Enhancing CS Degree Programs by Reducing Structural Complexity

Albert Lionelle
Director of Align, Associate Teaching Professor
Khoury College of Computer Sciences,
Northeastern University

Northeastern University
Khoury College of Computer Sciences

Northeastern
Center for Inclusive Computing

Does Structural Complexity Influence Diversity of Students?

Based on: Does Curricular Complexity in Computer Science Influence the Representation of Women CS Graduates?

By Albert Lionelle, McKenna Quam, Carla Brodley, and Catherine Gill

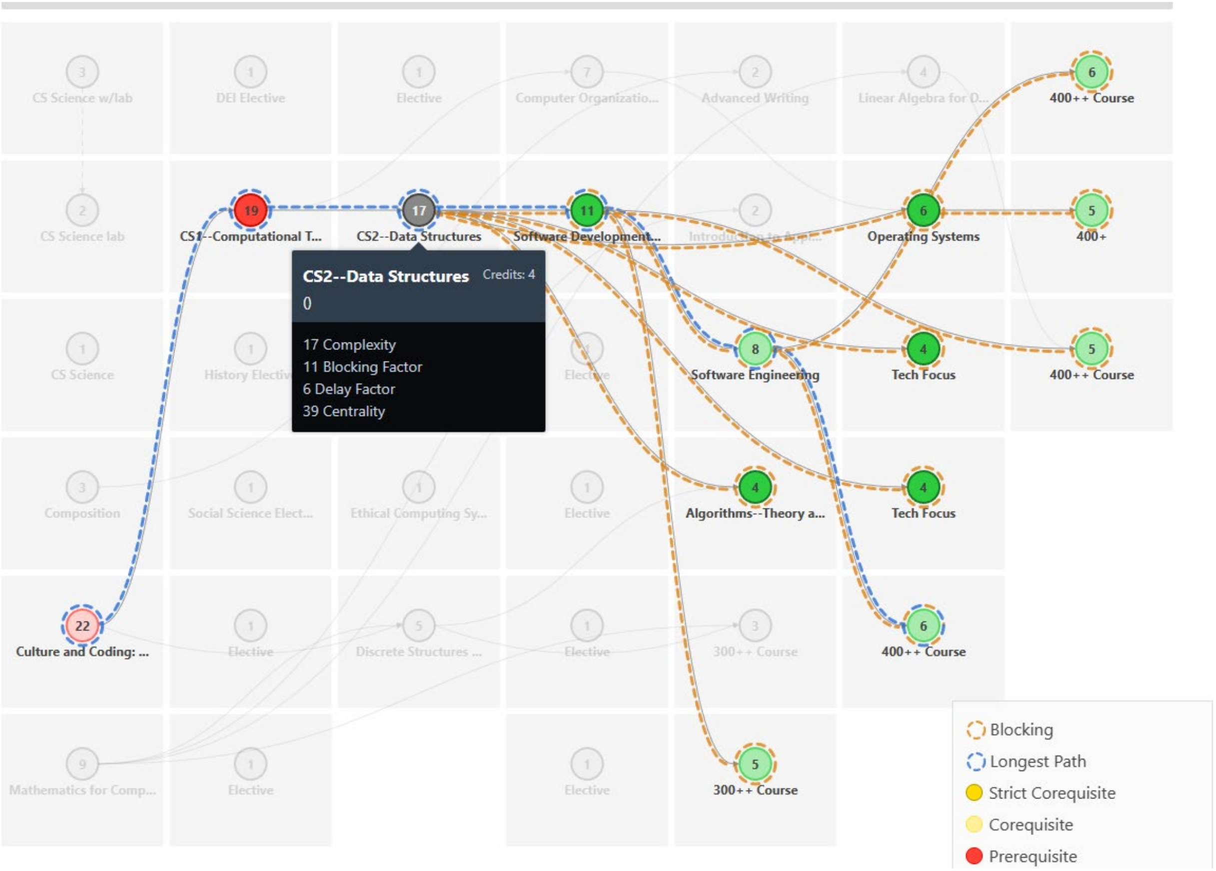
https://dl.acm.org/doi/10.1145/3626252.3630835

Northeastern Center for Inclusive Computing

What is Curricular Complexity?

