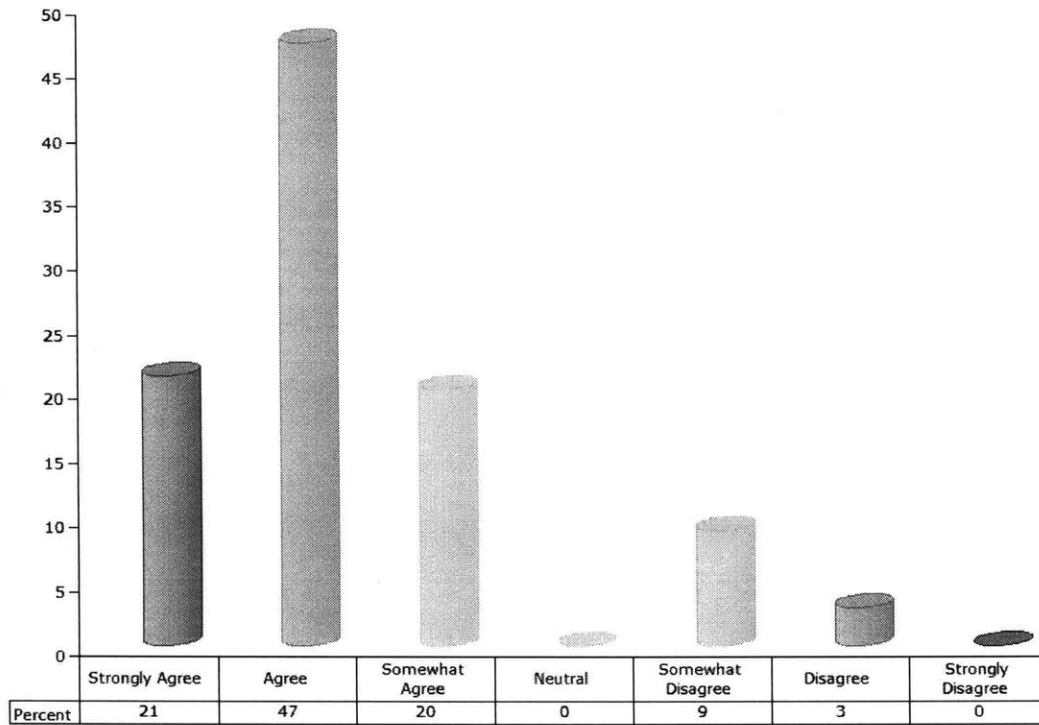


Figure 5-19 Did the subjects find Creo easy to use?

If my Web Browser included a feature to record actions, I would use it:

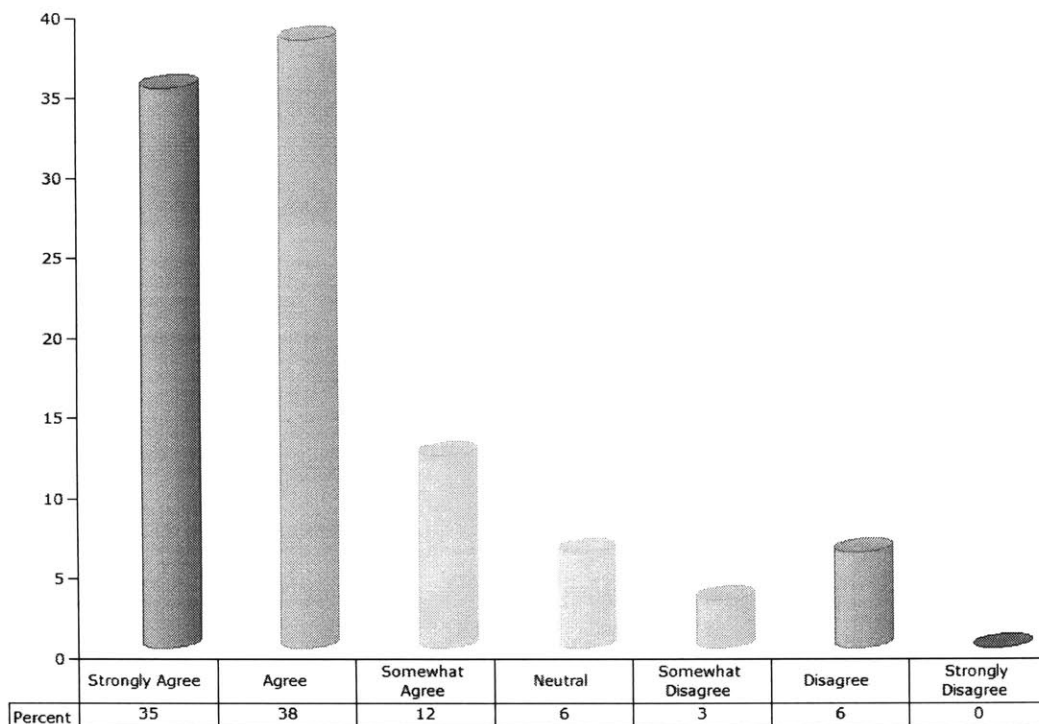


Figure 5-20 Would the subjects use Creo?

Final Question:

I understand the overall utility of this software:

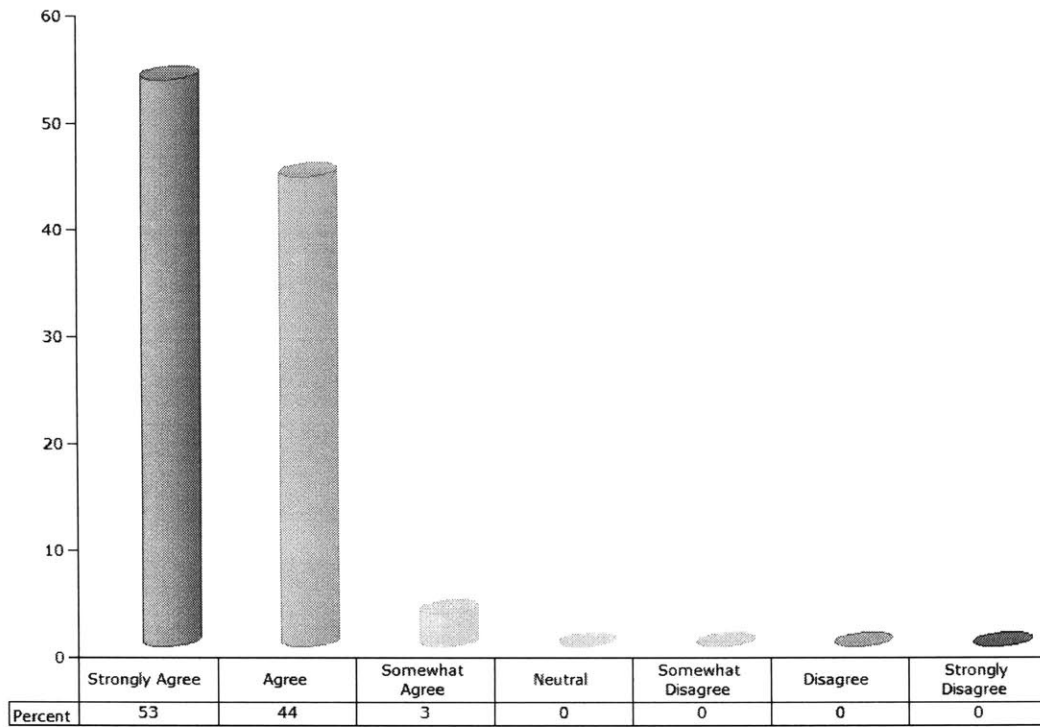


Figure 5-21 Did the subjects understand the software's overall utility?

More Miro Users than Creo Users

Shown in the previous figures, 85% of subjects responded they would use Creo, and 100% of subjects responded they would use Miro. We have implemented a way for users to easily share the functionality of recordings they create with Creo without sharing any of their personal information, (which Creo automatically detects and stores separately). So it is technically possible for a subset of users to use Creo, and for everyone to use Miro.

Ethnographic Results

During the study, the experimenter noticed several notable observations that were not captured by the quantitative measures or questionnaires. These observations, along with a discussion of remaining usability problems, are discussed below.

