University of Alberta

Visualization of Course Requisites

by

Camilo Arango Moreno

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2009
Edmonton, Alberta

Examining Committee

H. James Hoover, Computing Science

Stan Ruecker, Humanities Computing

Walter Bischof, Computing Science

Geoffrey Rockwell, Humanities Computing

Abstract

We present an interactive tool for browsing the requisites between courses in the University of Alberta as a case study of dependency visualization. This tool uses multiple interactive visualizations to allow the user to explore the courses' dependencies. We performed a usability study that showed that students perform planning tasks faster and more accurately using our tool compared to Bear Tracks, the current registration system used at the university, which only provides textual descriptions of the course requisites.

Acknowledgements

I would like to extend my thanks to Dr. James Hoover and Dr. Stan Ruecker for their valuable supervision in this project, to Dr. Walter Bischof for sharing his extensive knowledge in user studies, and to Business Objects and the ARC Program for their support. Without their collaboration, completing this work would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The dependency visualization problem

A dependency is a type of relationship between two objects, in which the first requires the second to accomplish some action. A simple example of this situation is a hierarchy, where the elements are associated with a single type of relationship, namely, parent-child. The relationship is constrained so that each of the elements has at most one parent but can have multiple children. Other problem domains require a more expressive model to represent their dependencies. Dependency problems appear in a variety of fields. In software development, for example, a piece of software may depend on many libraries, which in turn may depend on others. In project management, certain tasks of the project depend on other tasks to complete or on events that must be executed simultaneously. In workflow modeling, some decisions may require the approval of one or more executives in a certain order, followed by approval by an oversight committee.

Some dependency problems may be modeled by adjacency relationships among elements and, therefore, can be represented by a simple directed graph. Others require a more complex model, as they contain different types of relationships or dependencies may have rules associated with them. A good example of complex dependency graphs can be found in course requisites. Requisites among courses are formed by courses that must be taken before, called *prerequisites*, and courses that may be taken before or simultaneously, called *corequisites*. Furthermore, prerequisites and corequisites can have conditions associated to them; for example, it is possible that, in order to take a given course A, a student has to choose between taking courses B and C, or D, E and F. In turn, courses B, C, D, E and F can have their own requisites and conditions. In consequence, the complex relationships formed by these courses' requisites cannot be modeled by a simple directed graph.

People working on problems that contain dependencies are often given just the relationship information among elements, being forced to understand complex dependency systems on their own. Using visual representations of dependency graphs can provide a valuable tool to understand the relationships and extract additional information from them.

In this work, we further explore the problem of complex dependency visualizations using the requisites of courses of the University of Alberta as our case study. We present a tool to visualize the relationships among courses, called the Course Browser, which can be used to assist course-planning tasks. We also present a usability study that compares our tool against the current course registration system of the University of Alberta. The study shows that participants performed planning tasks significantly faster and more accurately using the Course Browser.

1.2 Rich prospect browsing of dependencies

In this work, we applied the principles of rich prospect browsing to visualize the dependencies defined by the course requisites in the University of Alberta. Rich prospect visualizations, defined by Ruecker in [28], are domain specific visualizations of a collection where each item is represented in a meaningful way, and emergent tools are available for manipulating the display of the items. A browsing interface is intended to support the tasks of understanding, interpreting, or systemizing the material in a domain. This kind of interfaces differ from retrieval interfaces in that the objective of the latter is to search for an specific item from a collection given a set of criteria.

The concept of prospect visualization comes from two notions borrowed from habitat theory and landscape painting: first, that people have a predilection for finding a prospect in a landscape, and, second, that people are capable of identifying opportunities for action in the environment.

Rich prospect interfaces can include functions and affordances present in other types of interfaces, such as search boxes. In addition, rich prospect interfaces may include affordances to zoom, pan, sort, select, group, subset, rename, annotate, open or structure items.

In order to create a rich prospect visualization of a dependency system, we must create a domain specific visualization that gives a meaningful representation of the dependencies of elements in addition to the representation of the elements themselves. In many cases, a directed graph drawing can serve as an appropriate visualization. Elements in the collection are represented by the nodes on the graph, and the dependencies by the edges. However, some simplification of the dependency model may be necessary to express complex dependencies with edges.

We also need new affordances for understanding the dependencies. In the prototype presented in this work we employed two of them: navigation and highlighting. Navigation enables the user to traverse the dependency relationships in the same way hyperlinks enable users to navigate web documents. Highlighting allows them to filter the connections to display only those that are relevant to the user. This operation can be done by modifying different visual characteristics of the connections, for example by changing the color, alignment or adding movement.

Each of the Course Browser prototypes, described in Chapter 3, adapt the concept of rich prospect interfaces for solving the problem of visualizing course dependencies. They, however, do not comply with all the characteristics of a rich prospect visualization. In particular, not all the elements of the collection are displayed in the visualization at all times. In some cases, some elements are omitted or compressed into a single element to allow the handling of a large course catalog.

1.3 Organization of the document

This thesis is organized as follows:

Chapter 2 provides a literature review of the topics that form the background for the development of interactive visualization of dependencies, referring to important contributions in the literature. It is organized into three sections. First we present an overview of user interface design. Second, we present an overview of information visualization, putting special emphasis in graph drawing. Finally, we present related projects.

Chapter 3 describes the design of the Course Browser, an interactive tool for exploring the dependencies of courses using their requisite information. It discusses the objectives of the Course Browser program, introduces the requisite description language and describes the design of the Course Browser tool, showing its evolution throughout different prototypes.

Chapter 4 describes the design and execution of the usability study we used to evaluate the Course Browser. First, it provides a description of the course catalog on Bear Tracks, the application we used as a benchmark. Then, it details the methodology used in the usability test. Finally, it discusses the pilot tests performed before the full-fledged testing. Chapter 5 presents the results of the study and provides an analysis of them.

Finally, Chapter 6 provides insights about the problem of representing dependencies and presents ideas about future work.

Chapter 2

Literature Review

2.1 User interface Design

2.1.1 Principles, patterns and guidelines

Good user interface design has been documented in different levels of granularity by using *design principles*, *design patterns* and *guideline documents*.

Preece *et al.* defines design principles as “generalizable abstractions intended to orient designers towards thinking about different aspects of their designs” [26]. The following design principles, presented by Norman in [24], are applicable to any kind of user interface:

Visibility: the user should be able to understand the information presented by the user interface and easily discover the actions available to control it. A good example of this principle can be seen in most modern elevators. A panel on top of the elevator’s door indicates the floor in which the cart is located. In addition, the buttons for calling the elevator on each floor indicate the two possible actions: “go up” and “go down”.

Feedback: Each action done must give some information back to the user. Otherwise, the user cannot be sure whether or not the action was recognized by the system. The feedback can be sent using stimuli for different senses: vision, audio, and tact. For example, most elevators have lit buttons. When a user presses a button, it illuminates to indicate that the action was recognized by the system and the elevator is on the way. When the feedback is not present, users tend to become confused and press the buttons multiple times.

Constraints: The number of actions available to the user at a given moment must be restricted to prevent them from making mistakes. In the elevator example, the buttons for opening and closing the doors should be restricted from working while the cart is moving. This would prevent accidents caused by users pressing the button by mistake.

Mapping: The way controls work must be consistent with their effects on the world. For example, consider a car's steering wheel. The car turns right when the wheel is turned clockwise, and left when it is turned counter-clockwise. This relationship is natural for the user as the whole car reacts consistently with the action. Another example is the elevator's calling panel. The button to go up should be on top of the panel and the button to go down on the bottom. If they were positioned otherwise (side by side, for example) the user is likely to become confused.

Consistency: The interface elements that represent similar operations must be similar. For example, imagine how strange it would be if some of the buttons for the floors in the elevator needed to be pulled and others pushed. The same principle applies for the representation of information. Imagine how confusing it would be if the panel that shows the elevator's location displayed roman numerals for some floors and arabic numerals for others.

Affordances: The physical characteristics of an object should imply its use. For example, a button *affords* pushing and a lever *affords* pulling. The appearance of the controls in an interface should give clues to the user on how to use it. Some of the affordances are social conventions. For example, today an underlined text is generally associated with a hyperlink and users often know that they can be selected. The interpretation of an underlined text was different before the popularization of the Internet.

While design principles are general and abstract, good design practices for software development have been documented in a more concrete way in the form of design patterns. Design patterns are "best practice" solutions for well-known design problems. Using patterns for software architecture was first done by Gamma *et al.* (commonly known as the *Gang of Four*) in [10]. Tidwell [34] provides a compilation of design patterns for user interface design. These patterns are divided into the nine categories: user behavior, content organization, navigation, layout, actions and commands, complex data presentation, input, builders and editors, and visual style.

Good user interface design has also been fostered by *guideline documents*, such as [23] and [1]. These documents are specific to particular software platforms and seek to bring consistency to all applications written for them.

2.1.2 Interaction Design

The process of designing interactive products is called *interaction design*. Unlike traditional software engineering practices that concentrate on functional requirements, interaction design follows a user-centered approach.

In [26], Preece *et al.* describes three key elements of Interaction design: focus on users, specific usability and user experience goals and iteration. Focus on users refers to the need of involving users in the design of a new product. Users are consulted to validate the design during each of the different phases of the design process. Specific usability and user experience goals are used to evaluate the design according to measurable criteria. Usability goals fall into the following categories:

Effectiveness: can a task be performed successfully with the product?

Efficiency: is the product faster to use? Does the product requires less resources?

Safety: does use of the product have undesirable side effects or increase the risk of undesirable events? (Note an efficient product could reduce safety).

Utility: does the product help the user perform the tasks that they need to perform?

Learnability: is the product easy to learn?

Memorability: can the user easily remember how to use the product?

Iteration refers to the development and refinement of prototypes based on testing and feedback from the users. Prototypes may be developed in low fidelity or high fidelity. Low fidelity prototypes are simple representations of a concept. They may consist of a storyboard or a mock version of the product built, for example, using cardboard or wood. These preliminary designs are presented to users to identify problems before spending further time and effort on their development. The major advantage of this type of prototypes is that they are simple and cheap to produce, enabling the creative team to produce many of them in a short period of time. Low fidelity prototypes need not be very defined or detailed. As mentioned by Buxton in [5], gaps in prototypes are an important part of the creative process. Voids in the prototype often lead to discussion and creation of new ideas. High fidelity prototypes, on the other hand, are partially or fully functional versions of the product. Their purpose is to provide an early view of the final product. These prototypes require more time and effort to produce, but can identify problems that can not be spotted with the low fidelity models. Eventually, the high fidelity prototypes evolve into the finished product.

2.1.3 Testing paradigms

Testing a design is a fundamental factor to ensure that it meets the proposed usability goals. The evaluation can be done for both the prototypes and the final product following one or more of these evaluation paradigms: *quick and dirty*, *usability testing*, *field studies*, *predictive evaluation* and *expert review*. The first four of these approaches are described by Preece in [26], and the last one is described by Shneiderman in [32]:

Quick and dirty refers to testing a product in an informal way. It does not require any preparation or specific setting. It is a good way to validate the design and gather new ideas and suggestions to improve it.

Usability testing is a technique that requires users to perform carefully designed tasks in a lab setting. The main characteristic of this technique is that it is carefully controlled by the researcher to produce measurable results. While the users perform the tasks, several aspects of their performance are measured using metrics such as time-to-complete and number of mistakes made. After solving the tasks, users may be required to answer a questionnaire to gather more information about the experience with the product.

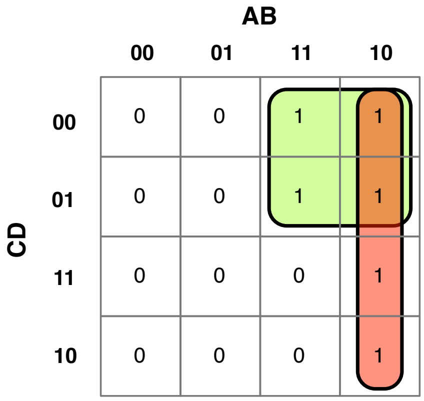
Field studies aim to test the design in its natural setting. The product is deployed into a realistic scenario and user activity is recorded using different means, which may include automatic logging, videotaping, answering questionnaires, and interviews. Field studies are a versatile tool that may be used for more than just testing a design: according to Bly [3], field studies can be used to identify opportunities for new technologies, determine requirements for design, facilitate introduction of technology and evaluate design of technology. However, field studies are more difficult to conduct and analyze than usability testing.

Predictive evaluation uses models of user behavior and heuristics to predict usability problems. Unlike the previous approaches, this one does not require the presence of the user. The accuracy of this approach is limited by the accuracy of the models used and the level of experience of the evaluator. This type of approach is recommended for situations were interaction with users is difficult or impractical.

Expert reviews are revisions of a model made by usability experts. An expert review can be conducted in different ways: heuristic evaluation (checking that the design conforms to design heuristics), guidelines review (ensure that the design conforms with the guidelines established by the organization), consistency inspection (ensure that all the controls in the interface are consistent), cognitive walkthrough (follow the typical tasks performed by the user), and formal usability inspection. The use of expert reviews can be very helpful for identifying flaws in the design. However, experts must be knowledgeable in both interaction design and the domain of the application in order to provide accurate feedback.

2.2 Information Visualization

The purpose of Information visualization is to exploit the natural abilities of the human visual perception in order to understand data that is not necessarily visual in nature. The human brain is permanently analyzing and interpreting the images it receives from the senses and memory. Visual characteristics like color, shape and alignment are processed much faster than textual descriptions. An example of a visualization that solves a problem using perception is a Karnaugh map. These types of maps are used in digital design to simplify Boolean algebra functions [17]. They solve the problem of finding a concise algebraic representation of a Boolean



CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	0	1
10	0	0	0	1

$F = AC' + AB'$

Figure 2.1: A Karnaugh map transforms the problem of reducing a Boolean function into one of grouping shapes. The function F has value 1 exactly when a cell contains a 1. Each cell is a term in the + (Boolean-or) equation for F. Complete rows, columns, or squares, in the Karnaugh map allow terms to be merged, resulting in a more compact representation of F.

function represented by a truth table. While the same task can be done by algebraic manipulation, using the Karnaugh map does not require remembering theorems of algebra. Instead, the problem is translated into one of grouping shapes together (see Figure 2.1). Larkin explains that diagrammatic representations are effective because the information needed to make inferences about the problem is present and explicit at a particular location [20]. This unloads the brain from doing searching and computing operations that are necessary in textual representations.

The use of visualization has the following advantages [36]: 1) It allows the viewer to comprehend huge amounts of data. 2) It allows the viewer to identify emerging properties that are not evident in textual representations. 3) It can reveal problems with the data itself, for example, errors in measurement can become obvious with the right representation. 4) It enables the understanding of both large - scale and small-scale aspects of the data. 5) Visualization facilitates the formulation of hypothesis.

Still, finding the right visualization can be a challenge. In [6], Chen *et al.* identifies three research issues for information visualization. First, finding the correct strategies and tools to visualize a set of data. Second, the need of generic criteria to assess the value of a visualization. Third, the use of visualizations as communication medium in multi-user environments.

Choosing the right representation is crucial for creating an effective visualization. While an effective representation can greatly simplify a problem–think of the Karnaugh map example–, others can make it even more confusing. As Tufte says, “Confusion and clutter are failures of design, not attributes of infomation”[35].

Moreover, the best representation may be different for different people. It has been demonstrated that some types are more effective for certain people based on some of their cognitive attributes like perceptual speed [7].

No less important are the choices of layout, colors, and fonts in the visualization. The right choice of visual attributes requires understanding of the human perception. Gestalt psychologists have studied these phenomena by considering a holistic approach, *i.e.* explaining a system by considering the relation of different parts as a whole as opposed to each individual part in isolation. The four Gestalt principles [8] used as a basis for graphic design are:

Proximity: elements that are close together are perceived as groups.

Similarity: elements that share the same shape, color, size or orientation are associated. Each one of these aspects can be used as a different "channels" to communicate information.

Continuity: the viewer tends to see lines and curves formed by the alignment of elements.

Closure: the viewer tends to complete or close shapes formed by elements and whitespace. Tufte calls this effect *"1 + 1 = 3 or more"* [35].

Multiple guides for good design have been published. To name a few, in [35], Tufte explains correct and incorrect ways of representing information based on extensive experience and examples from graphical design, cartography and art. Likewise In [21], Lidwell *et al.* present a set of principles of design aimed for multiple disciplines.

2.2.1 Graph drawing

A graph is an abstract representation of a set of objects that are connected together by a set of edges. Graphs can be used to represent problems that have the notion of adjacency among elements, as is the case of simple dependencies. Graph drawing techniques are a powerful tool for analyzing problems representable by a graph.

The drawing of graph structures has been widely studied and documented by graph theorists. Graph theory is described in [4, 12]. Different methods for drawing graphs are described in [2] and [33].

In [2], Di Battista *et al.* describe a methodology for drawing general graphs following a divide-and-conquer approach. This approach is motivated by the fact that many graph drawing algorithms are restricted to operate on a particular class of graph. Using this approach, if a graph does not fit into the requirements to apply a particular layout algorithm, a transformation is applied before using it and reverted at the end of the layout process.

Based on previous work on graph drawing, Sugiyama proposes a framework to classify automatic graph drawing methods in [33]. His framework considers five

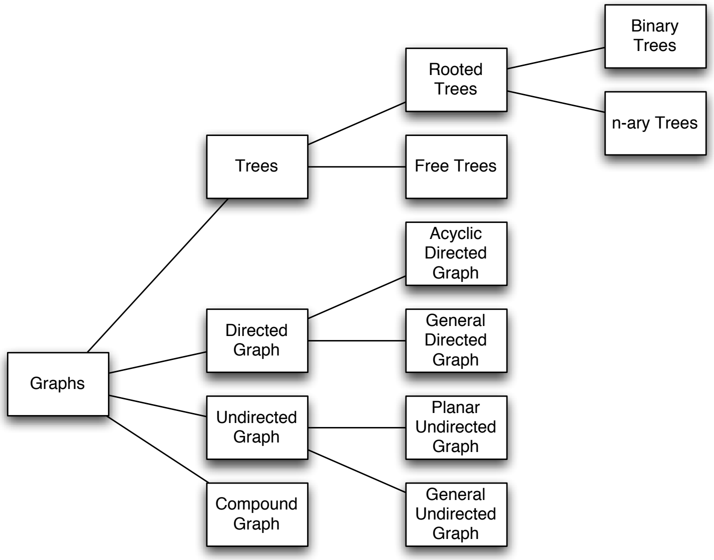


Figure 2.2: Classification of graphs[33].

different aspects: the type of graph, drawing conventions, drawing rules, priority of the relationships and feature of drawing algorithms.

Sugiyama's graph classification is shown in Figure 2.2. He distinguishes four main types of graph: trees, directed graphs, undirected graph and compound graphs. Trees can contain a defined root (called rooted) or be free. Directed graphs are divided in acyclic and general. Undirected graphs are classified in general and planar (*i.e.* graphs that can be drawn in a plane with no crossing between the edges). Finally, compound graphs have two different kinds of edges for expressing adjacency and inclusion. Distinguishing among the different types of graph is important as many drawing algorithms exploit the characteristics of each of them.

Drawing conventions refer to the way vertices and edges are represented and positioned in the drawing. The position of the edges can be free, allowing them to be located anywhere in the space, or constrained to a specific coordinate system. Coordinate systems include parallel lines, concentric circles, circle radii, and orthogonal grids (see Figure 2.3). Some coordinate systems can be obtained by combining some of the above; for example, the radii and concentric circles form a polar grid. The way Edges are represented is called routing conventions. The most common convention is to use straight, polygonal or curved lines to represent edges. However, other conventions, such as shape inclusion, can be used as well.

Drawing rules refer to specific criteria that aim to be met in the drawing of the diagram. These rules try to capture the aspects that make the diagram useful and esthetically pleasing. Some examples of drawing rules are shown in Table 2.1. Some the drawing rules conflict with each other. For example, displaying symmetries can cause the crossing of edges to be less than optimal. Therefore, a

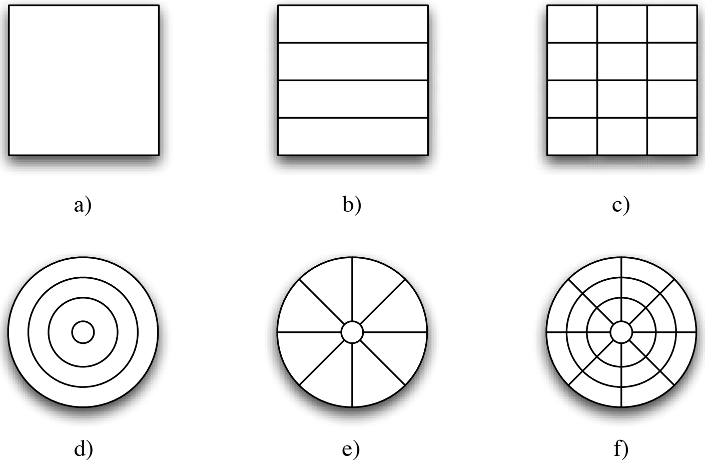


Figure 2.3: Different coordinate systems[33]. a) Free placement. b) Parallel lines. c) Grid. d) Circular. e) Radial. f) Combination of circular and radial (polar).

priority must be established among them.

Finally, a graph-drawing algorithm is a procedure that puts all these components together. It defines a placement for vertices and edges of a particular type of graph, following the drawing conventions and trying to respect the drawing rules as much as possible, according to the established criteria.

Some representative examples of graph drawing are dots and lines diagrams, circular diagrams, and tree-maps (see Figure 2.4).

Dots and lines is the most common way used to represent general directed or undirected graphs. They represent vertices as dots and edges as lines on free placement coordinate system. As routing convention, they may use straight lines, polygonal lines, or curves. If the graph is directed, edges are drawn as arrows.

Circular diagrams show general graphs using dots and lines in a polar coordi-

Table 2.1: Examples of drawing rules

	Drawing Rules
1.	Vertices are drawn in a straight line.
2.	Vertices are drawn in a circle.
3.	The size of the vertices is 10.
4.	The maximum number of allowed edge crossings is 10.
5.	The maximum number of allowed edge bends is 2.
6.	Vertices of high degree are positioned near the center.
7.	Drawing area is minimized.
8.	Angle between edges is maximized.
9.	Isomorphic subgraphs are shown identically.

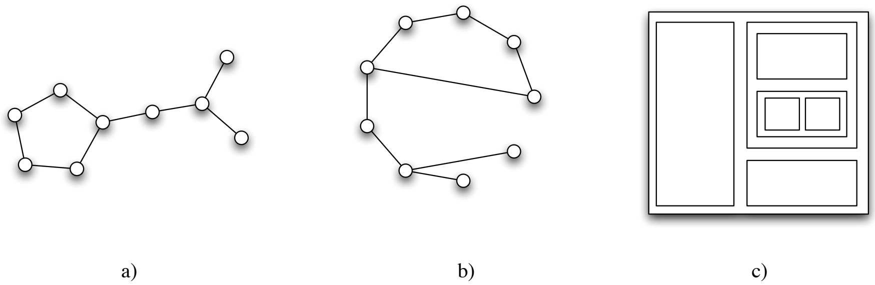


Figure 2.4: Graph drawing examples. a) Dot and lines. b) Circular layout using straight lines as edges. c) Tree-map.

nate system. In this type of diagrams, vertices are arranged around a circumference and edges are drawn using straight or curved lines. Edges may be bundled together to reduce clutter and allow to easily identifying their source and target. This technique is inspired by the way cables are bundled together in data centers. Algorithms for bundling edges are presented in [11], [14] and [15].

Tree-maps is a method for drawing n-ary trees, proposed by Shneiderman in [31]. Unlike the previous examples, this method does not use lines as drawing conventions for the edges of the tree. Instead, edges are represented by inclusion: nodes are represented as boxes and children nodes are included inside their parent's boxes. Tree-maps partition the design space, and the area of the box can represent an attribute of the node, for example, it's size.

When a graph is visualized in a computer screen, the user experience can be enhanced by adding interactivity. Interactive controls dynamically modify and animate the visual features of the graph (position, color and saturation of nodes and edges) to allow the user to easily understand and navigate it. A basic navigation technique is pan and zoom. More sophisticated techniques like Fisheye view [29] allow the user to zoom on a particular section of a diagram without loosing a global view of the graph. Other techniques extend static layouts by animating transitions. For example, in [41], Yee *et al.* explains an algorithm to add interactivity to a graph displayed using a radial layout to animate the change of a focus node. Graph drawing toolkits like Prefuse [13] and Flare [16] include support for interactive features.

2.3 Related projects

The following are visualization projects related to the Course Browser in diverse ways. Some of them tackle the same problem domain, while others provided inspiration for the design and evaluation of the user interface of the prototypes.

Visualization of the education system in Alberta

In [9], Fuite presented a visualization of the provincial education system in Alberta Canada. The purpose of this work is to study the education system using network analysis techniques. This work uses a graph visualization where the nodes represent courses and edges represent the prerequisites among them. To position the nodes, it uses a force directed layout, where the length of the edges is bound to how related the courses are to each other. The algorithm used by Fuite to calculate the weights of the edges is similar to the one presented in section 3.3.2; however, both algorithms were developed independently.

This work is similar to the Course Browser in that both visualize the prerequisites of courses at the University of Alberta. However, the Course Browser was designed as a tool for helping students to plan courses while Fuite's project was designed to perform high level analysis of the education system in the province. Another difference is that the Course Browser stores the complete requisite tree of each course while Fuite's tool simplifies the structure to create the network. Having the complete requisite tree is essential for performing course planning.

Thread arcs

The Thread Arcs project, presented by Kerr in [18], is a visualization technique for describing the structure of email threads (see Figure 2.5). It displays the messages as dots in a straight line, sorted by chronological order and then draws arcs among them to represent the *reply-to* relationship. The visualization offer some interactive features to allow the user to highlight the diagram according to the characteristics of the message, and inspect the messages. The advantages of the thread arcs visualization over other tree visualization are its compactness and its emphasis in chronology, which is fundamental in following an email conversation. One disadvantage is this visualization is that the diagram does not contain any text information about the messages, forcing the user to use the interactive features to discover this information. This also makes usefulness of the diagram limited when it is printed.

The evaluation criteria used to test the Course Browser visualizations were inspired in part by the one used by the Thread Arcs authors.

Circos

Circos[19] is a software tool that generates visualizations of genomic data and general 2-dimensional data. In the visualizations produced by this tool, data is displayed in a circular layout and the relationships among elements are drawn as arcs. The project was conceived to visualize relationships between genomes. The use of a circular layout was chosen to minimize the overlap among the lines that represent relationships, making the relationships easier to visualize.

Circos provides great flexibility for generating graphs in circular layout. Several

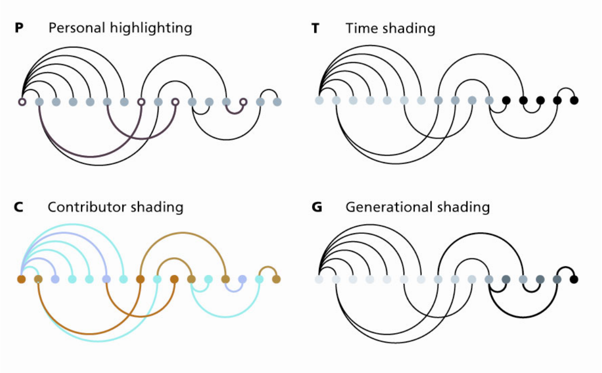


Figure 2.5: Email thread visualized using Thread Arcs using different highlight schemes. *Originally published by Kerr in [18].* © *2003 IEEE. Reproduced under permission.*

types of plots, like histograms, scatter plots and text, to name a few, can be embedded inside the circular graph. One disadvantage of the tool, however, is that the images generated are static. Circos is distributed as a command line tool with no graphical user interface for its operation, although the project offers an online version of the tool, called Table Viewer[1]. Figure 2.6 shows a table generated by this tool.

Flare

Flare[16] is a toolkit for creating interactive data visualizations that can be published on the Internet using the Flash plugin. It is based on the Prefuse toolkit, developed for the Java platform[13]. The Flare toolkit includes tools for performing data management, visual encoding, animation, and interaction. The framework is highly extensible and customizable. The visualizations for the Course Browser were created using this toolkit.

[1] http://mkweb.bcgsc.ca/circos/tableviewer/

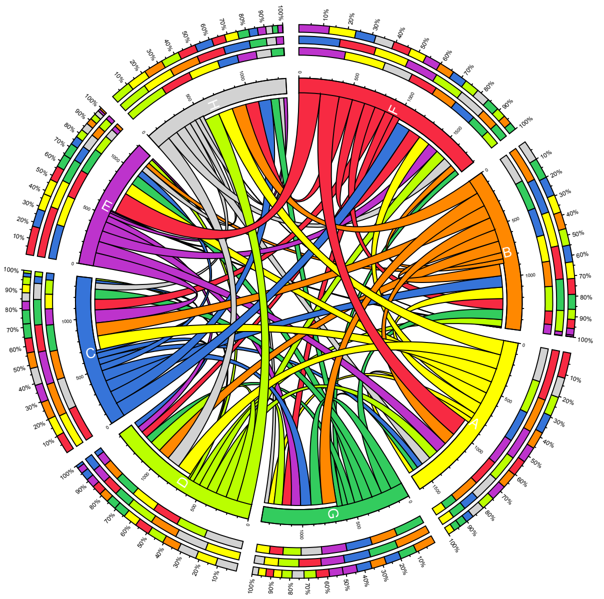


Figure 2.6: Random table visualization generated using the Circos table viewer.