- Curriculums have an innate structure
  - Prerequisites, course requirements
- Curricular Complexity – the complexity of that *structure*
  - Not to be confused with course complexity
- Four measurements to compare curricular structure
    - Developed by Gregory L. Heileman and Ahmad Slim
  - Complexity
  - Centrality
  - Delay Factor
  - Blocking Factor

Curricular Analytics



- Blocking Factor
  - Number of courses that require course. 11 in example
- Delay Factor
  - Measures sequential ordering (max) that it is a member. 6 in the example
- Centrality
  - Sum of delay factors “how many course chains include this course”.
- Complexity
  - Combination of Blocking + Delay
- Curricular Complexity
  - Sum of all course complexities. 175 in example

https://curricularanalytics.org/

Northeastern Center for Inclusive Computing

Why does this matter?

- Curricular Structure
  - Influences students directly
    - How long until graduation, ordering, etc
  - Measure to compare “best practices” in curricular design
- Heileman et al. found[1]
  - Higher ranked Engineering and CS programs had lower curricular complexity
- Meaning
  - The structure of the curriculum was less complex
  - Making it easier to
    - Take courses in different orders
    - Transfer into the program later
    - Less assumptions about previous knowledge going into courses (hopefully)

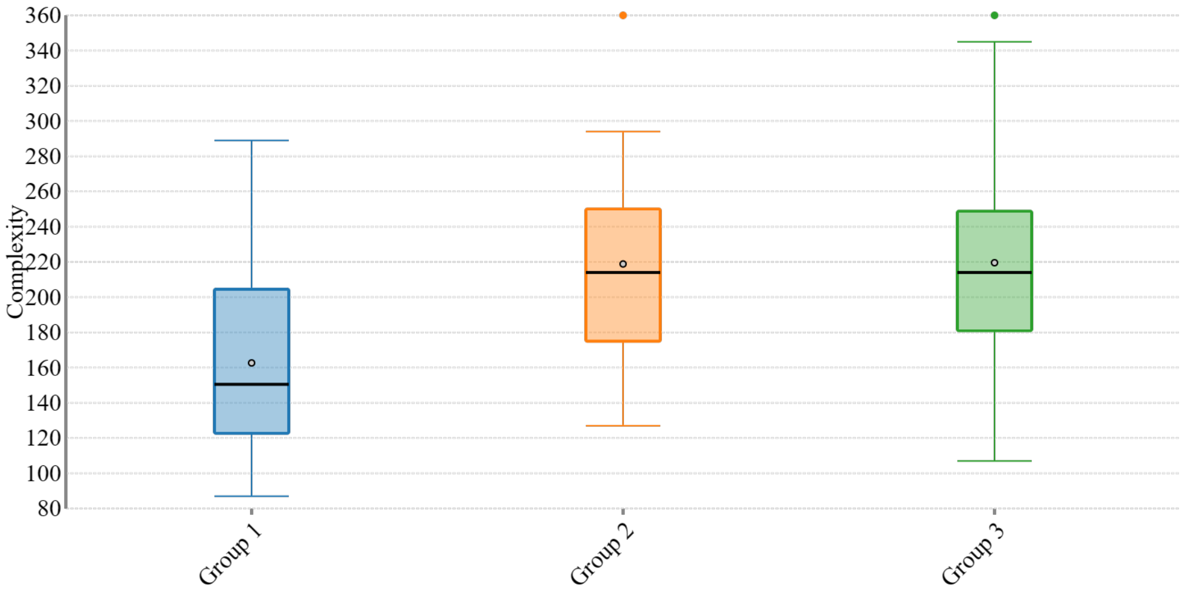
[1] https://doi.org/10.18260/1-2--32677

Degree Complexity Related to % Women

- Sampled 60 programs (20 each group)
- Built 60 degree maps
- By public facing websites
- Assumed calculus ready, no AP credits, bias towards reduced prerequisites, and 'quickest path' to graduation

	% women	Mean Complexity	Median longest delay factor
Group 1	n > 20%	162.7 ± 12.8	5
Group 2	20% >= n > 15%	218.9 ± 13.1	6
Group 3	n < 15%	219.6 ± 14.4	6