Confused Programmers:

The first and most interesting was a strong correlation between a subject's level of technical experience, and usability errors when using Creo. Unexpectedly, subjects with a technical background had considerably more difficulty using Creo than subjects who did not have a technical background. Nearly all of the usability problems revolved around understanding that Creo was automatically generalizing input. For instance, after training Creo on the example of ordering "diet coke," subjects with a technical background would often name the recording something to the effect of "Order diet coke." So even though Creo correctly generalized the input to any type of food, the subjects did not correctly generalize their recording name. Also, many of the subjects with a technical background would finish Part 2 and ask, "Ok, now do I need to create recordings for each of the ingredients in the recipe?" When the experimenter explained that only one example was required, and now the Miro toolbar would be able to link any type of food to the recording, several of the technical subjects protested. One subject pointedly argued "What do mean...it can't do that...how is it supposed to know that diet coke is a food?" The experimenter noted at least six subjects had difficulty

understanding that Creo was generalizing input. Of those six, only one subject reported that they had taken zero programming classes, and had zero years of programming experience. Normally, subjects with a technical background will perform better in usability experiments than non-technical subjects, due to their increased familiarity with computers and various types of software. In this case, the usability errors encountered by technical subjects seem to be a result of the technical subject's expectations. The experimenter showed every subject how to use Creo by creating a recording for IMDB with an example movie (Star Wars), and then testing the recording with a different example movie (The 5th Element). However, for some technical subjects, this single example was not enough to overcome their preconceived expectation that software applications cannot understand the semantic context of various types of input. Even some of the technical subjects who completed Part 2 without any usability problems remained skeptical about Creo's abilities. Several subjects asked, "So you programmed this to only work with movies and foods?" It is impossible to prove that there is a correlation between technical experience and usability problems since there were considerably less non-technical subjects in the experiment. Only 26% of all subjects reported that they did not know how to program (an unfortunate result of recruiting subjects in the MIT campus). However, non-technical subjects seemed far more accepting of Creo's abilities, both in their ability to successfully interact with the software, and in comments they made to the experimenter.

Faster is Not Better:

The experiment was designed with the assumed belief that allowing users to complete tasks faster was universally better. However, one subject's comments contradicted this notion. She stated, "I understand why people would really like this, and I think this is great, but I would never use it myself." Her reason was that automatically linking products to a user's favorite stores would lead to uncontrollable compulsive shopping. She also stated that she would never allow Web sites to remember her personal

information because she liked the extra time it took to type it all in. However, when asked if she would use the software for non-ecommerce tasks, like research, she responded, "Oh yes, of course."

Efficiency vs. Privacy:

Several users commented that they didn't feel they should have to use Creo to make recordings. One subject stated, "I don't understand why you have the recording interface. I shouldn't have to record actions for it to know what to do." When the experimenter commented, "We are considering having Creo constantly monitor the user's browsing habits so they never have to explicitly start or stop a recording," the subject found the privacy implications of this to be significantly worse than having to use Creo to make recordings.

Efficiency vs. Security:

Several subjects suggested the idea of storing recordings in a central repository. One subject stated, "Why do I even have to create the recording? Shouldn't the site do it, or someone else, and I can just download it?" Public repositories of services already exist, such as UDDI [129]. The user's browser could check a central repository for each site it went to, and could use other people's recordings to automate actions. The obvious downside to this model is security implications associated with sensitive tasks like transferring money from one bank account to another. Every subject that suggested this idea agreed that for tasks such as those, they would rather create the recording themselves using Creo, or receive the recording from a trusted friend.

Remaining Usability Problems:

The three initial evaluations of Creo's user interface improved the design enough that by the fourth evaluation, many of the obvious problems had been removed. However, a few usability problems (beyond technical subjects' problems with generalization, discussed earlier) did occur during

testing. The window users see when they complete a recording prompts them to enter a name with the question, "Your goal for using this recording is...". The text that the user enters here will appear in Creo's *Player* tab, and in buttons that appear on the Miro toolbar when the user hits *Scan Page*. This dialog box was also asking users to enter their goals because originally this text was going to be mined. The usability problem was that some subjects simply completed the sentence. For instance, one subject entered the name "to see the price." This name is, of course, missing any form of context, and does not adequately describe the recording.

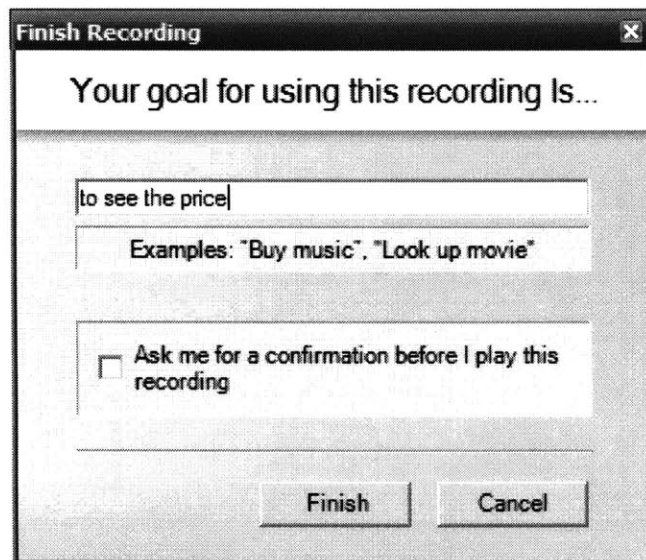


Figure 5-22 Usability problem with saving recordings in Creo

The prompt has now been changed to "Enter a name for this recording."

Another usability problem with Creo was that to submit a form, users had to click *Record Form and Submit*, instead of submitting the form in the Web page's interface. This was due to a technical problem with telling the difference between form submission events, and navigation events.

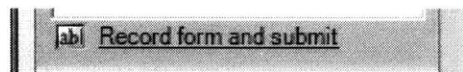


Figure 5-23 Usability problem with submitting forms with Creo

This is a serious usability problem. Subjects found it to be very confusing, and it should definitely be fixed in future versions of Creo. This problem alone likely caused some of the low scores on the question, "I felt that the user interface for recording actions on the Web was easy to use." (Figure 5-18).

The only remaining usability problems with Creo related to subjects not noticing its changes in state (From *Ready*, to *Recording*, to *Recording Complete*). While the color change is rather drastic, perhaps some form of animation or motion during recording would better inform the user of the change in state.

The only usability problem with Miro to occur during the evaluation was that the hyperlinks that it generates do not change color after the user clicks them. This can be easily fixed.

5.3.4 User Study Discussion and Summary

Overall, the results of the evaluation were better than anticipated. Subjects completed the task of searching on each ingredient in a recipe significantly faster using Miro compared to a normal Web browser (68 seconds compared to 139 seconds). Additionally, the difference of the two groups' average times (71 seconds) was considerably greater than the time it took subjects to create the recording being used by the experimental group (26 seconds). This means that even for interacting with a list of 11 items, it would be faster to train Creo first, and then use Miro to turn the information into hyperlinks.

