Semiotic Approaches & Software Development

STS Research Paper
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia

By

Scott Lutz

February 28, 2020

Benjamin J. Laugelli, Assistant Professor, Department of Engineering and Society

Introduction

University of Virginia's (UVA) Student Information System (SIS) is a web application that over 20,000 students primarily use to create their class schedule for the next semester ("Statistics"). A majority of students express dissatisfaction with the technology because it is very tedious and difficult to create a schedule with (Weigand, 2019). To get around this, students use workarounds developed by other students and faculty. The most popular tools are Lou's List and UVA schedule | me. Lou's List provides a well-organized catalog of all the courses offered (Bloomfield). UVA schedule | me allows users to build a schedule with real time visual aid ("uva schedule | me"). Some scholars would argue that a technology like SIS needed to sacrifice functionality to enhance its usability in order to create a better experience for its users (Goodwin, 1987). Other scholars argue that SIS is poorly designed due to biased decisions that software developers make during development (Tang, 2011). Both of these arguments fail to take into account how the ideas about users embedded into SIS by the designers constrain what the users can do and how these constraints have caused users to adapt and use workarounds to enhance their experience with the software. By exploring how the constraints embedded into SIS have caused users to have poor experiences and create their own solutions software developers stand to gain a better understanding of the way misconfiguring a technology to its user can negatively effect user experiences. Drawing on user configuration, I argue that the designers misunderstanding of the user's identity unintentionally embedded constraints into SIS which caused users to adopt the use of third-party tools when attempting to create a schedule. Software engineers stand to gain a better understanding of the importance of having an accurate image of the user's identity when attempting to develop successful software. I will use user configuration to explore SIS as a technological script in order to show how it does not match the script of a typical user and how this has caused users to express user agency and use workarounds rather than use the technology directly.

Literature Review

While several scholars have examined how the balance of functionality and usability and the biases of software designers' affect software development, scholars have not yet adequality considered how software designers may have a flawed understanding of their user's identity which ends up being built into the core of their software.

Goodwin argues that good software does not sacrifice usability for functionality. A feature filled software application will be less useful to its users if they are unable to use it easily (Goodwin, 1987). Particularly if they have to memorize a lot of commands, many users tend to learn only a few that help them accomplish the same goal but may not be optimal to use. Goodwin focuses on how increasing usability requires matching functions provided by the software and task requirements to help lower task completion rates and the amount of user errors (Goodwin, 1987). If the system is easier to use, then more users will want to use the software; and Goodwin argues that this in itself is improving the functionality.

Tang explains that software designers have cognitive bias, illogical reasoning, and low-quality premises. Cognitive bias is, "a distortion of judgement in particular situations due to psychological effects and insufficient regards of probability" (Tang, 2011). An example of this is that software designers may pick an inappropriate design solution that does not address the problem well because they are more familiar with it. Illogical reasoning happens primarily when designers do not consider whether the premises of their argument are true or if the conclusion is reasonable. Low-quality premises occur when designers have inaccurate premises about something and this causes them to make incorrect decisions. All of these issues contribute to poor software design reasoning and lead to poor decision making. Tang argues that software solutions turn out poorly because individual designers can make bad design decisions due to these biases and decision making is a key element in designing quality software (Tang, 2011).

Both of these arguments consider different reasons that software might turn out poorly. Goodwin believes there is too much focus on providing more functionality while compromising usability and Tang believes that software designers' poor decision-making due to their inherent biases will lead to poor software development. Both of them fail to consider who the designer is designing the software for. They never discuss how the designer's solutions may be poor because they are not designing with the user in mind or that their idea of the user's identity may be flawed.

In my analysis, I will focus on addressing how the designers' inadequate understanding of the user can lead to a mismatch between the user and the technology. If the technology does not match the user, then the user will resist the technology or be forced to adapt the technology. If the user resists the technology it will be more difficult for them to be productive or enjoy using it. If the user adapts the technology it will create more work for the users and show that the technology was not adequately addressing the user's problem.

Conceptual Framework

My analysis of UVA's web application SIS draws on user configuration, which I will use to show how the designers' flawed understanding of the user's identity was built into its core and the negative effect it has had on the user's relationship with the technology.

I will use the concept of user configuration to analyze the case of SIS as a technological script. User configuration examines ways designers embed ideas about users into technologies, which then function as scripts that direct what users can or cannot do (Akrich, 1992; Woolgar, 1991). The script, then can be described as the workflow forced upon the user by design. The user follows the script in order to use the software; however, in the case of SIS the users' script does not match the technological script. The scripts do not match because the designer has incorrect ideas about the user that get embedded into the technology (Lindsay, 2003). This conflict causes the user to resist and create workarounds. I will also draw on the concept of user agency, which analyzes how users take action to modify existing technology to fit their needs. The concept of antiprogram, the extent that designers' and users' scripts conflict, will allow me to analyze how the tools created by users, such as UVA schedule | me, were an expression of user agency to address major issues in the design of SIS (Oudshoorn & Pinch, 2003, p.7-11). In this case, the users have created tools such as Lou's List and UVA schedule | me to help them use SIS more efficiently.

