Adapting the “Unessay” for Use in Computer Science

John Aycock
Hannah Wright
Jannik Hildebrandt
David Kenny
Nicole Lefebvre
Mark Lin
Maria Mamaclay
Shanel Sayson
Aidan Stewart
Amy Yuen
aycock@ucalgary.ca
hannah.wright@ucalgary.ca
jhildebrandt@student.tugraz.at
djkenny@ucalgary.ca
njlefebv@ucalgary.ca
ling@ucalgary.ca
mmamacla@ucalgary.ca
ssayson@ucalgary.ca
aidan.stewart@ucalgary.ca
yuena@ucalgary.ca

Department of Computer Science, University of Calgary
Calgary, Alberta, Canada

ABSTRACT
The “unessay” is an extremely open-ended form of assignment that has primarily seen use in the humanities. This paper is an experience report discussing how a traditional computer science programming assignment was replaced with a suitably-adapted unessay assignment. We present the assignment specification, assessment criteria, as well as three sample submissions. Importantly, the experience is captured from multiple perspectives: the instructor’s, as well as firsthand reflections from the students who created the three submissions.

CCS CONCEPTS
• Social and professional topics → Student assessment.

KEYWORDS
assignment, unessay, open-ended, creativity

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WCCCE ’19, May 3–4, 2019, Calgary, AB, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6715-8/19/05...
https://doi.org/10.1145/3314994.3325073

1 INTRODUCTION
Every computer science instructor is familiar with the process of assessment using traditional programming assignments. Needing to assess students’ knowledge of topic X, the instructor will devise an appropriate assignment. The instructor will perhaps complete the assignment themselves and tune the assignment, then write a specification of said assignment for students. The students, in turn, will complete the assignment; a sufficiently good grade on a student’s submission is interpreted to mean that the student has exhibited some level of mastery of X.

There are other ways to assess X, however. The first author taught a fourth-year (senior) computer science undergraduate course, and several iterations of one particular traditional programming assignment seemed to fall flat: overly contrived, unsatisfying, and uninteresting. Inspired by seeing uses of the extremely open-ended “unessay” in the humanities (e.g., [7, 18]), with some use in business computing [5], the last offering of the course recast the troublesome assignment as a computer science unessay. The assignment covered the same ground as a traditional one, yet threw open the doors to the students’ creativity.
In the end, this experimental assignment produced very interesting and creative results: submissions included programs, videos, card games, comics, poems, and adventure games. \(^1\) We present the assignment “specification” in Section 3, followed by three student submissions and their reflections on the experience in Section 4. Discussion and conclusions follow in Section 5.

This paper adopts an unconventional structure due to the nature of its multiple authors’ different contributions. The first author (JA) was the course instructor and the instigator of this assignment; he/I wrote the bulk of the paper’s sections with certain exceptions. The descriptions of the example submissions and reflections on the assignment in Section 4 were written by the students themselves. These are voices that are seldom heard in computer science education research: direct responses to the assignment, not filtered or reduced down to survey data. Additionally, the second author (HW) contributed the related work section, which delves into the definition of an unessay.

2 RELATED WORK

Coined by Daniel Paul O’Donnell in 2012, the term unessay refers to “an assignment that attempts to undo the damage done by [traditional essays]” \(^1\). O’Donnell asks his students to “[t]hrow out all the rules you have learned about essay writing in the course of your primary, secondary, and post secondary education” \(^1\) in preparation for his unessay, which lets the students pick their own topic as well as how they want to present it \(^1\). He explains that the skills practiced with an unessay (matching the form to the topic, writing about topics of interest, etc.) are just as useful when writing “real” essays and, as a result, unessays help students develop transferable skills without being constrained by the formalities of traditional essays \(^1\). While O’Donnell’s definition of the unessay focuses on written assignments for English students, the idea of eliminating rules in favor of more open-ended, creative assignments can be applied in other ways to be suitable for other disciplines as well.

As early as 1998, professors at the Engineering School of Virginia Commonwealth University implemented open-ended lab assignments into an introductory statics course. Rather than simply stepping through pre-determined instructions for each lab as was done previously, students were challenged to complete traditional laboratory assignments with “little more than an apparatus and a hypothesis” \(^1\). While still following a traditional assignment structure, these open-ended labs left the main problem solving challenge up to the students (with assistance from the instructors and teaching assistants). Brauner took a similar approach, removing formal instructions from their laboratory assignments in favor of an open-ended approach in an attempt to encourage student responsibility \(^3\). However, the unessay can be pushed further than simply removing information or guidelines that are typically included as part of traditional assignments.