Chapter 3

The Course Browser

3.1 Objectives

The development of the Course Browser had two main objectives: 1) Explore different ways to visualize the dependencies among courses in the University, gathering useful knowledge that could be used for the visualization of dependencies in other similar problem domains; 2) Implement a tool that students at the University of Alberta could use to plan their program.

The central part of the project was the development of visualizations to represent course requisites. To evaluate the different visualization designs, we used the following criteria:

Stability: Adding and removing relationships should not modify the layout of unaffected elements in the diagram.

Compactness: The diagram should be able to scale to thumbnails, if possible.

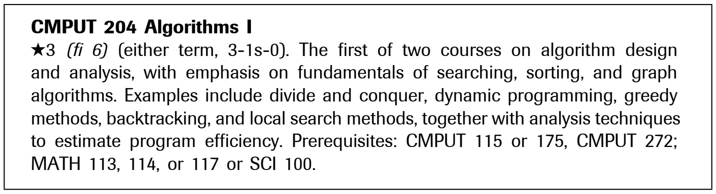
Attribute Highlighting: The nodes in the diagram should be able to display additional information about the course.

Scalability: The readability of the diagram should decrease gracefully as the complexity of the dependency tree increases.

Sense/Scanability: The user should be able to quickly understand the relationships between elements on the diagram by scanning it.

3.2 Requisite description language

All the information about the courses in the University of Alberta, including their prerequisites and corequisites are recorded in the university calendar. This information is available in printed and electronic form as well as in the course registration system of the university, called Bear Tracks. In all of these resources,



CMPUT 204 Algorithms I
★3 *(fi 6)* (either term, 3-1s-0). The first of two courses on algorithm design and analysis, with emphasis on fundamentals of searching, sorting, and graph algorithms. Examples include divide and conquer, dynamic programming, greedy methods, backtracking, and local search methods, together with analysis techniques to estimate program efficiency. Prerequisites: CMPUT 115 or 175, CMPUT 272; MATH 113, 114, or 117 or SCI 100.

Figure 3.1: Example of how courses are defined in the University of Alberta calendar[25]. The requisites of courses are contained in the description of the course and expressed in natural language.

the course requisites are specified as part of the description of each course, as shown in Figure 3.1.

Currently, there is no standard format for writing the course requisite descriptions across the university. Some departments have some conventions for using semi-colons and commas to separate the different elements, but no generalization is possible. Hence, some of the descriptions are ambiguous, making it unfeasible to build visualizations with this data right away. To solve this, the first step previous to creating the Course Browser was to define a language capable of expressing course requisites.

Expressing these requisites is not a trivial task. Some of them are formed with complex logical expressions. For example, in order to take the course *CMPUT 204: Algorithms I*–shown in Figure 3.1–the student must have taken *"CMPUT 115 or 175, CMPUT 272; MATH 113, 114, or 117 or SCI 100"* . Analyzing this expression we see that we have to take 3 courses: the first is a choice between CMPUT 115 and CMPUT 175, the second is always CMPUT 272, and the third is the one of MATH 113, MATH 114, MATH 117 and SCI 100. Thus, the language must be able to handle multiple level of nesting of logical expressions, using *AND* and *OR* operators.

Although the vast majority of course requisites are composed of expressions similar to that, there are also a few that include additional elements. For instance, some courses restrict two courses to be taken simultaneously, like *GENET 420*, that states that it *"may not be taken concurrently with BIOL 391"*; some courses, like *PHYS 261* require the that the student had previously taken some credits in 100-level courses from the same department; some of them, like *CHEM 299*, require students to have *"GPA of 2.5 or higher"*; others allow requisites to be waived with consent of the instructor.

In general, we identified three different types of information contained in the requisites of courses as given in the calendar: *requisites*, *equivalent courses* and *restrictions*. *Requisites* refer strictly to prerequisites and corequisites, *i.e.* courses that must be taken before or at the same time as the course in question in order to register for it. *Equivalent courses* are courses that cover similar content and therefore are not intended to be taken by the same students. In the calendar, these courses usually contain a sentence similar to this in their description: *"this course* *may not be taken for credit if credit has already been obtained in either MATH 115 or 118.” Restrictions* are additional conditions that may prevent the student from registering in the course; for example, requiring that the student hold a GPA greater than 2.5.

For the purpose of visualization of course dependencies, only the course requisite information is relevant. Therefore, we focused on defining a simple language capable of expressing this information well, rather than creating a complex language that tried to express all the information, including requisites, equivalent courses and restrictions. The elements of the requisite information are well defined, as they contain just courses and logical operators. In contrast, encoding restrictions would require an extensible rule engine capable of encoding restrictions according to different attributes of the student, course and program. That would enable administrators to add new rules to cover each special case, like requiring a minimum GPA, or restricting a course to a particular program of study. This task is outside the scope of this project. Courses whose descriptions contained equivalent courses and restrictions were flagged to inform the user to read the description to obtain this information.

The elements of the requisite description language are described in Table 3.1. The basic elements of the language are *pre*, *co* and *sync*. Each one of them must have a course code as content. The *pre* element represents a prerequisite, *i.e.* a course that must be taken before the course being described. The *co* element represents a course that must be taken at the same time or before the given course. The *sync* element is a strict version of the *co* element, as it represents a course that most be taken simultaneously with the given course. The *allOf*, *anyOf* and *atLeast* allow the construction of nested structures. Each one can contain one or more of the former elements. Finally, the *none* element is used to express that the course has no requisites.

Figures 3.2 and 3.3 show some examples of how to represent a course requisite using the requisite description language. In the first example, the containing element is of type *anyOf*, as the corse has two possible choices of requisites. The first of this choices is represented by an *allOf* element that contains three prerequisites, BIOCH 310, BIOCH 330 and BIOCH 330. The second choice is another *allOf* containing two requisites, BIOCH 203 and BIOCH 205. In the second example, the course requires the student to take three courses before and, therefore, the outer element is of type *allOf*. Inside, there is a *pre* element for the mandatory course and two *anyOf* elements containing the choices for two remaining courses.

The course codes contained by the basic elements may contain wild cards, expressed by the '*' symbol. The wild card may replace the course's department or any of the code digits. These syntactic elements are very practical for expressing requisites that include all the courses in a particular department and/or level. For example, *“Any 200-level CMPUT course”* is expressed as *“CMPUT 2**”*, *“any physics course”* can be expressed as *“PHYS ***”*, and *“Any 100-level course”* can

Table 3.1: Elements of the course requisite language

Element	Description	Content
<none/>	No requisites.	–
<pre> *code* </pre>	Prerequisite.	A string containing a course code.
<co> *code* </co>	Corequisites.	A string containing a course code.
<sync> *code* </sync>	Synchronized course.	A string containing a course code.
<allOf> ... </allOf>	All of. Satisfied if all of the sub-elements are satisfied.	[pre, co, sync, allOf, anyOf, atLeast]+
<anyOf> ... </anyOf>	Any of. Satisfied if at least one of the sub-elements is satisfied.	[pre, co, sync, allOf, anyOf, atLeast]+
<atLeast n=".."> ... </atLeast>	At least n of. Satisfied if at least n of the sub-elements are satisfied. n is a positive integer.	[pre, co, sync, allOf, anyOf, atLeast]+

expressed as “* 1**”.

The course requisite language presented is simple yet expressive enough to encode the course requisite information. Having the requisites encoded in a structured way enables the creation of graphical representations, automatic dependency checkers and even the generation of text that describes the requisites in natural language in a consistent and clear way.

3.2.1 Canonical Ordering

One characteristic the the requisite language is that all container elements are commutative. For example, the expression `allOf(1, 2, 3)` has the same meaning as the expression `allOf(3, 2, 1)`. This can be a problem for comparing and visualizing requisite trees, where, ideally, each possible requisite tree should have a unique representation. To address this, we propose a canonical order for the requisite nodes, defined by the following rules:

1. *Operators must obey the following precedence:* pre, co, sync, allOf, anyOf, atLeast, none.
2. *For leaf expressions of the same type, the order is lexicographical.*
3. *For non-leaf expressions of the same type, the ordering is done by comparing the left-most leaf.*

These rules guarantee that any tree has a unique canonical representation in the requisite language.

> **BIOCH 455:** Prerequisites: BIOCH 310, 320, and 330, or BIOCH 203 and 205, all with a minimum grade of B- or consent of Department.

```
<anyOf>
  <allOf>
      <pre>BIOCH 310</pre>
      <pre>BIOCH 320</pre>
      <pre>BIOCH 330</pre>
  </allOf>
  <allOf>
      <pre>BIOCH 203</pre>
      <pre>BIOCH 205</pre>
  </allOf>
</anyOf>
```

Figure 3.2: Example of a requisite encoding using the course requisite language. On the left: the course description for BIOCH 455 as it appears in the University of Alberta calendar. On the right: the representation of this requisites using the requisite language.

3.3 Course Browser's User Interface

The Course Browser is an interactive tool to explore a catalog of courses. It is specially designed to allow an easy exploration of the course dependencies through the use of multiple visualizations. A screen shot showing the main elements of the course browser user interface can be seen in Figure 3.4.

The prototype described in this section was developed during the third iteration of a prototyping process. Unlike previous prototypes, this version was intended to be fully functional for students. In order to make the software easier to learn, user interface elements in this version include a more familiar layout that resembles an email client like *Thunderbird*[1] or a music collection manager like *iTunes*[2].

This prototype contained information about all the courses in the faculty of Science at the University of Alberta, with approximately 1000 entries. However, the design is scalable and capable of managing the entire catalog of the university, which contains approximately 7500 courses.

To be able to browse a catalog of this magnitude, the Course Browser groups courses into collections. A *collection* is a group of courses that share common characteristics, for example, belonging to the same department. The prototype includes predefined collections by department, number of credits and number of requisites. Each collection is defined by a logical statement, allowing the support of user-defined collections in future versions.

Once a collection is selected, the user can explore the courses that belong to it

[1] Thunderbird is an email application developed by the Mozilla Foundation. For more information, visit http://www.mozillamessaging.com/en-US/thunderbird/

[2] iTunes is music collection manager and player developed by Apple Inc. For more information, visit http://www.apple.com/itunes/

CMPUT 204: Prerequisites: CMPUT 115 or 175, CMPUT 272; MATH 113, 114, or 117.

```
<allOf>
  <anyOf>
       <pre>CMPUT 115</pre>
       <pre>CMPUT 175</pre>
  </anyOf>
  <pre>CMPUT 272</pre>
  <anyOf>
       <pre>MATH 113</pre>
       <pre>MATH 114</pre>
       <pre>MATH 117</pre>
  </anyOf>
</allOf>
```

Figure 3.3: Example of a requisite encoding using the course requisite language. On the left: the course description for CMPUT 204 as it appears in the University of Alberta calendar. On the right: the representation of this requisites using the requisite language.

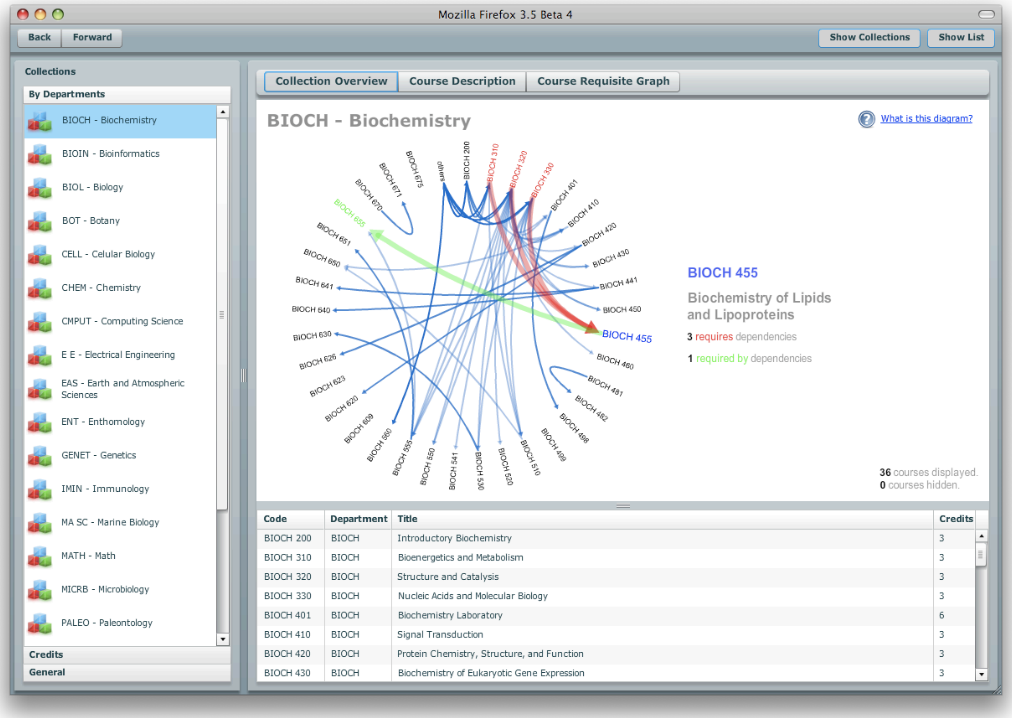
using an overview diagram or a list view. The *collection overview* is an interactive visualization of the courses in a collection. This diagram displays how courses inside the collection relate to each other, according to their requisite information. The *course list* uses a more conventional data grid to display the courses in a tabular way. The grid contains columns for the code, title and number of credits of each course. The columns are sortable, allowing the user to quickly organize and locate courses.

When the user selects a course, either by using the *collection overview diagram* or the *course list*, two more views become available to explore the details of the selection. First, the *course description* view (see Figure 3.5) shows all the details of the course available in the course calendar: course code, department, number of credits, frequency and description. In addition, the requisite information is displayed using a *requisite box diagram*. Second, a *Course Requisite Graph* (see Figure 3.6) displays all the requisites of the course, including indirect dependencies.

All the diagrams are interactive and allow the user to navigate to the related courses by clicking on their code. The navigation history is recorded to allow the user to revisit previously consulted courses using the Back/Forward buttons, located on the top-left corner of the screen.

3.3.1 Requisite box diagram

The *requisite box diagram* is a graphical representation of the requisites of a course, encoded using the language introduced at the beginning of the chapter. In the Course Browser, it is used in both the *course description* and *course requisite graph* views. Some examples of this diagrams are shown in Figure 3.7.



BIOCH 455

Biochemistry of Lipids and Lipoproteins

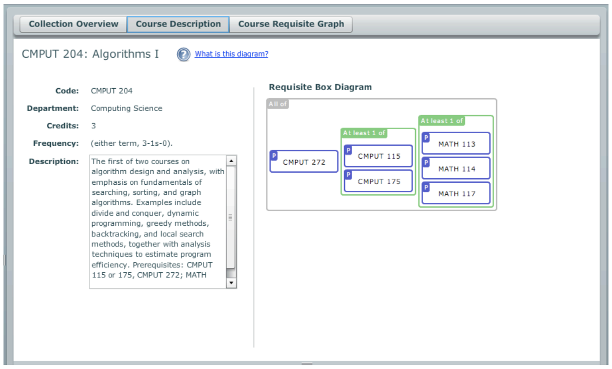
3 requires dependencies

1 required by dependencies

36 courses displayed.
0 courses hidden.

Code	Department	Title	Credits
BIOCH 200	BIOCH	Introductory Biochemistry	3
BIOCH 310	BIOCH	Bioenergetics and Metabolism	3
BIOCH 320	BIOCH	Structure and Catalysis	3
BIOCH 330	BIOCH	Nucleic Acids and Molecular Biology	3
BIOCH 401	BIOCH	Biochemistry Laboratory	6
BIOCH 410	BIOCH	Signal Transduction	3
BIOCH 420	BIOCH	Protein Chemistry, Structure, and Function	3
BIOCH 430	BIOCH	Biochemistry of Eukaryotic Gene Expression	3

Figure 3.4: Screen shot of the user interface of the Course Browser.



CMPUT 204: Algorithms I

Code: CMPUT 204

Department: Computing Science

Credits: 3

Frequency: (either term, 3-1s-0).

Description: The first of two courses on algorithm design and analysis, with emphasis on fundamentals of searching, sorting, and graph algorithms. Examples include divide and conquer, dynamic programming, greedy methods, backtracking, and local search methods, together with analysis techniques to estimate program efficiency. Prerequisites: CMPUT 115 or 175, CMPUT 272; MATH

Figure 3.5: The Course description view of the Course Browser.

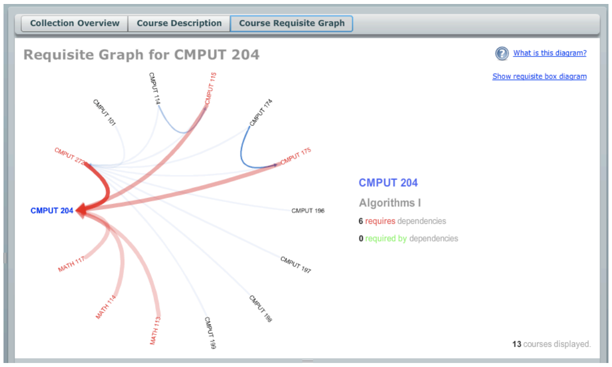


Figure 3.6: The Course requisite graph view of the Course Browser.

In the diagram, a box labeled with the course code represents each leaf element of the requisite tree, *i.e.* prerequisites and corequisites. Container elements, like *allOf*, *atLeast* and *anyOf*, are also drawn as boxes, with their children elements contained inside. The way that children are positioned inside the box depends on the type of the language element. *AllOf* boxes arrange their children in an horizontal way, while *anyOf* and *atLeast* boxes organize them in a vertical way. This way, the user can easily identify choices from mandatory requirements. When the requisite tree is composed of just *allOf* and *anyOf* elements, the number of columns in the diagram indicates the number of mandatory courses that the user most take. This way, courses that require few courses will have thin diagrams, and courses that have lots of choices will have tall diagrams. These features make the diagram easy to scan for basic information.

Each type of box in the diagram is drawn in a distinctive color. Thus, elements can be distinguished using multiple perceptual channels: shape, color and alignment. In addition, a title with the type of the element is drawn in the top left corner of each box, to make it easier for users to understand the meaning of the diagram and preserve all its semantics even when it is displayed in monochromatic media, for example, if printed in black and white.

This type of diagram shares some similarities with the treemap layout [31]. Both of them represent a node of the tree by a box that contains their children. Their main difference is that treemaps are a general purpose diagram for representing trees. In the treemap, there is no differentiation of the types of the nodes. The requisite box diagram, on the other hand, was created specifically for representing course requisites. It differentiates among different types of subtrees in the requisite language and defines coloring and arrangement rules for each of them.

The requisite box diagram complies with the criteria presented at the beginning of the chapter. It is compact, as it does not take a lot of space and it scales gracefully as the complexity of the requisite tree increases. The canonical ordering of the requisite language makes the diagram stable, as there is only one representation for any given requisite tree. It provides good sense and scalability as it uses multiple

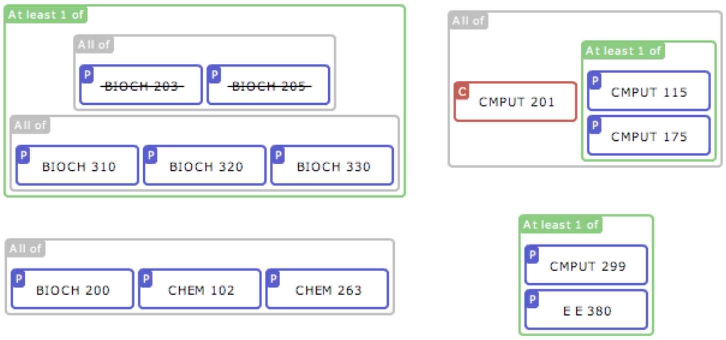


Figure 3.7: Four examples of the *requisite box diagram*.

cognitive channels to encode information. Finally, It is easy to be augment in order to highlight attributes. For example, the codes of obsolete courses (courses that are no longer part of the catalog, but still appear in requisites), are emphasized with a strikethrough line.

3.3.2 Circular diagrams

The Course Browser prototype contains two diagrams designed to display direct and indirect dependencies between courses: the *collection overview* and the *course requisite graph*. Both of them share the same design principles, so we refer to them as *circular diagrams*. Some examples of these diagrams are shown in Figures 3.4 and 3.6.

The purpose of these diagrams is not to encode all the complete information about the tree, but to give the user an overview of how courses are related. The arcs hide part of the complexity of the tree, but in exchange offer a transitive view of the course requisites.

Circular diagrams position the nodes along a circumference and display curved arrows between them to express their relationship. This way, no overlapping between the nodes and arrows is possible, as is the case with other graph layouts. Nodes are sorted alphabetically along the circumference to allow the user to quickly locate the desire course. Arrows show a relationship between two courses in the dependency tree; more precisely, each arrow represents a path from the root of the requisite tree to one of its leaves, which must contain a *pre*, *co* or *sync* element. This path can traverse multiple nodes of type *allOf*, *anyOf* or *atLeast*.

In the diagram, choices and mandatory requirements are characterized by the saturation of their arrows. This property is determined by a recursive weight distribution algorithm. The alpha channel of the node is defined by interpolating the weight of each target node in the interval $[0.5, 1]$. The weight assignment works as follows: The root of the dependency tree is assigned a weight of 1; then, the weight is distributed to each of their children following the following rules in a

	Weight
`<allOf>`	1
`  <anyOf>`	1
`    <pre>CMPUT 115</pre>`	1/2
`    <pre>CMPUT 175</pre>`	1/2
`  </anyOf>`	
`  <pre>CMPUT 272</pre>`	1
`  <atLeast n="2">`	1
`    <pre>MATH 113</pre>`	2/3
`    <pre>MATH 114</pre>`	2/3
`    <pre>MATH 117</pre>`	2/3
`  </anyOf>`	
`</allOf>`	

Figure 3.8: Example of the weight assignment for calculating the saturation of the arrows in the circular diagrams.

recursive way:

1. If the node is of type *allOf* and its weight is w, each of its immediate children are given the weight w.

2. If the node is of type *anyOf* and its weight is w, each of its immediate children are given the weight w/c, where c is the number of children of the node.

3. If the node is of type *atLeast* and its weight is w, each of its immediate children are given the weight $n * w/c$, where c is the number of children of the node and n is the parameter of the node.

An example of the weight assignment is shown in Figure 3.8: The root node gets assigned a value of 1. As the root node is of type *allOf*, each of its children is assigned a value of 1 as well. The first children is of type *anyOf*, therefore, its weight is distributed evenly among its children, giving each of them a weight of $1/2$. The second child of the root is a leaf, so it has no children to assign weights to. Finally, the third child of the root is an *atLeast* element with a parameter value of 2, and 3 children. Thus, each of its children is given a weight of 2/3.

The design of the circular diagrams meets the design objectives proposed at the beginning of the chapter. As the course titles are shown sorted at the edge of the circle, the diagram remains stable as new nodes are added, and it is easy to scan to locate a particular course. The size of the diagram is independent of the number of nodes and it can be adjusted to fill the size of the screen. Its complexity increases gracefully as more nodes are added. Finally, it allows the user to distinguish courses with different characteristics by scanning the diagram: fundamental courses have many arrows pointing to them; courses with no requisite have arrows originating from them; courses that are not required by other courses have no arrows pointing to them; finally, courses that require many courses have many opaque arrows originating from them while courses that require a few courses but have many choices have many translucent arrows.

Unfortunately, as the number of courses increases, the circular diagrams become complex and difficult to read. To solve this, the number of nodes in each diagram was limited to 150 elements and interactive features were implemented. For example the user may highlight a course code by positioning the mouse pointer on top of its code to consult basic information about it. Other interactive features particular to each of the diagrams are described in the following sections.

Collection Overview

The *Collection Overview* diagram is a type of circular diagram that displays the courses of a particular collection. Its objective is to show how the courses inside the collection depend on each other. This diagram is useful to identify the fundamental courses in a collection as well as courses with no requisites, and courses not required by other courses.

In the diagram, all courses that belong to the given collection are shown in the edge, and all the remaining courses in the library are represented by a single node labeled “others”. This node is important as it allows the user to identify when a course requires courses outside the collection.

The diagram has several interactive features. When the user hovers a course code with the mouse pointer, a panel appears at the right of the diagram displaying basic information about the course. In addition, the arrows are highlighted using different colors for incoming and outgoing requisites. When the user selects a course–by clicking its code on the diagram or using a different view–the diagram filters the nodes that are related directly or indirectly to the selection and hides all unrelated the arcs. This considerably enhances the readability of the diagram.

Course Requisite Graph

The *Course Requisite Graph* is a type of circular diagram that displays all the direct and indirect requisites of the selected course. This type of diagram is useful in course planning as it displays all the possible courses that must be taken before a given course. A course must be selected in order to display this diagram.

This diagram shares some of interactive features of the *Collection Overview Diagram*. When the course code is hovered by the mouse, the course information is displayed and the arrows are highlighted. Clicking a node changes the selection its respective course. In addition to this, an optional popup window displays the *Requisite box diagram* of the highlighted course.

3.4 Early Prototypes

The Course Browser prototype presented in the last section was designed in the third iteration of a prototyping process. The following sections describe the earlier designs of the application, discussing their advantages and disadvantages.

3.4.1 Card Prototype

The first prototype of the Course Browser used playing cards as a metaphor for displaying courses. In this metaphor, cards represent courses and stacks of cards represent groupings of courses which share common characteristics, for example, belonging to the same department.

The card metaphor supports the following operations (shown in Figures 3.9 and 3.10):

Move: both cards and stacks may be moved in the canvas.

Create card: creates a card to represent a piece of information.

Create stack : Creates new card stack. Stacks can be created in free style– allowing any card to be added, or have a selection criteria, in which case all the cards that comply with the criteria will be attracted to it.

Add card to stack : Adds a card to stack.

Bring card to top: brings a card from the middle of the stack to the top of the stack.

Expand stack: opens the stack to allow the user to view a portion of each card, named *header*. The header usually contains the identification of the element that the card represents.

Expand stack with fish-eye effect: If the stack contains many items, the expand view may overflow the screen. This version of the expand just expands just a segment of the stack, but allows the user to navigate it.

Close stack: The stack is closed, meaning that each card is positioned on top of each other. A small offset is left to give an illusion of depth.

Flip card: A card can be flipped to hide its contents. This can be used for a variety of purposes: for example, while filtering a stack, all filtered cards could be flipped, allowing the user to quickly identify the pertinent cards. Another possible use is in lazy-loading systems: cards can be displayed upside down and only when the user flips them would the system fetch their content and reveal them to the user.

Mark card: The card is marked. A small red spot is shown in one of the corners.

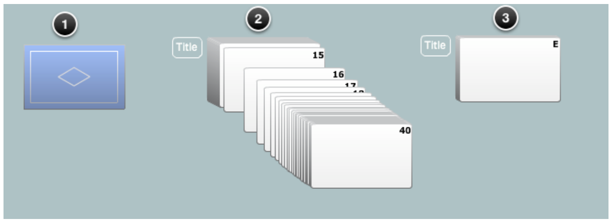


Figure 3.9: Some operations of the card UI metaphor. 1) Card in Flipped state. 2) Expanded stack with fisheye effect. 3) Closed stack.

Select card: The deck splits to reveal just the selected card. The rest of the cards remain in their previous state (closed or expanded).

Wiggle card: A card wiggles to represent that it will be affected by an action. This effect provides a sense of the impact of an action before it happens. In our prototype, if the user positions the mouse pointer over a selected course, all related courses would wiggle indicating its relationship with it.

In this prototype, each course was represented by a card. The front face of the card contained key information about the course, distributed in different areas of the card surface: code, department, title and credits. Stacks grouped courses by department. A screenshot of this prototype is shown in Figure 3.10.

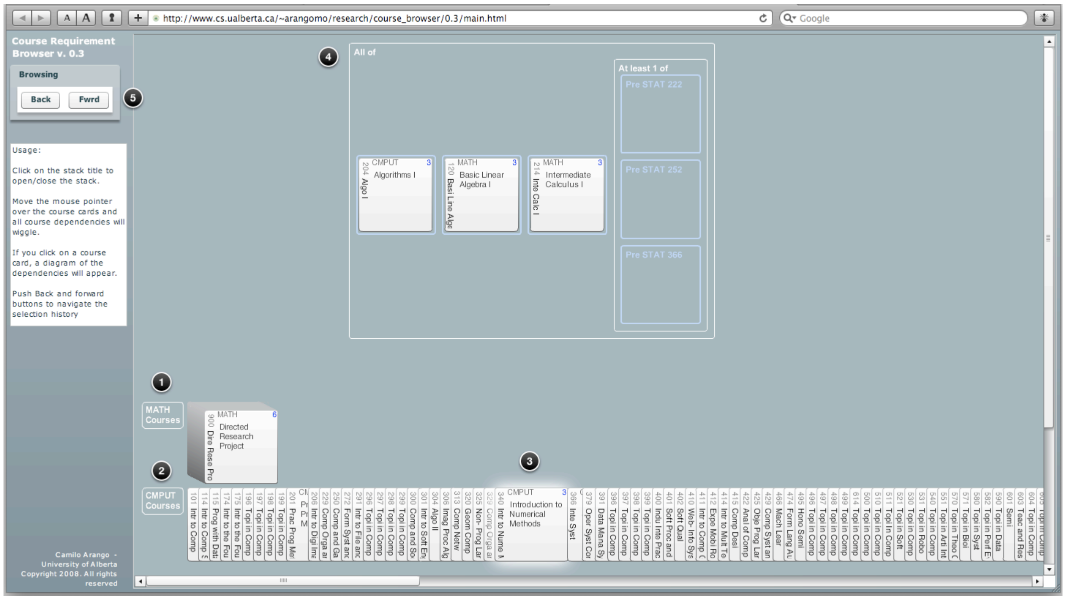
The user interface showed one stack per department. In this prototype we only included information of two of the departments in the university: Computing Science and Mathematics. When the user clicked on the title of the deck, it will expand, showing all the titles of the courses. Whenever the mouse pointer hovered a card, the dependencies of the respective course (directly and indirectly) wiggled to indicate that those courses are related. Upon click, the card was selected and all related cards moved to create a dependency diagram on top of the deck. This diagram was an early version of the *requisite box diagram* described in Section 3.3.1.

