ICSI 680 : Master's Project

Project Description – Fall 2023

SEMESTER PROJECT DESCRIPTION

Your semester-long project is to specify, design, build, and deploy a software project. In keeping with my teaching philosophy of divergent thinking and creative problem solving, you have numerous choices in your project and will have full responsibility to select features; to design the UI, UX, and the software internals; and to choose implementation technologies.

Many class projects give more detailed and strict requirements than in this class. That is not the way the real world works! For this project, you will have the opportunity to develop your creative thinking skills, to learn how to refine and resolve open-ended requirements, and take responsibility for the design of the project, not just the implementation. This may be uncomfortable at first, but these are valuable abilities for professional software engineers.

Your project is to design and develop a software test tool to support a widely used test criterion known as *input-space partitioning* (ISP). You do not need to know ISP for this project. In particular, you do not need to know the theory. I will share algorithms that you can use in your back-end software.

ISP starts with a finite (usually small) set of *characteristics*. For example, characteristics to describe students might be major, year started, status, and visa. Each characteristic is divided into a finite (usually small) set of *blocks*. For example, the characteristic **student** may have the following blocks:

- **major** = [cs, ece, ese, other]
- **year** started = [2023, 2022, 2021 or earlier]
- **status** = [part-time, full-time]
- **visa** = [US, student, other]

Software testers often use one letter abstract names for characteristics and blocks. For example, A = *major*, B = *year started*, C = *status*, and D = *visa*. That allows the blocks above to be rewritten in a compact form: A = [A1, A2, A3, A4]; B = [B1, B2, B3]; C = [C1, C2]; and D = [D1, D2, D3]. Software testers create test inputs for software by designing characteristics and blocks for the system interfaces, then *combining* blocks in one of several ways to create complete tests. With the student example, the blocks can be combined to form a maximum of 4*3*2*3 = 72 possible tests (more examples are below under **ISP criteria**).

Your assignment is to design and build software to:

1. Allow users to enter characteristics and blocks
2. Compute and return combinations based on several test criteria as described below under **ISP criteria**
3. Design the UI/UX to make the data entry as convenient as possible
4. Design the software so that the result is flexible and, in particular, convenient to turn into inputs to the software

It is completely up to you to design the user interface and the software. You may discuss ideas with your classmates or anyone else. You may also use any technologies you want and design the UI and software as you wish. You are free to make as many assumptions and decisions as you like within the parameters given. Be creative and have fun! Part of the final grade will be determined by the usability and overall functionality of your project.

Teams and project parameters

- Each team can have 3 to 5 students. You can choose your own teammates, then I will assign remaining students arbitrarily. Each student will submit a one-page evaluation of your teammates at the end of the project. These evaluations will be used as part of your grade.

- You may use any software technologies, package, and support software.
- You may deploy your system on any compute. This includes, but is not limited to, cloud services such as AWS or github-heroku, tablets and other mobile devices, university servers, or any other server.
- Your user interface may be of any style.
- You must demo your project to me at the end of the semester. All teammates must be present at the demo to receive full credit.
- You will submit all source files at the end of the semester.

Alternate project choice

If you wish to design and build a different software system, you may propose one. If it is an appropriate size and complexity (not too big and not too small), I will approve it. You may **not** propose something that is already built.

Project milestones

Milestone 1: Project definition and team composition

Milestone 2: Initial UX/UI design

Milestone 3: Software architecture & design

Milestone 4: Automated tests

Milestone 5: Final project demo

I will provide details as to specific deliverables later in the semester.

Input space partitioning criteria

The primary output of the test support tool will be the characteristics and blocks combined into specific combinations based on combination *criteria*. Your system should implement three criteria, as described here. The criteria are described with the following abstract example:

- A = [A1, A2, A3] (3 blocks)
- B = [B1, B2, B3] (3 blocks)
- C = [C1, C2] (2 blocks)

1. **All combinations (ACoC)**: Every block in every characteristic is combined with every other block. Our example needs 3*3*2 = 18 combinations for ACoC:
[A1,B1,C1]; [A1,B1,C2]; [A1,B2,C1]; [A1,B2,C2]; [A1,B3,C1]; [A1,B3,C2];
[A2,B1,C1]; [A2,B1,C2]; [A2,B2,C1]; [A2,B2,C2]; [A2,B3,C1]; [A2,B3,C2];
[A3,B1,C1]; [A3,B1,C2]; [A3,B2,C1]; [A3,B2,C2]; [A3,B3,C1]; [A3,B3,C2]

2. **Each choice (ECC)**: Every block in every characteristic appears in at least one combination. The number of combinations is equal to the number of blocks in the largest characteristic. Our example needs 3 combinations for ECC:
[A1,B1,C1]; [A2,B2,C2]; [A3,B3,*]
Note the "*" in the third combination. Since both C1 and C2 have already been used, it does not matter which C block is used, so the "*" means "any block from characteristic C," or "don't care."

3. **Base choice (BCC)**: The tester chooses one "base" block from each characteristic. Which block and why that block is chosen does not matter for this project. Base choice combinations include the base combination (all the base blocks), plus one combination for each other block, swapped in one at a time. Our example needs 7 combinations for BCC:
[A1,B1,C1] (base block);
[A1,B1,C2]; [A1,B2,C1]; [A1,B3,C1]; [A2,B1,C1]; [A3,B1,C1]