Advising Planning Program: A Usability Study

Allie Deford

Mescon Paper

March 15, 2014

Advising Planning Program: A Usability Study

One of the biggest problems a university must handle is how to track the classes each student takes on their path towards a degree. At the University of Evansville, there have been numerous attempts to create systems to adequately track this, but as it stands, there is no good tool to allow students to plan how they will complete their degrees. The job of making sure students meet all requirements falls to their advisor until their final year. This ultimately leads to some oversights, occasionally even forcing students to graduate later than they expect to. Tracking which students have completed what graduation requirements has been a problem in the College of Engineering and Computer Science for some time. Dr. Deborah Hwang presented the need for an application that solves the advising problem for students and faculty. This project will attempt to solve this problem by creating a web-based application for students and their faculty advisors that tracks the classes they have taken and allows them to check these classes against the graduation requirements for their given program. This program will focus specifically on the needs of the College of Engineering and Computer Science, with the hope that it can be adapted easily for other colleges within the university. This paper focuses on the front end usability and design on the application.

The Problem:

Graduation requirements come in three parts: general education requirements, major requirements, and electives. General education requirements are classes that the university requires every student to take. Most of these classes must be outside the student's major. This program consists of eleven Outcomes that the student must meet by taking courses that have been approved to meet that Outcome. Each Outcome requires a certain number of credit hours before it is considered to be completed. For example, Outcome 2 states that students show "Engagement with imaginative expressions of the human identity" and requires that students take three hours. The Registrar's office maintains a list of classes that fulfill each Outcome.

Additionally, each student must complete four Overlays. Overlays contain broader themes than Outcomes that span across the curriculum. Each Overlay has its own set of acceptable courses to complete it and requires a different number of hours, similar to the Outcomes. Many Overlay courses also fulfill Outcome courses, though this is not always the case. More information can be found in Appendix A.

Each major maintains its own set of requirements that a student must complete to earn a degree. These courses vary in content and quantity from major to major. Each department at the university decides what the requirements will be for the majors that it covers. Within each major there is generally a set of core required classes that must be completed by every student, as well as major-related electives, which gives each student the choice to tailor the major to their interests. To view the Computer Science degree requirements, please see Appendix B.

The last part of the degree requirements is free electives. Different majors have different numbers of free electives that must be taken. Free electives are defined as classes that fall outside of general education classes and major-related classes. Some programs require students to take some portion of their free electives at the 300/400 level.

Currently, each student is in charge of keeping track of completed classes by hand. Each student maintains a checklist containing all the classes they must complete in order to earn their desired degree. They fill out this sheet as the years progress, making sure that they earn all the credits they need. Because this was tricky, the advisors suggested a digital planning tool, which would allow students to map out what classes they want to take over their four years. This also could handle the problem of handling different sets of requirements for each student. If a student is pursuing two majors, or even a minor, under the current system, they must keep track of multiple sheets, cross-checking those papers to make sure they have completed all requirements.

Requirements:

To make sure the developer properly understood the project, she met with several faculty and staff members formally to talk about what they wanted in an application like this. She also had informal conversations with students about their needs in an application like this. She found that the same issues kept coming up in conversation. Those themes were refined into the requirements that define the project.

Broadly, the application needs to be easily accessed from a variety of platforms, to allow for the greatest number of users. Additionally, a number of different kinds of users will need to be able to use the application to complete different kinds of tasks. The application will allow for three different kinds of users, each with different abilities.

Students are the first group of users to be considered. They should be able to view each year of their education and the classes they have, are, or are planning to take in a readable format. The application also should allow students to plan their future semesters by moving classes around to explore different possible plans. Additionally, they should be able to declare one or more majors and zero or more minors. These majors and minors will determine their compiled list of classes, which they should be able to easily view. When planning the upcoming semester, students should be able to view a list of all available classes. Those classes in the upcoming semester that meet requirements the student has yet to fulfill will be noted.