In addition, the interface provided *back* and *forward* controls to navigate the selection history. This allowed the user to quickly revisit courses consulted previously.

Discussion

The card metaphor is intuitive and provides a rich set of interactions. The different operations are easily understood by the user. The wiggle animation provides an effective way to show related courses.

However, this metaphor presents several limitations that make it unpractical for representing course dependencies. In the first place, the cards have a small fixed area to show information. To be readable, each card must be rendered at a large scale, which limits the number of cards that can be shown in the screen at



Usage:

Click on the stack title to open/close the stack.

Move the mouse pointer over the course cards and all course dependencies will wiggle.

If you click on a course card, a diagram of the dependencies will appear.

Push Back and forward buttons to navigate the selection history

Camilo Arango - University of Alberta Copyright 2008. All rights reserved

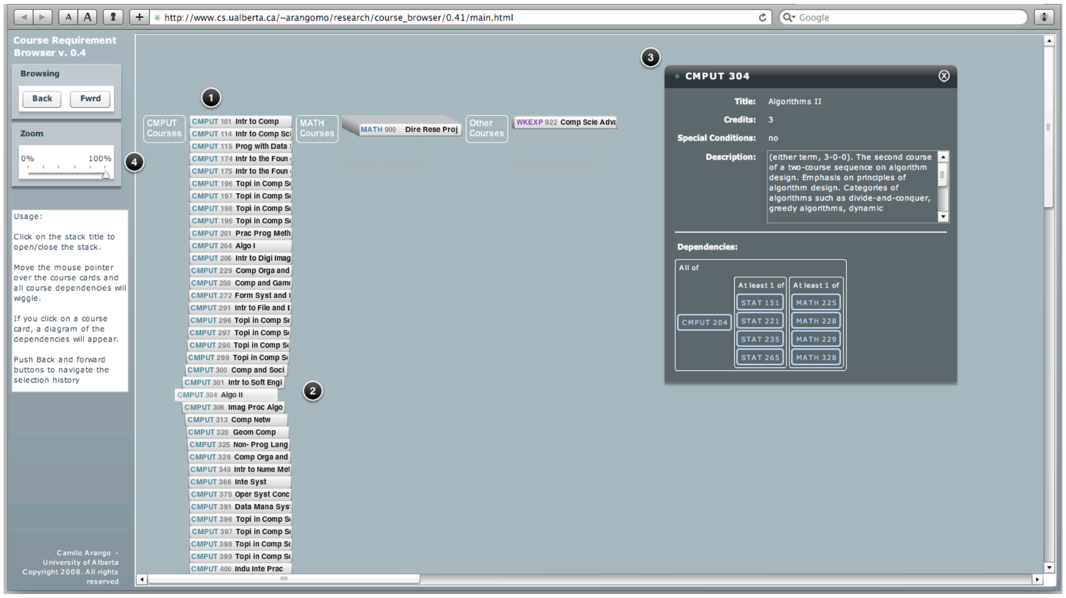
Figure 3.10: The Course Browser Card Prototype. 1) A closed stack containing courses from the Math department. 2) An expanded stack. 3) The selected card. 4) A Dependency box diagram. 5) Buttons for navigating browsing history.

a given time. For collections of hundreds to thousands of elements, as is the case of the course catalog, this is a big disadvantage. Opening a stack with the courses of a department would always overflow the screen. Another caveat of the card metaphor is that, to be consistent, each element should be represented by a single card. In the course catalog, it is possible for a course to appear more than once in the dependency diagram. Duplicating cards can cause confusion as it breaks the normal rules of physical objects.

Another problem identified with the prototype, was that the wiggle animation showed all related courses, direct and indirect, while the diagram could only show direct dependencies. This made the interaction confusing as only a fraction of the cards that wiggled would move to form the diagram. However, creating a diagram that included indirect dependencies would contain too many elements and easily overflow the screen.

3.4.2 Tiles Prototype

The tile browser prototype aimed to correct some of the problems with the previous prototype, tweaking some of the characteristics of the cards metaphor. A screenshot of this prototype is shown in Figure 3.11. In this version, instead of displaying all the course information in cards, smaller tiles were used. A tile was essentially a small version of the card, including only its header which, in this case, contained the code and title of the course. The department name was identified by a distinctive color. All the text was displayed horizontally to enhance readability. In addition,



http://www.cs.ualberta.ca/~arangomo/research/course_browser/0.41/main.html

Course Requirement Browser v. 0.4

Usage:

Click on the stack title to open/close the stack.

Move the mouse pointer over the course cards and all course dependencies will wiggle.

If you click on a course card, a diagram of the dependencies will appear.

Push Back and forward buttons to navigate the selection history

Camilo Arango - University of Alberta Copyright 2008. All rights reserved

CMPUT 304

Title: Algorithms II

Credits: 3

Special Conditions: no

Description: (either term, 3-0-0). The second course of a two-course sequence on algorithm design. Emphasis on principles of algorithm design. Categories of algorithms such as divide-and-conquer, greedy algorithms, dynamic

Dependencies:

Figure 3.11: The Course Browser Tiles Prototype. 1) Expanded tile. 2) Selected tile. 3) Inspector window displaying information of the selected course. 4) Zoom control.

the expansion of the deck was performed from top to bottom instead of left to right.

All the remaining information about a course was displayed in an inspector window. This window was a semi-transparent panel floating on top of the canvas. It displayed the title, number of credits, detailed description of the course and the second version of the requisite box diagram. The inspector window displayed the information of the currently selected course. It was possible to open extra inspector windows for different courses to compare their characteristics.

A zoom control allowed the user to change the magnification of the display. In addition, a layout effect was added to ease the identification of the selected course at low magnification levels: all the cards in the vicinity of the selection were pulled to the left a distance proportional to the distance between them and the selection. This would create an arc, directing the user visual attention to the selection.

Finally, the wiggle animation was preserved to reveal related courses, but the cards no longer moved to show the dependency diagram. This last step was unnecessary as the diagram was already being displayed on the inspector window.

Discussion

This prototype solved some of the problems with the previous prototype. The space was used more efficiently as more elements could be seen at the same time. The zoom feature allowed the user to choose the granularity of the view.

Displaying the dependency diagram in the inspector window sacrificed some of the interactivity, but eliminated the problem of having duplicate courses in the same diagram. It also made the diagram more compact and easy to read.

Expanding the stacks vertically and displaying the header information horizontally made the course title easier to read, but the expanded stacks overflowed the screen faster, as most computer displays have a wide aspect ratio. A better mechanism for browsing the courses inside a department was necessary.

The major limitation of this prototype was the lack of visualization of indirect dependencies. The wiggle effect gave the user some clue of which courses are related, but did not tell how. In particular, with courses that required just one course but gave many choices (*e.g.* courses that required any 100-level course), many tiles would wiggle, giving the wrong impression that the course required many courses. Even worse, as the wiggling movement included indirect dependencies, this effect propagated to any course that require a course with that characteristic.

Another aspect to improve was scalability. Loading all the courses in a university would require up to a hundred stacks. This imposes the problem of how to layout the stacks in the canvas. A horizontal layout, as done in the prototype, avoids overlapping the tiles when the stacks expand, but would require the user to scroll the view find a particular collection.

Both the cards and tile metaphors proposed an interesting paradigm suitable for collections with the following characteristics: 1) each element should be represented only once; 2) the total number of elements in the display must be from tens to hundreds; 3) the amount of information to display about the element may be fitted in a small area. These limitations made this metaphor inadequate for displaying all the courses in the university and, therefore, the final prototype of the course browser took a different approach for visualizing the course catalog. Nevertheless, this prototype preserved some of the concepts introduced in the cards and tiles prototype. In particular, the requisite diagram was further developed and included. This diagram preserves the representation of courses as small tiles. Likewise, the stacks evolved into the concept of collections.

Chapter 4

Usability Study: Methodology

We conducted a usability study to assess the Course Browser according to ease to use, easiness to learn, effectiveness to perform course planning tasks and confidence of the results. For comparison, we used the course catalog of Bear Tracks, a web-based tool that students presently use to browse the course catalog at the university. We asked students to perform equivalent tasks with both tools and measured the time-to-solve and accuracy of their answers. Then we asked the students about their experience and gathered their suggestions on how to improve the tool.

It is important to note that the Course Browser should not be considered a replacement for Bear Tracks. It was designed to leverage and enhance the functionality of only one of its modules, namely, the course catalog. We wanted to test whether the Course Browser provides a better set of tools for course planning than the current tools available. We believe that, ultimately, many of the features of the Course Browser could be integrated into Bear Tracks itself.

4.1 Bear Tracks

Bear Tracks is the course information and registration system of the University of Alberta. It hosts the course catalog, handles the registration of courses and keeps record of the grades. It also provides services related to admissions, academics, financial information for students, and payroll for employees.

For the usability study, we chose to compare the Course Browser against the feature of Bear Tracks that allows the user to search the course catalog. Students use this feature while planning their program to find which courses are available and what are their requisites.

The navigation scheme for the *browse course catalog* feature is described by Figure 4.1. It follows these steps: 1) the user inputs a search for a course; 2) the system displays a list of the courses that matches the search criteria; 3) the user accesses the detail information of the course; 4) the user may go back to the search results or start a new search.

When the *browse course catalog* option is selected from the main menu, a

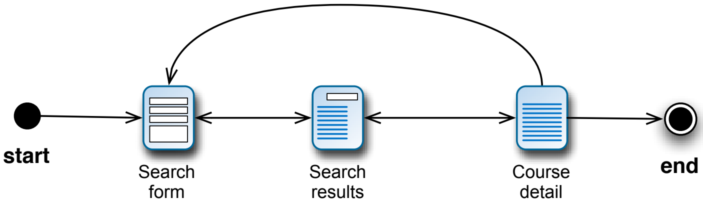
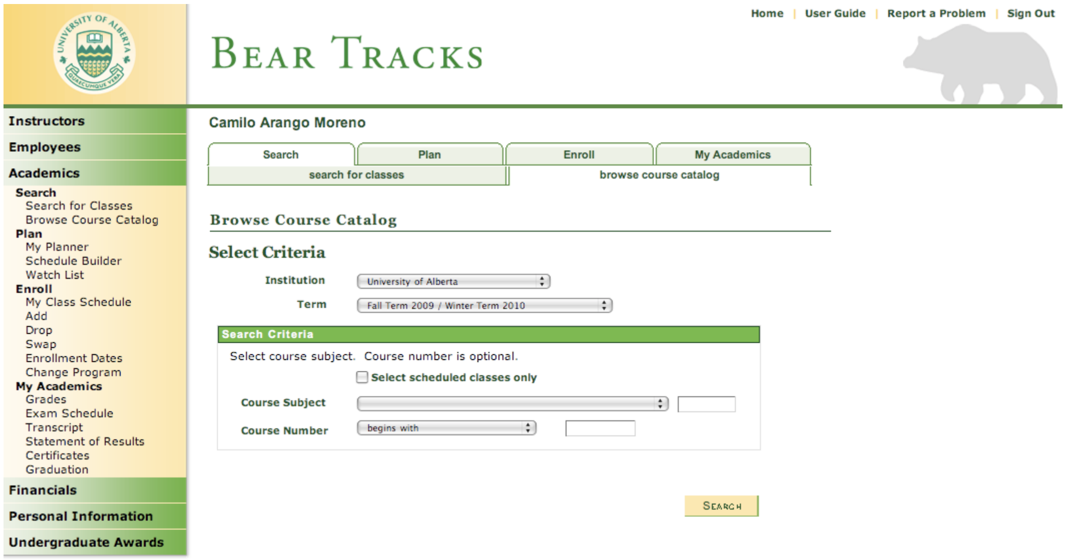


Figure 4.1: Bear Tracks' course catalog navigation chart.



Camilo Arango Moreno

Browse Course Catalog

Select Criteria

Institution: University of Alberta

Term: Fall Term 2009 / Winter Term 2010

Search Criteria

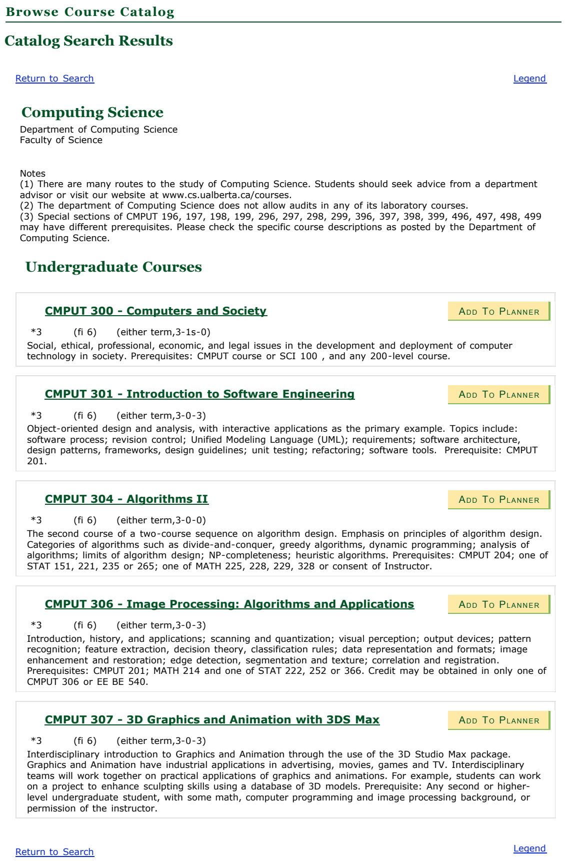
Select course subject. Course number is optional.

□ Select scheduled classes only

Course Subject

Course Number: begins with

Figure 4.2: Bear Tracks course catalog search screen



Browse Course Catalog

Catalog Search Results

Computing Science
Department of Computing Science
Faculty of Science

Notes
(1) There are many routes to the study of Computing Science. Students should seek advice from a department advisor or visit our website at www.cs.ualberta.ca/courses.
(2) The department of Computing Science does not allow audits in any of its laboratory courses.
(3) Special sections of CMPUT 196, 197, 198, 199, 296, 297, 298, 299, 396, 397, 398, 399, 496, 497, 498, 499 may have different prerequisites. Please check the specific course descriptions as posted by the Department of Computing Science.

Undergraduate Courses

CMPUT 300 - Computers and Society
*3 (fi 6) (either term,3-1s-0)
Social, ethical, professional, economic, and legal issues in the development and deployment of computer technology in society. Prerequisites: CMPUT course or SCI 100 , and any 200-level course.

CMPUT 301 - Introduction to Software Engineering
*3 (fi 6) (either term,3-0-3)
Object-oriented design and analysis, with interactive applications as the primary example. Topics include: software process; revision control; Unified Modeling Language (UML); requirements; software architecture, design patterns, frameworks, design guidelines; unit testing; refactoring; software tools. Prerequisite: CMPUT 201.

CMPUT 304 - Algorithms II
*3 (fi 6) (either term,3-0-0)
The second course of a two-course sequence on algorithm design. Emphasis on principles of algorithm design. Categories of algorithms such as divide-and-conquer, greedy algorithms, dynamic programming; analysis of algorithms; limits of algorithm design; NP-completeness; heuristic algorithms. Prerequisites: CMPUT 204; one of STAT 151, 221, 235 or 265; one of MATH 225, 228, 229, 328 or consent of Instructor.

CMPUT 306 - Image Processing: Algorithms and Applications
*3 (fi 6) (either term,3-0-3)
Introduction, history, and applications; scanning and quantization; visual perception; output devices; pattern recognition; feature extraction, decision theory, classification rules; data representation and formats; image enhancement and restoration; edge detection, segmentation and texture; correlation and registration. Prerequisites: CMPUT 201; MATH 214 and one of STAT 222, 252 or 366. Credit may be obtained in only one of CMPUT 306 or EE BE 540.

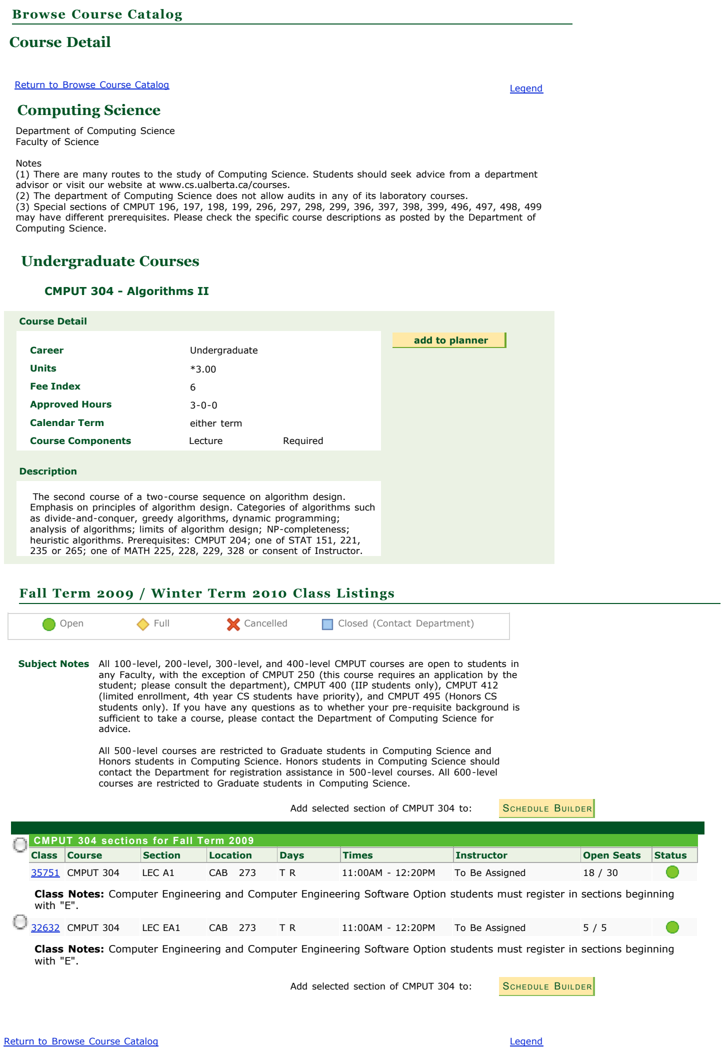
CMPUT 307 - 3D Graphics and Animation with 3DS Max
*3 (fi 6) (either term,3-0-3)
Interdisciplinary introduction to Graphics and Animation through the use of the 3D Studio Max package. Graphics and Animation have industrial applications in advertising, movies, games and TV. Interdisciplinary teams will work together on practical applications of graphics and animations. For example, students can work on a project to enhance sculpting skills using a database of 3D models. Prerequisite: Any second or higher-level undergraduate student, with some math, computer programming and image processing background, or permission of the instructor.

Figure 4.3: Bear Tracks course catalog result screen. This shows the results for courses in the Computing Science department with codes beginning with "30".

search form is displayed. The search form is filled in two phases: Initially, the form displays only a term selector that restricts the search to the courses taught during the spring/summer terms or the fall/summer terms; once the desired term is chosen, the system shows the rest of the search fields (see Figure 4.2). The system allows the user to search using filters by department code and course number. The course number filter is flexible, allowing the user to choose among different operators, like *begins with*, *greater than or equal to*, *exactly*, and *less or equal to*. This feature allows the user to search for an specific course as well as perform more comprehensive queries like *"search all the courses in the Physics department"* or *"search for courses in the Computing Science department with codes beginning with the number 30"*.

The results of the query are presented as a list that contains the course's code, title, and description, as shown in Figure 4.3. To see the complete information of a course, the student can click on the course title's hyperlink.

The course view shows the full calendar description of the course (Figure 4.4). The first section of the page displays basic course details including units, fee index,


Return to Browse Course Catalog
Legend
add to planner
Schedule Builder
Schedule Builder
Return to Browse Course Catalog
Legend

Browse Course Catalog

Course Detail

Computing Science

Department of Computing Science
Faculty of Science

Notes
(1) There are many routes to the study of Computing Science. Students should seek advice from a department advisor or visit our website at www.cs.ualberta.ca/courses.
(2) The department of Computing Science does not allow audits in any of its laboratory courses.
(3) Special sections of CMPUT 196, 197, 198, 199, 296, 297, 298, 299, 396, 397, 398, 399, 496, 497, 498, 499 may have different prerequisites. Please check the specific course descriptions as posted by the Department of Computing Science.

Undergraduate Courses

CMPUT 304 - Algorithms II

Course Detail

Career	Undergraduate	
Units	*3.00	
Fee Index	6	
Approved Hours	3-0-0	
Calendar Term	either term	
Course Components	Lecture	Required

Description

The second course of a two-course sequence on algorithm design. Emphasis on principles of algorithm design. Categories of algorithms such as divide-and-conquer, greedy algorithms, dynamic programming; analysis of algorithms; limits of algorithm design; NP-completeness; heuristic algorithms. Prerequisites: CMPUT 204; one of STAT 151, 221, 235 or 265; one of MATH 225, 228, 229, 328 or consent of Instructor.

Fall Term 2009 / Winter Term 2010 Class Listings

● Open ◆ Full ✗ Cancelled ■ Closed (Contact Department)

Subject Notes All 100-level, 200-level, 300-level, and 400-level CMPUT courses are open to students in any Faculty, with the exception of CMPUT 250 (this course requires an application by the student; please consult the department), CMPUT 400 (IIP students only), CMPUT 412 (limited enrollment, 4th year CS students have priority), and CMPUT 495 (Honors CS students only). If you have any questions as to whether your pre-requisite background is sufficient to take a course, please contact the Department of Computing Science for advice.

All 500-level courses are restricted to Graduate students in Computing Science and Honors students in Computing Science. Honors students in Computing Science should contact the Department for registration assistance in 500-level courses. All 600-level courses are restricted to Graduate students in Computing Science.

Add selected section of CMPUT 304 to:

CMPUT 304 sections for Fall Term 2009

Class	Course	Section	Location	Days	Times	Instructor	Open Seats	Status
○ 35751	CMPUT 304	LEC A1	CAB 273	T R	11:00AM - 12:20PM	To Be Assigned	18 / 30	●

Class Notes: Computer Engineering and Computer Engineering Software Option students must register in sections beginning with "E".

Class	Course	Section	Location	Days	Times	Instructor	Open Seats	Status
○ 32632	CMPUT 304	LEC EA1	CAB 273	T R	11:00AM - 12:20PM	To Be Assigned	5 / 5	●

Class Notes: Computer Engineering and Computer Engineering Software Option students must register in sections beginning with "E".

Add selected section of CMPUT 304 to:

Figure 4.4: Bear Tracks detail page for CMPUT 304

approved hours, calendar terms, course components and a general description. The rest of the page is dedicated to display the schedule of the course's sections.

The course requisites are contained inside the course description, usually at the end, and are written in natural language. Although some departments follow the same format to express them, there are no consistent rules on how to express the requisites among the entire university. Moreover, only immediate requisites are included in the description of each course. That implies that in order to discover indirect requisites, the user has to start with a search of the basic course, read the requisites of that course, and repeat the process for each of them. It is not uncommon having to run five or more individual searches in order to find the complete requisites for a given course.

4.2 Design of the usability study

The usability study was divided into three parts: 1) we performed an interview to gather information about the participants and how they conducted course planning; 2) each participant was asked to perform a series of tasks with both the Course Browser and Bear Tracks; 3) we performed a second interview to gather impressions of the experience.

4.2.1 First interview

At the beginning of the session each participant was asked to sign a consent form that explained the methodology of the experiment. Afterwards, we performed a five-minute interview to gather information about the participant and their experience with course planning. This interview was audiotaped for record keeping and further analysis.

The interviewer asked the participant information about: 1) current year of study and the number of courses they had taken at the university; 2) the participant's understanding of the terms *prerequisite* and *corequisite* in the context of course registration; 3) methods used to find information about the courses and their requisites and if they had experienced difficulties understanding the requisites of courses; 5) participant's current practices for planning his program. The complete questionnaire of the interview is given in Appendix 1.

If the participant did not have a precise understanding of the terms, the interviewer explained the meaning of them, as this knowledge was crucial for the task-solving phase.

4.2.2 Course Planning Tasks

The second part of the session consisted of a series of typical course planning tasks that the participant must solve using both the Course Browser and Bear Tracks. With each tool, the participant was asked to answer seven questions. The questions had increasing levels of difficulty, starting with a very simple task of counting courses in certain department and ending with the planning of all the courses required to take a fourth year course, term by term. The questionnaires for both tools were analogous in the sense that both contained the same form of questions, in the same order, but with different data.

The purpose of this part of the session was to objectively measure the effectiveness of the user interface and give the users a scenario as close as possible to reality to compare the tools. Therefore, we measured the time to complete each of the tasks and, afterwards, each of the questions was verified to find out if the user reached a correct answer.

Before solving the tasks with the Course Browser, each participant was given five minutes for free exploration of the user interface. After that, a short tutorial video was shown and they were given the chance to ask questions about the application. All the participants had already being exposed to the use of Bear Tracks in the past, so no introduction of this tool was necessary. Simply, they were instructed to use just the *browse course catalog* feature to answer the questions.

To avoid biasing the results, the order of the tools was alternated, so for each question, around half of the participants used Bear Tracks first, and the remaining half used the Course Browser first. For the participants that used Bear Tracks first, this phase was performed with the following steps:

1. *Solve tasks 1 to 7 using the Bear Tracks.*

2. *Watch introductory video about the Course Browser.*

3. *Do a 5-minute exploration of the Course Browser.*

4. *Solve tasks 1 to 7 (with different data) using the Course Browser.*

For the participants that used the Course Browser first, this phase was performed with the following steps:

1. *Watch introductory video about the Course Browser.*

2. *Do a 5-minute exploration of the Course Browser.*

3. *Solve tasks 1 to 7 using the Course Browser.*

4. *Solve tasks 1 to 7 (with different data) using the Bear Tracks.*

Both the task solving and the free exploration phases were recorded using a screen capture software called *Screenflow*[1]. This software recorded all the user activity, including audio and video of the participant's face. This information was kept for further analysis and record keeping.

4.2.3 Second interview

After finishing the tasks, we conducted a post interview. Just like the first interview, this part of the session was audiotaped. The questionnaire contained questions to test the user experience about both of the applications used. Some of the questions were formulated using a 5-level Likert scale [22] to facilitate the statistical analysis and comparison. The remaining questions were open-ended to identify elements that the participants liked and disliked about the user interface and gather suggestions on how to improve the application.

The goal for this interview was to measure if the users perceived the application as easy to use, easy to learn, and find out the level of confidence they had with the answers they found. To enable us to compare the results effectively and avoid bias, most of the questions that compared the applications were formulated in exactly the

[1] http://www.telestream.net/screen-flow/overview.htm

Table 4.1: User study questions formulated in parallel form for Bear Tracks and the Course Browser. Answers were represented using a five-level Likert scale.

Question Identifier	Question text
bt.easynd	It is easy to find a course on Bear Tracks
cb.easynd	It is easy to find a course on the Course Browser.
bt. enoughinfo	The course catalog on Bear Tracks displays enough information about the course for planning your program.
cb. enoughinfo	The Course Description view displays enough information about the course for planning your program.
bt.easyreq	It was easy to identify the requisites of a course using Bear Tracks
cb.easyreq	It was easy to identify the requisites of a course using the Course Browser
bt.confident	I felt very confident with the answers I got from Bear Tracks.
cb.confident	I felt very confident with the answers I got from Course Browser
bt.clear	The requisites of courses are presented in a clear way on Bear Tracks
cb.clear	The requisites of courses are presented in a clear way on the Course Browser
bt.planning	The course catalog on Bear Tracks is adequate for performing course planning
cb.planning	I would like to use a tool similar to the Course Browser for my course planning.

Prefix “bt.” indicates questions that refer to Bear Tracks.

Prefix “cb.” indicates questions that refer to the Course Browser.

same way for both of the systems. These paired questions are shown in Table 4.1. Some of the questions, however, were formulated to only one of the systems as they made reference to a specific part of the application, for example a particular visualization of the Course Browser. These questions are displayed in Table 4.2

4.3 Pilot testing

Before conducting the study, we performed four pilot trials. Three of the participants for these pilot tests were graduate students from the Software Engineering Research Lab and one was an undergraduate student in Computing Science. These trials served two purposes: first, they helped us to identify problems with the questionnairef or refinement and, second, they identifyed obvious problems with the prototype that could be easily fixed before the study.

One problems identified during the pilot testing was that some of the participants reached a wrong answer for some types of the questions when they

Table 4.2: User study questions formulated in individual form. Answers were represented using a five-level Likert scale.