For example, in 2016, Gribetz assigned a final project where her students were “free to create almost anything, as long as it seriously and meaningfully engaged with the texts and themes that [they] read and discussed together in class” \(^8\). This assignment pushed past the restrictions that a traditional assignment structure enforces, giving the students full creative control over how they wished to communicate their knowledge. In the end, Gribetz received a wide array of final projects including a recording of a fake radio show, a dance routine, a sculpture, and a spoken-word poem \(^8\). Through this process, Gribetz said that she “realized that when students become personally invested in a project, they dig deep, they read carefully, they notice details” \(^8\). And, despite the wide variety of assignment outcomes from her students, Gribetz mentioned that the marking process was not that different from marking a traditional assignment \(^8\).

While most examples of pushing the rules of assignments can be found in the social sciences or the arts, flexible assessment methods have been adopted in computer science classrooms to some degree. For example, some instructors chose to take a problem-based learning approach rather than following the typical known-answer assignment structure typically found in introductory computer science classes \(^9, 10, 13\). The belief behind this change was that “open-ended, authentic, substantial problems ... [drives] the learning” \(^10\) of the students, with O’Kelly and Gibson stating that their motivation was to increase retention of first-year computer science students by engaging their interest \(^13\). Pirker, Riffnaller-Schiefer, and Gütl took a motivational active learning approach in order to make their classroom more engaging and increase student motivation \(^15\). While all of these methods proved to be successful in their classroom applications, none of the above examples strayed too far from a typical assignment structure. In fact, we were unable to find any publications that discussed computer science assignments throwing away the rules in the same way that Gribets’ assignment did for her social science students \(^8\).

Considering the emphasis that computer science education researchers put on creativity \(^1, 9, 11, 13, 16\), it is interesting that the instructors often limit their own creativity by sticking to traditional assignment structures when designing their assessment methods. In a 2015 paper, Sullivan argues that creativity should be “nurtured in our composition classrooms, grades 6–13—and not hidden away” \(^17\); however, our contention is that the value of creativity extends to later education as well, especially in computer science. As Knoebelsdorff and Romeike put it, computer science is a “creative field ... where creativity is demanded and encouraged” \(^11\). Hence, we believe it is important for computer science educators to embrace the creativity that they strive to instill in their students as they design their assignments.

3 A COMPUTER SCIENCE UNESSAY ASSIGNMENT

The course where we used the unessay is a systems course on the implementation of old, “retro” computer games \(^2\). One aspect of the course is pointing out modern applications of old design and implementation techniques; for this particular assignment, the modern applications were internationalization (i18n) and embedded extension languages. They share a deep connection, one of language, and are also interesting in the sense that they are not topics likely to be otherwise encountered by students during their degree. The problem is how one assignment can meaningfully incorporate both.

\(^1\) Inclusion of this statement is exempt from ethics review, as verified with the University of Calgary’s Conjoint Faculties Research Ethics Board.
The first time the course was offered, this was presented as a traditional programming assignment. Starting with a simple “Hello, world” program, students had to add i18n support for English and at least two other languages. Additionally, they had to embed Lua, Tcl, or Python into the program to support commands in a text-based configuration file. Strictly speaking, it covered the two topics, but was abstracted so far from real applications as to feel contrived.

For the second offering of the course, the assignment was switched to a text-based tic-tac-toe game. The i18n requirements were to support English and one other language; the embedded extension language portion required using Duktape, an embeddable JavaScript interpreter, to implement AI algorithms for a computer tic-tac-toe opponent. Less contrived, but also less than inspiring.

Enter the unessay. For the third and most recent instance of the course, the assignment became an unessay adapted to this computer science context. O’Donnell’s original unessay concept would permit students to use any topic, as long as there was some connection to the course material [12].

While this unbounded scope might be appropriate for a term project, it seemed too open-ended to replace this specific assignment. Instead, we deliberately narrowed its scope to match that of the traditional assignment, and shifted the onus onto students to demonstrate their understanding.

The assignment specification was simply ‘demonstrate convincingly to me that you understand the following two things:

(1) Internationalizing a program in terms of human language.
(2) Embedding an extension (programming) language into a program.