In what follows I examine the case of UVA's SIS through the context of user configuration by first examining the technological script and the user's script to see where they do not match. Then I will incorporate the concept of antiprogram and user agency to strengthen my argument that there is a mismatch between the designers' ideas about users built into SIS and the user's true identity.

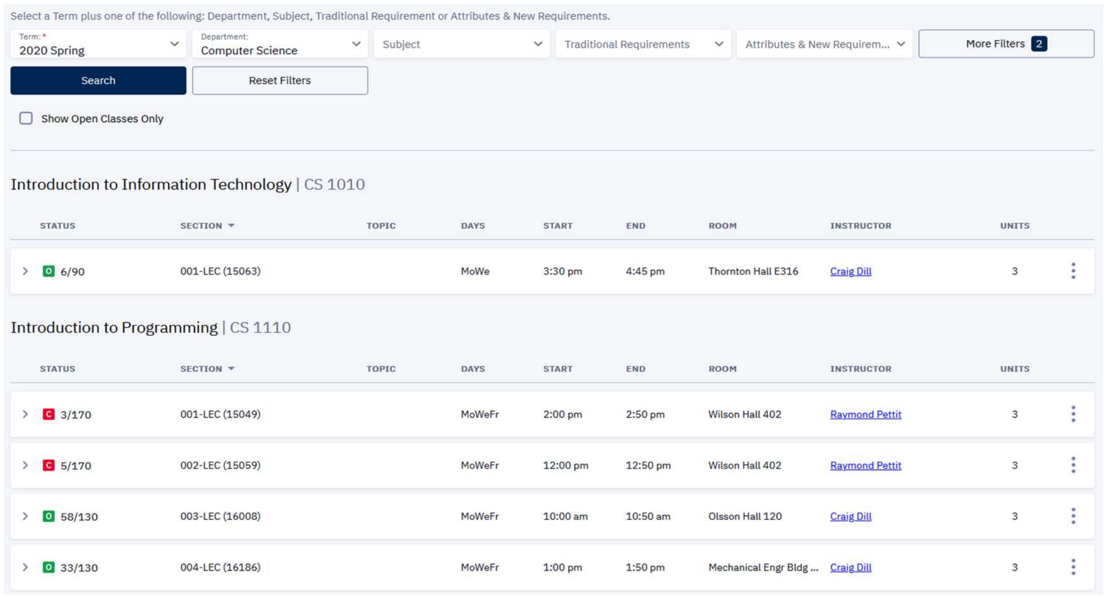
Analysis

The designers of UVA SIS failed to create software that is helpful and enjoyable for students to use. Through the lens of user configuration, I will demonstrate that the designers of SIS misconfigured the technology to their users by showing the poor workflow SIS constrains users to follow and how users have resisted using SIS by creating and using many workarounds.

Step-by-Step Workflow of SIS

Students primarily use SIS in order to plan their curriculum and create a schedule for the next semester. I will next present a plausible process of picking out classes and creating a schedule on SIS in order to highlight how the technological script of SIS is extremely tedious and frustrating and does not match the script of the users.

1) A user opens up a list of their remaining course requirements (See Appendix A).
    a. In the box you can see an example of a requirement that has not yet been fulfilled and a list of possible courses to choose from that would fulfill that requirement.
2) The user needs to remember which courses they need to take or they can write them down somewhere.
    a. Using SIS alone, it is not possible to view the list of required courses at the same time of selecting them. Due to how the web application is implemented, a user can only view one page at a time. Trying to open a new page in a different tab will break the system.


Select a Term plus one of the following: Department, Subject, Traditional Requirement or Attributes & New Requirements.

Term: * 2020 Spring | Department: Computer Science | Subject | Traditional Requirements | Attributes & New Requirem... | More Filters 2