Question Identifier	Question text
bt.easydesc	The description of the courses on Bear Tracks is easy to understand
cb.video	My understanding of the Course Browser improved significantly after watching the Quick Tour video
cb.box1	I understood the meaning of the Requisite Box Diagram
cb.box2	The Requisite Box Diagram is an effective way to visualize course requisites.
cb.collections	The Collections offer an effective way to group the courses in the course catalog
cb.list	The Course List offers an effective way to browse the courses
cb.reqgraph1	I understood the meaning of the Course Requisite Graph
cb.reqgraph2	The Course Requisite Graph offers an effective way to visualize the relationships between courses
cb.overview1	I understood the meaning of the Collection Overview diagram
cb.overview2	The Collection Overview diagram offers an effective way to visualize the relationships between courses
cb.better	The Course Browser is better than Bear Tracks for performing course-planning tasks

Prefix "bt." indicates questions that refer to Bear Tracks.

Prefix "cb." indicates questions that refer to the Course Browser.

chose to solve them using the *collection overview* diagram. This diagram was designed to present only the requisites of courses that belong to a given collection, and therefore, all dependencies to courses outside the collection were simply not displayed. The participants, however, tried to use the diagram to find out whether a course had dependencies in general. Therefore, by using this diagram, they sometimes reached to the wrong conclusion that a course had no dependencies at all, when the requisites of the course were outside the collection. While this problem was partially due to a wrong interpretation of the diagram, it became clear that the diagram could be improved to prevent the user from making that mistake. The problem was not that the diagram was showing a subset of the requisite information, but that this fact was not obvious to the user. To fix the problem, a new node labeled "*other*" was introduced in the diagram to represent all the courses outside the collection. Whenever a course had a dependency to another course outside the collection, an arc was drawn to the *other* node. Furthermore, whenever the user highlighted the *other* node, a text explicitly explained that the node represented courses outside the collection.

Another issue identified during the pilot testing was that some the participants often tried to double-click the items on the course list of the Course Browser to find more information about the course. This is a very common user interface behavior in other applications. As it was very simple to implement this feature as a shortcut to open the course description tab and it was not intrusive to other features, it was fixed before performing the actual study.

The pilot test also helped us to refine the questionnaire for the tests. The questions that were identified as confusing were revised or replaced altogether. Different styles were tried for the tasks, ranging from multiple selection questions to open-ended ones, as we tried to make them simple to solve using the tools but difficult to guess. Finally, times were recorded to ensure the length of the study was no greater than two hours.

The pilot testing proved to be a very useful practice to refine both the prototype and the questionnaire.

4.4 Target population and recruiting

The target population of the study consisted of undergraduate students at the University of Alberta. To recruit the participants, we attached advertisement posters in the university campus. We also sent e-mails to the mailing lists of different departments in the faculty of Arts and Sciences. We offered to give a movie ticket to each participant at the end of the session as an incentive for participating in the study.

In addition to the tests with the students, we also organized a few sessions with student advisors to gather their impressions about the Course Browser prototype.

Chapter 5

Usability Study: Results

5.1 First interview

The usability study was conducted with fifteen undergraduate students from the University of Alberta. They were enrolled in a variety of programs, and had different levels of experience with registration and course planning. As Figure 5.1 a shows, the years of enrollment for the participants ranged from 2005 to 2009, in a uniform way. Since there is no fixed number of courses that a student has to take per year, a better indication of their experience with course planning is the number of courses taken. This information, displayed in Figure 5.1 b, shows a wide range of courses taken, from 0 to 5 to more than 30.

Each participant was asked to define the concepts of *prerequisites* and *corequisites*. These concepts were central for performing the tasks in the following phase of the study. The results for this question are shown Figure 5.1 c. All of the participants had a good idea of what the term *prerequisite* means, as all but one of them could give a precise definition of the concept. The notion of *corequisite*, on the other hand, proved to be quite confusing: only four of the participants knew precisely what it means; six of the participants knew that corequisites are meant to be taken simultaneously with the course they refer to, but did not know that they can be taken before; and one third of them gave completely erroneous definitions.

For the question *"I have trouble understanding the prerequisites and corequisites of courses"* a significant number of participants disagreed [$V = 6.5, p < .050$] (see Table 5.1).

Some of the questions were aimed at identifying the resources that students use to perform course planning. There are a variety of ways in which the students can access the catalog of courses of the university. One is them is by using the course catalog feature on Bear Tracks. In addition, each year the university publishes a printed calendar containing all courses and programs. A digital version of the calendar is also available to all students on the university website. An alternative to these resources is Bear Scat, a system originally developed by students to complement the functionality of Bear Tracks. Bear Scat is now considered to be

Table 5.1: Results for the question: "I have trouble understanding the prerequisites and corequisites of courses" in the first interview

	Rating						Wilcoxon Test	
Question Identifier	SD	D	N	A	SA	median	V	p
pre.understand	2	11	1	1	0	2	6.5	$< .050$

SD=Strongly disagree, D=Disagree, N=Neutral, A=Agree, SA=Strongly Agree

obsolete, as Bear Tracks was recently updated to include most of its functionality. In addition, some departments publish and maintain information about the courses they offer. Finally, students often consult professors, advisors and their peers to find information.

We asked the participants which resources they used to browse courses and find out about course requisites. The results to this question, as seen on Figure 5.1 d, show that students use all of the printed and electronic resources to search for courses, Bear Tracks and the printed version of the university calendar being the most popular. For finding about the requisites about courses, however, Bear Tracks was less popular, as only six of the participants claimed to use it. The same number of participants mentioned the departments website as their source of information, and calendar and seeking help from advisors were next in the ranking of preferred method.

Finally, students were asked about their practices in performing course planning. Their answers are displayed in 5.1 e. Six of the students said that they perform course planning on a yearly basis, choosing their courses for the two terms ahead. Five of the students did not do planning and simply picked their courses at the beginning of each term.

5.2 Course-Planning Tasks

During the task-solving phase, participants were asked to perform seven tasks related to course planning. The time of completion and the correctness of the answers were recorded for each of the tasks. Table 5.3 displays the results for correctness of the tasks, separately for each system used. The time-to-solve is presented on Table 5.4 and displayed in scatter plots, on Figure 5.2.

5.2.1 Statistical Analysis

To test the significance of the correctness results, we applied the McNemar's Chi-squared test [30]. In this case, the null hypothesis was that the number of participants that obtained an incorrect answer to a given task using Bear Tracks, but reached a correct answer with the Course Browser, is the same as the number

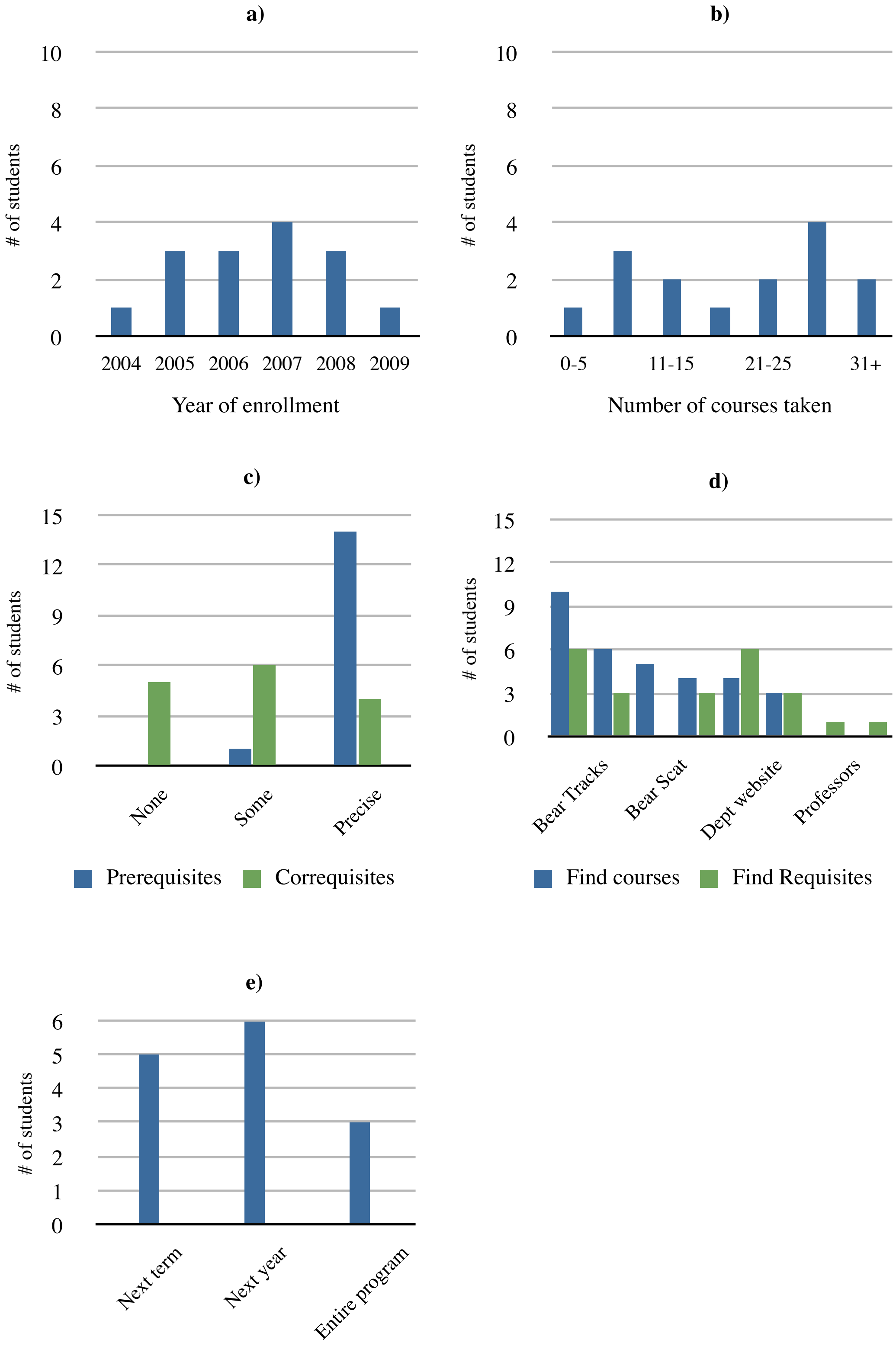


Figure 5.1: Results for first interview with fifteen undergraduate students. a) Number of students by year of enrollment in the university. b) Number of courses taken. c) Understanding of the concepts of *prerequisite* and *corequisite*. d) Resources used for course planning. e) Planning horizon for students.

of participants that gave the correct answer with Bear Tracks, and a wrong answer with the Course Browser. If the probability of the null hypothesis being true is significantly small, we reject the null hypothesis and conclude that the increase or decrease in correctness is related to the tool used.

With the time-to-solve data, we performed a mean and a variance test for each of the tasks, comparing the results with Bear Tracks and the Course Browser. To test the equality of means, we used the paired version of the Student's t-test [39]. To drop the assumption that the variances of the samples are equal, we used the Welch approximation. The alternative hypothesis used was that the means of the two samples were different. The equality of the variances was tested using the F test, with the alternative hypothesis being that the variance of the times with each of the systems are different [30]. The results for these tests are shown in the bottom row of Tables 5.3 and 5.4.

No statistical test was performed to assess order effects. It is reasonable to assume that the ordering does not produce a signicant effect in this case, since all the participants had prior experience with Bear Tracks, all of them received training with the Course Browser before using it, and the tasks performed were the same except for specific choices of courses involved. Table 5.2 shows the system each participant used first to solve the tasks.

All the statistical computations were performed using the *R* statistical software [27]. For the analysis, we considered any value of p greater than 0.05 to be *not significant* (labeled *n.s.*).

5.2.2 Results

Task 1 required the participants to count the number of courses offered by a certain department of the university. The average time taken by the participants to complete this task was significantly lower with the Course Browser than with Bear Tracks [$t(14) = 3.03, p < .01$] and had a significantly smaller spread [$F(14, 14) = 4.62, p < .01$]. No significant improvements in correctness were observed [$\chi^2(1) = 0.5, n.s.$].

Task 2 asked the participants to identify the courses with no requisites among a list of four. The average time-to-complete times were significantly smaller with the Course Browser [$t(14) = 9.39, p < .001$] and had a significantly higher spread with Bear Tracks [$F(14, 14) = 6.68, p < .001$]. The improvement in correctness with the Course Browser was significant [$\chi^2(1) = 4.17, p < .05$].

Task 3 had the purpose of testing the understanding of the concepts *prerequisite* and *corequisite*. The participants were given a course and a list of courses. First, they were asked to choose the ones that could be taken *before* in order to meet the requirements of the course. Then, they were supposed to choose the ones that could be taken *simultaneously* to meet its requirements. Some of the courses in the list were part of the prerequisites, some of the corequisites, and some were not related at all. The results showed no significant differences in time-to-complete

[$t(14) = -0.52, n.s.$], [$F(14, 14) = 0.36, n.s.$] or correctness among the systems [$\chi^2(1) = 0, n.s.$]. With both tools, 13 out of 15 participants were able to identify the correct answer.

In Task 4, participants were asked to create two different lists of courses that could be taken in order to meet the requisites of a given course. The time-to-complete for this task were not significantly different between the two applications [$t(13) = -0.26, n.s.$], [$F(13, 13) = 0.68, n.s.$]. One of the participants did not complete the exercise with Bear Tracks and marked the *not possible* box, hence the lost of one degree of freedom in the analysis. The difference in correctness was not significant either [$\chi^2(1) = 2.25, n.s.$]. In total, 3 of the participants were able to complete the task successfully Bear Tracks.

Task 5, consisted in completing the requisites for a course. Participants were told to assume that they have taken some courses and were asked which courses are they missing in order to meet the requisites of a given course. The average time-to-complete was significantly lower with the Course Browser than with Bear Tracks [$t(14) = 2.25, p < .05$] and had a significantly smaller spread [$F(14, 14) = 3.13, p < .05$]. In terms of correctness, we did not observe any significant difference between the two tools [$\chi^2(1) = .10, n.s.$].

Task 6 required the participants to perform a reverse dependency lookup of the course requisites. For this exercise, they were given a course and were asked to find other courses within that department that directly required it. The average time-to-complete was significantly lower with the Course Browser than with Bear Tracks [$t(10) = 2.80, p < .05$] and had a significantly smaller spread [$F(10, 10) = 11.80, p < .05$]. Five of the participants did not complete the exercise with Bear Tracks and marked the *not possible* box, hence the lost of degrees of freedom in the analysis. The improvement in correctness with Course Browser was significant [$\chi^2(1) = 9.10, p < .05$].

Finally, in Task 7 the participants were given the description of a program as shown in the university calendar, and were asked to make a full planning, term by term, of the courses they would take in order to register for a four level course as soon as possible. The time-to-complete results for this task were not significantly different among the two applications [$t(14) = 0.16, n.s.$], [$F(14, 14) = 0.64, n.s.$]. The improvement in correctness with the Course Browser was significant [$\chi^2(1) = 5.10, p < .05$].

5.2.3 Analysis and Discussion

The Course Browser proved to be superior to Bear Tracks for tasks that required looking up indirect requisites and doing reverse requisite lookups. It also proved to be significantly faster to navigate.

For simple tasks that did not require students to look for more than one course, both tools performed in a similar way. In Task 3, which was meant to test the understanding of the concepts *prerequisite* and *corequisite* no significant differences

Table 5.2: System used first by each participant during the task solving phase.

Participant	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
System	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB	BT	BT	CB	BT	CB

BT = Bear Tracks, CB = Course Browser

Table 5.3: Correctness of the results for each participant by task and system used. The bottom row shows the results of the McNemar's Chi-squared test for each task.

	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Task 7	
Part.	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB
1	C	C	C	C	C	C	I	C	C	C	I	C	C	C
2	C	C	C	C	C	I	I	I	I	C	I	C	I	C
3	C	C	C	C	C	C	I	I	C	I	I	C	I	C
4	C	C	I	C	C	C	I	I	I	I	I	I	I	I
5	C	C	C	C	C	C	I	C	I	I	I	C	I	C
6	C	C	I	C	C	C	I	C	C	C	C	C	I	I
7	C	C	I	I	C	C	I	I	I	C	C	C	I	I
8	C	C	C	C	C	C	I	I	C	I	I	C	I	I
9	C	C	I	C	I	I	C	C	C	I	I	C	C	C
10	C	C	C	C	C	C	I	I	I	C	I	C	I	C
11	C	C	I	C	C	C	C	C	C	C	I	C	I	C
12	C	C	I	C	C	C	C	C	I	C	I	C	C	C
13	C	C	C	C	I	C	I	I	C	I	I	C	C	C
14	C	I	C	C	C	C	I	I	C	I	I	C	I	C
15	C	I	I	C	C	C	I	C	C	I	I	I	I	C
χ^2	0.5		4.17		0		2.25		0.10		9.10		5.1	
df	1		1		1		1		1		1		1	
p	$n.s.$		$< .05$		$n.s.$		$n.s.$		$n.s.$		$< .01$		$< .05$	

BT = Bear Tracks, CB = Course Browser, C = Correct answer, I = Incorrect Answer

Table 5.4: Time-to-complete, in seconds, taken by each participant to solve the tasks.

	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Task 7	
Part.	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB	BT	CB
1	228	27	158	72	86	49	301	119	139	40	294	63	571	305
2	141	43	348	107	193	247	459	357	249	160	169	88	267	432
3	131	72	289	76	85	96	381	600	321	153	146	46	1340	1272
4	59	36	144	74	69	115	–	132	51	55	–	45	334	347
5	69	33	294	133	54	47	128	155	71	40	57	31	318	193
6	87	30	345	75	82	181	150	145	116	141	98	74	350	560
7	46	53	298	155	82	91	129	111	72	124	192	47	552	677
8	94	56	232	70	116	66	124	96	73	62	–	72	592	371
9	49	40	268	107	87	100	258	142	191	86	126	47	435	321
10	83	51	178	63	102	66	218	180	114	121	54	90	470	436
11	52	107	255	103	147	63	351	324	109	88	64	87	805	584
12	46	22	282	72	87	134	148	280	148	148	–	62	383	236
13	59	25	141	69	82	48	140	257	62	54	–	52	280	213
14	63	17	182	92	63	71	221	226	134	80	228	42	500	310
15	107	42	220	111	115	176	130	252	95	85	85	91	461	1237
mean	87.60	43.60	242.27	91.93	96.67	103.33	224.14	225.07	129.67	95.80	137.55	62.47	510.53	499.60
std dev	49.06	22.82	69.43	26.87	35.21	58.49	110.42	131.57	74.57	42.17	77.47	20.04	269.84	336.20
t-test	$t(14) = 3.07$ $p < .01$		$t(14) = 9.39$ $p < .001$		$t(14) = -0.52$ *n*.s.		$t(13) = -0.26$ *n*.s.		$t(14) = 2.25$ $p < .05$		$t(10) = 2.80$ $p < .05$		$t(14) = 0.16$ *n*.s.	
F-test	$F(14,14) = 4.62$ $p < .01$		$F(14,14) = 6.68$ $p < .001$		$F(14,14) = 0.36$ *n*.s.		$F(13,13) = 0.68$ *n*.s.		$F(14,14) = 3.13$ $p < .05$		$F(10,10) = 11.80$ $p < .001$		$F(14,14) = 0.64$ *n*.s.	

BT = Bear Tracks, CB = Course Browser

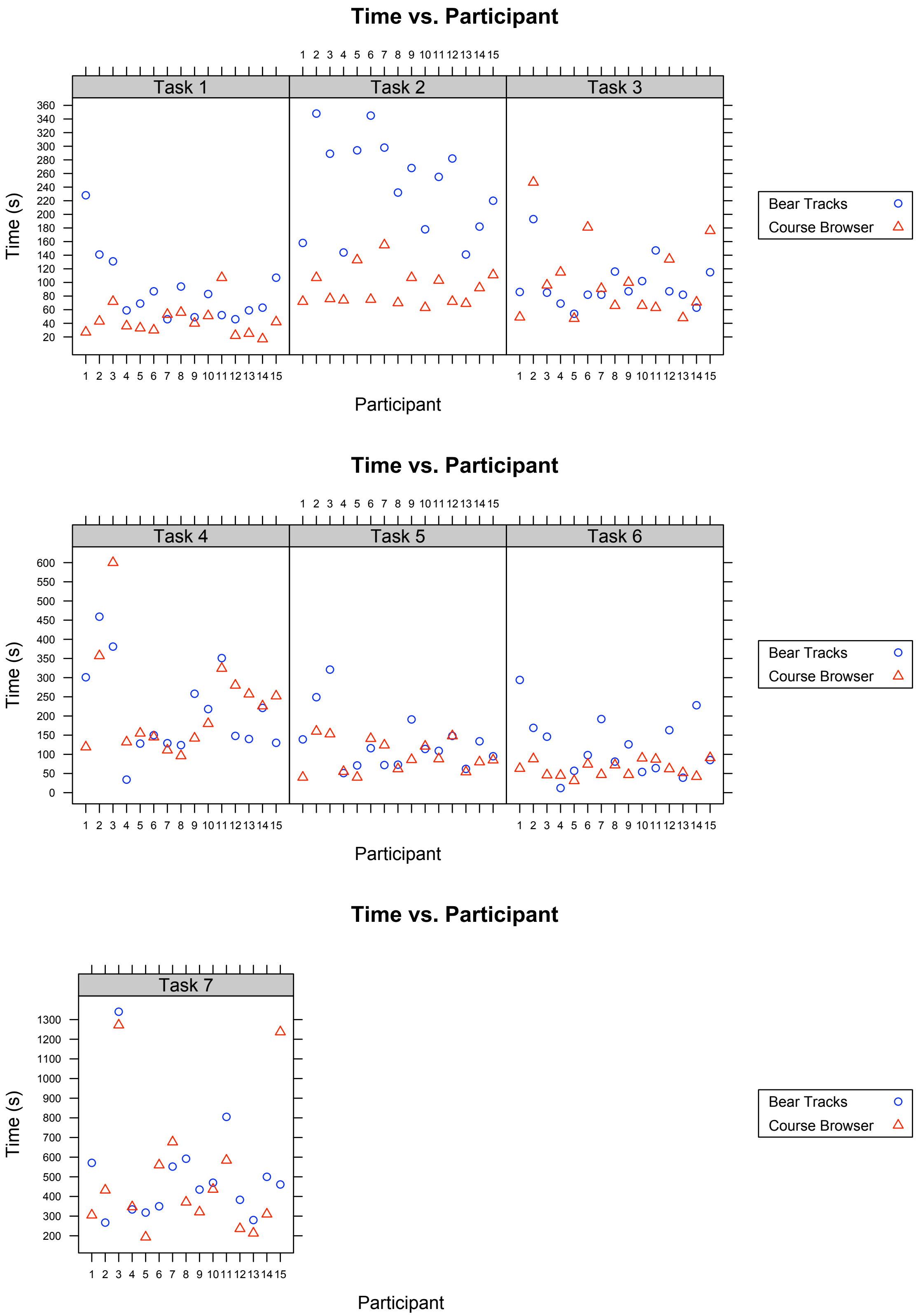


Figure 5.2: Time taken by each participant to complete each of the tasks with Bear Tracks and the Course Browser

were observed in terms of time-to-complete or correctness.

Using the Course Browser was faster than Bear Tracks for navigating and finding courses. The representation of requisites in the former was easier to understand. This was shown by the results of Task 2, where we asked the participants to identify the courses with no requisites among a list of four. Task solving with Course Browser were approximately 2.5 times faster than task solving with Bear Tracks and significantly more accurate. The reason for this is that, with Bear Tracks, the participants had to start a new search for each of the courses, adding up the loading times of the search form and the result page. These operations could take several seconds despite that fact that we where using a high-speed connection to connect to the server. In addition, many of the participants did not know where to find the information about the requisites once they reached the calendar description of the course. In Bear Tracks, it is not mentioned explicitly that a course has no requisites. Some of the users had to, first, find a course with requisites to realize that this information was given as part of the description, and, then, repeat the operation with the previous courses to verify their answer. In contrast, the Course Browser offers participants multiple ways to find courses. Most participants started by choosing the department from the *collections view*. From there, they could either use the *Collection Overview diagram* and see if the course had any connections or consult the *Course description view* and take a look at the requisite diagram, which will inform them right away if the course had no requisites. There is no network delay for any of these operations in our system.

For the reverse dependency lookup task, using the Course Browser was better than Bear Tracks in both speed and correctness. This operation is key to answer the question *"If I have taken this course, what courses can I take next?"*. In Task 6, only two participants reached the correct answer; with the Course Browser thirteen succeeded. The cause for these contrasting results is that Bear Tracks does not offer any means to perform reverse requirement searches. Four of the participants stated that the task was impossible; the rest tried to look for the answers manually, missing some of the courses. One of the successful participants used the browser integrated search feature to quickly look for matches of the course code in the course descriptions. This ingenious solution, however was not obvious for other participants. Conversely, the Course Browser was designed to simplify this type of task. All of the participants identified that the *Collection Overview diagram* allowed them to solve the question easily.

For tasks that required looking for indirect requisites, we observed better results using the Course Browser than using Bear Tracks. In Task 4, participants were ask to create two different lists of courses that could be taken in order to meet the requisites of a given course. This is a basic task in course planning, as students often have to adjust the courses as some of the courses may not be taught each term and they may have conflicts in schedules. As these tasks required looking for indirect dependencies of the courses and building a requisite tree, they were more difficult than the previous ones. No significant differences in time-to-complete or correctness were observed in this task. However, the rate of failure was particularly high for Bear Tracks, as only 3 of the participants were able to complete the task successfully. With the Course Browser, almost half of the participants were successful. Solving this task with Bear Tracks required the participant to perform several searches to discover the complete requisite tree for the course. In contrast, the Course Browser, offers different tools for the job: both the box diagrams in the *Course description view*, and the *Course requisite graph* could be used to solve the question. We noted that some of the participants were confused by this task, as they were not used to look for indirect requisites. The following tasks had better results as the participants were more familiar with the concept and had improved their knowledge of the tools offered by the Course Browser. A common mistake for some participants was to assume that all the arrows in the circular diagrams expressed *allOf* dependencies.

Task 5, consisted in completing the requisites for a course. Like in the previous task, participants had to find out the indirect requisites of the course. In this case, however, they had to match the courses that they supposedly had taken and complete the path of courses using them. In terms of correctness, we did not observe any significant difference between the two tools. In contrast, times were significantly lower when using Course Browser.

Finally, in Task 7 the participants were given the description of a program as shown in the university calendar, and were asked to make a full planning, term by term, of the courses they would take in order to register for a four level course as soon as possible. This task was the most difficult task of the session. Although we did not observe significant differences in time to complete for this task, we did see a significant improvement in correctness while using the Course Browser. In this task, the participants could really take advantage of the information exposed by our tool. Some of the participants chose to use the box diagrams and the back and forward navigation, while others felt more comfortable with the *course requisite graph*. Still, this task still involved a lot of effort to complete and shows the need for a more complete tool for planning.

5.3 Second interview

The second interview captured the subjective impressions of each participant after completing the tasks. This interview comprised questions asked using a five-level Likert scale and some additional open-ended questions. Six of the questions were asked in the same form for both the Course Browser and Bear Tracks to allow for a comparison. The results for this set of questions are shown in Table 5.5. The remaining eleven questions referred exclusively to one of the systems. The results for this set of questions are shown in Table 5.6.

5.3.1 Statistical Analysis

To analyze the questions formulated in parallel form, we calculated the median and performed a Mann-Witney U-Test [38]. Using this test, it is possible to determine if the distributions of the answers in the Likert scale are significantly different. For the non-paired questions, we used the Wilcoxon's signed-rank test. With this test we can determine if the median of the tests is significantly greater than the neutral point of the scale [40]. These tests were chosen because we assumed the Likert scale to be an ordered categorical data. In other words, we cannot tell whether the distances between the adjacent units on the scale are the same, but we can define an order; for example, we can tell that 5 is better than 3, but we don't know if the distance between 3 and 4 is the same as the one between 4 and 5 [37].

All the statistical computations were performed using the *R* statistical software [27]. For the analysis, we considered any value of p greater than 0.05 to be *not significant* (labeled *n.s.*).

5.3.2 Results

Paired Questions

For the question *"It is easy to find a course on [Bear Tracks/Course Browser]"*, we obtained a significantly higher score for the Course Browser [$W = 60.5, p < .05$].

For the question, *"The [course catalog on Bear Tracks / Course Description view on the Course Browser] displays enough information about the course for planning your program."* we had a significantly higher score for the Course Browser [$W = 40, p < .010$].

For question *"It was easy to identify the requisites of a course using [Bear Tracks/Course Browser]"* we had a significantly higher score for the Course Browser [$W = 9.5, p < .001$].

The question *"The requisites of courses are presented in a clear way on [Bear Tracks/Course Browser]"* we had a significantly higher score for the Course Browser [$W = 16, p < .001$].

The question, *"I felt very confident with the answers I got from [Bear Tracks/Course Browser]"* showed that participants felt more confident with the information obtained with the Course Browser. We observed a significantly higher score for the Course Browser [$W = 31, p < .001$].

The last paired question compared the adequacy of the systems for performing course-planning tasks. For Bear Tracks, it was formulated as *"The course catalog on Bear Tracks is adequate for performing course planning"*. In the case of the Course Browser, we gave them the following sentence: *"I would like to use a tool similar to the Course Browser for my course planning"*. Comparing the two, we observed a significantly higher score for the Course Browser [$W = 6, p < .001$].