The way you complete this assignment and demonstrate these things is limited only by your imagination. Here are some examples, but this is far from a complete list: epic poem; song; graphic novel; board game; essay; a play in iambic pentameter; card game; comic book; tutorial video; a pop-up book; animated tutorial video; a screenplay; about a bazillion other ways; oh yeah, program(s).

Especially considering a computer science audience, the intent was to retain program-based solutions as a possibility, while trying to emphasize that other approaches might be more desirable.

Pragmatically speaking, consideration needed to be given to accepting both physical submissions as well as electronic ones. Also, a substantial amount of thought was given to the grading rubric, which was published along with the assignment specification, and was based on O’Donnell [12]. Four criteria were used, with four points allotted to each – again quoting from the specification:

- Correct and convincing: any obviously wrong things? Am I convinced that you understand both i18n and embedding?
- Focused and judicious: are important things included, and unimportant ones excluded? Do I need to dig to find out if you met the requirements?
- Cohesive and appropriate: have both i18n and embedding been seamlessly combined into one? Does the presentation method chosen suit the topic?
- Interesting and presentable: is creativity demonstrated? Is the result understandable/running/playable/readable/watchable/listenable (i.e. the production values are appropriately high and the audience is not distracted by avoidable lapses in presentation)?

There are obviously some challenges assessing a variety of submissions in different formats, a topic we return to in Section 5, but the experience from the instructor point of view was markedly different than grading traditional assignments: a potential slog through same-y submissions was replaced by the surprise and delight of seeing students’ creativity on display.

4 EXAMPLE SUBMISSIONS AND STUDENT REFLECTIONS

Here we present some exemplary solutions that were submitted for this assignment, along with first-hand reflections written by the students who created them. The first two are physical games – Boss Bash (by NL, MM, and SS) and Embedded I18N (JH, DK, AS, AY) – and provide not just examples but counterpoints to one another, showing substantially different interpretations of the assignment using the same medium. The third, Tales of Two Kingdoms (ML, HW), is a radical departure from both games and traditional computer science fare.

4.1 Boss Bash

The Boss Bash game we created for this assignment is based on the card game Smash Up (Alderac Entertainment Group 2017). We used

Figure 1: Boss Bash cards (unblurred images are CC0)
internationalization was more difficult to incorporate into the card game, but we eventually came up with an interesting way to include multiple languages on both the cards and in the instructions. Once we had a general idea of the direction we wanted to take, the assignment came together quickly. Being in a group also gave us opportunities to discuss not only the topics themselves but also how to present these topics comprehensively, which helped our understanding of internationalization and extension languages even further. This assignment made internationalization and extension languages much clearer than when we briefly learned it in lecture.

4.2 Embedded i18N: The Card Game

The game we designed (Figure 2) consists of 8 different kinds of cards with 64 cards in total. Each player starts with 7 cards and the goal is to be the first one with no cards left. There are special cards that change the language of the game, which influences other cards. This is our approach to integrate i18n in our game. The embedded extension language is addressed by the cards themselves. The whole game can be seen as a machine and each card has a code snippet written on it. When the card is played this code is executed and changes the machine (the game).

We chose a game because it was the most appealing idea to us out of the example options. We were originally going to go with a board game instead of a card game but we were having trouble figuring out how to incorporate the extension language portion of the assignment. The i18n portion was basically the same from conception to end result: our first idea was to make a game where
Figur 3: Sample pages from Tales of Two Kingdoms

the players have to switch languages every so often. One of us suggested a card game instead of a board game so that we could have the cards as extension languages themselves and we basically worked outwards from there.

At first, we felt overwhelmed by the open-ended nature of the assignment. As university computer science students, we’re used to having more structure. So, we spent a lot of time figuring out what we wanted to do. However, we were glad that it wasn’t another coding assignment. Now, being done the project, we’re glad that we were able to make something more creative. It also made us think in a different way than we were used to.

4.3 Tales of Two Kingdoms

For many of our courses, assignments are accompanied with requirements that outline the most “important” aspects. This helps make the expectations of the course transparent to the students, since one can easily identify the areas they are unfamiliar with based on whether or not they have met these requirements. Our assignment in this course was different in this sense – due to its open-ended nature, it encouraged us to be creative and was vague in its requirements. This led to initial feelings of frustration, since we were uncertain on how to approach the assignment. A large concern of ours was that we would misconstrue what the professor deemed to be important aspects of the concept he was teaching versus what we personally felt was important.