Search | Reset Filters

☐ Show Open Classes Only

Introduction to Information Technology | CS 1010

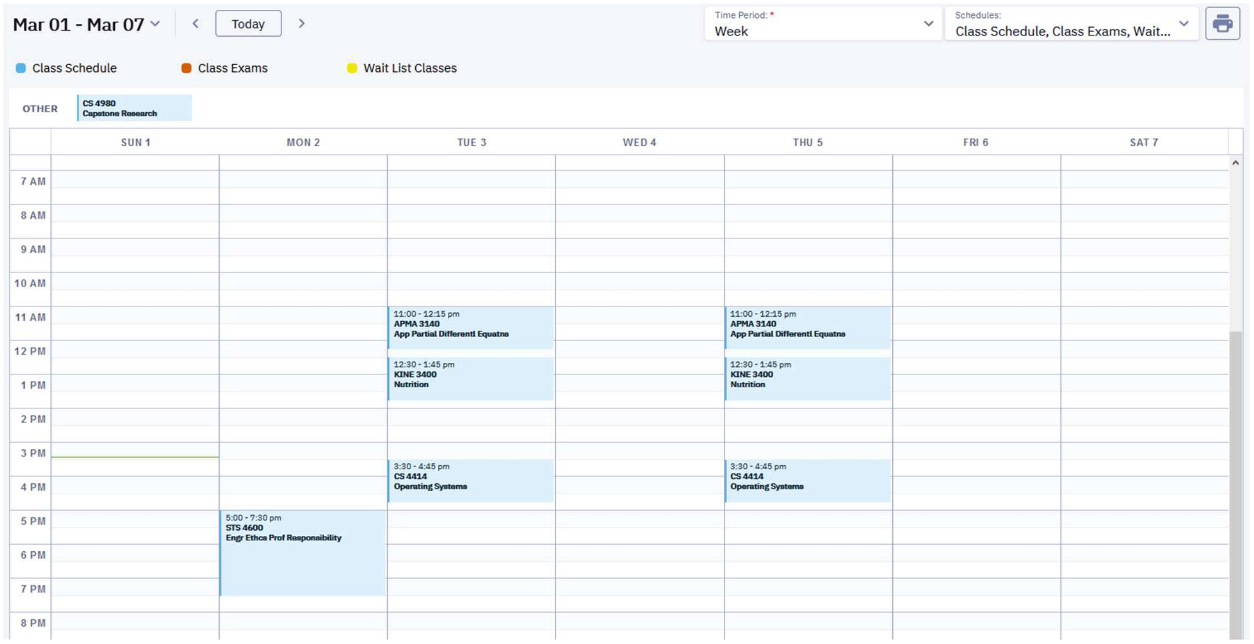
STATUS	SECTION	TOPIC	DAYS	START	END	ROOM	INSTRUCTOR	UNITS
O 6/90	001-LEC (15063)		MoWe	3:30 pm	4:45 pm	Thornton Hall E316	Craig Dill	3

Introduction to Programming | CS 1110

STATUS	SECTION	TOPIC	DAYS	START	END	ROOM	INSTRUCTOR	UNITS
C 3/170	001-LEC (15049)		MoWeFr	2:00 pm	2:50 pm	Wilson Hall 402	Raymond Pettit	3
C 5/170	002-LEC (15059)		MoWeFr	12:00 pm	12:50 pm	Wilson Hall 402	Raymond Pettit	3
O 58/130	003-LEC (16008)		MoWeFr	10:00 am	10:50 am	Olsson Hall 120	Craig Dill	3
O 33/130	004-LEC (16186)		MoWeFr	1:00 pm	1:50 pm	Mechanical Engr Bldg ...	Craig Dill	3

Figure 1 - A list of computer science courses on SIS ("Student information system")

3) The user navigates to a list of courses (Figure 1), chooses one, and adds it to his or her cart. The user cannot view multiple versions of this page with different filters at the same time. Filters narrow down which courses are shown in the list from the full list.



CS 4980
Capstone Research

5:00 - 7:30 pm
STS 4600
Engr Ethcs Prof Responsibility

11:00 - 12:15 pm
APMA 3140
App Partial Differentl Equatns

12:30 - 1:45 pm
KINE 3400
Nutrition

3:30 - 4:45 pm
CS 4414
Operating Systems

11:00 - 12:15 pm
APMA 3140
App Partial Differentl Equatns

12:30 - 1:45 pm
KINE 3400
Nutrition

3:30 - 4:45 pm
CS 4414
Operating Systems

Figure 2 - A visual representation of a student’s schedule on SIS ("Student information system")

4) The user can navigate to a visual representation (Figure 2) of their schedule so they can see how the time slots for different classes line up and if there are any conflicts. This page cannot be viewed at the same time a student is viewing a list of all the courses.