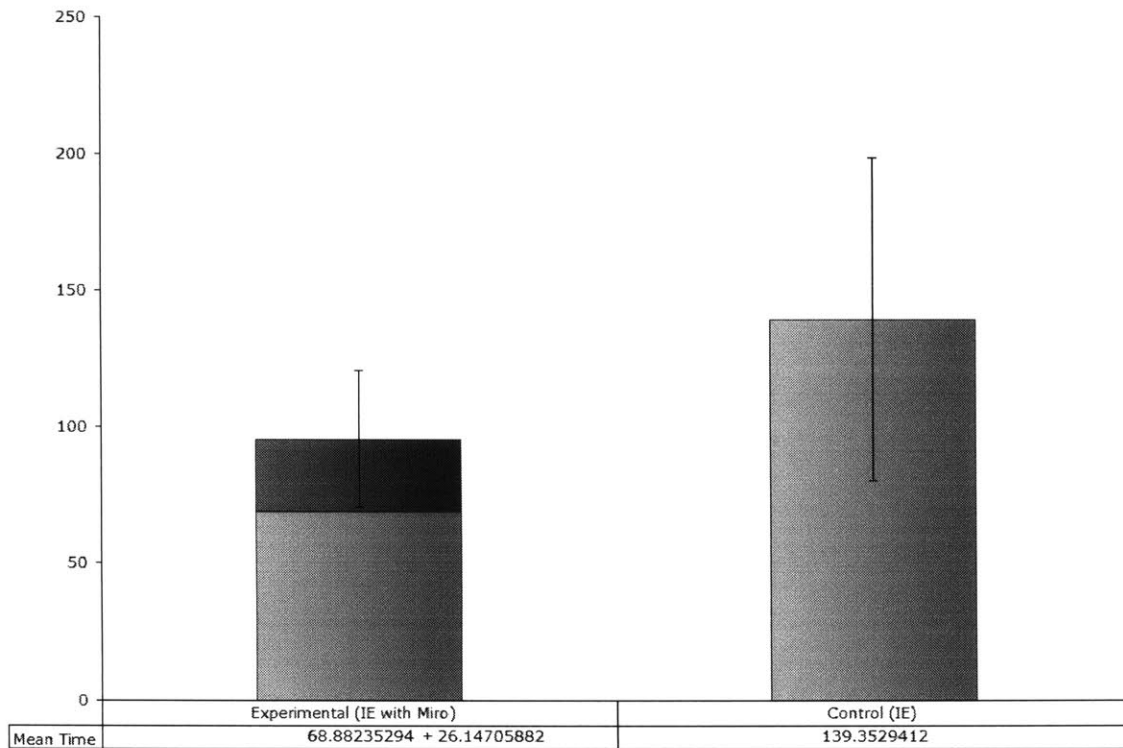


Figure 5-24 The time to complete Part 2 added to the experimental group's time in Part 1

The range of results from the control group in Part 1 is due to the fact that subjects were asked to complete the task however they naturally would. And even though subjects were instructed not to race, some subjects in the control completed the task at a very efficient pace. There was a large amount of variability in the way subjects transferred information between the recipe and the grocery store site. Some subjects relied heavily on keyboard shortcuts, using alt-tab to switch windows and tab to switch which control on the grocery store page had the focus. Some subjects double clicked to select a word, and triple clicked to select a full line. Other subjects retyped every ingredient instead of copying and pasting. Since they would often hold three to four ingredients in their own memory at a time, this often turned out to be faster. The times it took subjects in the control group to complete the task ranged from 73 seconds to 268 seconds, because of the varying ways that subjects chose to complete the task. Despite the varied approaches of the

control group, the experimental group was consistently faster at completing the task. 82% of the subjects in the experimental group completed the task in less than 75 seconds, while only 12% in the control group were able to complete the task in less than 75 seconds.

The results of the post-experiment questionnaire were also better than expected. The lowest scoring interface was Creo, in part due to technical problems related to form submission, and in part due to the fact that creating recordings is a fundamentally more difficult task. Only 6% of subjects stated that they found Miro hard to use, while 12% of subjects found Creo difficult to use. Interestingly, 100% of the subjects responded that they would use Miro if they had access to it, while only 85% responded that they would use Creo if they had access to it. This difference was also represented in the users' comments. Several subjects said that Miro should be able to function without having to use Creo. Since recordings are shareable, this will be possible. Even though Creo does not require that the user has the ability to program, the idea of using it to automate procedures may only appeal to more technically minded users (which is ironic, since they had more difficulty understanding how to use it). Technical users can use Creo to create recordings, and then they can share them with their less technical friends.

For the final question, "I understand the overall utility of this software," all of the subjects checked "somewhat agree" or higher. In retrospect, the question should have been worded, "I believe this software has utility," since "I understand" implies an intelligence test. Additionally, subjects answered this question directly before receiving \$10. However, from both the written and verbal comments made by the subjects, subjects demonstrated a lot of enthusiasm for Creo and Miro. Many subjects wrote comments like, "Great work! I look forward to this becoming available," and "I would like such a feature."

Study Conclusion

We hypothesized that in Part 1 of the experiment, the subjects in the experimental group would be able to complete the task considerably faster than the control group. This hypothesis was confirmed. Additionally, our hypothesis that subjects would have a favorable opinion of Creo and Miro was also confirmed.

Chapter 6

Future Directions and Conclusion

6.1 Future Work

This section discusses several interesting directions this research may take in the future. These future research directions include passively monitoring the user, creating a central repository of recordings, building different types of user interfaces on top of the Web, and the field of Ambient Intelligence.

6.1.1 Passively Monitoring the User

Making Creo a Proactive Interface

Currently, users must launch Creo and actively create recordings. It would be interesting to explore ways to make Creo a more proactive interface. For instance, Creo could notice a repetitive sequence of actions performed by the user, and then ask the user if they would like to automate these actions in the future. This would be similar to the way Letizia by Henry Lieberman monitors the user's Web browsing over long periods of time in an attempt to learn the types of information they are interested in [50]. One advantage of this approach is that Creo would get more training data when generalizing procedures. For instance, if Creo watched the user search for 10 different kinds of food at a grocery store Web site, Creo would be able to generate a very long and accurate list of generalizations for the search field. While Creo is able to create a general-purpose recording with a single example, more data still results in a better set of generalizations.

Detecting potential recordings that return a specific piece of information from a Web site would be harder to observe. For instance, if a user regularly logs into their bank's Web site, navigates to their checking account and reads the balance, there is no way for Creo to know (without performing eye tracking) that the user is regularly retrieving a specific piece of information from the Web page.

Improving Automatic Form Filling

Creo relies on observing the user's input to forms on the Web because usually a field's name attribute consists of terms like "T1" or "sParams," which do not provide any useful information.

A number of Web browser extensions currently exist to automatically fill out forms for the user. A popular example of this is the AutoFill feature of the Google Toolbar. All of these AutoFill-like systems work by analyzing the name attributes of form elements, looking for common names like "zipcode" and "firstname." If a text field actually takes a zip code, but its field name is "zc," "zcode," or "T7," AutoFill will fail to recognize the field.

This problem can be solved by passively monitoring the types of data users put into undetected fields, and then reporting the association of these data types and obscure field name attributes to a central server. For instance, if the user typed their phone number into a form field that had the name attribute "T12," the toolbar could report to a central server a message like "*T12 at URL takes phoneNumber.*" (Note that the user's personal information itself is not being sent to the server, just the type of information.) If the server aggregated these messages and cached the obscure field name attributes of the most popularly filled out forms, systems like AutoFill could greatly increase their accuracy and coverage.

Detecting Information Authorities

This approach can be taken a step farther, aggregating a richer variety of semantic information being submitted into forms by a community of users. By aggregating the semantics of users' queries at specific sites, the system could detect information authorities. For instance, the central server aggregating this information could easily infer that IMDB is a popular information source for movies (due to the number of movie searches users perform at the IMDB Web site), CNET is a popular information source for consumer electronics and gadgets, and IGN is a popular information source for video games. Detecting these types of information authorities could be used to re-rank search results, or augment search results, similar to the ABS (Activity Based Search) system created by the Stanford Knowledge System Laboratory [11-15, 32]. However, unlike ABS, the information authorities would be selected through the democratic means of aggregating users' behavior, instead of being arbitrarily selected by the system's creator.

Exploring the Deep Web

This type of semantic analysis of form submissions would represent an important step toward exploring the deep Web. The deep Web consists of pages that are dynamically generated, and are therefore difficult for a Web crawler to access. Currently, Google's Web crawler is only able to explore the deep Web when it encounters a hyperlink to search results [130]. Yahoo currently allows users to perform searches of the deep Web, but they only support nine participating Web sites [131]. By aggregating the types of searches users are performing at popular information authorities, a Web crawler could anticipate which types of queries to try submitting to these sites, and then index the dynamically generated Web pages.

Helping Users Manage their Personal Information

In the future work of Earl Wagner's masters thesis on Woodstein [61], he proposes the idea of a Web browser helping users manage which Web sites have received their personal information.

As part of tracking information the user enters, Woodstein could track who knows what. We've all experienced the hassle of having to update contact information when we've moved, switched jobs, or experienced other changes. An agent that tracked what information we've entered would know exactly who needs to be updated [61].