Faculty members are the second set of core users. Faculty users have different needs than students, so they should be able to perform different tasks. The main ability of the faculty will be to view information about their advisees. They should be able to view the classes each of their advisees has taken and is planning to take so that they can make sure their students are taking the classes they need to in order to graduate on time. Next, faculty advisors should have the ability to set the catalog of their advisees. Initially, students will be assigned the default catalog of their freshman year, but occasionally, changes have to be made based on the changing circumstances of some students. Along those same lines, advisors should be able to

grant a variety of exceptions to their students, including allowing them to bypass certain prerequisites or major classes. This allows the system to meet the needs of all students, even the ones who have special circumstances. Lastly, faculty should be able to pick one of their classes and have the application check to see if every student in the class has completed the prerequisites for that class.

The final group of users is the administrator group. These users are the ones who will control the system itself. They should have the ability to add majors and minors to the system, as well as modifying already existing majors and minors. Administrators also should have the ability to add new requirements to the system, including general education requirements and other university-ordained requirements. Finally, administrators should have the ability to modify some aspects of currently existing accounts, as may be required from time to time.

Design:

Broadly speaking, the visual look of the site will be the same for all users, but it will be populated with different content based on the user's personal data. Each task will be contained in a page. Students will have pages for viewing and editing their current proposed schedule, viewing their major and minor checklists, searching for classes, and changing personal options, such as major. Faculty users will have pages for searching for and viewing an advisee's schedule, viewing rosters for current classes being taught, and adding exceptions for a given student. Admins will have pages for adding new majors and minors, modifying existing data, and other similar pages.

The exact layout and function of the pages will be determined through scenario-based user-interface design. Initially, the developer will create some hypothetical stakeholders for the project. These stakeholders will be fictional future users of the system. From this, the developer will write design scenarios using the stakeholders, which describe a single action the user will complete using the system. For example, one of the hypothetical stakeholders could be Mindy

Right. Mindy is a sophomore computer science major. She is trying to figure out if she has time to take a minor and still graduate on time. She needs help laying out her classes to see if she has enough time. In one design scenario, Mindy wants to add a class to her proposed schedule for her junior year. She should be able to select a class from a drop-down and add it to her schedule in the right block.

These design scenarios will influence the design of two prototype pages. These prototypes will cover important features, as well as the overall design on the site. The prototypes will not hook into the backend, but instead will contain static data for testing purposes. For example, Figure 3 contains some mockups of layouts for the overall site design that will be tested. An example of the splash screen of a completed prototype can be seen in Figure 4. The two prototypes will be fully completed sites, though they will share some common features. Each prototype will have a different overall design, as well as different functionality on some of the pages.