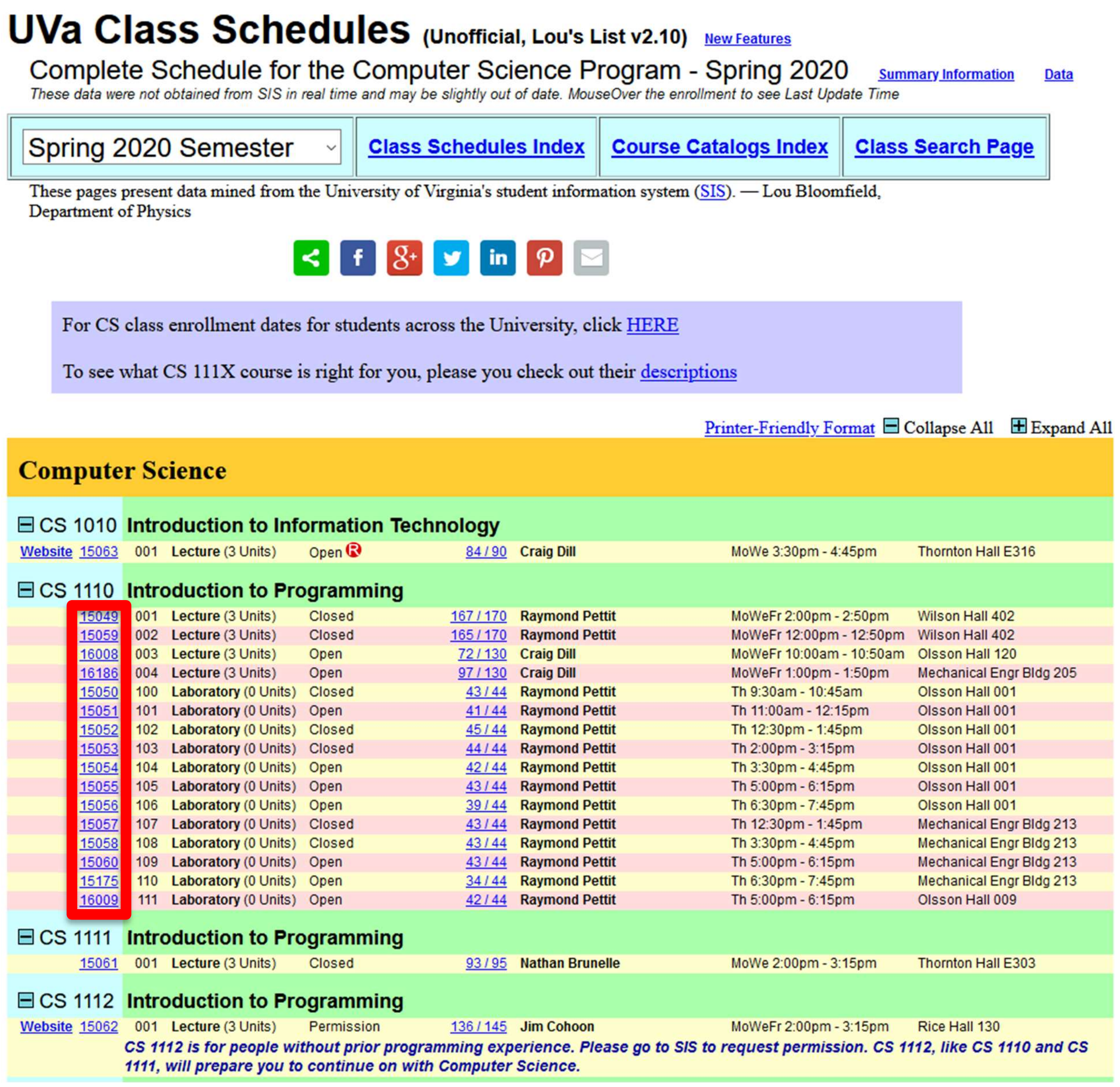
5) The user continues to go back and forth between selecting different classes (Figure 1) and looking at the visual representation (Figure 2) until they have built a schedule they are satisfied with.

The most important thing to notice on all of these pages is separation of information. Appendix A, Figure 1, and Figure 2 can never be viewed at the same time. The technological script of SIS for schedule creation can be broken down as follows. A student looks at their requirements, finds those courses on a list and adds them to a schedule and views that schedule to make sure there are no time conflicts. This process is then repeated until a complete schedule is built or a user decides to write down some of this information so that they can remember it. From my own experience and the experiences described to me by many fellow students the users' script is completely different. Users want to see a list of their requirements, a list of offered courses, and a visual representation of their schedule at the same time in order to greatly reduce the amount of information that needs to be remembered or written down. Students are forced to remember a lot of information or write it down as they navigate back and forth between different pages, rather than having all of the information they need provided to them at once. Schedule creation requires visual aid so that students can have any idea of what classes can fit together and what their daily schedule will be like. This highlights how the designers of SIS had a flawed understanding of the user's identity and how that has been embedded into the technological script. It indicates that they overestimated the abilities of the user's memory and visualization skills by forcing that work onto the user rather than the software. It is important to notice that all of the information a student would need to create a schedule are present, but it is presented to the user poorly. The designers had a good understanding of the requirements of the software and created something that was fully functional; however, the workflow it creates does not match the user's desires and therefore has led many to resist the technological script of SIS by expressing user agency and utilizing work arounds.