This is a fantastic idea, and could be easily integrated into a future version of Creo. Creo already automatically detects personal information, and Creo's profile.xml file contains all of the personal information a user has submitted to Web sites in the process of creating recordings. When the user changes their profile, the new information is submitted when they play recordings in the future. However, there is currently no user interface for viewing this information in the way Earl Wagner proposes. For instance, a user could potentially view all of the Web sites that have received a specific piece of personal information, like the user's address or phone number.

Privacy Implications

All of these ideas are based around the activity of passively monitoring the user, or to use a more familiar term, spyware. However, unlike traditional spyware, the intention of monitoring the user is to help them. In the case of creating an interface that allows users to track who has received their personal information, users are being directly helped. In the case of aggregating the semantic value of form submissions to improve automatic form filling and explore the deep Web, users are being indirectly helped by advancing these features for everyone. The ideas of improving form filling systems and exploring the deep Web are more controversial, since they involve reporting the semantic value of information to a central server. Currently many users believe that systems that aggregate non-personally identifiable information are acceptable. However, several users during the user studies discussed in Chapter 5 were adamantly opposed to the idea of their Web browser passively monitoring their actions. Any future

implementation of Creo that involves passive monitoring will inform the user exactly what types of information are being collected or shared, and why. Creo will also allow the user to easily disable these features.

6.1.2 Creating a Central Repository of Recordings

Currently Creo has been designed to allow users to share their recordings with their friends. Users can share a recording's functionality without sharing any of their specific personal information. An obvious next step would be to create a central repository of recordings similar to UDDI [129]. This would allow users to share the recordings they create with everyone on the Web. Such a system would have to solve a number of problems related to security and privacy. For instance, when Creo displays to the user the specific actions a recording is going to take, a percentage of users may still disregard the confirmation window, and choose to play a malicious recording that transfers out all of the money in their bank account, or sends their credit card number to a third party. By allowing users to share recordings with their immediate friends, they know exactly who to blame when things go wrong, and malicious recordings will not spread very far.

One potential compromise would be to create a central repository for recordings that do not involve any sensitive information or actions. The server could automatically delete any submitted recordings that violate a list of warning signs, like referencing any information in the user's profile, or using https. This would automatically rule out any legitimate recordings that interact with the user's bank, or complete actions like ordering a pizza. However, less dangerous recordings like accessing a food at a grocery store, or looking up movies at IMDB, would still be available.

A central server of recordings would significantly streamline the process of creating recordings for each the user's favorite sites. Miro could display an icon to the user indicating that it can automatically learn how to interact with

the site, and the user would not have to open Creo to create the recording. However, dealing with the implications of such a system is left to future work.

6.1.3 Building Different Types of User Interfaces on Top of the Web

Once a user teaches their computer how to interact with a Web site using Creo, their computer has enough knowledge to automate future interactions. The abstract representation of a Web site's content or services generated by Creo facilitates exposing this functionality through new types of user interfaces. Miro and Adeo are two examples of this. However, many other types of user interfaces could be built on top of the Web.

The 10-Foot Interface

The user experience when seated away from a computer, without input devices like a keyboard and mouse, and viewing output on a large display, is often referred to as the "10-foot interface." Examples of this type of interface include interacting with set top boxes, digital video recorders, and video game systems. Ironically, many users currently have about the same input and output bandwidth interacting with the 10-foot interface as they do interacting with present-day mobile devices. This is because remote controls contain roughly the same input bandwidth as the physical controls found on a Smartphone, and the resolution of non-high definition televisions is roughly equivalent to an (albeit considerably smaller) Smartphone screen. While several systems exist to allow users to browse the Web on their television sets, the process can be just as frustrating as browsing the Web on a mobile device. A system like Adeo could be easily adapted to the 10-foot interface, allowing users to check the balance of their bank account, look up a movie, or order a pizza using their TiVo or Xbox.

Conversational and Embodied Interfaces

A more natural way of interacting with recordings could be created through the use of conversational interfaces. The creators of WebViews [116] have explored using VoiceXML as an interface to content and services on the Web, allowing users to access information using speech generation and recognition. A similar conversational interface could also be created using text. Users could invoke recordings and provide input through SMS messages on a mobile device, or using an instant messaging client.

A conversational interface to the user's recordings could also be accessed through an embodied interactive intermediary, posing as a cute squirrel.



Figure 6-1 The Cellular Squirrel, by Stefan Marti

The Cellular Squirrel [132], created by Stefan Marti, is a small portable animatronic device that mediates communication between the caller and user, finding out the purpose of the call, and deciding if the user should be interrupted. It could also be adapted to act as an intermediary between the user and their favorite content and services on the Web. However, any embodied interface must also deal with the challenges facing anthropomorphism, discussed in Chapter 2.

6.1.4 Integrating Scan into Adeo

Miro provides two primary features, (1) the ability to perform semantic searches, and (2) the ability to scan a page, linking the content of the page to contextually relevant recordings. Adeo currently only provides the first feature, the ability to perform semantic searches. It is interesting to consider how the second feature, *Scan*, could be integrated into an application like Adeo.

In Miro, the *Scan* feature works by looking up the semantic context of each term on a Web page, and matching this context against the user's set of recordings. Adeo of course does not have the text of a Web page to base suggestions on. However, there are many other forms of context in the environment in which a mobile application like Adeo might be used, including the user's location, surrounding objects, and audio. A mobile device can detect these forms of context through technologies like GPS, RFID, and speech recognition. Each of these forms of context has been explored in other research projects.

Location: CyberDesk

The Data Detector CyberDesk [110] by Anind Dey, Gregory Abowd and Andrew Wood, provided an infrastructure for creating context aware mobile devices, prototyped with a Newton MessagePad. Beyond the forms of context relevant to a desktop environment (like phone numbers and email addresses), CyberDesk also performed data detection on the user's GPS position. They demonstrated this with a service that took GPS coordinates of a location on the Georgia Tech campus, and returned a URL corresponding to that location [110]. This service was programmed by the creators of CyberDesk, and user's could not define their own services.

Objects: ReachMedia

ReachMedia [133], by Assaf Feldman, aims to merge the virtual world and the physical world, providing the user with just-in-time information about physical objects. Based on the idea that RFID will soon be replacing barcodes, ReachMedia detects what object the user is holding through an RFID reader worn on the user's wrist.



Figure 6-2 ReachMedia, by Assaf Feldman

ReachMedia is designed to provide information without demanding too much of the user's attention, and in a manner that (unlike wearing an augmented reality headset) is socially acceptable. The user provides input through slight hand gestures, which are detected with an accelerometer located on the wristband, and output is provided with audio through the user's Bluetooth headset. ReachMedia accesses contextually relevant services on the Internet through the user's Smartphone, which can remain in the user's pocket. An example use of ReachMedia would be hearing reviews of books from Amazon.com while picking up books at a bookstore. Like many of the Data Detectors discussed in Chapter 4, the contextual services provided by ReachMedia are programmed by the system's creators, and can not be created by non-technical end-users.

Audio: GISTER and OVERHEAR

GISTER and OVERHEAR [73, 74] by Nathan Eagle, Push Singh and Alex Pentland, infer the user's context by performing spontaneous speech

recognition on the user's conversations. The system uses a Zarus Linux handheld computer, and by leveraging ConceptNet [9] and LifeNet [39], the device can correctly determine the gist of the user's conversation, even when spontaneous speech recognition rates fall below 35%. The creators of GISTER and OVERHEAR do not directly link the context detection to a end-user interface, but simply note that the additional context "will become invaluable for a variety of applications" [73].

All of these systems act as Data Detectors, like the Intel Selection Recognition Agent, or Miro, except instead of detecting text, they are based on detecting the context of the user's physical environment.

Adeo could streamline mobile browsing for the user even more if it leveraged contextual information like the user's location, surrounding objects, and the current topic of conversation. For instance, in the user scenario described at the start of Chapter 2, Adeo could have proactively displayed the recording to order a pizza from the local pizza place on Pete's Smartphone as soon as Ian said, "I'm getting hungry, let's order a pizza." Context could also play a role in specifying input to selected recordings: the actions of sending PayPal and Evites to other people in the room also could have been streamlined by detecting who was nearby.