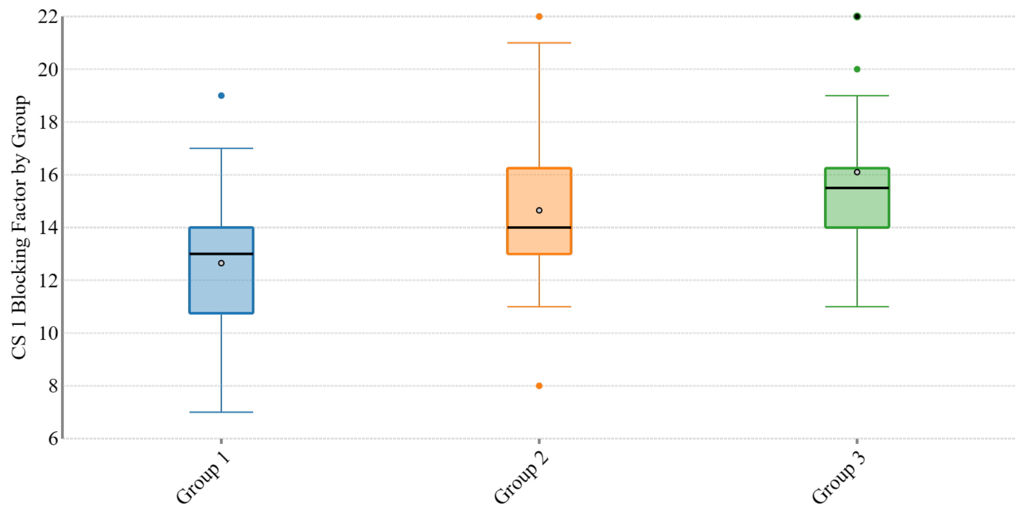
p = 0.003

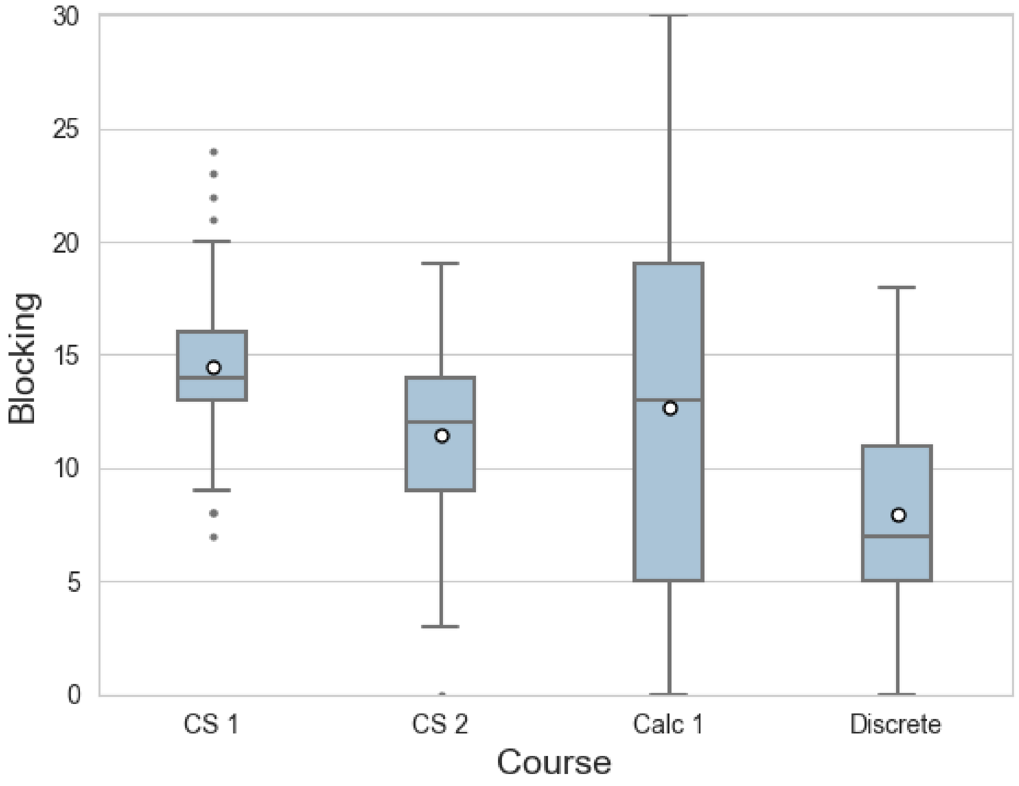


Northeastern
Center for Inclusive Computing

Blocking Overall

- Blocking prevents progress
- Across all 60 schools
  - CS 1 should be blocking (mostly)
  - Calculus I, no consensus!