Table 5.5: Results for paired questions in the second interview

	Rating						U-Test	
Question Identifier	SD	D	N	A	SA	median	W	p
bt.easyfind	0	1	1	13	0	4	60.5	$< .050$
cb.easyfind	0	1	1	5	8	5		
bt.enoughinfo	1	6	4	4	0	3	40	$< .010$
cb.enoughinfo	0	2	0	9	4	4		
bt.easyreq	0	8	4	3	0	2	9.5	$< .001$
cb.easyreq	0	0	1	3	11	5		
bt.confident	0	4	7	4	0	3	31	$< .001$
cb.confident	0	1	0	9	5	4		
bt.clear	1	9	2	3	0	2	16	$< .001$
cb.clear	0	0	1	8	6	4		
bt.planning	2	11	0	2	0	2	6	$< .001$
cb.planning	0	0	0	6	9	5		

SD=Strongly disagree, D=Disagree, N=Neutral, A=Agree, SA=Strongly Agree

Individual questions

Bear Tracks For the question *"The description of the courses on Bear Tracks is easy to understand"* we observed no significant agreement [$V = 45.5, n.s.$].

Course Browser's user interface In order to explain the user interface of the Course Browser to the participants, we recorded a three-and-a-half-minute screencast. This multimedia presentation about the application guided the user through the most important features of the application. To check the user response to this approach of training we asked the participants if they agreed with the statement *"My understanding of the Course Browser improved significantly after watching the Quick Tour video"*. A significant number of the participants agreed with it [$V = 80, p < .01$].

Requisite Box diagrams A significant number of participants answered agreed or strongly agreed with the statement *"I understood the meaning of the Requisite Box Diagram"* [$V = 105, p < .001$]. Similarly, significant number of participants answered agreed or strongly agreed with the statement *"The Requisite Box Diagram is an effective way to visualize course requisites."* [$V = 120, p < .001$].

Navigations Features A significant number of participants agreed or strongly agreed with the statement *"The Collections offer an effective way to group the*

Table 5.6: Results for non-paired questions in the second interview.

	Rating						Wilcoxon Test	
Question Identifier	SD	D	N	A	SA	median	V	p
bt.easydesc	0	5	3	7	0	3	45.5	$n.s.$
cb.video	0	2	2	8	3	4	80	$< .010$
cb.box1	0	0	1	8	6	4	105	$< .001$
cb.box2	0	0	0	6	9	5	120	$< .001$
cb.collections	0	1	0	7	7	4	115.5	$< .001$
cb.list	1	0	1	6	7	4	94.5	$< .010$
cb.reqgraph1	0	0	1	13	1	4	105	$< .001$
cb.reqgraph2	0	0	0	9	6	4	120	$< .001$
cb.overview1	0	0	0	12	3	4	120	$< .001$
cb.overview2	0	0	1	11	3	4	105	$< .001$
cb.better	0	0	2	3	10	5	91	$< .001$

SD=Strongly disagree, D=Disagree, N=Neutral, A=Agree, SA=Strongly Agree

courses in the course catalog" [$V = 115.5, p < .001$].

A significant number of participants agreed or strongly agreed with the statement *"The Course List offers an effective way to browse the courses"* [$V = 94.5, p < .010$].

Course requisite Graph A significant number of participants agreed or strongly agreed with the statement *"I understood the meaning of the Course Requisite Graph"* [$V = 105, p < .001$].

A significant number of participants agreed or strongly agreed with the statement *"The Course Requisite Graph offers an effective way to visualize the relationships between courses"* [$V = 120, p < .001$].

Collection overview Diagram A significant number of participants agreed or strongly agreed with the statement *"I understood the meaning of the Collection Overview diagram"* [$V = 120, p < .001$].

A significant number of participants agreed or strongly agreed with the statement *"The Collection Overview diagram offers an effective way to visualize the relationships between courses"* [$V = 105, p < .001$].

Overall We asked the participants if they agreed with the sentence *"The Course Browser is better than Bear Tracks for performing course-planning tasks"*. A significant number of participants answered agreed or strongly agreed with it [$V = 91, p < .001$].

5.3.3 Analysis and Discussion

Participants expressed a strong acceptance of the Course Browser and considered it superior to Bear Tracks for performing course-planning tasks. We obtained significant results in favor of the Course Browser in all questions. We also identified problems with the current prototype and gather important suggestions on how to improve it.

When asked if it was easy to find courses with Bear Tracks and the Course Browser, most of the participants expressed agreement for both of the systems. However, they had stronger preference towards the Course Browser as it allowed them to search for courses faster. Having said that, still some of the participants suggested incorporating a search feature into the Course Browser. During the development of the prototype we were aware of the value of this feature, but could not implement it due to task constraints. Having a search could further improve the navigability of the application.

Participants also found the way the of displaying information for courses in the Course Browser adequate for performing course planning and consulting course requisites. The requisite information was easier to understand using the Course Browser. They also agreed that the visualization and navigation tools provided in our system simplified the interpretation of results. Conversely, when we asked the descriptions of the courses on Bear Tracks were easy to understand, we had mixed responses (see question *bt.easydesc* on Table 5.6). About this, a participant suggested that the requisite information of Bear Tracks should be separated from description of the course and a larger font should be used. Several of them also suggested the inclusion of diagrams on Bear Tracks to represent requisites in a graphical form, using the box diagrams.

In terms of confidence, we had a higher level of agreement with the Course Browser. Participants felt better about that the answers they got when working with our system because it often displayed all the information they needed in one place. They could see in a diagram the relations between courses.

The participants concurred with the usefulness and relevance of the different elements of the Course Browser's user interface. We had significant results supporting the use of collections to group the courses in the course calendar. Most participants used the collections that grouped courses by departments. Other collections were only used by a small number of them. The course list also demonstrated to be a useful feature among participants.

Users also expressed their approval of the circular diagrams: a significant number of them agreed when asked if they understood their meaning of both the *collection overview* and the *course requisite graph*. They also agreed that these visualizations offered an effective way to visualize relationships. Nonetheless, a few observations and comments revealed that there is still room for improvement. Some of the participants were confused by the representation of *anyOf* relationships in these diagrams and assumed that all the arrows expressed *allOf* dependencies. Some of them also expressed that the different intensities of blue in the arrows were difficult to distinguish and suggested to change them by dashed lines or different colors. We also identified that the use of different intensities of a color were not practical with some displays, namely projectors. Finally, for some of the courses, the diagrams became too cluttered. To improve this, a filter feature could help focus on the relevant elements.

Using a *screencast* to show the functionality of the course browser was also welcomed by the participants. Many of them agreed that they improve their understanding of the application after watching the three-and-a-half-minute video. We also included a number of help features inside the application. Each of the diagrams had a button in the top left corner labeled *"What is this diagram"* that, when pressed, would display a popup window with an explanation of the diagram. This documentation consisted of an annotation version of a sample diagram. Even though this feature was mentioned in the screencast and most of the participants were aware of it, only a few of them read it. Despite the efforts to provide an easily accessible and concise help, only a small fraction of the participants made use of it.

5.4 Conclusions

We performed a study in course planning with fifteen students of the University of Alberta. The participants had diverse levels of experience with course planning and registration. We found that course planning is not a widespread practice among students. Most of them plan their courses on a year or term basis and only a small fraction of them know which courses they want to take for their whole program.

Performing course planning is not a trivial task. The requisites of courses form graphs that can be difficult to understand for students. Moreover, the calendar browsing feature in Bear Tracks is not suited for performing course planning as it does not offer an efficient way to see the indirect requisites of courses. Similarly, performing reverse dependency lookups is not possible with this tool. This activity is important to answer a key question in doing planning: *"If I have taken these courses, what courses can I take next?"*.

Our prototype, the Course Browser, implements different visualization techniques to display relationships between courses. During the user study, the participants agreed that these techniques ease the process of performing course planning. These results were in consonance with the empirical evidence. For planning tasks we observed a significant improvement in time-to-complete and correctness using the Course Browser, compared to Bear Tracks.

Still, there is room for improvement in our prototype. Performing complete planning tasks is still a cumbersome operation. For the most difficult tasks in the experiment we observed a high error rate. In particular, we identified that the circular diagrams were subject to misinterpretation by the students for courses that contained choices in their requisites. An improvement of this diagram is recommended as a future work.

Chapter 6

Insights and Future work

6.1 Insights about visualizing dependencies

Dependency visualizations can be a powerful tool for understanding problems, when they are well defined. In our experience with the Course Browser we learnt valuable lessons on how to create useful visualization of dependency graphs:

A good starting point for creating a dependency visualization (and even any kind of visualization) is to define a set of questions about the domain that the visualization must help answer. These questions would serve as evaluation criteria to compare different prototypes.

Take advantage of the particular characteristics of the problem domain. For example, if the dependencies form a tree, use specific algorithms for trees; if the dependencies form a graph, it would be relevant to check the particular characteristics of it, *e.g.* if it is acyclic, connected, bi-connected or bipartite, and choose algorithms suited for those types of graphs. Another aspect is the typical number of elements to visualize. Dealing with hundreds of elements requires different strategies than dealing with thousands or hundreds of thousands. If the exact number can not be defined, at least the order of magnitude will be useful.

Let the user decide what view is the best. Different views will appeal to different users. In the Course Browser study, we observed that some users preferred to use the overview diagram while others preferred to navigate the requirements using the hyperlinks in the requisite box diagram. Providing different ways to visualize information and different affordances to manipulate the views will maximize the utility of the user interface.

Make good use of the different perceptual channels, like shape, color, alignment and movement. If possible, use more than one of these channels to encode the same property. That will help preserve the semantics of your diagram even if transferred to a different media, *e.g.* from screen to paper. It will also be helpful for people with disabilities, like color blindness, to completely understand the diagram.

Take advantage of the media you are using to display a visualization. Each medium has its advantage and limitations. For example, if you are using a computer screen, you may be able to include interactive features to help the user navigate the data, but your screen may be small. If you are using paper, you typically have higher resolution than a screen, but your content must be static and you may not be able to use color.

Exercise caution when you simplify the information. Removing some of the complexity to provide an overview of the problem can help the user understand the information better, as long as they are aware of the simplification. Therefore, it is important to make the simplifications as explicit as possible; otherwise, the user might take the simplified model for the real model.

Make diagrams self explanatory. Users are unenthusiastic about reading documentation. Therefore, the diagram should provide the user clues on how to interpret and manipulate it. One way to do this is to provide tooltips or rollover messages. In addition, choosing a metaphor that is familiar to the user will help them guess the possible actions.

6.2 Future Work

6.2.1 Improvements to the Course Browser

These are some possible improvements to the Course Browser:

Search: One of the features that many of the participants in the user study suggested is a search box. This box would instantly filter the course information by code and title.

Custom collections: The current prototype only supports predefined collections. With custom collections, the user would be capable of defining new collections based on their needs. A custom collection would be defined by a query string: for example, users can look for courses in Physics and Math with 3 credits and no requisites.

Reverse dependency lookup: The current prototype of the Course Browser only allows the user to see reverse dependencies, *i.e.* courses that require a given course, within a collection. Sometimes it is useful to see the reverse dependencies across different collections. This can be done by extending the Course Requisite graph to explore the dependency graph in reverse.

Improve readability of circular diagrams: Some of the participants were confused by the use of different saturation levels to represent the strength of course dependency in circular diagrams. The difference between mandatory requisites and choices can be made more explicit by using different colors or combining dashed and continuous lines.

Improve scalability of circular diagrams: Right now the circular diagrams are limited to work with 150 elements or less. Beyond that point, the diagrams become too cluttered to be useful. New techniques can be explored to allow the visualization of up to thousands of elements. This may be done by dynamically collapsing groups of courses into single nodes and expanding them as necessary.

Integration with Bear Tracks: The functionality of the Course Browser is complementary to Bear Tracks in the sense that the former allows students to efficiently explore the course catalog and the latter provides the ability to register in courses. If the functionality of the Course Browser was integrated into Bear Tracks, the users would be able to perform these tasks using a single interface.

Extension of the box diagram to show indirect dependencies: Some of the participants suggested that the box diagram be extended to show indirect dependencies. Figure 6.1 shows a concept drawing of how this could be done. The diagram is separated into layers. The first layer of the diagram shows the selected course. Each subsequent level is used to show a step in the indirect requisites tree. For simplicity, each of the elements of an All of dependency is represented by an arrow, and Any of relationships are shown by stacking courses one on top of the other; however, using boxes around them is also possible. A blue arrow indicates a prerequisite, and an orange arrow a co-requisite.

In order to see the next level, the user must select one or more of the courses in each of the choice stacks. All of the selected courses are then aggregated to define the courses that can be taken in the next level. The choice is necessary to aggregate the requisites of courses in the next level maintaining the same semantics. In addition, this removes the clutter of showing multiple paths at the same time. To display another path, the user can simply select a different choice.

In addition, the nodes can be annotated to distinguish between different characteristics. For example, the color of the nodes can show selected courses and courses already taken, and the border of the nodes can be solid if the course can be taken immediately (*i.e.*, all its dependencies are satisfied), or dashed, otherwise.

6.2.2 Course Planner

While the Course Browser proved to be a useful tool for performing basic course planning, it is still insufficient to accomplish the most difficult planning tasks. Therefore, we suggest the creation of a tool that extends the functionality of the Course Browser for performing course planning. This section provides some hints of how this could be accomplished.

The planning tool should have access to the student's academic record and desired program of study. With this information, it can suggest to the student which courses to take next, taking into account the requisites of each of them,

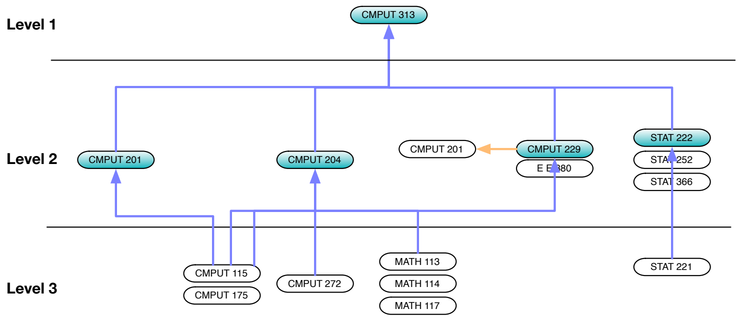


Figure 6.1: Concept drawing of a box diagram that allows the exploration of indirect dependencies.

courses already taken by the student, and the conditions of the program. The information about each of the programs will be contained in templates, supplied by each department. Each template will contain the requisites of graduation as well as some sample paths to accomplish them.

We envision this tool to have a drag and drop user interface, where courses would be represented with tiles and requisites by arrows, using the semantics described in the previous section for the indirect requisite box diagram. Initially, the tool will show the suggested template of the program along with all the courses already taken by the student. Then, the user can modify the program, by adding, removing or swapping courses. This would be done using the Course Browser's views. Once a course is added or swapped, the planner would immediately search for unfulfilled dependencies and prompt the user to select required courses using a box diagram. Courses that are not required by other courses, and courses with unsatisfied requisites would be highlighted.

Finally, the user can group courses into terms and choose particular sections using the scheduling information provided by Bear Tracks. At any time, the users may save their work and even keep track of alternative plans for comparison.

6.2.3 Dependency visualization framework

Each dependency visualization domain has a variety of ways to visualize its particular kinds of dependencies. Since each software tool in each of these domains has to deal with dependencies, it is conceivable to define a common unified visualization framework.

A dependency visualization framework would provide the ability to model and visualize a wider range of relationships between elements. For example, in an organizational chart, it is often useful to document the work relationships among coworkers in teams. These relationships crosscut the tree hierarchy and, therefore, do not fit into the simple hierarchy model. Furthermore, a given individual may participate in a variety of projects, and how their relationship is visualized depends on the particular view that is desired (*e.g.* project versus management reporting).

At this stage we do not fully understand how to visualize dependencies. The work presented here, however, lays a basis to develop a framework that will support both static and interactive diagrams and enable us to test out different visualization ideas. Because dependencies, in general, and even supposedly simple hierarchies, are challenging to lay out automatically, the visualization framework must allow for interaction. This interaction would be implemented using a rich-prospect browsing interface.

To create the visualization diagrams requires a layout engine capable of rendering different styles of diagram. Since a purely algorithmic layout approach rarely works to complete satisfaction, it is important to permit users to specify hints and constraints. As this increases the complexity of the users' interaction with the visualization, the system must have the ability to remember past hints and layouts from other users, possibly experts.

Bibliography

[1] APPLE INC, *Introduction to apple human interface guidelines*. `http://developer.apple.com/documentation/userexperience/Conceptual/AppleHIGuidelines/XHIGIntro/XHIGIntro.html`, 2009.

[2] G. D. BATTISTA, I. G. TOLLIS, P. EADES, AND R. TAMASSIA, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1998.

[3] S. BLY, *Field work: is it product work?*, interactions, 4 (1997), pp. 25–30.

[4] J. A. BONDY AND U. S. R. MURTY, *Graph theory with applications*, North Holland, 1976.

[5] B. BUXTON, *Sketching and experience design*, in HCI Seminar on People, Computers and Design, Stanford University, 2007. `http://www.youtube.com/watch?v=xx1WveKV7aE`.

[6] C. CHEN, *Information visualization: beyond the horizon*, Springer, 2004.

[7] C. CONATI AND H. MACLAREN, *Exploring the role of individual differences in information visualization*, in AVI '08: Proceedings of the working conference on Advanced visual interfaces, New York, NY, USA, 2008, ACM, pp. 199–206.

[8] W. D. ELLIS, *A source book of Gestalt psychology*, Routledge, 2 ed., 1999.

[9] J. FUITE, *A Large-scale Education System, K-16, Visualized and Navigated*, Journal of Education, Informatics and Cybernetics, 1 (2009), p. 7.

[10] E. GAMMA, R. HELM, R. JOHNSON, AND J. M. VLISSIDES, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1 ed., 1994.

[11] E. GANSNER AND Y. KOREN, *Improved Circular Layouts*, in Graph Drawing: 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006: Revised Papers, Springer Verlag, 2007, p. 386.

[12] F. HARARY, *Graph Theory*, Perseus Books, 15 ed., 2001.

[13] J. HEER, S. K. CARD, AND J. A. LANDAY, *prefuse: a toolkit for interactive information visualization*, in CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 2005, ACM, pp. 421–430.

[14] D. HOLTEN, *Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data*, IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, (2006), pp. 741–748.

[15] D. HOLTEN AND J. J. VAN WIJK, *Force-directed edge bundling for graph visualization*, in Eurographics/ IEEE-VGTC Symposium on Visualization, H.-C. Hege, I. Hotz, and T. Munzner, eds., vol. 28, IEEE, 2009.

[16] JEFFREY MICHAEL HEER, *Flare: Data visualization for the web.* `http://flare.prefuse.org/`, 2009.

[17] M. KARNAUGH, *The map method for synthesis of combinational logic circuits*, AIEE Transactions Comm. Elec, 72 (1953), pp. 593–599.

[18] B. KERR, *Thread Arcs: An Email Thread Visualization*, Proceedings of the 2003 IEEE Symposium on Information Visualization, (2003), p. 27.

[19] M. KRZYWINSKI, *Circos: circularly composited genomic data and annotation imager.* `http://mkweb.bcgsc.ca/circos/`, Visited June 15, 2009.

[20] J. LARKIN AND H. SIMON, *Why a diagram is (sometimes) worth ten thousand words*, Cognitive Science, (1987).

[21] W. LIDWELL, J. BUTLER, AND K. HOLDEN, *Universal Principles of Design: A Cross Disciplinary Reference*, Rockport Publishers, 2003.

[22] R. LIKERT, *A technique for the measurement of attitudes*, sn, 1932.

[23] S. MICROSYSTEMS, *Java look and feel design guidelines: advanced topics*, vol. 2, Addison-Wesley, illustrated ed., 2001.

[24] D. A. NORMAN, *The Design of Everyday Things*, Basic Books, reprint, illustrated ed., 2002, pp. 1–34.

[25] U. OF ALBERTA, ed., *2009-2010 University of Alberta Calendar*, University of Alberta, 2009.

[26] J. PREECE, *Interaction Design*, J. Wiley & Sons, New York, 2002.

[27] R DEVELOPMENT CORE TEAM, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

[28] S. RUECKER, *Affordances of prospect for academic users of interpretively-tagged text collections*, PhD thesis, University of Alberta, 2003.

[29] M. SARKAR AND M. H. BROWN, *Graphical fisheye views of graphs*, in CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, 1992, ACM, pp. 83–91.

[30] D. J. SHESKIN, *Handbook of Parametric and Nonparametric Statistical Procedures, Third Edition*, Chapman & Hall/CRC, 2003, pp. 382–384, 633–638.

[31] B. SHNEIDERMAN, *Tree visualization with tree-maps: 2-d space-filling approach*, ACM Trans. Graph., 11 (1992), pp. 92–99.

[32] B. SHNEIDERMAN, *Designing the User Interface*, Addison Wesley Longman, 1998.

[33] K. SUGIYAMA, *Graph drawing and applications for software and knowledge engineers*, World Scientific, 2002.

[34] J. TIDWELL, *Designing interfaces*, O'Reilly, illustrated ed., 2005.

[35] E. R. TUFTE, *Envisioning information*, Graphics Press, 5, illustrated ed., 1995.

[36] C. WARE, *Information visualization: perception for design*, Morgan Kaufmann series in interactive technologies, Morgan Kaufman, 2004.

[37] WIKIPEDIA, *Likert scale — wikipedia, the free encyclopedia*, 2009. [Online; accessed 4-May-2009; `http://en.wikipedia.org/w/index.php?title=Likert_scale&oldid=287068650`].

[38] ——, *Mann–whitney u — wikipedia, the free encyclopedia*, 2009. [Online; accessed 4-May-2009; `http://en.wikipedia.org/w/index.php?title=Mann%E2%80%93Whitney_U&oldid=286892139`].

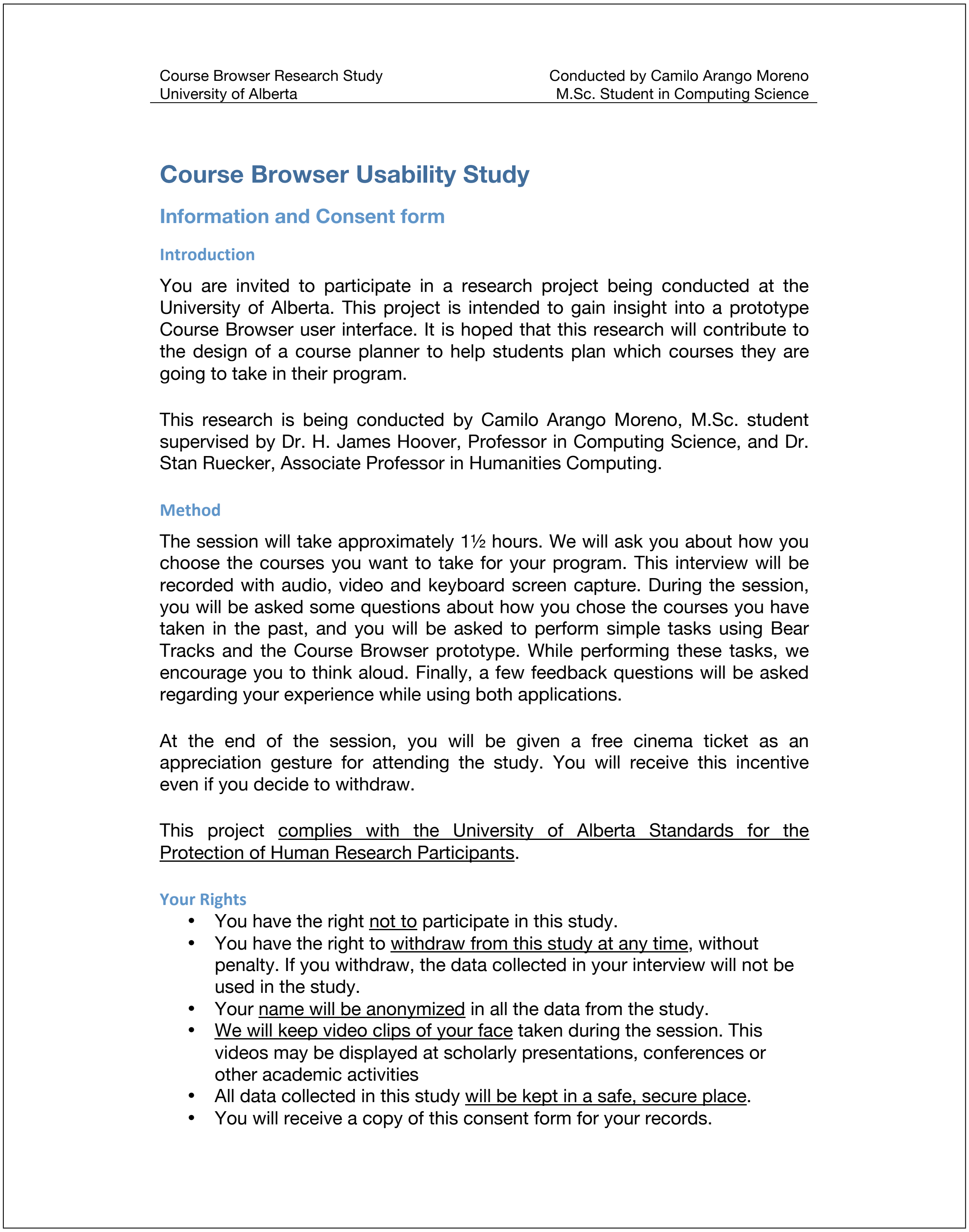
[39] ——, *Student's t-test — wikipedia, the free encyclopedia*, 2009. [Online; accessed 4-May-2009; `http://en.wikipedia.org/w/index.php?title=Student%27s_t-test&oldid=285704843`].

[40] ——, *Wilcoxon signed-rank test — wikipedia, the free encyclopedia*, 2009. [Online; accessed 4-May-2009; `http://en.wikipedia.org/w/index.php?title=Wilcoxon_signed-rank_test&oldid=280116509`].

[41] K.-P. YEE, D. FISHER, R. DHAMIJA, AND M. HEARST, *Animated exploration of dynamic graphs with radial layout*, in INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01), Washington, DC, USA, 2001, IEEE Computer Society, p. 43.

Appendix A

Usability study questionnaire



Course Browser Research Study
University of Alberta

Conducted by Camilo Arango Moreno
M.Sc. Student in Computing Science

Course Browser Usability Study

Information and Consent form

Introduction

You are invited to participate in a research project being conducted at the University of Alberta. This project is intended to gain insight into a prototype Course Browser user interface. It is hoped that this research will contribute to the design of a course planner to help students plan which courses they are going to take in their program.

This research is being conducted by Camilo Arango Moreno, M.Sc. student supervised by Dr. H. James Hoover, Professor in Computing Science, and Dr. Stan Ruecker, Associate Professor in Humanities Computing.

Method

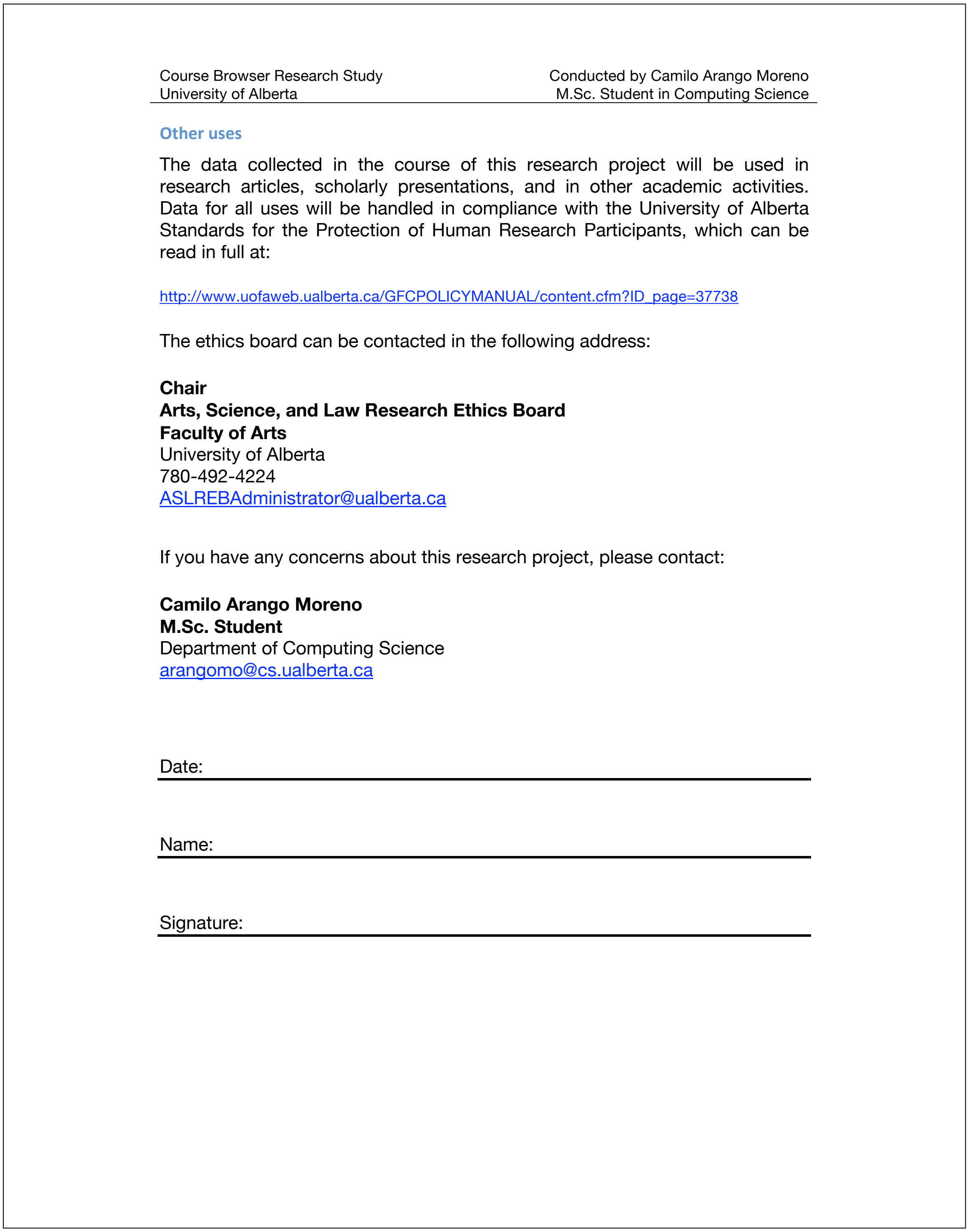
The session will take approximately 1½ hours. We will ask you about how you choose the courses you want to take for your program. This interview will be recorded with audio, video and keyboard screen capture. During the session, you will be asked some questions about how you chose the courses you have taken in the past, and you will be asked to perform simple tasks using Bear Tracks and the Course Browser prototype. While performing these tasks, we encourage you to think aloud. Finally, a few feedback questions will be asked regarding your experience while using both applications.