6.1.5 Ambient Intelligence and Context Aware Computing: Real World Data Detection

Detecting contextually relevant services based on information like the user's location, nearby objects, and audio, are all examples of leveraging the real world for input. In the future, output will exist in the real world as well. The Augmented Reality Kitchen by Leonardo Bonanni, Chia-Hsun Lee, and Ted Selker [134] contains a system called CounterActive that aids the user by projecting recipes onto countertops and cabinets as they prepare food.



Figure 6-3 The Augmented Reality Kitchen

This is similar to Miro's ability to link contextually relevant information and services to foods, demonstrated in Chapter 2, except applied to an actual physical kitchen. However, the information and services being projected in the Augmented Reality Kitchen are not personalized to the user. It is impossible in the Augmented Reality Kitchen for the user to associate the physical foods in their kitchen with ordering them at their personal grocery store, looking them up at their favorite site for recipes, or viewing their nutritional information at a particular site. By integrating Creo into the Augmented Reality Kitchen (perhaps through a projected Web browser and an RFID reader that converts physical objects to text), users could define their own augmented reality experiences by example, and then share them with their friends.

Unlike all of the research projects exploring these types of context aware applications, the services provided by an Ambient Intelligence version of Miro and Adeo could be personalized to each specific user, and users would be free to define their own services by example using a system like Creo. Additionally, regardless of if the concept being detected is a word (in text or audio) or the presence of a physical object, large knowledge bases of semantic information like ConceptNet and TAP are still required to

Chapter 6: Future Directions and Conclusion

dynamically associate the concept with contextually relevant services, and dealing with ambiguity remains a relevant issue.

6.2 Conclusion

The ultimate user interface would always be able to proactively anticipate the user's goals, and automate the procedures for them, regardless of the task. This is the essential idea that Vannevar Bush was getting at when he described *Selection by Association* [18], the idea wasn't just hypertext. *Selection by Association* describes an implicit mapping between the user's mental model of information and a computer's resources.

This thesis has demonstrated how large scale knowledge bases of semantic information can enable a computer to anticipate a user's goals, and how Programming by Example can enable a computer to automate procedures for the user. While this is still far from the ultimate user interface, it represents a step toward the vision of *Selection by Association*.

Every user has a different mental model of information. One user may associate the foods in a recipe with their online grocery store, and yet another user with a Web site that provides nutritional information. Working together, Creo and Miro allow each individual user to perform *Selection by Association* on data, receiving personalized and contextually relevant information and services, based on their own mental model of information.

In the future, this personalized and contextual mapping between data on the Web and Semantic Web Services may be achieved through standards such as RDF, OWL, and OWL-S, or it may be achieved through smarter Web browsers, like the one described in this thesis. How this mapping occurs is not particularly important; what is important is that it happens.

Computing is on the verge of breaking out of a machine sitting on a user's desk and expanding into the real world, where objects and gestures provide input, and any surface can provide output. However, even in this new era, the concept of personalized *Selection by Association* remains relevant.

Appendix A

User Study Recruitment Poster

Appendix B

User Study Consent Form and Questionnaires

6.2.1 Consent to Participate in Non-Biomedical Research

A Goal-Oriented User Interface for Personalized Semantic Search

You are asked to participate in a research study conducted by Henry Lieberman (PhD), and Alexander Faaborg (BA), from the Software Agents Group in the MIT Media Lab, at the Massachusetts Institute of Technology (M.I.T.). Results will contribute to Alexander Faaborg's Masters Thesis. You were selected as a possible participant in this study because you have previous experience using a Web browser. You should read the information below, and ask questions about anything you do not understand, before deciding whether or not to participate.

PARTICIPATION AND WITHDRAWAL

Your participation in this study is completely voluntary and you are free to choose whether to be in it or not. If you choose to be in this study, you may subsequently withdraw from it at any time without penalty or consequences of any kind. The investigator may withdraw you from this research if circumstances arise which warrant doing so.

PURPOSE OF THE STUDY

Tim Berners-Lee envisions the Semantic Web, where interface agents assist users with complex, multi-step, context-dependent tasks, such as arranging travel, doctors' appointments, and online purchases. This study will evaluate subjects' ability to complete several tasks using a modified Web browser that can automate actions and can understand the semantic context of

information being displayed to the user. Results from the study will help us determine the prototype software's effectiveness in real world situations.

PROCEDURES

If you volunteer to participate in this study, we would ask you to do the following things:

1. Use a modified version of Internet Explorer to complete several tasks
2. Fill out questionnaire forms before and after the experiment.

The experiment should last a total of 30 minutes. Participation is voluntary. You may leave the experiment at any time or refuse to answer any questions, and you will still be compensated.

POTENTIAL RISKS AND DISCOMFORTS

There are no foreseeable psychological or physiological risks or discomforts associated with participating in this experiment.

This is a usability experiment: if you have trouble with a task, it is not your fault. Any mistakes you may make using the software involved in this study are the fault of the software's designer. This is not an intelligence test, and you are not competing with other subjects. During the experiment, please keep in mind that we are testing the software, we are not testing you.

POTENTIAL BENEFITS

Benefits to you: You will learn about research to improve Web browsers; you may find this information interesting.

Benefits to Society: Society may eventually receive the benefits laid out in Tim Berners-Lee's vision of the Semantic Web. In particular, this project will ideally lead to people spending less time interacting with forms, and copying and pasting information into search boxes. In the future, it will be easier for people to quickly complete complex tasks on the Web.

PAYMENT FOR PARTICIPATION

You will receive \$10 for participating in this experiment. Participation is voluntary. You may leave the experiment at any time or refuse to answer any questions, and you will still be compensated.

CONFIDENTIALITY

Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law.

Your confidentiality is assured. You will be assigned a subject ID number. No personally identifiable information will be associated with data collected during this study.

IDENTIFICATION OF INVESTIGATORS

If you have any questions or concerns about the research, please feel free to contact:

Alexander Faaborg

Research Assistant
faaborg@media.mit.edu
617-899-5064

Henry Lieberman
Research Scientist
lieber@media.mit.edu
617-253-0315

EMERGENCY CARE AND COMPENSATION FOR INJURY

In the unlikely event of physical injury resulting from participation in this research, you may receive medical treatment from the M.I.T. Medical Department, including emergency treatment and follow-up care as needed. Your insurance carrier may be billed for the cost of such treatment. M.I.T. does not provide any other form of compensation for injury. Moreover, in either providing or making such medical care available, it does not imply the injury is the fault of the investigator. Further information may be obtained by calling the MIT Insurance and Legal Affairs Office at 1-617-253-2822.

RIGHTS OF RESEARCH SUBJECTS

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, M.I.T., Room E32-335, 77 Massachusetts Ave, Cambridge, MA 02139, phone 1-617-253-6787.

SIGNATURE OF RESEARCH SUBJECT OR LEGAL REPRESENTATIVE

I understand the procedures described above. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

Name of Subject

Name of Legal Representative (if applicable)

Signature of Subject or Legal Representative

Date

SIGNATURE OF INVESTIGATOR

In my judgment the subject is voluntarily and knowingly giving informed consent and possesses the legal capacity to give informed consent to participate in this research study.

Signature of Investigator

Date

6.2.2 A Goal-Oriented User Interface for Personalized Semantic Search Pre-Experiment Questionnaire

MIT Media Lab, Software Agents Group

General Information:

Subject Number: _____

Age: _____

Gender: _____

Computer Use:

On average, how many hours do you spend using the Web each week? _____

Which of the following things have you done on the Web?

- Used a search engine
- Used the search feature of a particular site
- Shopped
- Researched a product
- Participated in an auction
- Arranged travel
- Posted to a message board
- Signed up for an activity/appointment
- Accessed a bank account
- Looked up something you needed more information on, knowing which site to search first

On average, how many times per week do you interact with a form on the Web to complete an action? _____

Around how many different actions do you often repeat on the Web?

Some examples of an "often repeated action" include: ordering groceries, checking the balance of your bank account, transferring money between two bank accounts, checking the current weather, sending money with PayPal, etc... _____

Have you taken any classes involving programming? How many?

- 0 Classes
- 1 Class
- 2 Classes
- 3+ Classes

If you know how to program, how many years have you programmed?