After some discussion, we settled on writing a children’s book to express what we learned (Figure 3). The reason for this was twofold: it gave our assignment an artistic element (making use of a strength we had) and also supported the notion that the ability to simplify a difficult concept is an effective way to demonstrate understanding. Having decided on our goal, we were able to produce our own set of requirements for the assignment and became more confident. The idea of being able to create our own guidelines made it feel more personal and led to us becoming more invested in our work.

The overarching setting of our book, and the reason for the title “Tales of Two Kingdoms,” was a fantasy world with two kingdoms: the Kingdom of Users and the Kingdom of Software. The book had three chapters: one on internationalization, another on embedded languages, and the last brought the two concepts together in order to show how they were similar. While each chapter told a different story, they were all set in this same world. The first chapter told the story of the princess of the Kingdom of Users and the prince of the Kingdom of Software. The two were in love but, due to a language barrier, the prince could not communicate with the princess; luckily, he and his subjects used internationalization to overcome this. The second chapter described the rivalry between two knights of the same kingdom representing different programming languages. Amidst their feud over who was better, they united to battle a common enemy and discovered that the opposite parties both had unexpected, meritable traits. A friendship then emerges where both knights make use of each other’s strengths to overcome obstacles. We worked on chapters one and two individually, but we worked together on the final chapter in order to blend our ideas.

Although we had a clear idea as to what we wanted to do, we still struggled on many aspects of the assignment. One thing we frequently discussed was scope: what details were the most important, and what should we avoid talking about? We had to pick and choose what we felt were the most crucial elements, leaving behind more technical details that might not be suitable in a book for children. At the same time, we could not lose sight of the fact that it was still an assignment with the goal of convincing our professor that we understood the concepts. Since the book was aimed at kids, we wanted to keep both the text and images simple which, once again, meant excluding technical details. As a balance, we opted for a simpler text with the pictures expressing more technical detail. For example, chapter one showed how internationalization was useful for software prompts with an image but referred to this as Software subjects “giving instructions” to the Users.
One noteworthy benefit of the open-endedness of the assignment (or rather, the format we chose to do it in) was how much easier it was in identifying gaps in our knowledge. When we were unable to communicate the ideas in a simpler language, it immediately highlighted an area where we needed a deeper understanding. To further this point, we were able to supplement each other’s knowledge by describing the concepts in the context of our own work while still understanding the concept at the level it was taught but without the technical jargon. However, one reservation we shared after the assignment was that, while we were confident in explaining the concepts of internationalization and embedded languages, we were not familiar with how to apply these concepts. Consequently, equally important aspects such as implementation details are lost. For instance, we are unable to suggest what may or may not be a good choice for an embedded language or properly comment on the struggles of using/implementing one. Despite this, the assignment and the approach we decided on was a positive experience overall. It allowed us to freely exercise our creativity, making it more enjoyable and entertaining, as opposed to a chore that most students view assignments as. It prompted us to become more involved and invested in our work while also still being able to retain what was taught.

5 DISCUSSION AND CONCLUSIONS

Our experience with the unessay assignment was a positive one, an experiment that definitely bears repeating. However, there are also some refinements that could be made.

First, the unessay could be paired with a follow-up self-reflection exercise. There may be value for students in a “post mortem” that considers why they chose a particular medium, and whether they stayed in their comfort zone, or used the occasion to branch out.

Second, in that vein, the option of submitting a traditional program could be removed entirely to force students into different media types. At the very least, our experience was that compiling and running a random assembly of program submissions comprised a substantial amount of marking time. Less drastic alternatives to an outright ban on program submissions would be cases where the instructor need not run the program directly, such as showing a program in a tutorial video, or requiring any programs to run in-browser without installing plug-ins.

Third, Flick mentions requiring “a 1 paragraph explanation/summary/artistic statement” [5] as part of unessay submissions. We think that has merit: on the one hand, students are forced to articulate how their submission relates to the specification; on the other, it gives students a framing device for their unessay to assist others (notably the instructor) in its interpretation.

Fourth, although McLuhan is not normally required reading for computer scientists, the unessay is a place where ideas about media could be deliberately introduced. As Graham observes for unessays, “the focus shifts […] to how the medium and the argument intersect, reinforce, or contradict each other” [6].

Fifth, given our unessay’s origin in place of a traditional assignment, no attempt was made to showcase the results more broadly beyond the audience of one: the instructor. After seeing the creative results, however, this should be reconsidered in future.