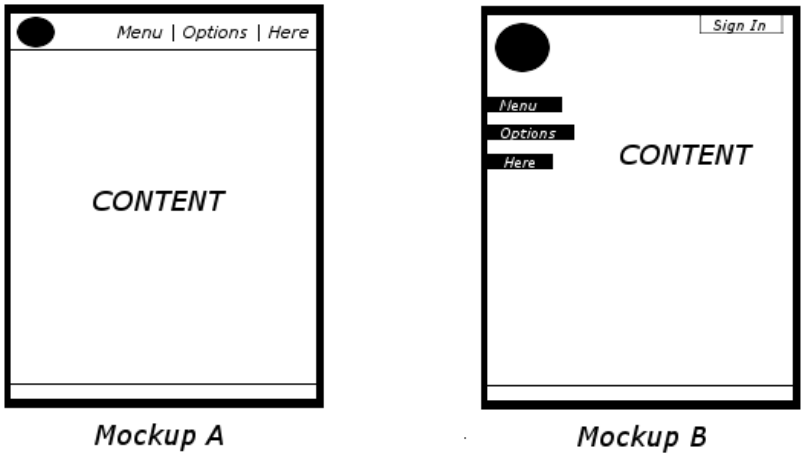

Figure 3: Mock layout prototypes

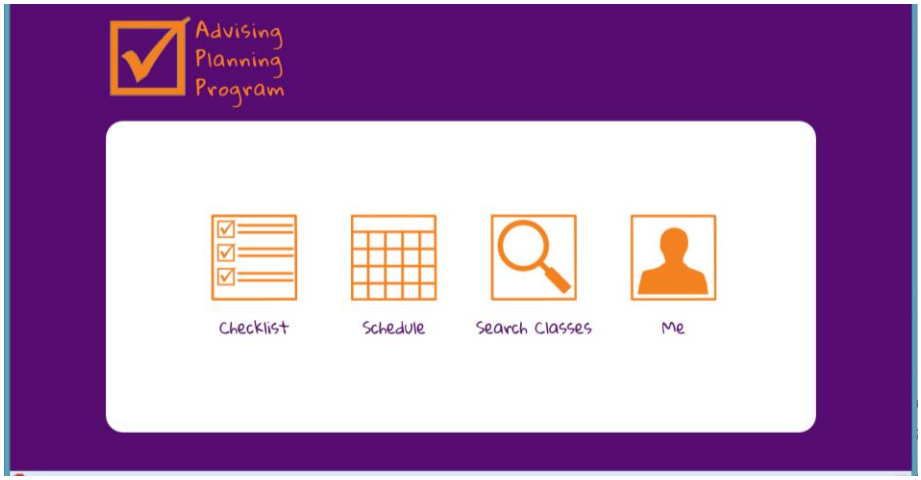

Figure 4: Home page of first completed prototype

Once the prototypes have been completed, the developer will complete a usability test on the system. The developer chose to use an empirical test, which observes real potential users of the system instead of an analytical test, which simply simulates the actions of a future user. This was chosen because, although it is more complicated to complete, it gives a better understanding of the usability of the system [1]. Specifically, the developer has chosen to use a laboratory usability test because it can be completed on a prototype, whereas a long-term field study, where the developer would spend several days watching the normal work flow of users, requires a completed program. The test will use within-subject design, which means that each participants will be exposed to all of the independent variables, instead of each user being exposed to only a single independent variable. This is to say, all subjects will be doing tests on both prototypes and following the same procedure, as opposed to each subject only looking at a single prototype [1]. During the test, both numerical data, in terms of number of errors and time taken to complete tasks, as well as subjective user thoughts will be gathered. These kinds of studies are common in the field of Human-Computer Interaction.

Future users of the system, including students and faculty, will be recruited for this study. Ideally, the participants will come from all majors and grades within the College of Engineering

and Computer Science. Testers will enter the Computer Science Project Lab 1 at a time. After reading a consent waiver, subjects will be asked to sit at one of the computers that has been prepared by the developer for the test. Subjects will be randomly assigned to begin with either Interface A or Interface B. This is to prevent familiarity with the questions and the system as being a deciding factor in which interface appears the most usable. The test subjects will be asked to complete a certain set of tasks, as seen in Appendix C, using each of the prototypes while the developer observes their actions. The developer will note what tasks they seem to complete easily, and which ones the subjects struggle with, specifically noting the number of errors that occur that occur while completing each task and noting the total amount of time it takes to complete each task. After completing the tasks using Interface A, the subjects will be asked to switch to Interface B and complete the same tasks. When they are finished, the subjects will be given a survey where they can voice any overall thoughts they have about each prototype. The survey can also be found in Appendix C. The developer will then analyze the results of the testing by comparing the observed number of errors and total time to complete each to a base decided on by the developer as the results for that task. The developer will then decide on the final design based on all the given input. The most usable features of each prototype will be combined and turned into a single, working application.

Conclusion:

Those testing has not been completed yet, the developer hopes to have preliminary results to present at during the presentation, pending approval from the Institutional Research Board. The developer expects the usability testing to be helpful in creating an application that will be used by future students and faculty in the College of Engineering and Computer Science.