At the end of the session, you will be given a free cinema ticket as an appreciation gesture for attending the study. You will receive this incentive even if you decide to withdraw.

This project complies with the University of Alberta Standards for the Protection of Human Research Participants.

Your Rights

- You have the right not to participate in this study.
- You have the right to withdraw from this study at any time, without penalty. If you withdraw, the data collected in your interview will not be used in the study.
- Your name will be anonymized in all the data from the study.
- We will keep video clips of your face taken during the session. This videos may be displayed at scholarly presentations, conferences or other academic activities
- All data collected in this study will be kept in a safe, secure place.
- You will receive a copy of this consent form for your records.



Course Browser Research Study
University of Alberta

Conducted by Camilo Arango Moreno
M.Sc. Student in Computing Science

Other uses

The data collected in the course of this research project will be used in research articles, scholarly presentations, and in other academic activities. Data for all uses will be handled in compliance with the University of Alberta Standards for the Protection of Human Research Participants, which can be read in full at:

http://www.uofaweb.ualberta.ca/GFCPOLICYMANUAL/content.cfm?ID_page=37738

The ethics board can be contacted in the following address:

Chair
Arts, Science, and Law Research Ethics Board
Faculty of Arts
University of Alberta
780-492-4224
ASLREBAdministrator@ualberta.ca

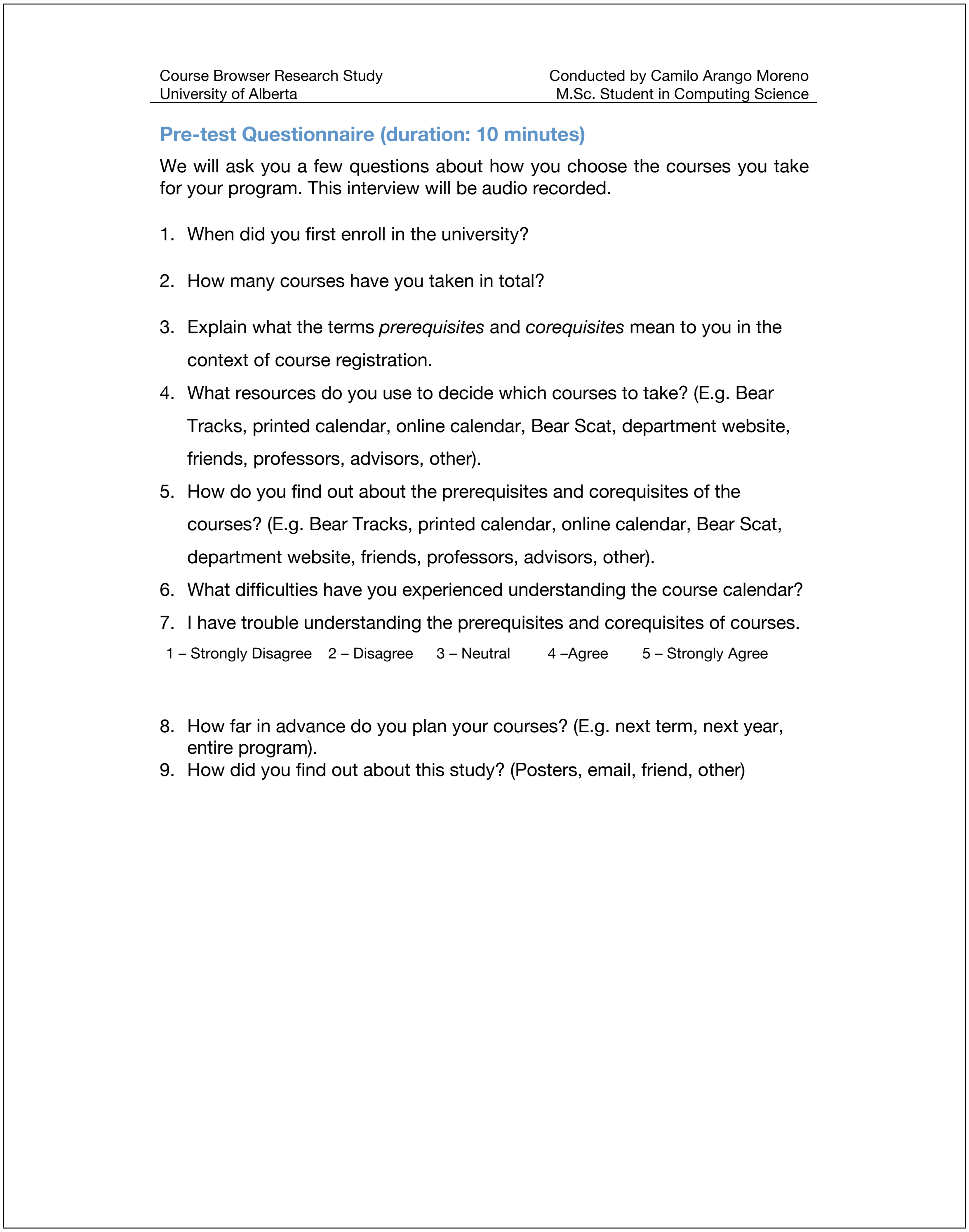
If you have any concerns about this research project, please contact:

Camilo Arango Moreno
M.Sc. Student
Department of Computing Science
arangomo@cs.ualberta.ca

Date: ________________________________

Name: ________________________________

Signature: ________________________________



Course Browser Research Study
University of Alberta

Conducted by Camilo Arango Moreno
M.Sc. Student in Computing Science

Pre-test Questionnaire (duration: 10 minutes)

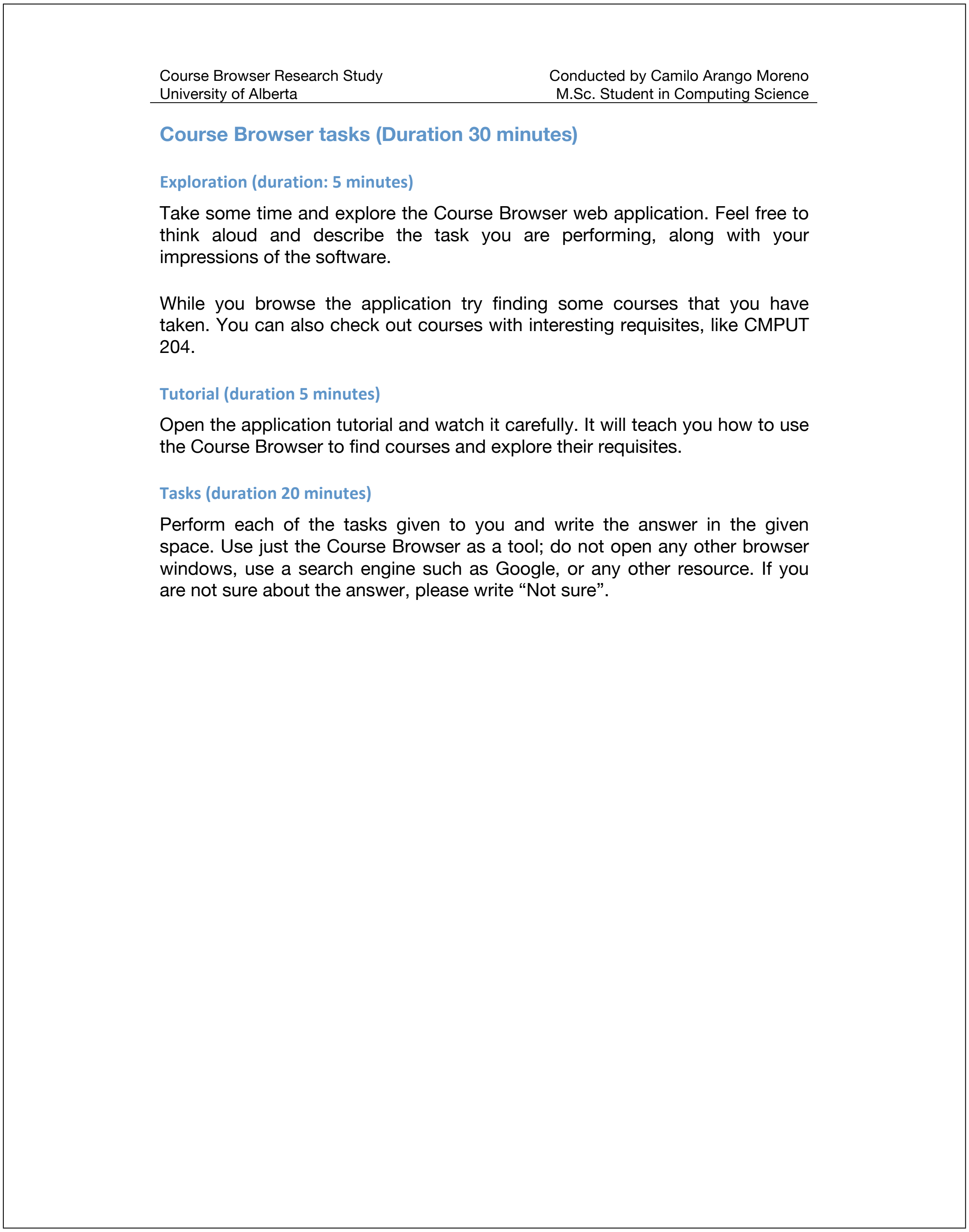
We will ask you a few questions about how you choose the courses you take for your program. This interview will be audio recorded.

1. When did you first enroll in the university?
2. How many courses have you taken in total?
3. Explain what the terms *prerequisites* and *corequisites* mean to you in the context of course registration.
4. What resources do you use to decide which courses to take? (E.g. Bear Tracks, printed calendar, online calendar, Bear Scat, department website, friends, professors, advisors, other).
5. How do you find out about the prerequisites and corequisites of the courses? (E.g. Bear Tracks, printed calendar, online calendar, Bear Scat, department website, friends, professors, advisors, other).
6. What difficulties have you experienced understanding the course calendar?
7. I have trouble understanding the prerequisites and corequisites of courses.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

8. How far in advance do you plan your courses? (E.g. next term, next year, entire program).
9. How did you find out about this study? (Posters, email, friend, other)

Course Browser Research Study | Conducted by Camilo Arango Moreno
University of Alberta | M.Sc. Student in Computing Science



Course Browser tasks (Duration 30 minutes)

Exploration (duration: 5 minutes)

Take some time and explore the Course Browser web application. Feel free to think aloud and describe the task you are performing, along with your impressions of the software.

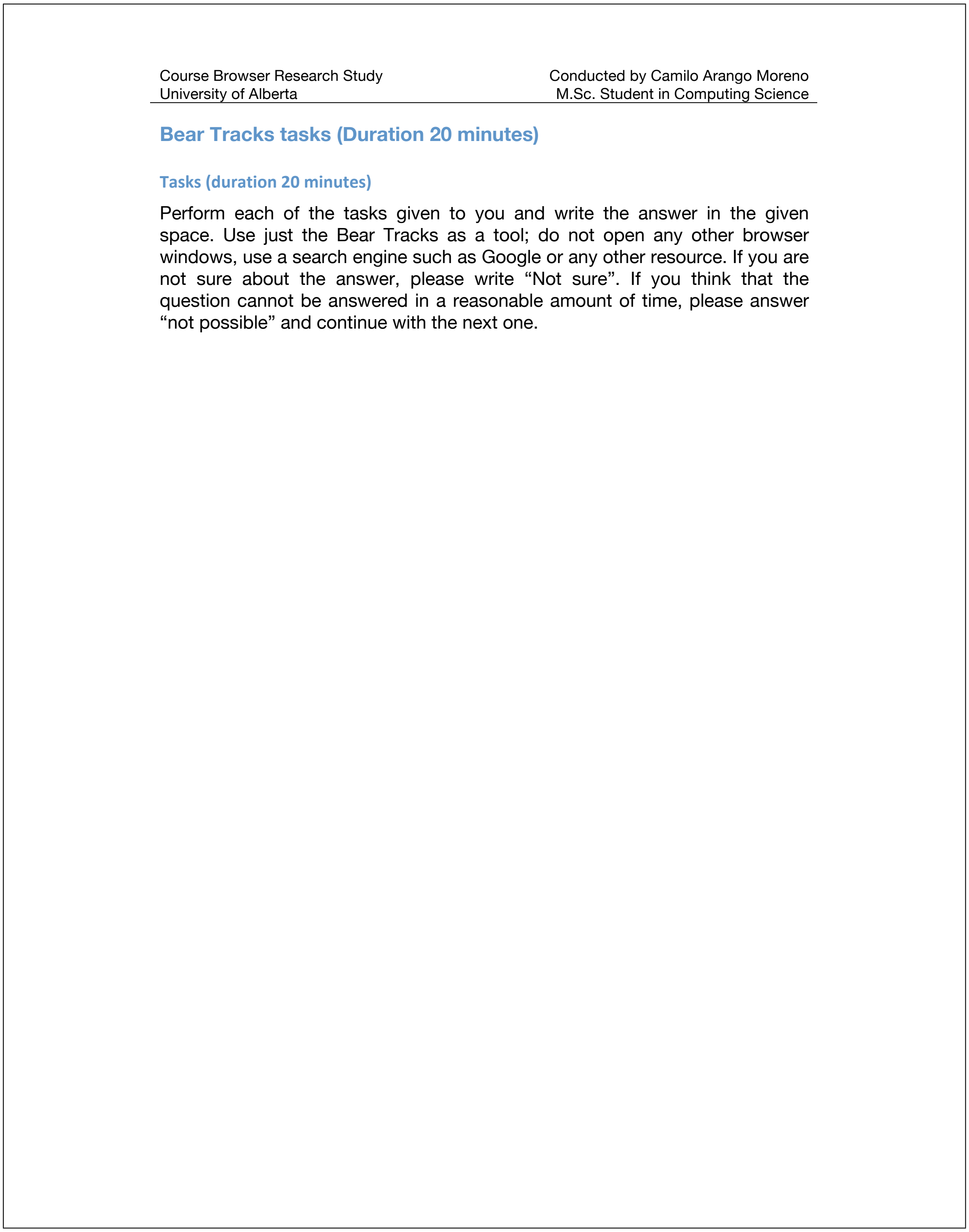
While you browse the application try finding some courses that you have taken. You can also check out courses with interesting requisites, like CMPUT 204.

Tutorial (duration 5 minutes)

Open the application tutorial and watch it carefully. It will teach you how to use the Course Browser to find courses and explore their requisites.

Tasks (duration 20 minutes)

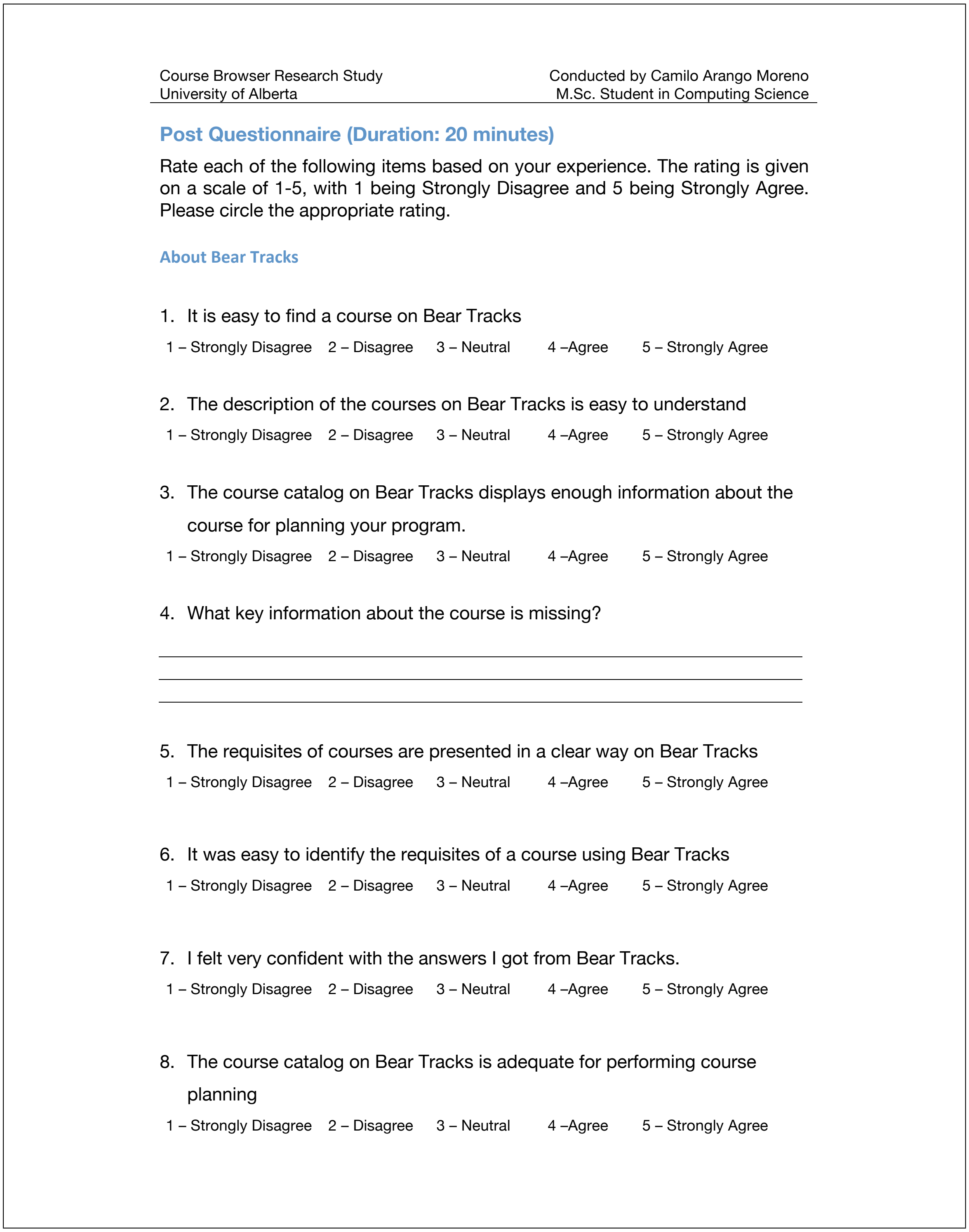
Perform each of the tasks given to you and write the answer in the given space. Use just the Course Browser as a tool; do not open any other browser windows, use a search engine such as Google, or any other resource. If you are not sure about the answer, please write “Not sure”.



Bear Tracks tasks (Duration 20 minutes)

Tasks (duration 20 minutes)

Perform each of the tasks given to you and write the answer in the given space. Use just the Bear Tracks as a tool; do not open any other browser windows, use a search engine such as Google or any other resource. If you are not sure about the answer, please write “Not sure”. If you think that the question cannot be answered in a reasonable amount of time, please answer “not possible” and continue with the next one.



Course Browser Research Study
University of Alberta

Conducted by Camilo Arango Moreno
M.Sc. Student in Computing Science

Post Questionnaire (Duration: 20 minutes)

Rate each of the following items based on your experience. The rating is given on a scale of 1-5, with 1 being Strongly Disagree and 5 being Strongly Agree. Please circle the appropriate rating.

About Bear Tracks

1. It is easy to find a course on Bear Tracks

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

2. The description of the courses on Bear Tracks is easy to understand

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

3. The course catalog on Bear Tracks displays enough information about the course for planning your program.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

4. What key information about the course is missing?

____

____

____

5. The requisites of courses are presented in a clear way on Bear Tracks

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

6. It was easy to identify the requisites of a course using Bear Tracks

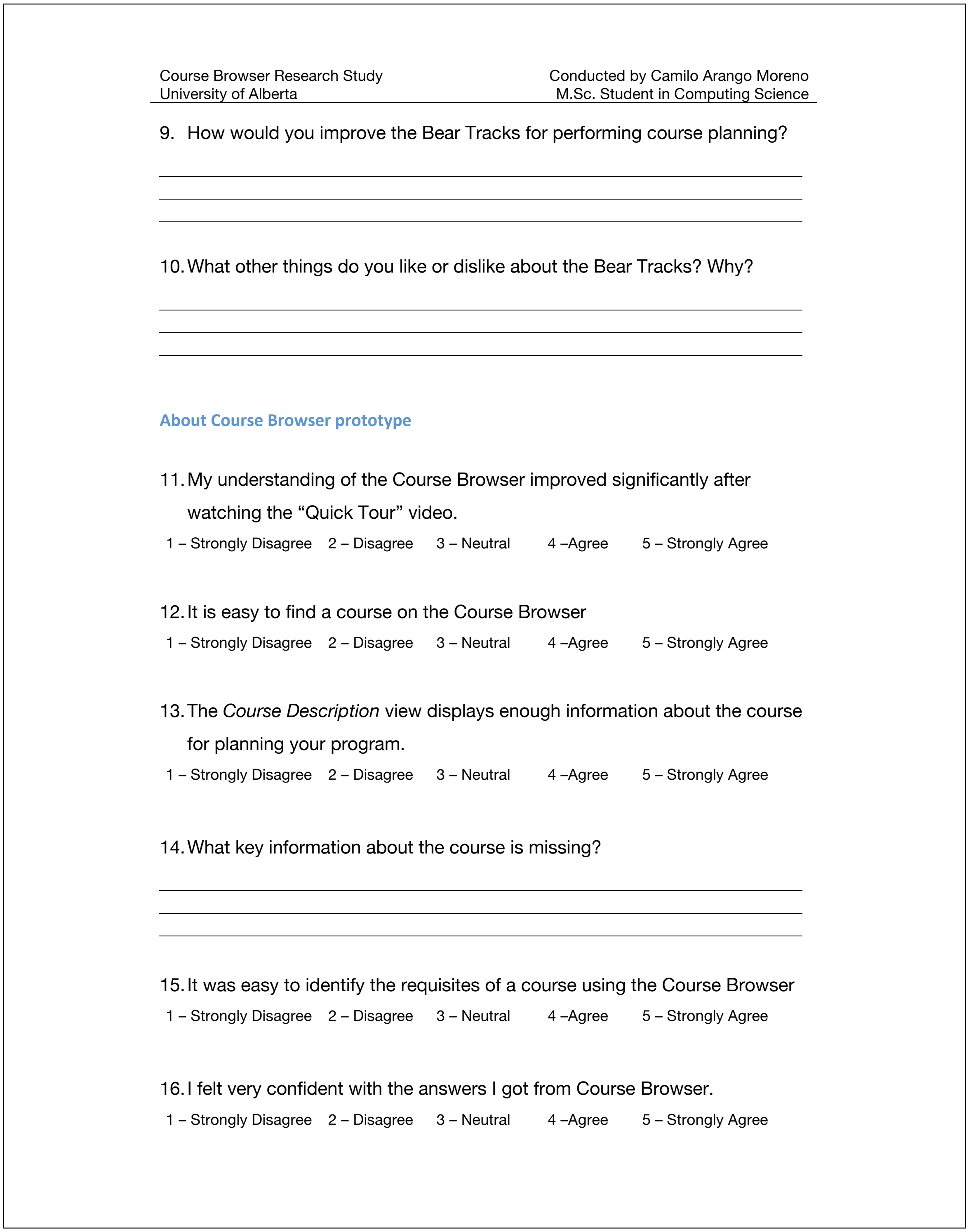
1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

7. I felt very confident with the answers I got from Bear Tracks.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

8. The course catalog on Bear Tracks is adequate for performing course planning

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree



9. How would you improve the Bear Tracks for performing course planning?

____

____

____

10. What other things do you like or dislike about the Bear Tracks? Why?

____

____

____

About Course Browser prototype

11. My understanding of the Course Browser improved significantly after watching the “Quick Tour” video.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

12. It is easy to find a course on the Course Browser

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

13. The *Course Description* view displays enough information about the course for planning your program.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

14. What key information about the course is missing?

____

____

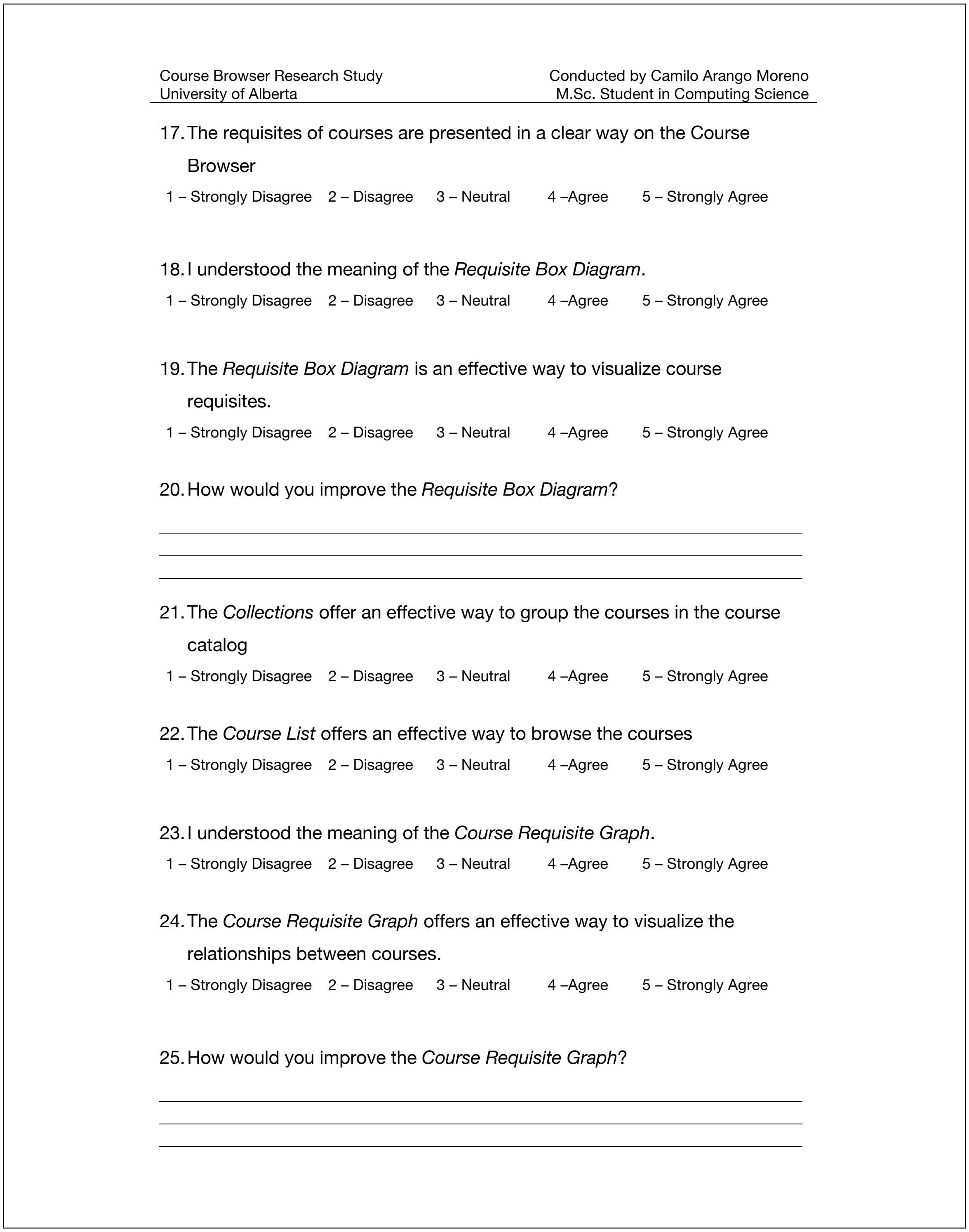
____

15. It was easy to identify the requisites of a course using the Course Browser

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

16. I felt very confident with the answers I got from Course Browser.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree



Course Browser Research Study
University of Alberta

Conducted by Camilo Arango Moreno
M.Sc. Student in Computing Science

17. The requisites of courses are presented in a clear way on the Course Browser

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

18. I understood the meaning of the *Requisite Box Diagram*.

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

19. The *Requisite Box Diagram* is an effective way to visualize course requisites.

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

20. How would you improve the *Requisite Box Diagram*?

21. The *Collections* offer an effective way to group the courses in the course catalog

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

22. The *Course List* offers an effective way to browse the courses

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

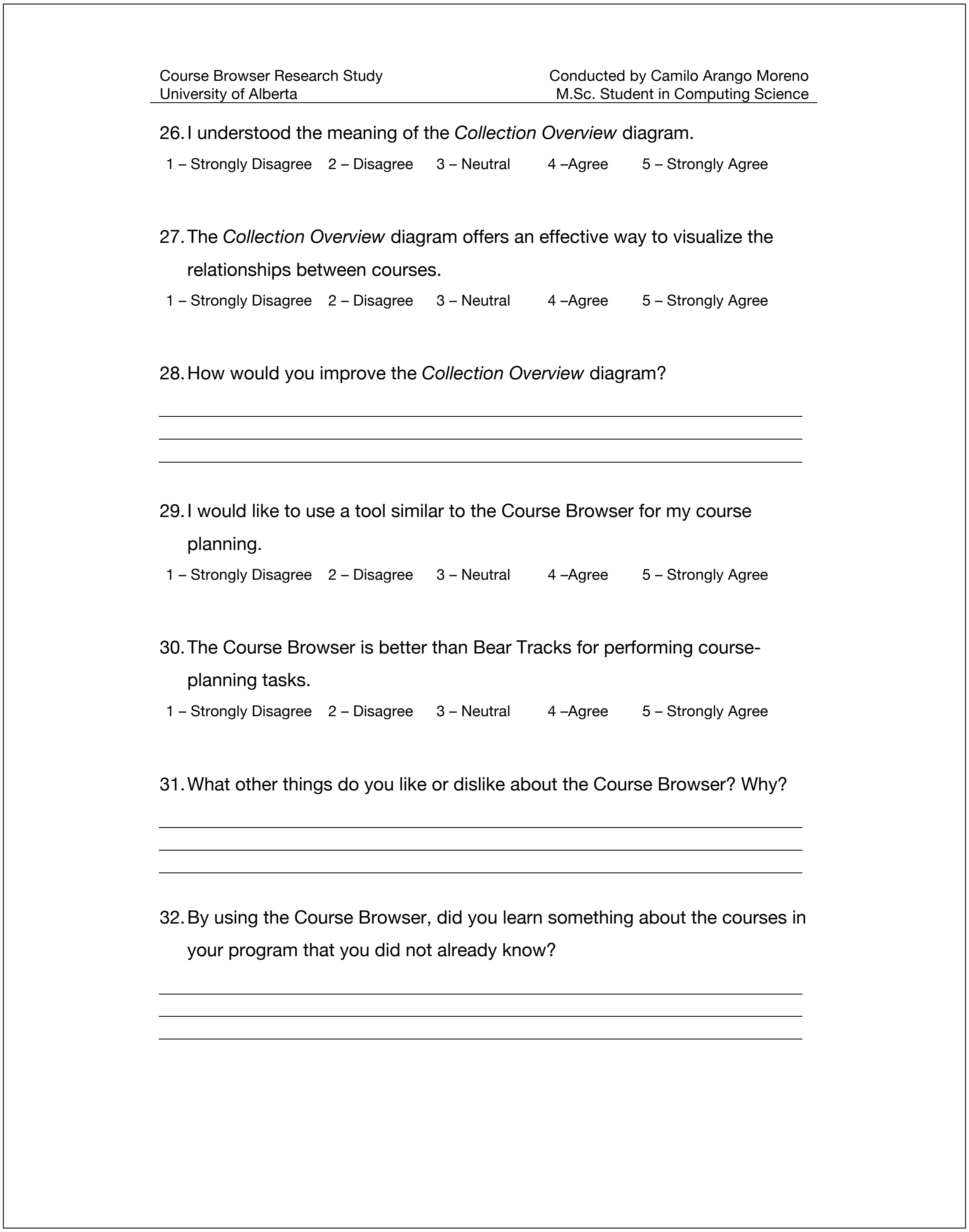
23. I understood the meaning of the *Course Requisite Graph*.