- Do not know how to program
- 1 year
- 2 years
- 3+ years

6.2.3 A Goal-Oriented User Interface for Personalized Semantic Search Post-Experiment Questionnaire

MIT Media Lab, Software Agents Group

About the Experiment:

We will be happy to answer any questions you have about the experiment you just participated in. For both experiments, we recorded the time it took you to complete particular tasks. None of the information we are collecting is personally identifiable.

Final Questions:

Part 1:

I felt that the user interface for selecting actions associated with the page I was viewing was easy to use:

Strongly Agree Agree Somewhat Agree Neutral Somewhat Disagree Disagree Strongly Disagree

If my Web Browser included a feature to select actions associated with the page I was viewing, I would use it:

Strongly Agree Agree Somewhat Agree Neutral Somewhat Disagree Disagree Strongly Disagree

Part 2:

I felt that the user interface for recording actions on the Web was easy to use:

Strongly Agree Agree Somewhat Agree Neutral Somewhat Disagree Disagree Strongly Disagree

If my Web Browser included a feature to record actions, I would use it:

Strongly Agree Agree Somewhat Agree Neutral Somewhat Disagree Disagree Strongly Disagree

Final Question:

I understand the overall utility of this software

Strongly Agree Agree Somewhat Agree Neutral Somewhat Disagree Disagree Strongly Disagree

Comments:

Bibliography

1. Tim Berners-Lee, James Hendler, Ora Lassila. *The Semantic Web*. Scientific American. (2001).
2. Alan Cooper. *About Face: The Essentials of User Interface Design*. IDG Books Worldwide, Inc. Foster City, California. (1995).
3. José Espinosa. *Reducing Complexity of Consumer Electronics Interfaces Using Commonsense Reasoning*. Masters Thesis. Media Arts and Sciences, Massachusetts Institute of Technology. Cambridge, Massachusetts. (2005).
4. *Open Mind Common Sense Project*. <http://commonsense.media.mit.edu/>. Accessed: 9/4/2005. (2005)
5. Push Singh. *KurzweilAI.net: The Open Mind Common Sense Project*. <http://www.kurzweilai.net/meme/frame.html?main=/articles/art0371.html>. Accessed: 9/5/2005. (2002)
6. Push Singh. *The Public Acquisition of Commonsense Knowledge*. Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access. Palo Alto, California. (2002).
7. Push Singh, Barbara Barry, Hugo Liu. *Teaching Machines about Everyday Life*. BT Technology Journal. (2004).
8. Push Singh, Thomas Lin, Erik Mueller, Grace Lim, Travell Perkins, Wan Li Zhu. *Open Mind Common Sense: Knowledge Acquisition from the General Public*. Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems. Irvine, California. (2002).
9. Hugo Liu, Push Singh. *ConceptNet: a Practical Commonsense Reasoning Toolkit*. BT Technology Journal. (2004).
10. Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press. Cambridge, Massachusetts. (1998).
11. *TAP: Building the Semantic Web*. <http://tap.stanford.edu/>. Accessed: 9/21/2005. (2005)
12. R. Guha, Rob McCool. *TAP: a Semantic Web Platform*. Computer Networks: The International Journal of Computer and Telecommunications Networking. Volume 42, Issue 5. (2003).
13. R. Guha, Rob McCool. *A System for Integrating Web Services into a Global Knowledge Base*. <http://tap.stanford.edu/sw002.html>. Accessed: 9/21/2005. (2005)
14. R. Guha, Rob McCool, Eric Miller. *Semantic Search*. Proceedings of the 12th International Conference on World Wide Web. Budapest, Hungary. (2003).
15. Rob McCool, Richard Fikes, Deborah McGuinness. *Semantic Web Tools for Enhanced Authoring*. http://ksl.stanford.edu/KSL_Abstracts/KSL-03-07.html. Accessed: 9/21/2005. (2005)

16. Rob McCool, R. Guha, Richard Fikes. *Contexts for the Semantic Web*. <http://tap.stanford.edu/contexts.pdf>. Accessed: 9/21/2005. (2005)
17. Douglas Lenat. *CYC: a Large-Scale Investment in Knowledge Infrastructure*. Communications of the ACM. Volume 38, Issue 11. (1995).
18. Vannevar Bush. *As We May Think*. The Atlantic Monthly. (1945).
19. Alexander Faaborg. *Semantic Browsing*. Undergraduate Honors Thesis. College Scholar Program, Cornell University. Ithaca, New York. (2003).
20. Alexander Faaborg, Carl Lagoze. *Semantic Browsing*. Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries. Trondheim, Norway. (2003).
21. H. Peter Alesso, Craig Smith. *Developing Semantic Web Services*. A K Peters. Natick, Massachusetts. (2005).
22. Grigoris Antoniou, Frank Harmelen. *A Semantic Web Primer*. MIT Press. Cambridge, Massachusetts. (2004).
23. Michael Daconta, Leo Obrst, Kevin Smith. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing Inc. Indianapolis, Indiana. (2003).
24. Thomas Passin. *Explorer's Guide to the Semantic Web*. Manning. Greenwich, Connecticut. (2004).
25. *The Semantic Web: An Interview with Tim Berners-Lee*. <http://www.consortiuminfo.org/bulletins/semanticweb.php>. Accessed: 9/25/2005. (2005)
26. *OWL-S: Semantic Markup for Web Services*. <http://www.w3.org/Submission/OWL-S/>. Accessed: 9/29/2005. (2005)
27. James Hendler. *Professor James A. Hendler*. <http://www.cs.umd.edu/users/hendler/>. Accessed: 8/18/2005. (2005)
28. Scott Ard, Steven Musil. *Microsoft clips Windows XP Smart Tags*. <http://news.com.com/2100-1001-269167.html?legacy=cnet>. Accessed: 8/19/2005. (2001)
29. Walter Mossberg. *Microsoft Will Abandon Controversial Smart Tags*. <http://ptech.wsj.com/archive/ptech-20010628.html>. Accessed: 8/19/2005. (2001)
30. Leslie Walker. *Google AutoLink Pits Convenience, Ownership Issues*. <http://www.washingtonpost.com/wp-dyn/articles/A660-2005Mar2.html>. Accessed: 8/19/2005. (2005)
31. Dan Gillmor. *Google Toolbar, an Update*. http://dangillmor.typepad.com/dan_gillmor_on_grassroots/2005/02/google_toolbar_.html. Accessed: 8/19/2005. (2005)
32. Chris Kaminski. *Much Ado About Smart Tags*. <http://www.alistapart.com/articles/smarttags/>. Accessed: 8/19/2005. (2001)
33. Paul Boutin. *When Good Search Engines Go Bad*. <http://slate.msn.com/id/2114308/>. Accessed: 8/19/2005. (2005)
34. *Mozilla Firefox - Smart Keywords*. <http://www.mozilla.org/products/firefox/smart-keywords>. Accessed: 8/20/2005. (2005)

35. Wikipedia. *Wireless Application Protocol*.
http://en.wikipedia.org/wiki/Wireless_Application_Protocol. Accessed: 8/24/2005. (2005)
36. Jakob Nielsen. *WAP Backlash*.
<http://www.useit.com/alertbox/20000709.html>. Accessed: 8/24/2005. (2000)
37. Jeremy Scott-Joynt. *The Secret of NTT's i-mode Success*.
<http://news.bbc.co.uk/1/hi/business/1835821.stm>. Accessed: 8/24/2005. (2002)
38. *2005 North Asian Mobile Communications and Mobile Data*.
<http://www.gii.co.jp/english/pa27575-mobile-communications.html>. Accessed: 8/24/2005. (2005)
39. Push Singh, William Williams. *LifeNet: a Propositional Model of Ordinary Human Activity*. Proceedings of the Workshop on Distributed and Collaborative Knowledge Capture (DC-KCAP). Sanibel Island, Florida. (2003).
40. *The CAPTCHA Project*. <http://www.captcha.net/>. Accessed: 8/27/2005. (2005)
41. Marti Hearst. *Automatic Acquisition of Hyponyms from Large Text Corpora*. Proceedings of the Fourteenth International Conference on Computational Linguistics. Nantes France. (1992).
42. *Software Agents Group - Projects*.
<http://agents.media.mit.edu/projects.html>. Accessed: 9/2/2005. (2005)
43. Henry Lieberman, Hugo Liu. *Adaptive Linking between Text and Photos Using Common Sense Reasoning*. Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002. Malaga, Spain. (2002).
44. Hugo Liu. *Semantic Understanding and Commonsense Reasoning in an Adaptive Photo Agent*. Masters Thesis. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Cambridge, Massachusetts. (2002).
45. Hugo Liu, Henry Lieberman. *Robust Photo Retrieval Using World Semantics*. Proceedings of the LREC 2002 Workshop on Creating and Using Semantics for Information Retrieval and Filtering: State-of-the-art and Future Research. Las Palmas, Canary Islands. (2002).
46. *Aria: An Agent for Integrated Annotation and Retrieval of Images*.
<http://agents.media.mit.edu/projects/aria/>. Accessed: 9/5/2005. (2003)
47. *Alexander Faaborg's Bank Account at the MIT Federal Credit Union*.
<http://www.mitfcu.org/>. Accessed: 9/3/2005. (2005)
48. *R.I.P. WYSIWYG*. <http://www.useit.com/alertbox/wysiwyg.html>. Accessed: 10/11/2005. (2005)
49. Henry Lieberman. *Autonomous Interface Agents*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 97). Atlanta, Georgia. (1997).