Northeastern
Center for Inclusive Computing

More on math requirements

Based on “An Analysis of the Math Requirements of 199 CS BS/BA Degrees at 158 U.S. Universities”

By Carla E. Brodley, McKenna Quam, and Mark Weiss

https://cacm.acm.org/research/an-analysis-of-the-math-requirements-of-199-cs-bs-ba-degrees-at-158-u-s-universities/

Math’s Roll in Computer Science

All U.S. CS programs with 150+ graduates with publicly available degree plans

199 degrees at 158 universities

- 80% Bachelor’s of Science, 20% Bachelor’s of Arts
- 100 in Engineering, 99 not
- 83 ABET-accredited
- 55 MSIs, 47 AAU, 4 online
- 60% of all CS graduates in the U.S.

Results

Course	# of Degrees	% of Degrees
Calculus 1	191	96.0%
Calculus 2	152	76.4%
Discrete	198	99.5%
Probability/Statistics	140	70.4%
Linear Algebra	116	58.3%
Calculus 3	43	21.6%

Additional Findings

- High inclusion of Discrete, no consensus on placement in the program.
  - 117 requires it for Algorithms and 73 data structures, but shows up across 29 different classes at substantially lower rates
- Calculus 1
  - 73 require for Discreet
  - 33 require for CS 1!
  - All assume calc ready
- Calculus 2
  - 108 have it, but not tied to any other computing course (pre/post req)

Forming Best Practices

Best Practices Observed

- Critically Evaluate Prerequisites and Assumptions
- Minimize Delay on Transfer Students
- Eliminate choke points preventing progress
- Offer flexibility around when calculus must be completed
- Small core, with flexible options after core
- How you communicate degree plans matter

Critically Evaluate Prerequisites

- Look at prerequisites
  - Is it really needed or is a proxy for something else
    - Proxy for “mathematical maturity”
    - Proxy for advising paths
- Don’t block progress unless majority of content is needed
  - Especially true for mathematics courses
    - Don’t let other departments determine who is in your major
  - Ensure there is time to discover CS
    - Ensure time to graduate after discovery
- Question college assumptions – example COE
  - Rethink common engineering core for first year
  - Does CS benefit from engineering core or just hurting discovery?

Minimize Delay on Transfer Students

- The number of transfer students
  - Increase every year
  - Primary concern: time to graduation
- Use curricular maps to look at time to graduation
- Reduce if transfer students have “added” complexity
  - Extremely common!
- Minimize barriers on transferring into the program
  - Also true for AP credits or partial transfers!

Flexibility

- Programs with greater representation had
  - More options for students at various points
  - Smaller core requirements
    - 26% - group 1 compared to 33% group 3
  - Group 1 is characterized by minimal prerequisite
    - Often 300/3000 lvl upper division only required Data Structures
  - Don’t assume calc-ready

Choke Points and Calculus

- Choke points
  - Very evident in a curricular map
  - Programs with greater diversity often had ‘pathways’
    - Allows progression to prevent frustration of a choke point
- Calculus 1 – if placed early becomes a choke point
  - Yet, there is NO consensus between programs
  - Suggestion:
    - Only require it *as needed* for a course
    - Delay when it is needed
      - Allows time for pre-calc requirements
    - CS2023: ACM/IEEE-CS/AAAI Computer Science Curricula

Communication Matters

- Students do use websites
- We found:
  - Some programs made it very difficult to find information
    - Requirements often listed across multiple pages (university, college, degree)
    - Often contradicting information
    - Sometimes prereqs listed, sometimes not
  - Nearly everyone had a single “calc ready” suggested plan
- Suggestion:
  - Have a clean page that lists / links to courses, prereqs and plans.
  - Have multiple degree plans
    - Calc ready
    - No mathematical background
    - Transfer student plans (internal and external transfers)
  - These present to the students that everyone belongs

Additional Reading: Top Ten List for Making a Great “Major Requirements” Web Page
McKenna Quam

Does a fluid degree structure keep quality?
Case study: Colorado State University

Does Reducing Curricular Complexity Impact Student Success in Computer Science?