Step-by-Step Workflow with Workarounds

Many users have been forced to express user agency and create workarounds due to antiprogram, the conflict between the technologies and the user's scripts. I will demonstrate how users have resisted SIS by using these workarounds and examining how they greatly enhance the experience of the users. It is important to note that some users decide to use these tools together while others may only use one, but for this analysis I will go through the workflow of using Lou's List and UVA schedule | me together because they are complementary. Lou's List has better search functionality and UVA schedule | me allows students to visualize their schedule.

1) A user goes to SIS and opens up a list of their required courses (Appendix A).



UVa Class Schedules (Unofficial, Lou's List v2.10) New Features

Complete Schedule for the Computer Science Program - Spring 2020 Summary Information Data

These data were not obtained from SIS in real time and may be slightly out of date. MouseOver the enrollment to see Last Update Time

Spring 2020 Semester | Class Schedules Index | Course Catalogs Index | Class Search Page

These pages present data mined from the University of Virginia's student information system (SIS). — Lou Bloomfield, Department of Physics

For CS class enrollment dates for students across the University, click HERE

To see what CS 111X course is right for you, please you check out their descriptions

Printer-Friendly Format ⊟ Collapse All ⊞ Expand All

Computer Science

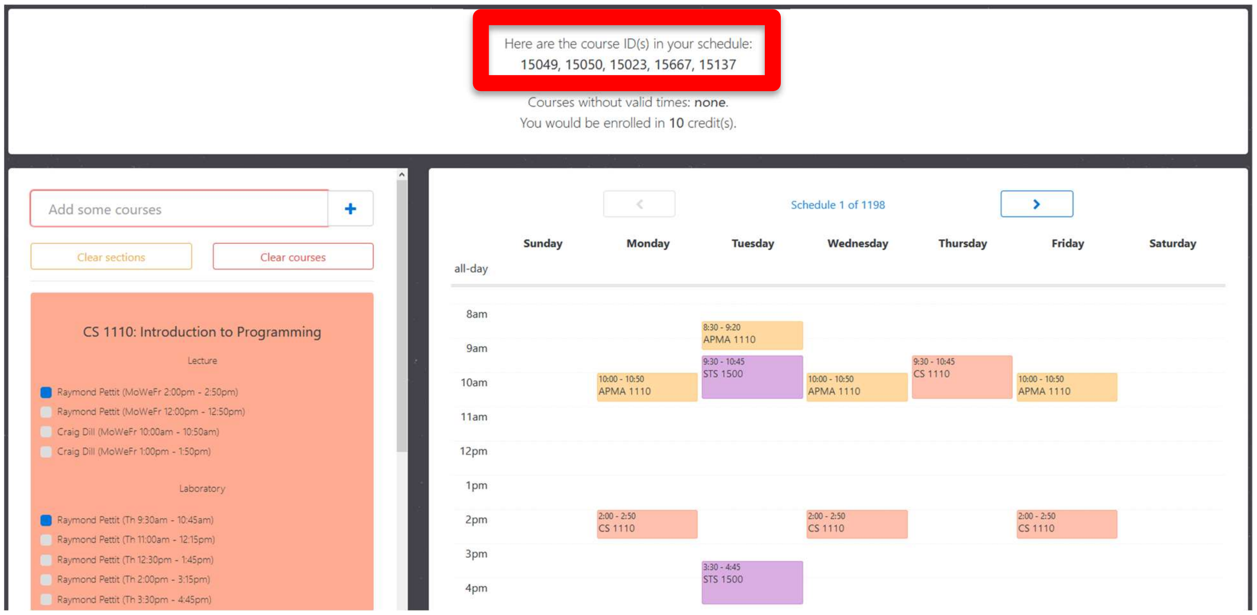
⊟ CS 1010	**Introduction to Information Technology**							
Website	15063	001	**Lecture** (3 Units)	Open	84 / 90	**Craig Dill**	MoWe 3:30pm - 4:45pm	Thornton Hall E316
⊟ CS 1110	**Introduction to Programming**							
	15049	001	**Lecture** (3 Units)	Closed	167 / 170	**Raymond Pettit**	MoWeFr 2:00pm - 2:50pm	Wilson Hall 402
	15059	002	**Lecture** (3 Units)	Closed	165 / 170	**Raymond Pettit**	MoWeFr 12:00pm - 12:50pm	Wilson Hall 402
	16008	003	**Lecture** (3 Units)	Open	72 / 130	**Craig Dill**	MoWeFr 10:00am - 10:50am	Olsson Hall 120
	16186	004	**Lecture** (3 Units)	Open	97 / 130	**Craig Dill**	MoWeFr 1:00pm - 1:50pm	Mechanical Engr Bldg 205
	15050	100	**Laboratory** (0 Units)	Closed	43 / 44	**Raymond Pettit**	Th 9:30am - 10:45am	Olsson Hall 001
	15051	101	**Laboratory** (0 Units)	Open	41 / 44	**Raymond Pettit**	Th 11:00am - 12:15pm	Olsson Hall 001
	15052	102	**Laboratory** (0 Units)	Closed	45 / 44	**Raymond Pettit**	Th 12:30pm - 1:45pm	Olsson Hall 001
	15053	103	**Laboratory** (0 Units)	Closed	44 / 44	**Raymond Pettit**	Th 2:00pm - 3:15pm	Olsson Hall 001
	15054	104	**Laboratory** (0 Units)	Open	42 / 44	**Raymond Pettit**	Th 3:30pm - 4:45pm	Olsson Hall 001
	15055	105	**Laboratory** (0 Units)	Open	43 / 44	**Raymond Pettit**	Th 5:00pm - 6:15pm	Olsson Hall 001
	15056	106	**Laboratory** (0 Units)	Open	39 / 44	**Raymond Pettit**	Th 6:30pm - 7:45pm	Olsson Hall 001
	15057	107	**Laboratory** (0 Units)	Closed	43 / 44	**Raymond Pettit**	Th 12:30pm - 1:45pm	Mechanical Engr Bldg 213
	15058	108	**Laboratory** (0 Units)	Closed	43 / 44	**Raymond Pettit**	Th 3:30pm - 4:45pm	Mechanical Engr Bldg 213
	15060	109	**Laboratory** (0 Units)	Open	43 / 44	**Raymond Pettit**	Th 5:00pm - 6:15pm	Mechanical Engr Bldg 213
	15175	110	**Laboratory** (0 Units)	Open	34 / 44	**Raymond Pettit**	Th 6:30pm - 7:45pm	Mechanical Engr Bldg 213
	16009	111	**Laboratory** (0 Units)	Open	42 / 44	**Raymond Pettit**	Th 5:00pm - 6:15pm	Olsson Hall 009
⊟ CS 1111	**Introduction to Programming**							
	15061	001	**Lecture** (3 Units)	Closed	93 / 95	**Nathan Brunelle**	MoWe 2:00pm - 3:15pm	Thornton Hall E303
⊟ CS 1112	**Introduction to Programming**							
Website	15062	001	**Lecture** (3 Units)	Permission	136 / 145	**Jim Cohoon**	MoWeFr 2:00pm - 3:15pm	Rice Hall 130