Appendix A:

University of Evansville General Education Requirements 2013-2015 [2]

Outcomes

Outcome 1: Critical reading and thinking (FYS)- 3 hours

Outcome 2: Engagement with imaginative expressions of the human condition- 3 hours

Outcome 3: Knowledge of human history and the historical context of knowledge- 3 hours

Outcome 4: Engagement with fundamental beliefs about human identity, core values, and humankind's place in the world- 3 hours

Outcome 5: Understanding of human aesthetic creation and artistic creativity- 3 hours

Outcome 6: Linguistic and cultural competence in a language other than one's own- 6 hours

Outcome 7: Quantitative literacy- 3 hours

Outcome 8: Scientific literacy- 7 hours

Outcome 9: Understanding of core concepts of society, human behavior, and civic knowledge- 6 hours

Outcome 10: Knowledge and responsibility in relation to health and wellness- 1 hour

Outcome 11: (Capstone) Ability to think critically and communicate effectively, orally and in writing- 3 hours

Overlays

Global Perspective: International Diversity (two course-equivalents)

Global Perspective: Domestic Diversity (one course-equivalent)

Social Responsibility (one course-equivalent)

Writing-Intensive (four course-equivalents)

Appendix B: Major Requirements

Below is a sample of the requirements for computer science.

Enduring Foundations General Education Requirements			
(43 hrs)	*Hrs*	*Grade Overlay*	
Outcome 1: Critical Reading and Thinking (3 hrs)			
FYS 112: First Year Seminar	3		E
Outcome 2: Engagement with Imaginative Expressions of the Human Condition (3 hrs)			
	3		
Outcome 3: Knowledge of Human History and the Historical Context of Knowledge (3 hrs)			
	3		
Outcome 4: Engagement with Fundamental Beliefs about Human Identity, Core Values, and Humankind's Place in the World (3 hrs)			
	3		
Outcome 5: Understanding of Human Aesthetic Creation and Artistic Creativity (3 hrs)			
	3		
Outcome 6: Linguistic and Cultural Competence in a Language Other than One's Own (6 hrs)			
	3		
	3		
Outcome 7: Quantitative Literacy (3 hrs)			
Math 221 – Calculus I	4		
Outcome 8: Scientific Literacy (8 hrs)			
(Lab) BIOL 107 or CHEM 118	4		
PHYS 210 – Calculus Physics I	4		
Outcome 9: Understanding of Core Concepts of Society, Human Behavior, and Civic Knowledge (6 hrs)			
	3		
	3		
Outcome 10: Knowledge and Responsibility in Relation to Health & Wellness (1 hr)			
	1		
Outcome 11: Ability to Think Critically and Communicate Effectively, Orally and in Writing (3 hrs)			
CS 495 - Senior Project Phase I	3		E
		TOTAL CREDITS	
TECHNICAL ELECTIVES (12 hrs minimum)			
Students must choose from CS 350, 355, 375, 376, 415, 430, 440, 475, 478, 480, 499; EE 310, 311, 354, 454, 456.			
	3		
	3		
	3		
	3		
		TOTAL CREDITS	

BASIC LEVEL REQUIREMENTS (33 hrs)	*Hrs*	*Grade*	*Overlay*
CS 101 – Intro to Computer Science	3		
CS 210 – Fund'ls of Programming I	3		
CS 215 – Fund'ls of Programming II	3		
CS 220 – Logic Design & Machine Org	3		
CS 290 – Object-Oriented Design	3		
ENGR 390 or MATH 365 or MATH 341	3		
MATH 222 – Calculus II	4		
MATH 323 – Calculus III	4		
MATH 370 – Discrete & Combinatorial Mathematics	3		
Computer Science majors must complete a two-semester sequence in biology, chemistry, or physics.			
BIOL 109; CHEM 240, 280; or PHYS 211	4		
	TOTAL CREDITS		