Interestingly, the unessay is diametrically opposed to the course in which we used it. The implementation of old computer games is in large part the story of developing under extreme constraints; the unessay, by comparison, removes constraints and forces students to confront the difficulty of choosing one thing to do from a universe of possibilities. That said, while our experience with the unessay was in a specialized fourth-year course, nothing precludes its use – and its ability to encourage creativity – in a wider range of computer science courses.

ACKNOWLEDGMENTS

The first author’s work is supported in part by the Natural Sciences and Engineering Council of Canada, grant RGPIN-2015-06359. Thanks to Michael Ulliot for sharing his unessay experiences.

REFERENCES

[1] M. Apiola, M. Lattu, and T. A. Pasanen. 2010. Creativity and intrinsic motivation in computer science education: experimenting with robots. In Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education. ACM, 199–203.
[2] J. Aycock. 2015. Applied Computer History: Teaching Systems Topics through Retrogames. In 20th ACM Annual Conference on Innovation and Technology in Computer Science Education. 105–110. https://doi.org/10.1145/2729094.2724583
[3] A. Brauner, J. Carey, M. Henriksen, M. Sumnerhagen, and E. Ehrenborg. 2007. Open-ended assignments and student responsibility. Biochemistry and Molecular Biology Education 35, 3 (2007), 187–192.
[4] J. Cavan. [n. d.]. https://make.wordpress.org/polyplots/handbook/about/what-we-do/logos-and-graphics/. CC BY-SA 4.0 license (https://creativecommons.org/licenses/by-sa/4.0/).
[5] C. Flick. 2018. Assessing Unessays. Workshop at the DMU Institute for Education Futures. https://ief.dmu.ac.uk/2018/01/15/january-workshop-assessing-unessays/ Presentation at https://slideplayer.com/slide/13999578/.
[6] S. Graham. [n. d.]. History. Digital Pedagogy in the Humanities: Concepts, Models, and Experiments. Retrieved 13 March 2019 from https://digitalpedagogy.mla.hcommons.org/keywords/history/
[7] S. Graham. 2016. No More Essays. https://github.com/shawngraham/ocdsb Accompanying presentation at https://shawngraham.github.io/presentations/ocdsb.html.
[8] S. K. Grubetz. 2016. Creation and Creativity: Teaching Critical Thinking Beyond the Term Paper. http://www.ancientjewreview.com/articles/2016/7/25/creation-and-creativity
[9] C. C. D. Head and S. A. Wolfman. 2008. Google and the Unknown-answer Assignment: Open-ended, Shareable CS1 Assignments. In Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE ‘08). ACM, New York, NY, USA, 133–137. https://doi.org/10.1145/1352135.1352183
[10] J. Kay. M. Barg, A. Fekete, T. Greening, O. Hollands, J. H. Kingston, and K. Crawford. 2000. Problem-based learning for foundation computer science courses. Computer Science Education 10, 2 (2000), 109–128.
[11] M. Knobelsdorff and R. Romeike. 2008. Creativity as a pathway to computer science. In Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education. ACM, 286–290.
[12] D. P. O’Donnell. 4 September 2012. The unessay. http://people.uleth.ca/~daniel. odonnell/Teaching/the-unessay Last modified 28 September 2018.
[13] J. O’Kelly and J. P. Gibson. 2006. RoboCode & problem-based learning: A non-prescriptive approach to teaching programming. In Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education. ACM, 217–221.
[14] M. A. Palmer, E. Sandgren, R. A. Heinz, A. Chatterji, and T. W. Haas. 1998. A novel science course. In 28th Annual Frontiers in Education Conference, Vol. 3. 1236–1241 vol.3. https://doi.org/10.1109/FIE.1998.738639
[15] J. Pirkner, M. Riffnaller-Schaefer, and C. Gütli. 2014. Motivational active learning, engaging university students in computer science education. In Proceedings of the 19th Annual Conference on Innovation and Technology in Computer Science Education. ACM, 297–302.
[16] R. Romeike. 2007. Three drivers for creativity in computer science education. Proc of Informatics, Mathematics and ICT: a ‘golden triangle’. Boston, USA (2007).
[17] F. Sullivan. 2015. The unessay: Making room for creativity in the composition classroom. College Composition and Communication (2015), 6–34.
[18] M. Ulliot. 4 October 2012. Ideas + Materials for Research + Teaching [Blog]. http://ullyot.ucalgaryblogs.ca/2012/10/04/the-unessay/