Sumukhi Ganesan, Albert Lionelle, Catherine Gill, and Carla Brodley

SIGCSE 2025, to appear. prepublication copy:
https://github.com/NeuCurricularAnalytics/papers

Northeastern
Center for Inclusive Computing

Curricular Revamp

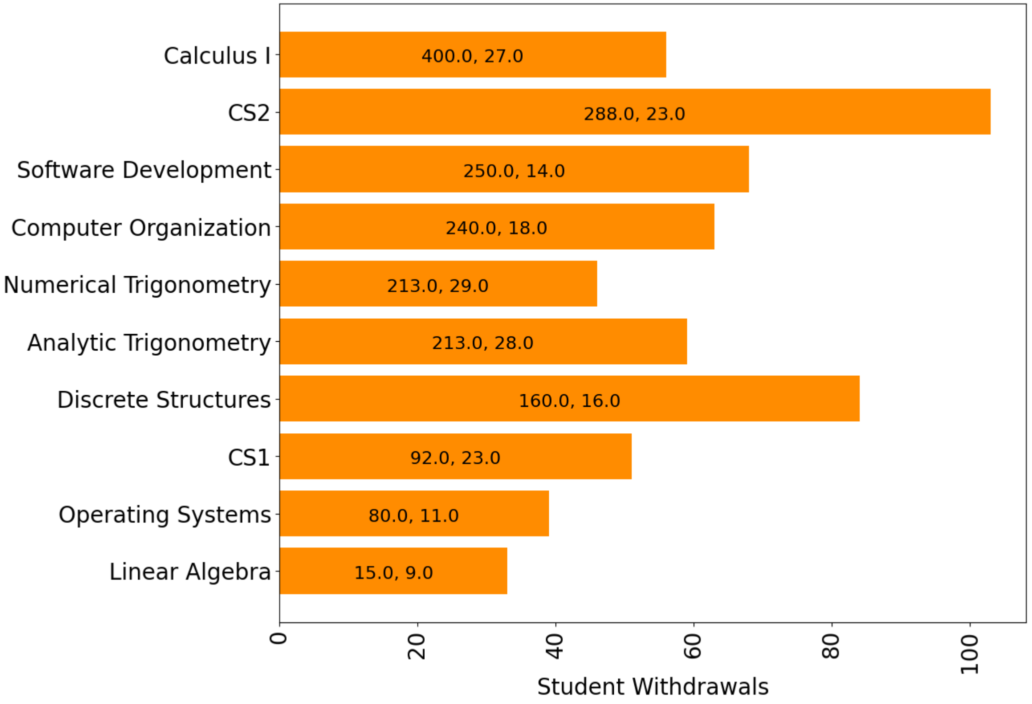
- For 20 years, only minor updates
- Discovered multiple ‘hidden’ prerequisites that
  - Drove away students
  - Made minoring in CS impossible
  - Created a very rigid structure and students were not graduating in 4 years
- Revamped curriculum in three phases
  1. Revamped prerequisites for individual courses
  2. Addressed overall degree program
  3. (in progress) Systematically updating weak points and barriers

     -- Arguably this should always be in progress

Old Curriculum Issues

- Math major barrier
  - 45% of CS majors NOT calc-ready
  - 55% of seeking CS NOT calc ready
  - Most students dropped while in math, often never taking CS
- Minor in CS 40 credits!
  - Should be 21-24 credits
  - 40 due to hidden pre-reqs
- Transfer Students
  - 3.5 years minimum
  - Created financial barrier both internal and external transfers
- Most students 4.5 or 5 years to graduate
- Overall – department was seeing minimum growth
  - And negative growth in URMs

Terminal Course(s)
The last course a student took before leaving program. Often math.



Northeastern Center for Inclusive Computing

Phase 1: Reduce Prerequisites

- Systematically
    - Approached faculty teaching courses – in small groups
    - Asked what were the *minimum* prerequisites needed
        - Focused on needed content, not “nice to have” content
        - Faculty had *many* misconceptions on what was taught where!
            - Focused on content itself, then curriculum expert pointed out which courses that content actually showed up in (especially in lower division courses)
- Redesigned Prerequisites to go from
    - “Interconnected” core of every 2xx and 3xx course to
    - Required core with three “pillars”
        - **Systems** (CS 1 → Computer Org → Upper Division Operating Systems)
        - **Software Engineering** (CS 1 → CS 2 → CS 3 → Upper Division SE)
        - **Algorithms and ML** (CS 1 → Discreet Structures → Upper Division Algorithms)
            - Calculus I, Linear Algebra, Statistics are part of this pillar
        - Data Structures technically in SE, but also acts as central course for most upper division courses

Phase 1: Reduce Prerequisites cont.

