Incorporating Resubmission into Skills-based Courses

UWTL 2021 Conference

Michael Liut, **Andrew Petersen**
Mathematical and Computational Sciences
University of Toronto Mississauga

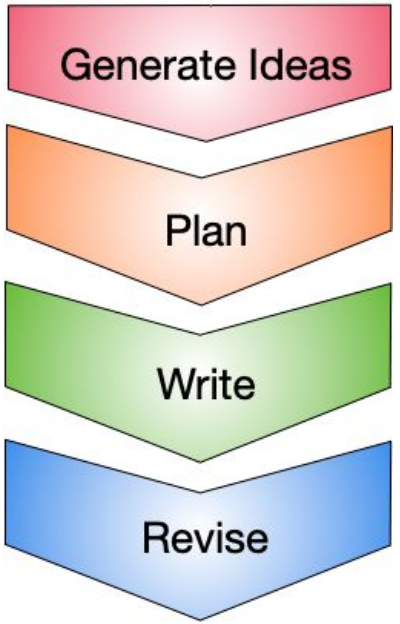
UNIVERSITY OF TORONTO MISSISSAUGA

Revision (in Writing)

Revision is a critical phase in the writing process, where the writer reconsiders and refines their work.

Revision is closely connected to “critical reading”, as the work is evaluated on a set of criteria.

While challenging, revision (and editing) are critical phases for generating truly effective artifacts.



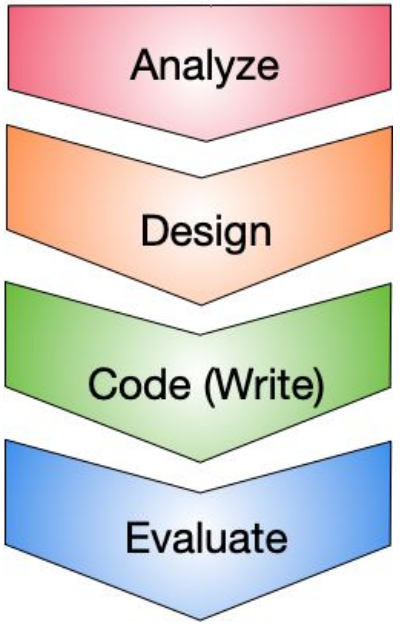
Revision (in Programming)

The programming process should look familiar!

The Evaluation phase is critical, as the programmer identifies whether the program is an effective solution.

Students often think of evaluation as “debugging” and “testing”: fixing problems.

But proper evaluation requires the programmer to critically read the code -- to consider a wider set of criteria.



Our Course (Background)

- Introductory Programming (CS1), assuming no prior experience.
  - Functions, Basic Logic, Loops, Data Structures, File I/O, Sorting, OOP
  - Test-driven development. Functional Design.
- 800-1200 students per semester.
- General first year course. ~50% are interested in pursuing Computer Science as their program of study.
- Active Learning in an Active Learning Classroom (and pods!)
  - Peer aided support and feedback through code review, analysis, and refactoring.
  - Course is scaffolded and builds off of prior fundamental knowledge.

Our Course (Structure)

Similar to the ISW's BOPPPS model, our course uses these 3 P's:

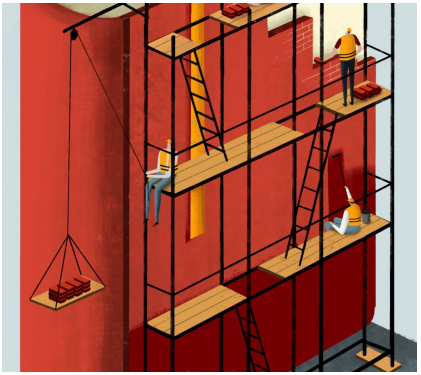
Prepare: completing mini-tasks targeting specific learning objectives

Practice: attending a synchronous lecture for a mini-lesson in an active learning classroom

Perform: a series of exercises that test students' understanding of prepared and practiced materials

This structure is implemented weekly to introduce new concepts and to increase the depth of previously introduced ideas.

Resources and Scaffolding



Credit: https://www.edutopia.org/blog/scaffolding-lessons-six-strategies-rebecca-alber

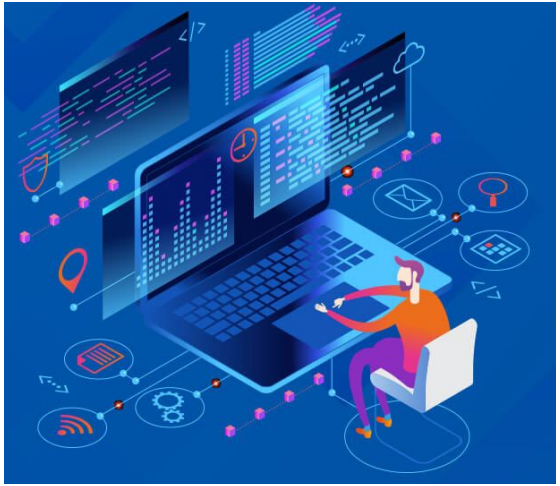
Available Resources

- General guides/processes to frame the approach to solve problems
- Short video modules demonstrating concepts and processes, with exercises
- Slides and exercises from lecture (with solutions)
- Live support in labs, asynchronous Q&A forum

Scaffolding

- The course is organized to repeatedly revisit previous material at a deeper level
- Problem solving processes are introduced, demonstrated, and then observed and critiqued
- The language is introduced in stages: syntactic, then semantic, then in combination with other language features

Role of Revision in Our Course



Credit: https://www.goodcore.co.uk/blog/coding-vs-programming/

We position programming as a process that involves:

- Analyzing the problem
- Designing a solution
- Writing tests
- **Writing a first draft of the code**
- Iteratively reviewing, assessing, and updating the code

Students naturally focus on the bolded step.