50. Henry Lieberman, Christopher Fry, Louis Weitzman. *Exploring the Web with Reconnaissance Agents*. Communications of the ACM, Volume 44, Issue 8. (2001).
51. *Google Web Accelerator Considered Overzealous*. http://radar.oreilly.com/archives/2005/05/google_web_acce_1.html. Accessed: 10/9/2005. (2005)
52. Henry Lieberman. *Integrating User Interface Agents with Conventional Applications*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 98). San Francisco, California. (1998).
53. Henry Lieberman, Ted Selker. *Agents for the User Interface*. Handbook of Agent Technology. Jeffrey Bradshaw, editor. MIT Press. Cambridge, Massachusetts. (2003).
54. Henry Lieberman, Hugo Liu, Push Singh, Barbara Barry. *Beating Some Common Sense into Interactive Applications*. AI Magazine, Winter 2004. (2004).
55. Pavel Zolnikov. *Extending Explorer with Band Objects Using .NET and Windows Forms*. <http://www.codeproject.com/csharp/dotnetbandobjects.asp>. Accessed: 10/14/2005. (2002)
56. vadaa. *IE Advanced Toolbar for Favorite Site Navigation and Log-In*. http://www.codeproject.com/csharp/IE_Advanced_Toolbar.asp. Accessed: 10/14/2005. (2005)
57. Simon Mourier. *.NET Html Agility Pack: How to Use Malformed HTML Just Like it was Well-Formed XML*. <http://smourier.blogspot.com/2005/05/net-html-agility-pack-how-to-use.html>. Accessed: 10/14/2005. (2005)
58. *Commonsense Computing @ Media*. <http://csc.media.mit.edu/>. Accessed: 9/10/2005. (2005)
59. *Woodstein*. <http://agents.media.mit.edu/projects/woodstein/>. Accessed: 9/5/2005. (2003)
60. Henry Lieberman, Earl Wagner. *End-User Debugging for Electronic Commerce*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 03). Miami, Florida. (2003).
61. Earl Wagner. *Woodstein: A Web Interface Agent for Debugging E-Commerce*. Masters Thesis. Media Arts and Sciences, Massachusetts Institute of Technology. Cambridge, Massachusetts. (2003).
62. Earl Wagner, Henry Lieberman. *Intelligent Interfaces for E-Commerce Problem Solving*. ACM Conference on Electronic Commerce. San Diego, California. (2003).
63. *Goal-Oriented Web Search User Interfaces*. <http://agents.media.mit.edu/projects/goose/>. Accessed: 9/5/2005. (2002)
64. Hugo Liu, Henry Lieberman, Ted Selker. *GOOSE: A Goal-Oriented Search Engine With Commonsense*. Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002. Malaga, Spain. (2002).
65. Tom Stocky, Alexander Faaborg, Henry Lieberman. *A Commonsense Approach to Predictive Text Entry*. Proceedings of the Conference on

- Human Factors in Computing Systems (CHI 04). Vienna, Austria. (2004).
66. Henry Lieberman, Alexander Faaborg, José Espinosa, Tom Stocky. *Common Sense on the Go: Giving Mobile Applications an Understanding of Everyday Life*. BT Technology Journal. (2004).
 67. *A Commonsense Approach to Predictive Text Entry*. <http://agents.media.mit.edu/projects/textentry/>. Accessed: 9/5/2005. (2004)
 68. *Using Commonsense Reasoning to Improve Voice Recognition*. <http://agents.media.mit.edu/projects/voice/>. Accessed: 9/5/2005. (2004)
 69. Henry Lieberman, Alexander Faaborg, Waseem Daher, José Espinosa. *How to Wreck a Nice Beach You Sing Calm Incense*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 05). San Diego, California. (2005).
 70. *Emotus Ponens*. <http://agents.media.mit.edu/projects/emotusponens/>. Accessed: 9/5/2005. (2003)
 71. Hugo Liu, Henry Lieberman, Ted Selker. *A Model of Textual Affect Sensing using Real-World Knowledge*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 03). Miami, Florida. (2003).
 72. Hugo Liu, Ted Selker, Henry Lieberman. *Visualizing the Affective Structure of a Text Document*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 03). Ft. Lauderdale, Florida. (2003).
 73. Nathan Eagle, Push Singh. *Context Sensing Using Speech and Common Sense*. Proceedings of the NAACL/HLT 2004 workshop on Higher-Level Linguistic and Other Knowledge for Automatic Speech Processing. (2004).
 74. Nathan Eagle, Push Singh, Alex Pentland. *Common Sense Conversations: Understanding Casual Conversation Using a Common Sense Database*. Proceedings of the Artificial Intelligence, Information Access, and Mobile Computing Workshop (IJCAI 2003). Acapulco, Mexico. (2003).
 75. *Real Time Searches on a Local Social Network*. <http://agents.media.mit.edu/projects/socialnetwork/>. Accessed: 9/5/2005. (2004)
 76. *Anticipating User Tasks Using Commonsense Reasoning*. <http://agents.media.mit.edu/projects/tasks/>. Accessed: 9/5/2005. (2005)
 77. *GloBuddy 2*. <http://agents.media.mit.edu/projects/globuddy2/>. Accessed: 9/5/2005. (2004)
 78. *Using Commonsense Reasoning to Find Cultural Differences in Text*. <http://agents.media.mit.edu/projects/culture/>. Accessed: 9/5/2005. (2004)
 79. *MakeBelieve: Interactive Computer Story Generation*. <http://agents.media.mit.edu/projects/makebelieve/>. Accessed: 9/5/2005. (2002)

80. Hugo Liu, Push Singh. *MAKEBELIEVE: Using Commonsense to Generate Stories*. Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-02). (2002).
81. *Using Commonsense Reasoning in Video Game Design*. <http://agents.media.mit.edu/projects/videogame/>. Accessed: 9/5/2005. (2004)
82. *Commonsense Investing: Bridging the Gap between Expert and Novice*. <http://agents.media.mit.edu/projects/investing/>. Accessed: 9/5/2005. (2004)
83. *i-Seek: An Intelligent System for Eliciting and Explaining Knowledge*. <http://agents.media.mit.edu/projects/iseek/>. Accessed: 9/5/2005. (2005)
84. Ashwani Kumar. *i-Seek: An Intelligent System for Eliciting and Explaining Knowledge*. Masters Thesis. Media Arts and Sciences, Massachusetts Institute of Technology. Cambridge, Massachusetts. (2005).
85. Ashwani Kumar, Sharad Sundararajan, Henry Lieberman. *Common Sense Investing: Bridging the Gap Between Expert and Novice*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 04). Vienna, Austria. (2004).
86. *Reducing Complexity in Consumer Electronic Interfaces*. <http://agents.media.mit.edu/projects/consumerelectronics/>. Accessed: 9/5/2005. (2005)
87. Henry Lieberman. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann. San Francisco, California. (2001).
88. Tessa Lau, Pedro Domingos, Daniel Weld. *Version Space Algebra and its Application to Programming by Demonstration*. Proceedings of the Seventeenth International Conference on Machine Learning. San Francisco, California. (2000).
89. Steven Wolfman, Tessa Lau, Pedro Domingos, Daniel Weld. *Mixed Initiative Interfaces for Learning Tasks: SMARTedit Talks Back*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 01). Santa Fe, New Mexico. (2001).
90. Jakob Nielsen. *Ten Usability Heuristics*. http://www.useit.com/papers/heuristic/heuristic_list.html. Accessed: 8/5/2005. (2005)
91. Henry Lieberman, Bonnie Nardi, David Wright. *Training Agents to Recognize Text by Example*. Proceedings of the Third Annual Conference on Autonomous Agents. Seattle, Washington. (1999).
92. Henry Lieberman, Bonnie Nardi, David Wright. *Grammex: Defining Grammars by Example*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 98). Los Angeles, California. (1998).
93. Mathias Bauer, Dietmar Dengler. *TriAs: Trainable Information Assistants for Cooperative Problem Solving*. Proceedings of the Third Annual Conference on Autonomous Agents. Seattle, Washington. (1999).