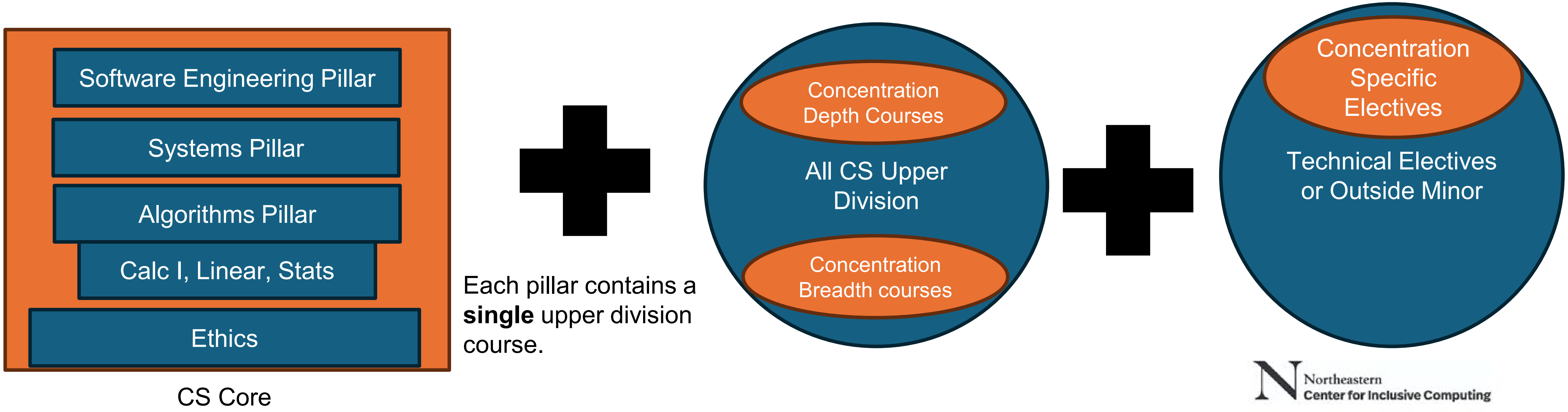
- CS majors required to take the full core
  - But we removed pre-req inter-connections among three pillars
  - Allows student progress in one pillar even if struggling in a topic in another pillar
  - Allowed ‘minors’ to be a single pillar.
- **Requirement: Keep course outcomes the <u>same</u>**
  - Put measures in place to compare performance between semesters (such as exam timing, content, etc)
- Was in place for a year before degree redesign
  - Used the year to measure outcomes and student performance
    - Students performed equally well

Phase 2: Degree Redesign

- Added more CS courses overall (13 courses → 15-18 courses)
  - Added CS 0 (optional) – As gen-ed and for majors new to coding
  - Added Ethics in CS – Also a gen-ed option
  - Added additional upper division CS electives
- Moved Calculus II as optional/part of tech elective picklist
- Reduced natural sciences (3 courses → 2 courses)
- Added Concentrations
  - All fall back into the ‘general’ concentration
  - General concentration had room for CS+x programs