CS 1112 is for people without prior programming experience. Please go to SIS to request permission. CS 1112, like CS 1110 and CS 1111, will prepare you to continue on with Computer Science.

Figure 3 - A list of computer science courses on Lou's List (Bloomfield)

2) In a new tab a user navigates to Lou's List (Figure 3) where they can easily search for any course and see what time it is offered. The list of courses in Figure 3 is exactly the same as the list of courses in Figure 2, but the advantage is that a user can have as many tabs of Lou's List open as they want and they can view Lou's List at the same time they are viewing a list of requirements.

3) After a student has decided on which courses they wish to take, they navigate to UVA schedule | me.


Here are the course ID(s) in your schedule:
15049, 15050, 15023, 15667, 15137

Courses without valid times: none.
You would be enrolled in 10 credit(s).

CS 1110: Introduction to Programming

Lecture

- [✓] Raymond Pettit (MoWeFr 2:00pm - 2:50pm)
- [ ] Raymond Pettit (MoWeFr 12:00pm - 12:50pm)
- [ ] Craig Dill (MoWeFr 10:00am - 10:50am)
- [ ] Craig Dill (MoWeFr 1:00pm - 1:50pm)

Laboratory

- [✓] Raymond Pettit (Th 9:30am - 10:45am)
- [ ] Raymond Pettit (Th 11:00am - 12:15pm)
- [ ] Raymond Pettit (Th 12:30pm - 1:45pm)
- [ ] Raymond Pettit (Th 2:00pm - 3:15pm)
- [ ] Raymond Pettit (Th 3:30pm - 4:45pm)

Figure 4 - A visual representation of a student's schedule on UVA schedule | me ("uva schedule | me")

4) On UVA schedule | me (Figure 4) they can add all of the courses they wish to take to their schedule. This is accomplished by searching for the course in the search bar that says "Add some courses" and clicking on the "+" button.

5) From here, students can check boxes to decide which time slot they want to take each course at, on the bottom left of Figure 4.

6) UVA schedule | me can automatically create valid schedules based on the courses chosen and the user can navigate through the options or the user can select their own time slots freely and see the visual representation of their schedule update dynamically in real time to reflect what they have chosen, on the right of Figure 4.
7) After a final schedule has been decided on, UVA schedule | me provides course IDs, highlighted in a box near the top of Figure 4, that can be easily entered into SIS in order to add the courses a student has chosen to his or her schedule.

According to Google Trends related queries, a large number of students at UVA tend to use these tools when it is time to create a new schedule ("Google trends: Lou's list"). It is important to mention that Lou's List was created by a professor at the university in order to help students search for classes more easily. UVA schedule | me was created by students in order to help other students create visual representation of their schedules dynamically without have to manually create an excel spreadsheet in order to see it.