They also focus on *functional* correctness -- but there are many other (more complex) criteria.

There are multiple ways to solve a problem. We want solutions that are “high quality”.

Revision in Class Meetings

Credit: https://www.elegantthemes.com/blog/tips-tricks/2-ways-to-fix-the-wordpress-mixed-content-error

Class meetings are meant to (a) model process and (b) provide opportunities for students to receive feedback as they collaborate in groups.

Example activities:

- Explicitly modeling the design process and translation to code, then emphasizing review and revision.
- Showing student solutions to an exercise, discussing and comparing the solutions, and then revising them live.
- Building multiple solutions to show differences in speed and readability.

Resubmission in Weekly Exercises

Students are *Prepared* for class meetings with mini-lessons (readings and videos) followed by reflection and comprehension activities (mini-exercises).

I*mmediate feedback* is provided to allow students to *revise and resubmit* exercises until no issues are detected.

We use a combination of functional tests and tools that analyze the structure of the code for the use of appropriate structures.

- But adherence to process can’t be effectively monitored in asynchronous sessions

Revision of Assignment
(24-hour Resubmissions → Resubmission Tokens)

Credit: https://blog.polymath.network/what-the-is-a-security-token-f4ff987620e8

Assignments involve more complex problems and require integration of multiple concepts or syntactic structures.

- The process is the same, but the problems are more difficult.

Students are expected to demonstrate process by submitting artifacts demonstrating their design work and evaluation of the code.

- Still, assessing evaluation and revision beyond “functional correctness” is challenging

Students are encouraged to submit multiple times to get feedback.

- Initial model: one resubmission allowed, 24 hours after initial submission.
- Later models: multiple resubmissions allowed but rate limited to encourage students to do their own evaluation.

Problem: Focusing on Functional Correctness

Since most immediate feedback focuses on functional correctness, style, and the use of problematic constructs, students tend to *edit* rather than *revise.*

- Changes tend to be small, and to focus on aspects of the current approach
- Sweeping changes, like refactoring, aren't encouraged by the feedback that can be provided quickly

We have introduced peer review of code, both in class and after assignment submission, to encourage reflection on more complex issues and to demonstrate larger-scale revision.

Problem: Modeling Revision (in ALCs)

How do we model the framework in a room designed for active work?

1. Present a problem
2. Individual students jot down their ideas of how to solve it.
3. Students work in their “pods” to come up with their "best" solution strategy.
4. The “pods” formulate (code) a solution, as TAs circulate to provide feedback.
5. “Pods” rotate and provide feedback on the solution at their new station.
6. TAs record solutions and comments (taking photos with phone/camera).
7. Instructor selects 3-5 solutions with different characteristics and then performs a whole-class code review in the following lecture.

Problems: Avoiding “Fixit” Focus

Credit: https://www.information-age.com/fixing-problems-save-smes-millions-hours-123466214/

Providing feedback with little penalty for resubmission is good practice but encourages students to fix issues, rather than to proactively avoid them.

Our approach is to gradually fade scaffolds -- both in this course and across the courses in our program. Students must take increasing responsibility for critically reading their code.

Problem: Participation

Credit: https://mindflash.com/blog/training-incentives-improve-participation

Our ongoing open question:
How do we motivate students to actually participate in the revision process?

- How do we provide grade-based incentives without creating a system to “game”?
- How do we authentically assess engagement in processes in an asynchronous activity?

We have provided grade-based incentives for resubmitting, but the top-performing students have little incentive to engage, and some lower-achieving students see this as an extension of the deadline.

Where Else?

In Computing:

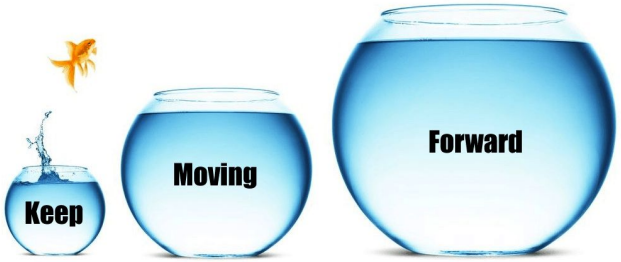
- Software Engineering: Refactoring, Agile Methods
- Theory: Constructing and revising proofs

These are topics that create an artifact through a process.

The analyze, write, evaluate loop is present in all of these processes.

Are these processes present in your own field?

Going Forward



Credit: https://netwiseprofits.com/just-keep-moving-forward-wealthy-affiliate-success/

We'd love to see resubmission evaluated in STEM contexts! 😊

- Ideally, across disciplines
- Also ideally, across educational contexts

If you're interested, please get in contact with us!

andrew.petersen@utoronto.ca & michael.liut@utoronto.ca