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

24. The *Course Requisite Graph* offers an effective way to visualize the relationships between courses.

1 – Strongly Disagree   2 – Disagree   3 – Neutral   4 –Agree   5 – Strongly Agree

25. How would you improve the *Course Requisite Graph*?



Course Browser Research Study
University of Alberta

Conducted by Camilo Arango Moreno
M.Sc. Student in Computing Science

26. I understood the meaning of the *Collection Overview* diagram.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

27. The *Collection Overview* diagram offers an effective way to visualize the relationships between courses.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

28. How would you improve the *Collection Overview* diagram?

29. I would like to use a tool similar to the Course Browser for my course planning.

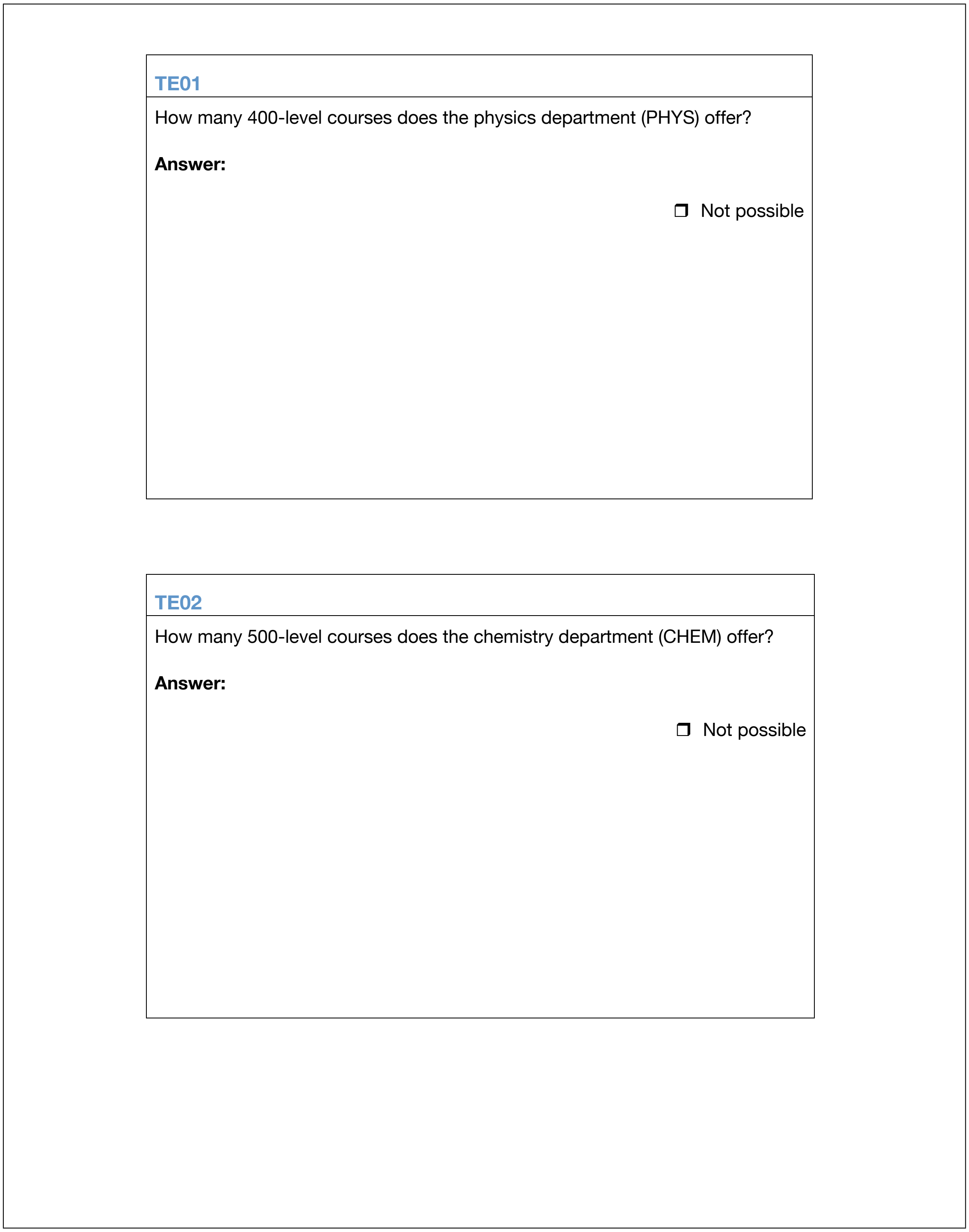
1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

30. The Course Browser is better than Bear Tracks for performing course-planning tasks.

1 – Strongly Disagree 2 – Disagree 3 – Neutral 4 –Agree 5 – Strongly Agree

31. What other things do you like or dislike about the Course Browser? Why?

32. By using the Course Browser, did you learn something about the courses in your program that you did not already know?



TE01

How many 400-level courses does the physics department (PHYS) offer?

Answer:

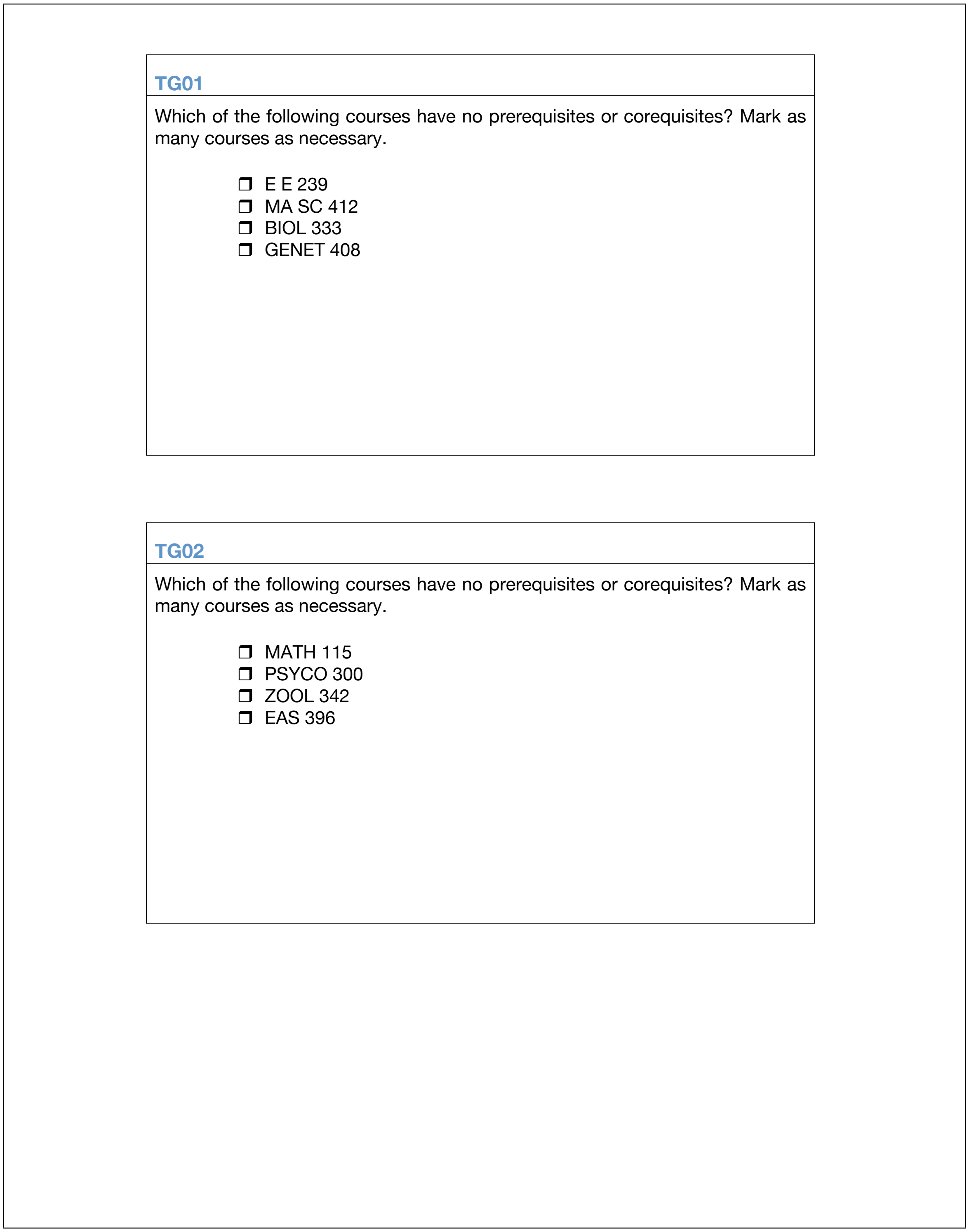
□ Not possible

TE02

How many 500-level courses does the chemistry department (CHEM) offer?

Answer:

□ Not possible



TG01

Which of the following courses have no prerequisites or corequisites? Mark as many courses as necessary.

- □ E E 239
- □ MA SC 412
- □ BIOL 333
- □ GENET 408

TG02

Which of the following courses have no prerequisites or corequisites? Mark as many courses as necessary.

- □ MATH 115
- □ PSYCO 300
- □ ZOOL 342
- □ EAS 396

TI03

1. Which of the following courses can be taken **before** BIOL 392 to meet its requirements?

   □ BIOL 201
   □ BIOL 207
   □ BIOL 208
   □ BIOL 380
   □ BIOL 592

2. Which of the following courses can be taken **simultaneously** with BIOL 392 to meet its requirements?

   □ BIOL 201
   □ BIOL 207
   □ BIOL 208
   □ BIOL 380
   □ BIOL 592

TI04

3. Which of the following courses can be taken **before** STAT 221 to meet its requirements?

   □ MATH 115
   □ MATH 118
   □ MATH 120
   □ STAT 141
   □ STAT 151

4. Which of the following courses can be taken **simultaneously** with STAT 221 to meet its requirements?

   □ MATH 115
   □ MATH 118
   □ MATH 120
   □ STAT 141
   □ STAT 151

TH01

Find the calendar description of PHYS 211 and create two different lists of courses can be taken before it in order to meet its requisites. Include indirect requisites in your lists.

Answer:

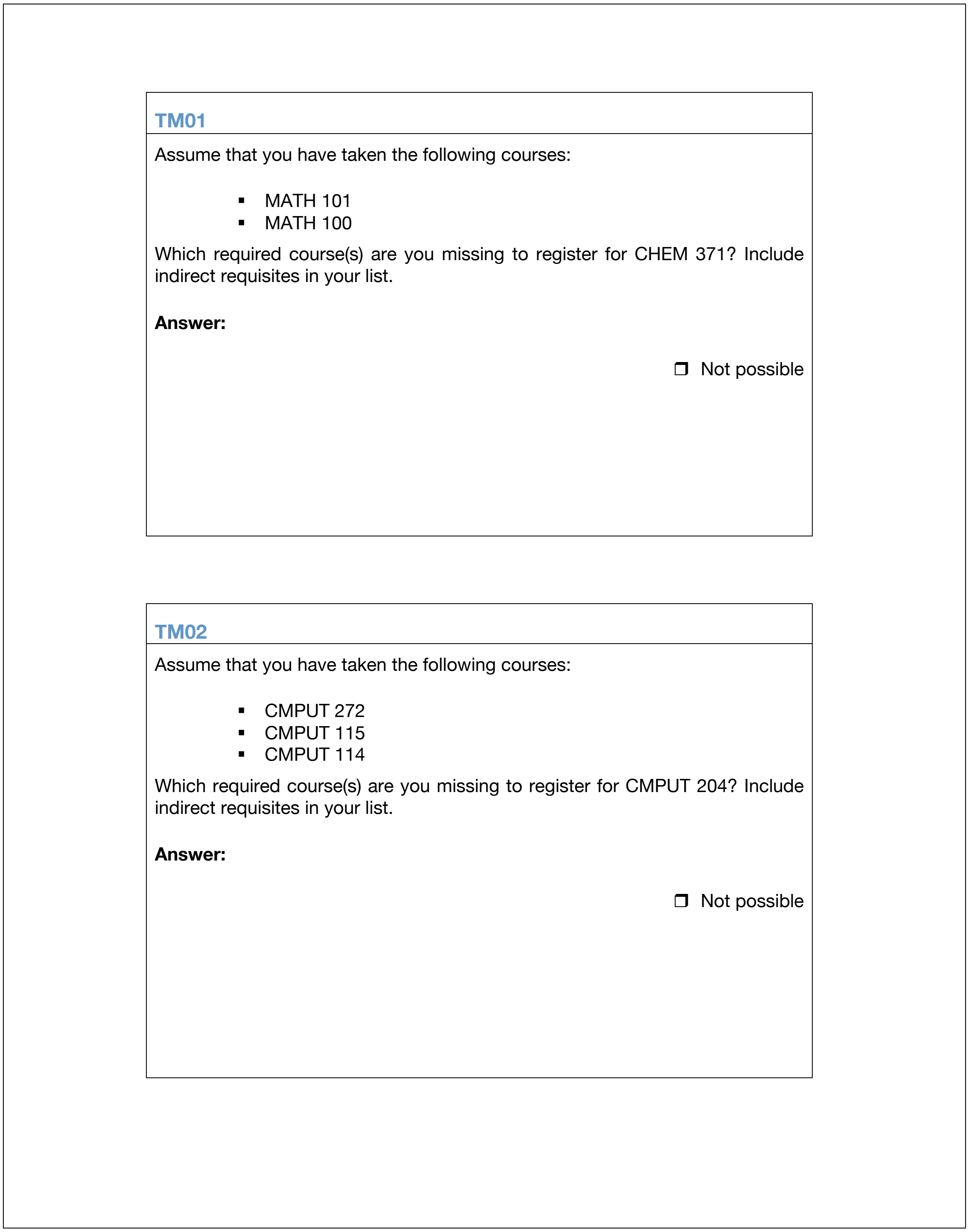
❐ Not possible

TH02

Find the calendar description of BOT 303 and create two different lists of courses can be taken before it in order to meet its requisites. Include indirect requisites in your lists.

Answer:

❐ Not possible



TM01

Assume that you have taken the following courses:

- MATH 101
- MATH 100

Which required course(s) are you missing to register for CHEM 371? Include indirect requisites in your list.

Answer:

□ Not possible

TM02

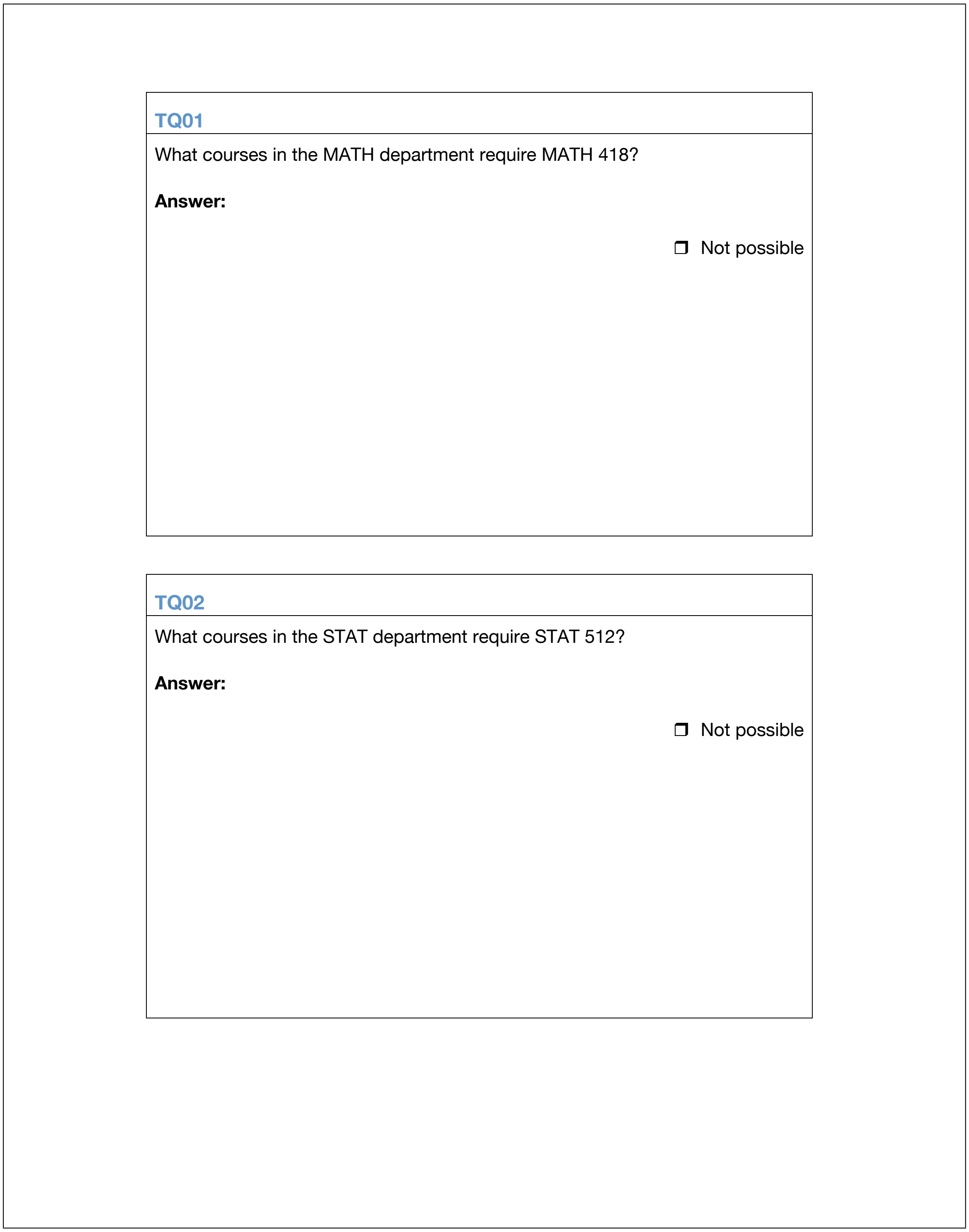
Assume that you have taken the following courses:

- CMPUT 272
- CMPUT 115
- CMPUT 114

Which required course(s) are you missing to register for CMPUT 204? Include indirect requisites in your list.

Answer:

□ Not possible



TQ01

What courses in the MATH department require MATH 418?

Answer:

□ Not possible

TQ02

What courses in the STAT department require STAT 512?

Answer:

□ Not possible

TP01

Assume you are student in the Computing Science program. You are interested in computer graphics. The University calendar provides the following description for your program [1]

> **Year 1**
>
> - CMPUT (114 and 115) or (174 and 175)
> - MATH 114, 115
> - *6 junior English
> - *12 in options (see Notes 1, 2)
>
> **Year 2**
>
> - *6 from CMPUT 201, 204, 229, 272, 291
> - MATH 120 or 125
> - *6 in Statistics (see Note 3)
> - *15 in options (see Notes 1, 2)
>
> **Year 3**
>
> - *12 in CMPUT at the 300-level or 400- level (see Note 4)
> - *18 in options (see Notes 1,2)
>
> **Year 4**
>
> - *12 in CMPUT at the 300-level or 400- level (see Note 4)
> - *18 in options (see Notes 1, 2)

During your first year of studies, you took the following mandatory courses:

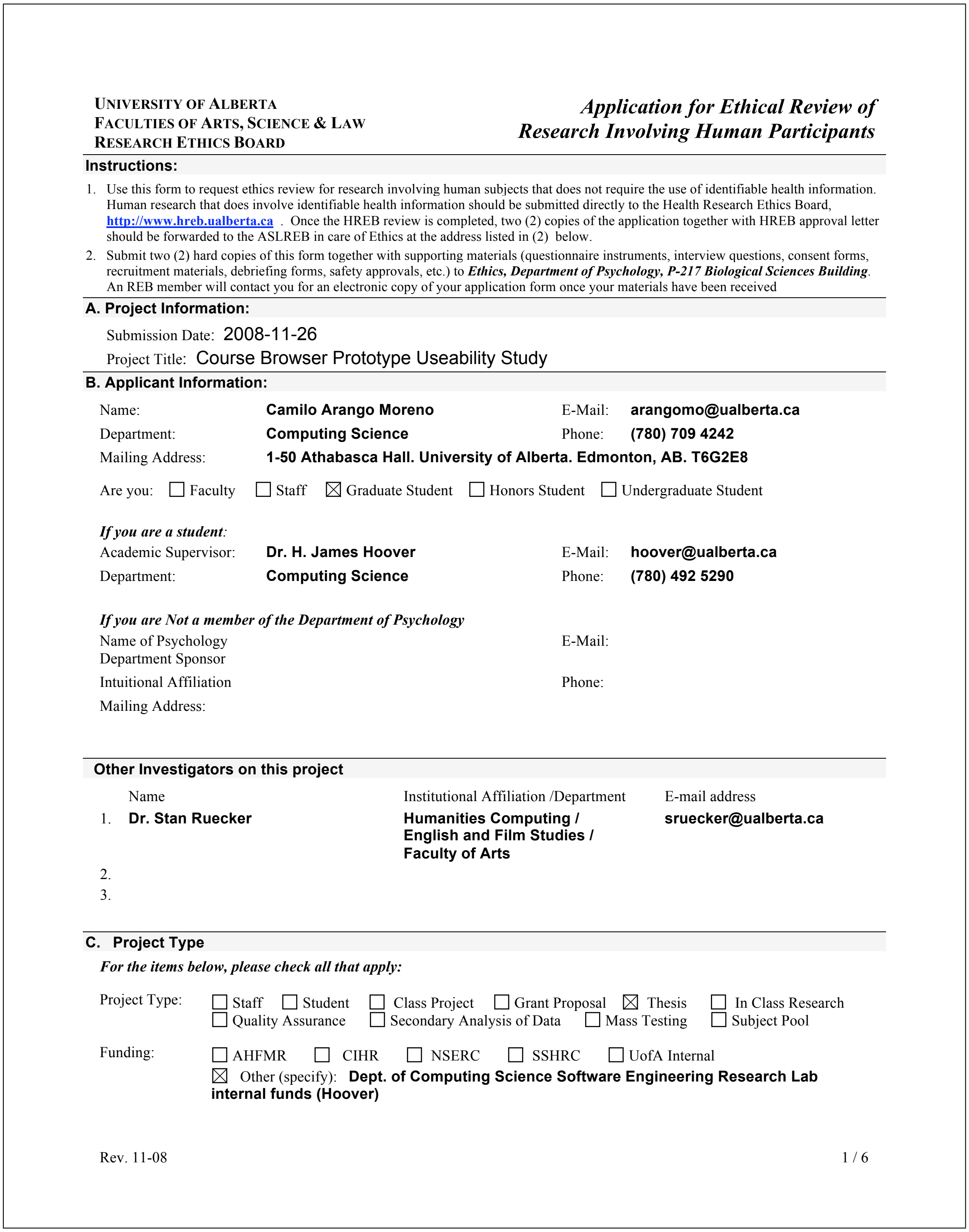
- CMPUT 174
- CMPUT 175
- MATH 114
- MATH 115

As you are starting your second year, you want to make sure you can take courses on computer graphics as soon as possible. As you browse though the course catalog, **CMPUT 411: Introduction to Computer Graphics** catches your attention.

Elaborate a plan of courses that you would take, term by term, in order to take CMPUT 411 as soon as possible. Use scratch paper as needed.