Both of these tools are an expression of user agency and highlight how users are resisting the technological script of SIS because it does not match their script. Lou's List and UVA schedule | me help students have a less frustrating experience creating a schedule by reducing the amount of time they have to spend navigating back and forth between a list of courses and a visual representation of their schedule. The users' script wanted to have all of the information that they needed at once and these workarounds allow for that to be possible by having multiple webpages open at once. The creation and widespread use of these tools illustrates how the designers' flawed understanding of the user's identity that was built into the core of SIS has constrained the user's abilities and has rendered the technology to be less useful for its intended user.

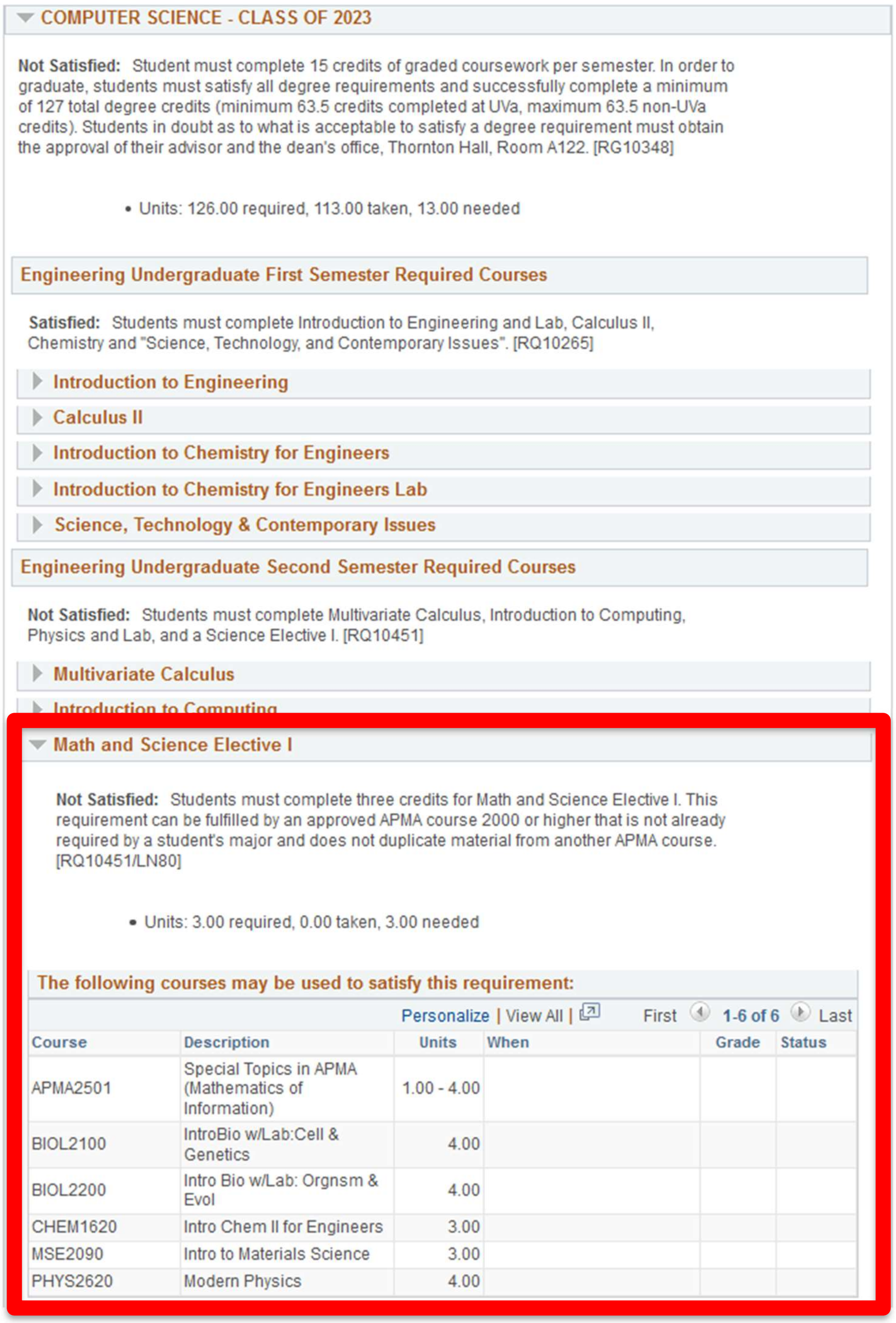
The technological script of SIS does not match the user's script which has caused an expression of user agency in order to create a less frustrating experience for students as they create schedules. Some may argue that the designers of SIS did not have a flawed understanding of the user's identity. Instead they had to comprise usability in order to promote security and ensure that student's data was safer. However, by creating such a frustrating system the designers have led more students to enter their information on other, potentially less secure, websites as a result of their design decisions.