94. Mathias Bauer, Dietmar Dengler. *InfoBeans - Configuration of Personalized Information Assistants*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 98). Los Angeles, California. (1998).
95. Mathias Bauer, Dietmar Dengler, Gabriele Paul. *Instructible Information Agents for Web Mining*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 00). New Orleans, Louisiana. (2000).
96. Mathias Bauer, Dietmar Dengler, Gabriele Paul, Markus Meyer. *Programming by Example: Programming by Demonstration for Information Agents*. Communications of the ACM, Volume 43, Issue 3. (2000).
97. Vinod Anupam, Juliana Freire, Bharat Kumar, Daniel Lieuwen. *Automating Web Navigation with the WebVCR*. Proceedings of the 9th International World Wide Web Conference on Computer Networks. Amsterdam, The Netherlands. (2000).
98. Robert Miller, Brad Myers. *Creating Dynamic World Wide Web Pages by Demonstration*. Technical Report CMU-CS-97-131 (and CMU-HCII-97-101), CMU School of Computer Science. (1997).
99. Atsushi Sugiura, Yoshiyuki Koseki. *Internet Scrapbook: Automating Web Browsing Tasks by Demonstration*. Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology. San Francisco, California. (1998).
100. Alex Safonov. *Web Macros by Example: Users Managing the WWW of Applications*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 99). Pittsburgh, Pennsylvania. (1999).
101. Alex Safonov, Joseph Konstan, John Carlis. *Towards Web Macros: a Model and a Prototype System for Automating Common Tasks on the Web*. Proceedings of the 5th Conference on Human Factors and the Web. Gaithersburg, Maryland. (1999).
102. Alex Safonov, Joseph Konstan, John Carlis. *Beyond Hard-to-Reach Pages: Interactive, Parametric Web Macros*. Proceedings of the 7th Conference on Human Factors and the Web. Madison, Wisconsin. (2001).
103. Alex Safonov, Joseph Konstan, John Carlis. *End-user Web Automation: Challenges, Experiences, Recommendations*. Proceedings of WebNet 2001. Orlando, Florida. (2001).
104. Robert Miller. *Lightweight Structure in Text*. PhD Thesis. School of Computer Science, Computer Science Department, Carnegie Mellon University. Pittsburgh, Pennsylvania. (2002).
105. Michael Bolin. *End-User Programming for the Web*. MEng Thesis. Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Cambridge, Massachusetts. (2005).
106. Nicholas Kushmerick. *Wrapper Induction: Efficiency and Expressiveness*. Artificial Intelligence, Volume 118, Issue 1-2. (2000).
107. Nicholas Kushmerick. *Wrapper Verification*. World Wide Web, Volume 3, Issue 2. (2000).

108. Milind Pandit, Sameer Kalbag. *The Selection Recognition Agent: Instant Access to Relevant Information and Operations*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 97). (1997).
109. Bonnie Nardi, James Miller, David Wright. *Collaborative, Programmable Intelligent Agents*. Communications of the ACM, Volume 41, Issue 3. (1998).
110. Anind Dey, Gregory Abowd, Andrew Wood. *CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 98). San Francisco, California. (1998).
111. Thomas Bonura, James Miller. *Drop Zones: an Extension to LiveDoc*. ACM SIGCHI Bulletin, Volume 30, Issue 2. (1998).
112. James Miller, Thomas Bonura. *From Documents to Objects: an Overview of LiveDoc*. ACM SIGCHI Bulletin, Volume 30, Issue 2. (1998).
113. Yu Chen, Wei-Ying Ma, Hong-Jiang Zhang. *Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices*. Proceedings of the 12th International Conference on World Wide Web. Budapest, Hungary. (2003).
114. Saikat Mukherjee, I.V. Ramakrishnan. *Browsing Fatigue in Handhelds: Semantic Bookmarking Spells Relief*. Proceedings of the 14th international conference on World Wide Web. Chiba, Japan. (2004).
115. Corin Anderson, Pedro Domingos, Daniel Weld. *Personalizing Web Sites for Mobile Users*. Proceedings of the 10th International World Wide Web Conference. Hong Kong, Hong Kong. (2001).
116. Juliana Freire, Bharat Kumar, Daniel Lieuwen. *WebViews: Accessing Personalized Web Content and Services*. Proceedings of the 10th International Conference on the World Wide Web. Hong Kong, Hong Kong. (2001).
117. Dennis Quan, David Karger. *How to Make a Semantic Web Browser*. Proceedings of the 13th international conference on World Wide Web. New York, New York. (2004).
118. *Haystack*. <http://haystack.csail.mit.edu/home.html>. Accessed: 9/28/2005. (2005)
119. *Piggy Bank*. <http://simile.mit.edu/piggy-bank/index.html>. Accessed: 9/28/2005. (2005)
120. David Huynh, Stefano Mazzocchi, David Karger. *Piggy Bank: Experience the Semantic Web Inside Your Web Browser*. Proceedings of the 4th International Semantic Web Conference. Galway, Ireland. (2005).
121. *Haystack Project*. <http://haystack.lcs.mit.edu/>. Accessed: 9/28/2005. (2005)
122. John Domingue, Martin Dzbor. *Magpie: Supporting Browsing and Navigation on the Semantic Web*. Proceedings of the International Conference on Intelligent User Interface (IUI 04). Madeira, Portugal. (2004).

123. Martin Dzbor, John Domingue, Enrico Motta. *Magpie – Towards a Semantic Web Browser*. Proceedings of the 2nd International Semantic Web Conference. Sanibel Island, Florida. (2003).
124. *Flickr*. <http://www.flickr.com/>. Accessed: 9/28/2005. (2005)
125. *del.icio.us*. <http://del.icio.us/>. Accessed: 9/28/2005. (2005)
126. Rob Miller. *6.831 User Interface Design and Implementation*. <http://classes.csail.mit.edu/6.831/>. Accessed: 8/10/2005. (2004)
127. *FreshDirect.com*. <http://www.freshdirect.com/>. Accessed: 8/10/2005. (2005)
128. *The Internet Movie Database*. <http://www.imdb.com/>. Accessed: 8/10/2005. (2005)
129. *UDDI*. <http://www.uddi.org/>. Accessed: 8/10/2005. (2005)
130. *Deep Web*. http://en.wikipedia.org/wiki/Deep_web. Accessed: 10/1/2005. (2005)
131. *Yahoo! Search Subscriptions Beta*. <http://search.yahoo.com/subscriptions>. Accessed: 10/6/2005. (2005)
132. Stefan Marti. *Autonomous Interactive Intermediaries: Social Intelligence for Mobile Communication Agents*. PhD Thesis. Media Arts and Sciences, Massachusetts Institute of Technology. Cambridge, Massachusetts. (2005).
133. *ReachMedia*. <http://interact.media.mit.edu/projects/ReachMedia.html>. Accessed: 10/1/2005. (2005)
134. Leonardo Bonanni, Chia-Hsun Lee, Ted Selker. *Counter Intelligence: Augmented Reality Kitchen*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 05). Portland, Oregon. (2005).