UPPER LEVEL REQUIREMENTS (21 hrs)	*Hrs*	*Grade*	*Overlay*
CS 315 – Algorithms & Data Structures	3		
CS 320 – Computer Architecture	3		
CS 380 – Programming Languages	3		
CS 381 – Formal Languages	3		
CS 390 – Software Engineering	3		
CS 470 – Operating Systems	3		
CS 494 – Senior Project Seminar	0		
CS 497 – Senior Project Phase II	3		E
		TOTAL CREDITS	
PROFESSIONAL DEVELOPMENT ELECTIVE - ***Computer Science majors must choose one course from ECON 101; COMM 210, 382, 485; PHIL 111, 121, 231,241, 316, 317; WRTG 330 when the topic is technical writing. This course may not be used to fulfill a general education course. (3 hrs)***			
	3		
		TOTAL CREDITS	
FREE ELECTIVES - ***At least 9 hours must be 300 level or higher. It is recommended that Computer Science majors use their free electives to minor in a field of application. Courses numbered MATH 222 or lower, CHEM 10x, CS 210 or lower, PHYS 100 level, SA courses, IT 120, and English language courses may not be used as free electives.*** *(18 hrs)*			
300/400 level			
300/400 level			
300/400 level			
		TOTAL CREDITS	

Appendix C: Usability Study

Directions:

1. Bring up the web browser and make sure the web browser is focused on the first tab.

2. Log in using the following username and password:

3. Imagine you are a computer science major trying to plan your course load for Fall 2014, which will be the fall of your junior year. First, view the checklist to determine which of the following class you still need to complete: CS290, CS315, CS350, CS390, CS380, CS381, MATH323, MATH324

4. Navigate to the schedule page and add the classes you have not yet completed to your Fall 2014 schedule.

5. After talking to some classmates, you decide that your schedule is too full for the upcoming semester. You decide to take CS315 in the spring of your junior year instead. Remove it from your current schedule and schedule it for the spring of your junior year.

6. Add a class from Outcome 7 to your schedule for the spring semester of your junior year.

7. You hear some of your friends talking about a class they call PSYC121. Find the title and description for this class number.

8. Upon viewing your schedule, you discover that you need to take 3 more technical electives. Search for classes that would fulfill the CS technical elective requirement. Write the course number for 3 of those courses below:

9. Let's say you decide that you want to add a psychology minor to your computer science degree. Using the Me page, as this minor to your requirements.

10. Can you determine how many classes you still need to take to complete this minor? If so, write the answer here:

11. View the Checklist page. How Outcomes must you still fulfill before graduation?

12. You have a question about how to officially add your new psychology minor. Find your advisor's email address and write it below so that you can send them an email and ask:

13. Repeat steps 3-12 in the second tab, which contains an alternate user interface. Fill in your answers to the questions using the blanks labeled Interface B.

14. When you have completed these steps, please turn the page and complete the post survey.

Please answer the following questions (separate form for each interface option). All questions will use the same scale printed in question 1.

1. I found this interface easy to use.

1	*2*	*3*	*4*	*5*	*6*	*7*
Strongly Disagree	*Disagree*	*Moderately Disagree*	*Neutral*	*Moderately Agree*	*Agree*	*Strongly Agree*

2. It was clear where I needed to click to complete each task.

3. I thought the Checklist page was easy to use.

   Additional Comments on the Checklist page:

4. I thought the Schedule page was easy to use.

   Additional Comments on the Schedule page:

5. I thought the Search Classes page was easy to use.

   Additional Comments on the Search Classes page:

6. I thought the About Me page was easy to use.

   Additional Comments on the About Me page:

With regards to both interfaces,

1. Which interface was easier to use? *Interface A* *Interface B*
2. Which Checklist page was easier to use? *Interface A* *Interface B*
3. Which Schedule page was easier to use? *Interface A* *Interface B*
4. Which Search Classes page was easier to use? *Interface A* *Interface B*
5. Which About Me page was easier to use? *Interface A* *Interface B*
6. Which overall site design was easier to use? *Interface A* *Interface B*
7. I am likely to use an application such as this one when planning my future classes.
8. Please include any overall comments you might have about either design:

References

[2] University of Evansville. (2013). *2013-2015 Undergraduate and Graduate Catalog*. Retrieved from http://www.evansville.edu/registrar/downloads/CourseCatalog2013-2015.pdf

[1] Rosson, Mary Beth., and John M. Carroll. Usability Engineering: Scenario-based Development of Human-computer Interaction. San Fancisco: Academic, 2002. Print.