Conclusion

I have argued that the designers of SIS created a frustrating tool for students to create schedules because they had a flawed understanding of the user's identity and capabilities built into its core. The workflow SIS forces upon the user does not match what the user desires and has forced the user to express agency by relying on workarounds to create a schedule. The user configuration framework offered insight into where the designer's ideas about a user were incorrect and the result it had on user experience with SIS. This is significant for software engineers because it highlights that having an inaccurate image of the user's identify during software development leads to the creation of a poor solution. It is not enough to consider functionality vs usability. Software designers needs to consider a diverse variety of users and try to create software solutions that match them and their expectations.

Word Count: 2910

Appendix



COMPUTER SCIENCE - CLASS OF 2023

Not Satisfied: Student must complete 15 credits of graded coursework per semester. In order to graduate, students must satisfy all degree requirements and successfully complete a minimum of 127 total degree credits (minimum 63.5 credits completed at UVa, maximum 63.5 non-UVa credits). Students in doubt as to what is acceptable to satisfy a degree requirement must obtain the approval of their advisor and the dean's office, Thornton Hall, Room A122. [RG10348]

- Units: 126.00 required, 113.00 taken, 13.00 needed

Engineering Undergraduate First Semester Required Courses

Satisfied: Students must complete Introduction to Engineering and Lab, Calculus II, Chemistry and "Science, Technology, and Contemporary Issues". [RQ10265]

Introduction to Engineering

Calculus II

Introduction to Chemistry for Engineers

Introduction to Chemistry for Engineers Lab

Science, Technology & Contemporary Issues

Engineering Undergraduate Second Semester Required Courses

Not Satisfied: Students must complete Multivariate Calculus, Introduction to Computing, Physics and Lab, and a Science Elective I. [RQ10451]

Multivariate Calculus

Introduction to Computing

Math and Science Elective I

Not Satisfied: Students must complete three credits for Math and Science Elective I. This requirement can be fulfilled by an approved APMA course 2000 or higher that is not already required by a student's major and does not duplicate material from another APMA course. [RQ10451/LN80]

- Units: 3.00 required, 0.00 taken, 3.00 needed

The following courses may be used to satisfy this requirement:

Personalize | View All | First 1-6 of 6 Last

Course	Description	Units	When	Grade	Status
APMA2501	Special Topics in APMA (Mathematics of Information)	1.00 - 4.00			
BIOL2100	IntroBio w/Lab:Cell & Genetics	4.00			
BIOL2200	Intro Bio w/Lab: Orgnsm & Evol	4.00			
CHEM1620	Intro Chem II for Engineers	3.00			
MSE2090	Intro to Materials Science	3.00			
PHYS2620	Modern Physics	4.00			

Appendix A - List of course requirements on SIS ("Student information system")

References

Akrich, M. (1992). The de-scription of technical objects. *Shaping Technology/Building Society*, 205–224.

Bloomfield, L. (n.d.). UVa class schedules. Retrieved from https://rabi.phys.virginia.edu/mySIS/CS2/.

Goodwin, N. C. (1987). Functionality and usability. *Communications of the ACM*, *30*(3), 229–233. doi: 10.1145/214748.214758

Google trends: Lou's list. (n.d.). Retrieved from https://trends.google.com/trends/explore?date=today 5-y&geo=US&q=lous list,uva lous list.

Lindsay, C. (2003). From the shadows: Users as designers, producers, marketeers, distributors, and technical support. In N. Oudshoorn and T. Pinch (Eds.), *How Users Matter* (pp. 29-51).

Oudshoorn, N. & Pinch, T. (2003). Introduction: How users and non-users matter. In N. Oudshoorn and T. Pinch (Eds.), *How Users Matter* (pp. 1-25).

Statistics. (n.d.). Retrieved from https://admission.virginia.edu/admission/statistics.

Student information system. (n.d.). Retrieved from https://msisuva.admin.virginia.edu/.

Tang, A. (2011). Software designers, are you biased? *Proceeding of the 6th International Workshop on Sharing and Reusing Architectural Knowledge - SHARK 11*. doi: 10.1145/1988676.1988678

uva schedule | me. (n.d.). Retrieved from https://www.uvaschedule.me/.

Weigand, S. (2019, May 20). UX case study: Process of redesigning UVA's student information system (SIS) page. Retrieved from https://medium.com/@smw5gn/ux-case-study-process-of-redesigning-uvas-student-information-system-sis-page-7ee7ecfabc88.

Woolgar, S. (1991). Configuring the user: The case of usability trials. *In A Sociology of Monsters: Essays on Power, Technology and Domination* (pp. 58–99). London: Routledge.