Appendix B

Usability study ethical review



UNIVERSITY OF ALBERTA
FACULTIES OF ARTS, SCIENCE & LAW
RESEARCH ETHICS BOARD

Application for Ethical Review of Research Involving Human Participants

Instructions:

1. Use this form to request ethics review for research involving human subjects that does not require the use of identifiable health information. Human research that does involve identifiable health information should be submitted directly to the Health Research Ethics Board, http://www.hreb.ualberta.ca . Once the HREB review is completed, two (2) copies of the application together with HREB approval letter should be forwarded to the ASLREB in care of Ethics at the address listed in (2) below.
2. Submit two (2) hard copies of this form together with supporting materials (questionnaire instruments, interview questions, consent forms, recruitment materials, debriefing forms, safety approvals, etc.) to ***Ethics, Department of Psychology, P-217 Biological Sciences Building***. An REB member will contact you for an electronic copy of your application form once your materials have been received

A. Project Information:

Submission Date: 2008-11-26

Project Title: Course Browser Prototype Useability Study

B. Applicant Information:

Name: **Camilo Arango Moreno** E-Mail: **arangomo@ualberta.ca**

Department: **Computing Science** Phone: **(780) 709 4242**

Mailing Address: **1-50 Athabasca Hall. University of Alberta. Edmonton, AB. T6G2E8**

Are you: □ Faculty □ Staff ☒ Graduate Student □ Honors Student □ Undergraduate Student

If you are a student:

Academic Supervisor: **Dr. H. James Hoover** E-Mail: **hoover@ualberta.ca**

Department: **Computing Science** Phone: **(780) 492 5290**

If you are Not a member of the Department of Psychology

Name of Psychology Department Sponsor E-Mail:

Intuitional Affiliation Phone:

Mailing Address:

Other Investigators on this project

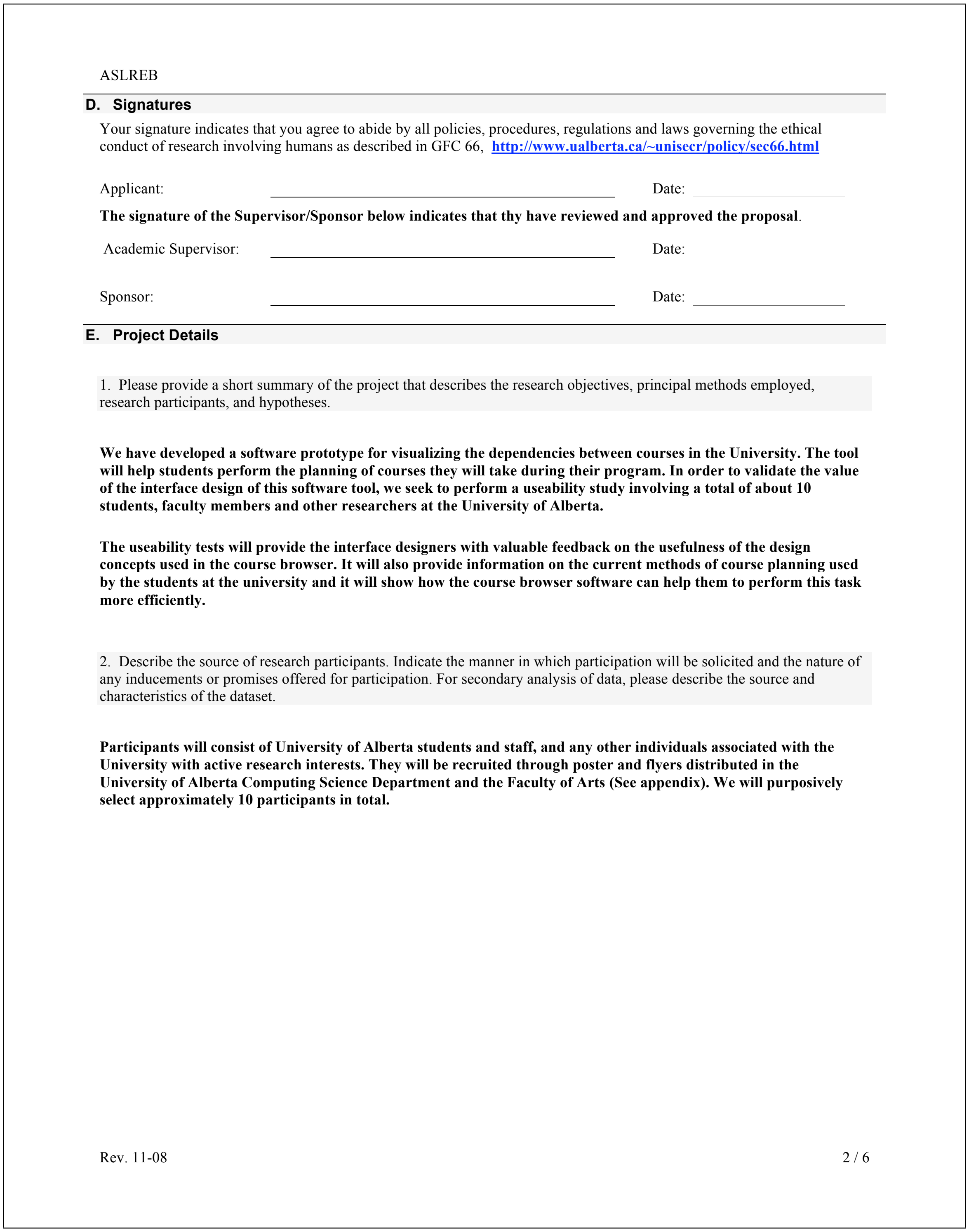
	Name	Institutional Affiliation /Department	E-mail address
1.	**Dr. Stan Ruecker**	**Humanities Computing / English and Film Studies / Faculty of Arts**	**sruecker@ualberta.ca**
2.			
3.			

C. Project Type

For the items below, please check all that apply:

Project Type: □ Staff □ Student □ Class Project □ Grant Proposal ☒ Thesis □ In Class Research
□ Quality Assurance □ Secondary Analysis of Data □ Mass Testing □ Subject Pool

Funding: □ AHFMR □ CIHR □ NSERC □ SSHRC □ UofA Internal
☒ Other (specify): **Dept. of Computing Science Software Engineering Research Lab internal funds (Hoover)**



ASLREB

D. Signatures

Your signature indicates that you agree to abide by all policies, procedures, regulations and laws governing the ethical conduct of research involving humans as described in GFC 66, **http://www.ualberta.ca/~unisecr/policy/sec66.html**

Applicant: ______________________________ Date: ______________

The signature of the Supervisor/Sponsor below indicates that thy have reviewed and approved the proposal.

Academic Supervisor: ______________________________ Date: ______________

Sponsor: ______________________________ Date: ______________

E. Project Details

1. Please provide a short summary of the project that describes the research objectives, principal methods employed, research participants, and hypotheses.

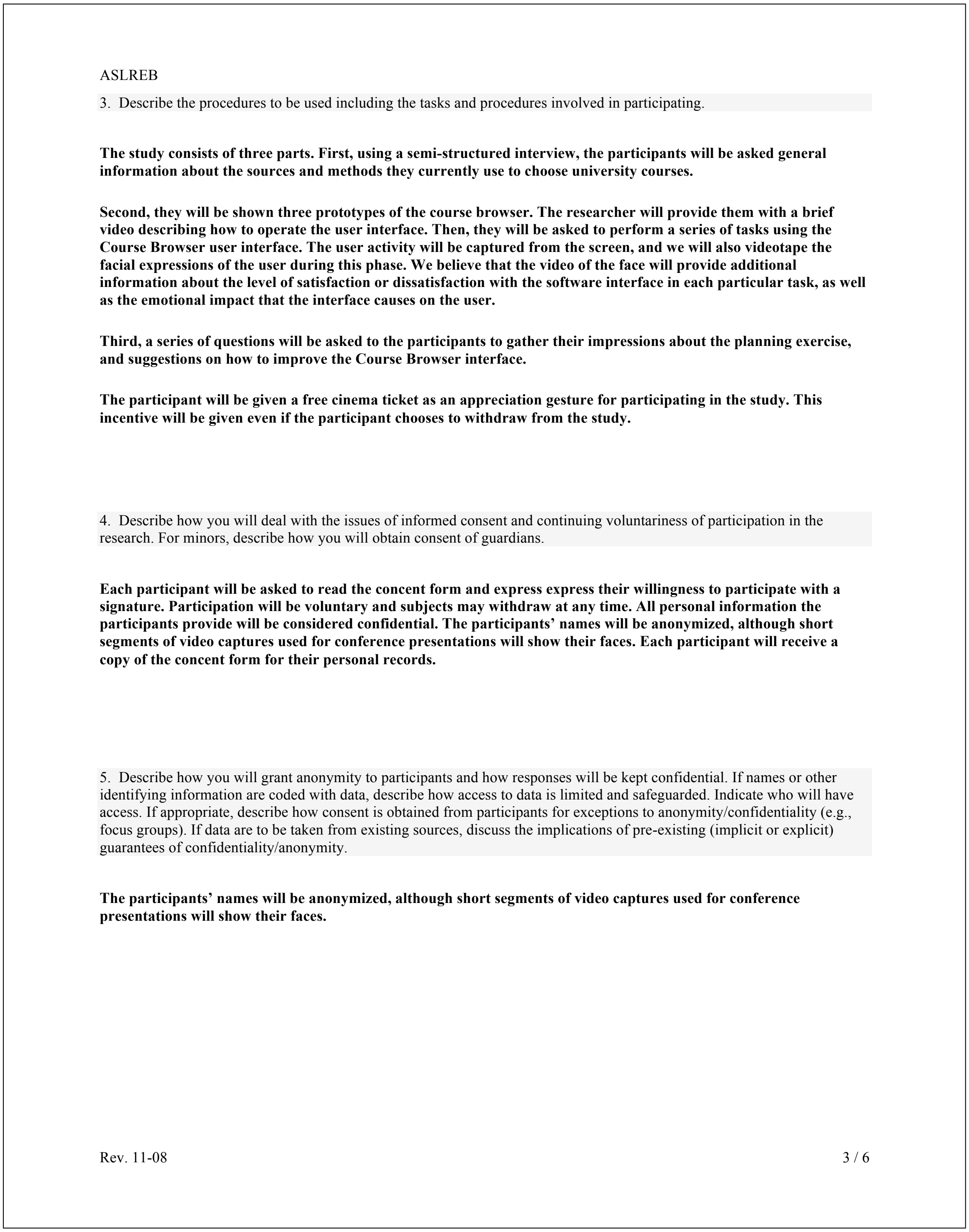
We have developed a software prototype for visualizing the dependencies between courses in the University. The tool will help students perform the planning of courses they will take during their program. In order to validate the value of the interface design of this software tool, we seek to perform a useability study involving a total of about 10 students, faculty members and other researchers at the University of Alberta.

The useability tests will provide the interface designers with valuable feedback on the usefulness of the design concepts used in the course browser. It will also provide information on the current methods of course planning used by the students at the university and it will show how the course browser software can help them to perform this task more efficiently.

2. Describe the source of research participants. Indicate the manner in which participation will be solicited and the nature of any inducements or promises offered for participation. For secondary analysis of data, please describe the source and characteristics of the dataset.

Participants will consist of University of Alberta students and staff, and any other individuals associated with the University with active research interests. They will be recruited through poster and flyers distributed in the University of Alberta Computing Science Department and the Faculty of Arts (See appendix). We will purposively select approximately 10 participants in total.

Rev. 11-08 2 / 6



3. Describe the procedures to be used including the tasks and procedures involved in participating.

The study consists of three parts. First, using a semi-structured interview, the participants will be asked general information about the sources and methods they currently use to choose university courses.

Second, they will be shown three prototypes of the course browser. The researcher will provide them with a brief video describing how to operate the user interface. Then, they will be asked to perform a series of tasks using the Course Browser user interface. The user activity will be captured from the screen, and we will also videotape the facial expressions of the user during this phase. We believe that the video of the face will provide additional information about the level of satisfaction or dissatisfaction with the software interface in each particular task, as well as the emotional impact that the interface causes on the user.

Third, a series of questions will be asked to the participants to gather their impressions about the planning exercise, and suggestions on how to improve the Course Browser interface.

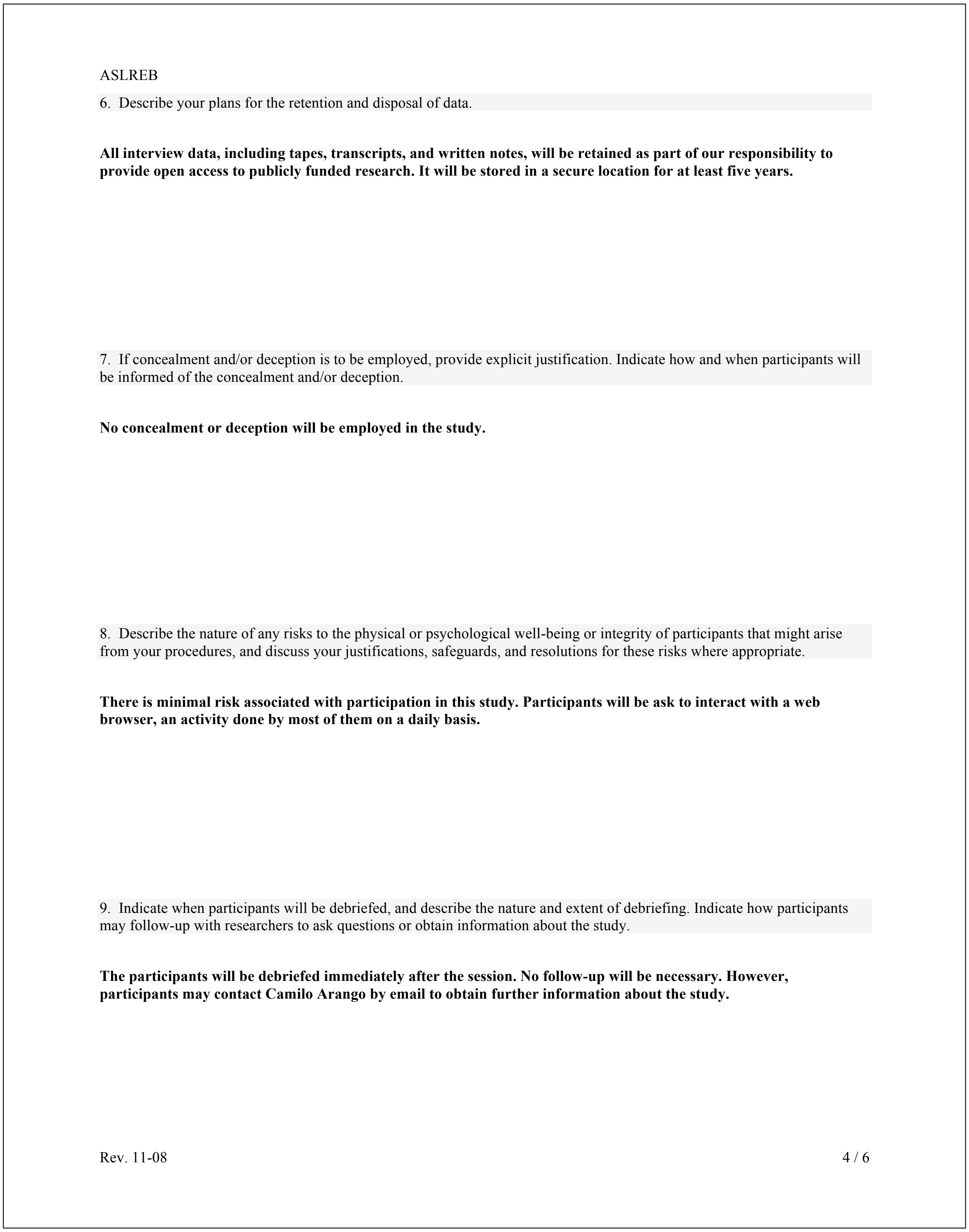
The participant will be given a free cinema ticket as an appreciation gesture for participating in the study. This incentive will be given even if the participant chooses to withdraw from the study.

4. Describe how you will deal with the issues of informed consent and continuing voluntariness of participation in the research. For minors, describe how you will obtain consent of guardians.

Each participant will be asked to read the concent form and express express their willingness to participate with a signature. Participation will be voluntary and subjects may withdraw at any time. All personal information the participants provide will be considered confidential. The participants' names will be anonymized, although short segments of video captures used for conference presentations will show their faces. Each participant will receive a copy of the concent form for their personal records.

5. Describe how you will grant anonymity to participants and how responses will be kept confidential. If names or other identifying information are coded with data, describe how access to data is limited and safeguarded. Indicate who will have access. If appropriate, describe how consent is obtained from participants for exceptions to anonymity/confidentiality (e.g., focus groups). If data are to be taken from existing sources, discuss the implications of pre-existing (implicit or explicit) guarantees of confidentiality/anonymity.

The participants' names will be anonymized, although short segments of video captures used for conference presentations will show their faces.



6. Describe your plans for the retention and disposal of data.

All interview data, including tapes, transcripts, and written notes, will be retained as part of our responsibility to provide open access to publicly funded research. It will be stored in a secure location for at least five years.

7. If concealment and/or deception is to be employed, provide explicit justification. Indicate how and when participants will be informed of the concealment and/or deception.

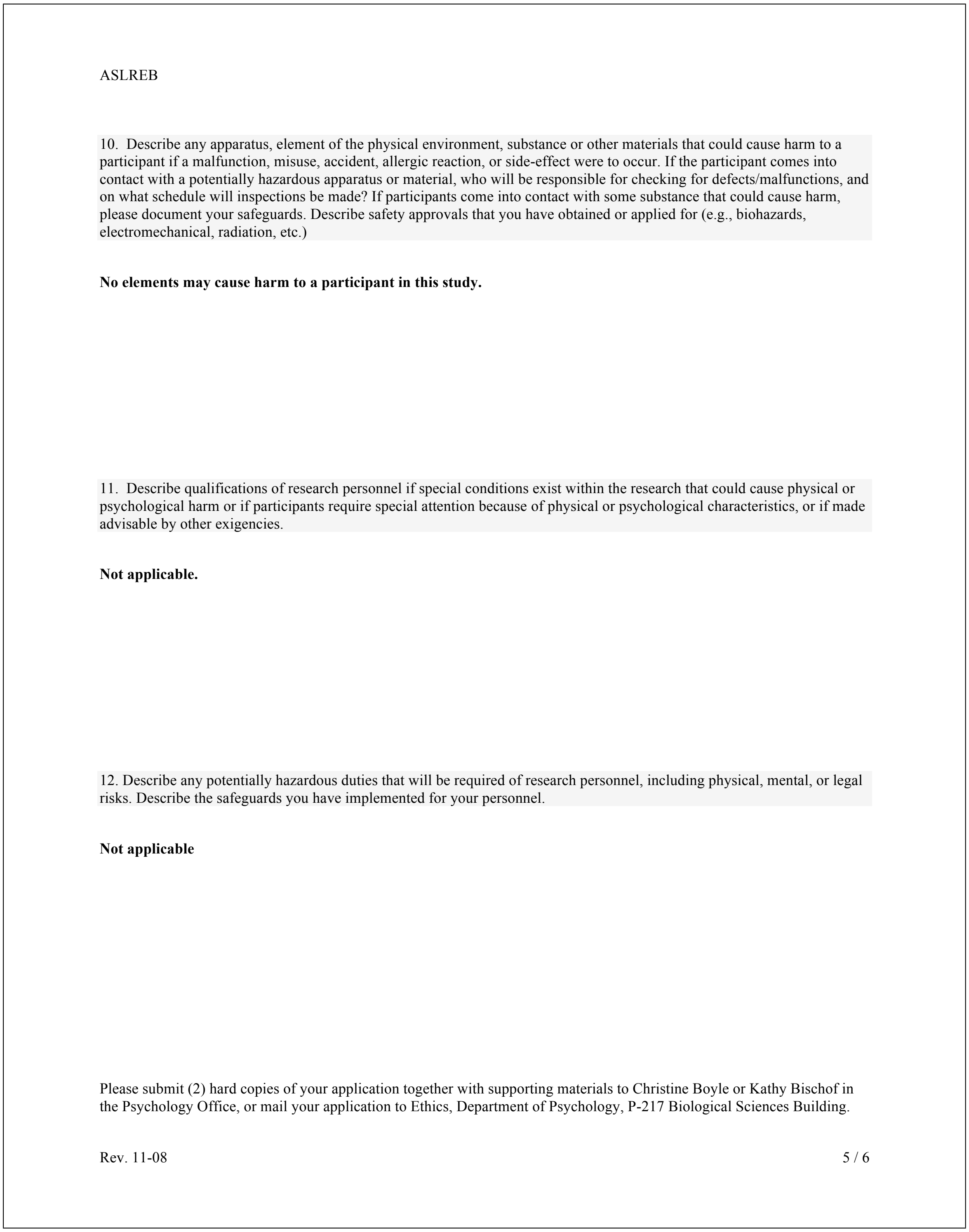
No concealment or deception will be employed in the study.

8. Describe the nature of any risks to the physical or psychological well-being or integrity of participants that might arise from your procedures, and discuss your justifications, safeguards, and resolutions for these risks where appropriate.

There is minimal risk associated with participation in this study. Participants will be ask to interact with a web browser, an activity done by most of them on a daily basis.

9. Indicate when participants will be debriefed, and describe the nature and extent of debriefing. Indicate how participants may follow-up with researchers to ask questions or obtain information about the study.

The participants will be debriefed immediately after the session. No follow-up will be necessary. However, participants may contact Camilo Arango by email to obtain further information about the study.



10. Describe any apparatus, element of the physical environment, substance or other materials that could cause harm to a participant if a malfunction, misuse, accident, allergic reaction, or side-effect were to occur. If the participant comes into contact with a potentially hazardous apparatus or material, who will be responsible for checking for defects/malfunctions, and on what schedule will inspections be made? If participants come into contact with some substance that could cause harm, please document your safeguards. Describe safety approvals that you have obtained or applied for (e.g., biohazards, electromechanical, radiation, etc.)

No elements may cause harm to a participant in this study.

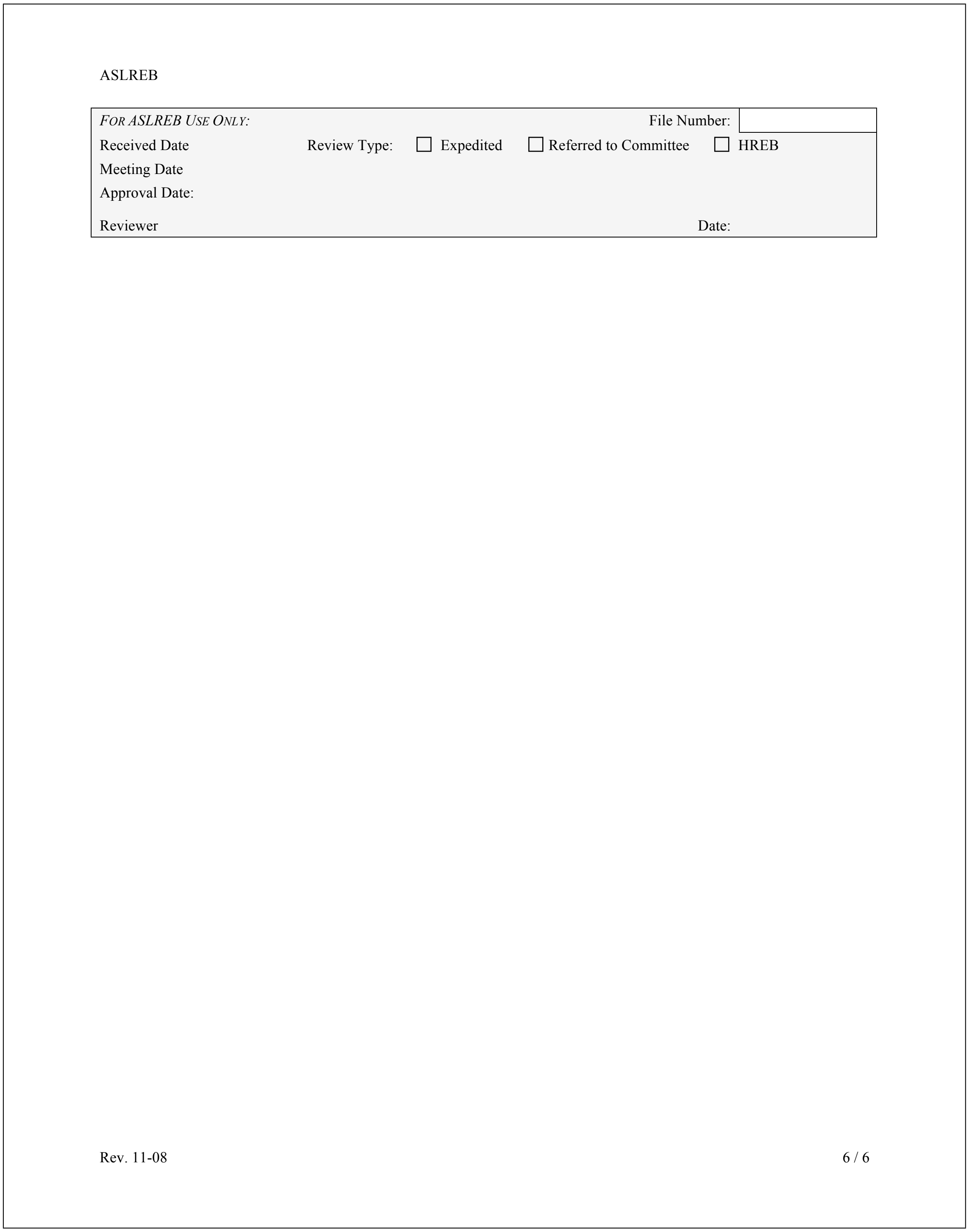
11. Describe qualifications of research personnel if special conditions exist within the research that could cause physical or psychological harm or if participants require special attention because of physical or psychological characteristics, or if made advisable by other exigencies.

Not applicable.

12. Describe any potentially hazardous duties that will be required of research personnel, including physical, mental, or legal risks. Describe the safeguards you have implemented for your personnel.

Not applicable

Please submit (2) hard copies of your application together with supporting materials to Christine Boyle or Kathy Bischof in the Psychology Office, or mail your application to Ethics, Department of Psychology, P-217 Biological Sciences Building.



ASLREB

FOR ASLREB USE ONLY: File Number:

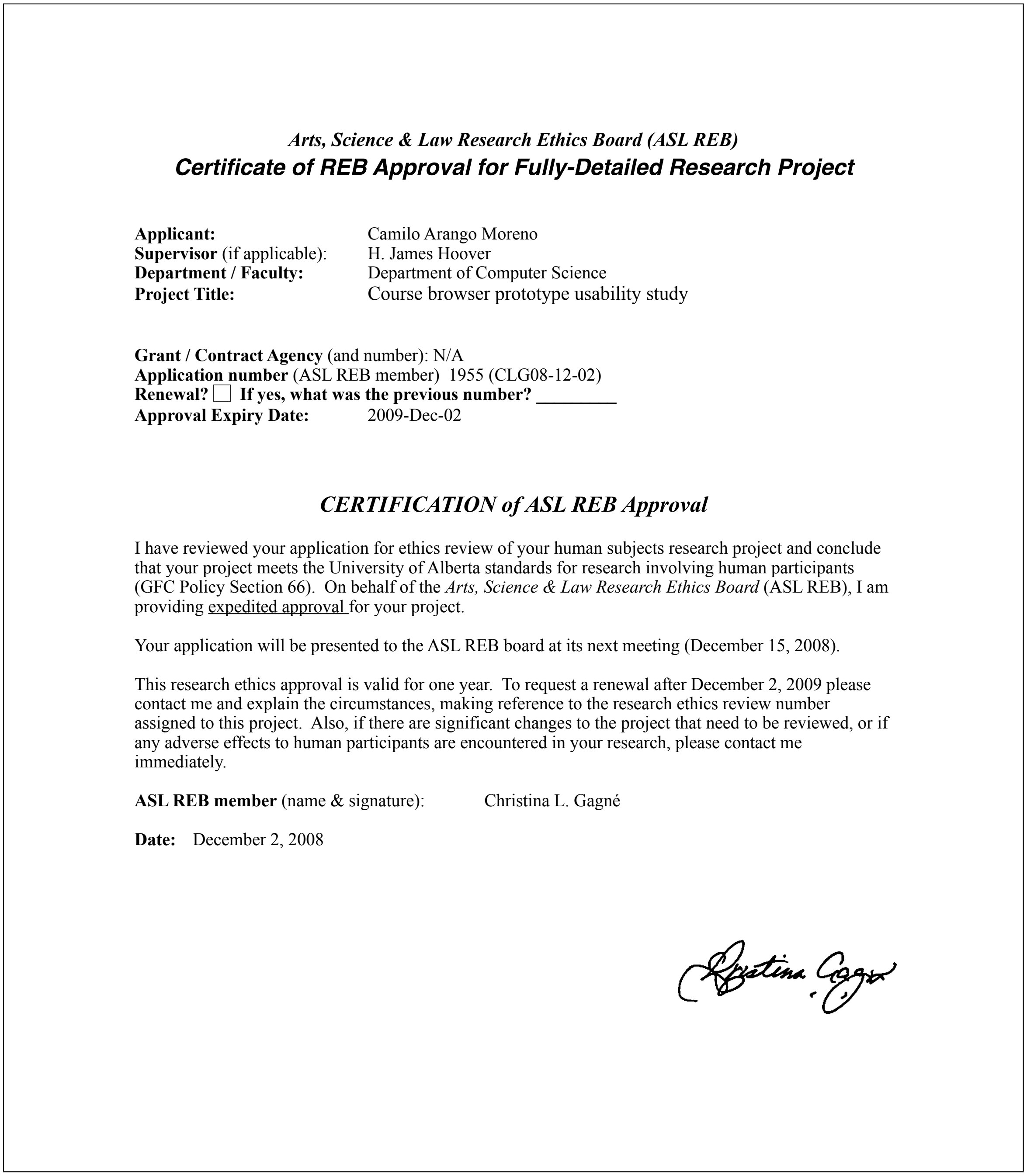
Received Date Review Type: □ Expedited □ Referred to Committee □ HREB

Meeting Date

Approval Date:

Reviewer Date:

Rev. 11-08 6 / 6



Arts, Science & Law Research Ethics Board (ASL REB)

Certificate of REB Approval for Fully-Detailed Research Project

Applicant: Camilo Arango Moreno
Supervisor (if applicable): H. James Hoover
Department / Faculty: Department of Computer Science
Project Title: Course browser prototype usability study

Grant / Contract Agency (and number): N/A
Application number (ASL REB member) 1955 (CLG08-12-02)
Renewal? □ **If yes, what was the previous number?** _________
Approval Expiry Date: 2009-Dec-02

CERTIFICATION of ASL REB Approval

I have reviewed your application for ethics review of your human subjects research project and conclude that your project meets the University of Alberta standards for research involving human participants (GFC Policy Section 66). On behalf of the *Arts, Science & Law Research Ethics Board* (ASL REB), I am providing expedited approval for your project.

Your application will be presented to the ASL REB board at its next meeting (December 15, 2008).

This research ethics approval is valid for one year. To request a renewal after December 2, 2009 please contact me and explain the circumstances, making reference to the research ethics review number assigned to this project. Also, if there are significant changes to the project that need to be reviewed, or if any adverse effects to human participants are encountered in your research, please contact me immediately.

ASL REB member (name & signature): Christina L. Gagné

Date: December 2, 2008

Christina Gagné