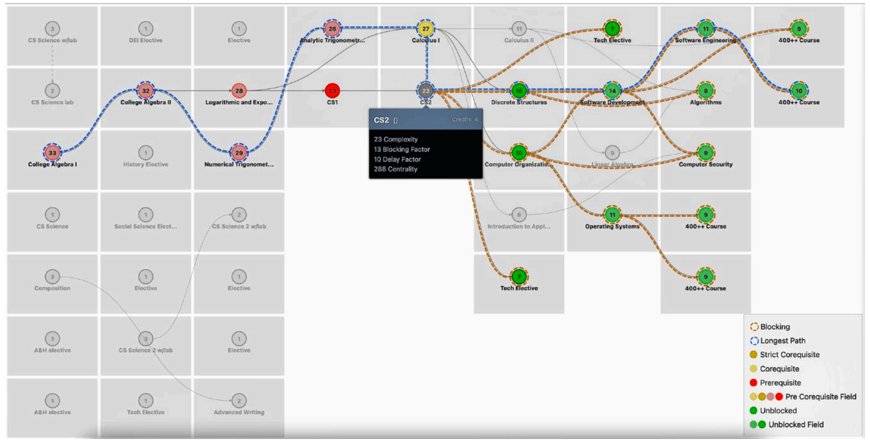
Concentration Design

- Core Requirement
- CS Upper Division
  - Specific to concentration – general concentration contains all of them
  - Most concentrations have two sub lists to pick from to ensure specific courses but with flexibility
  - Prerequisites kept to a minimum / only what is exactly needed.
- Technical Electives
  - Specific for concentration or general picklist of all of them
  - For general: could also be a minor in another field
- Concentrations added – specific to primary research areas in the department
  - General, AI/ML, Cybersecurity, Education, Human Centered Computing, Software Engineering, Systems



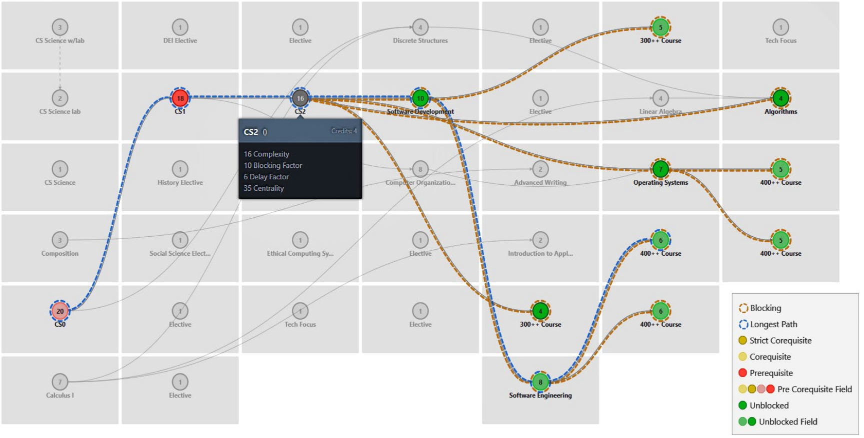
Northeastern Center for Inclusive Computing

Did it work? (As of Spring 23)

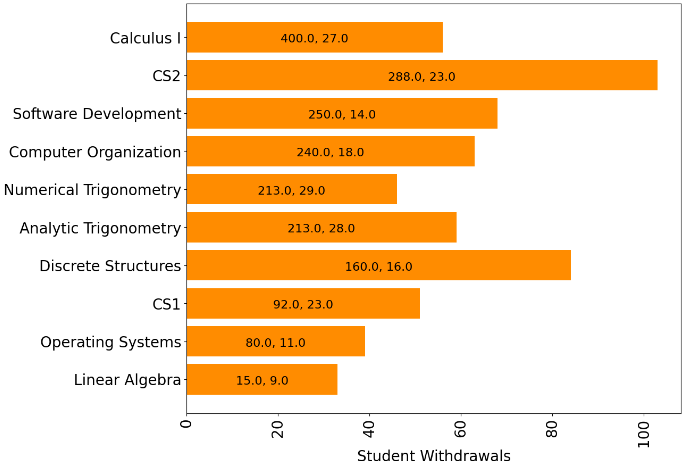


415 Complexity, multiple blocking courses

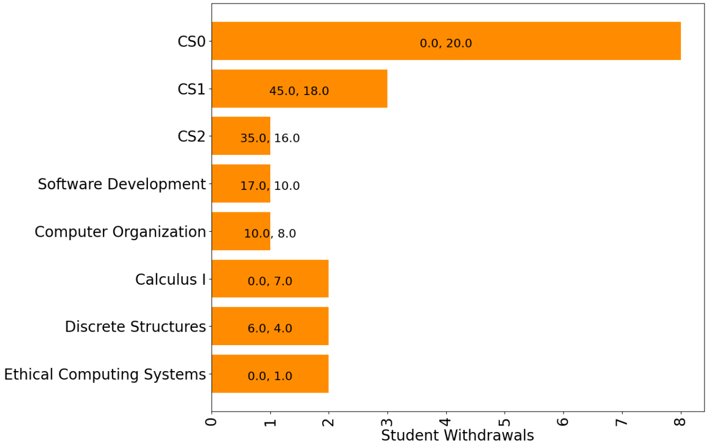
60% reduction!



164 Complexity, minimal blocking courses



Terminal Courses – Mainly Math

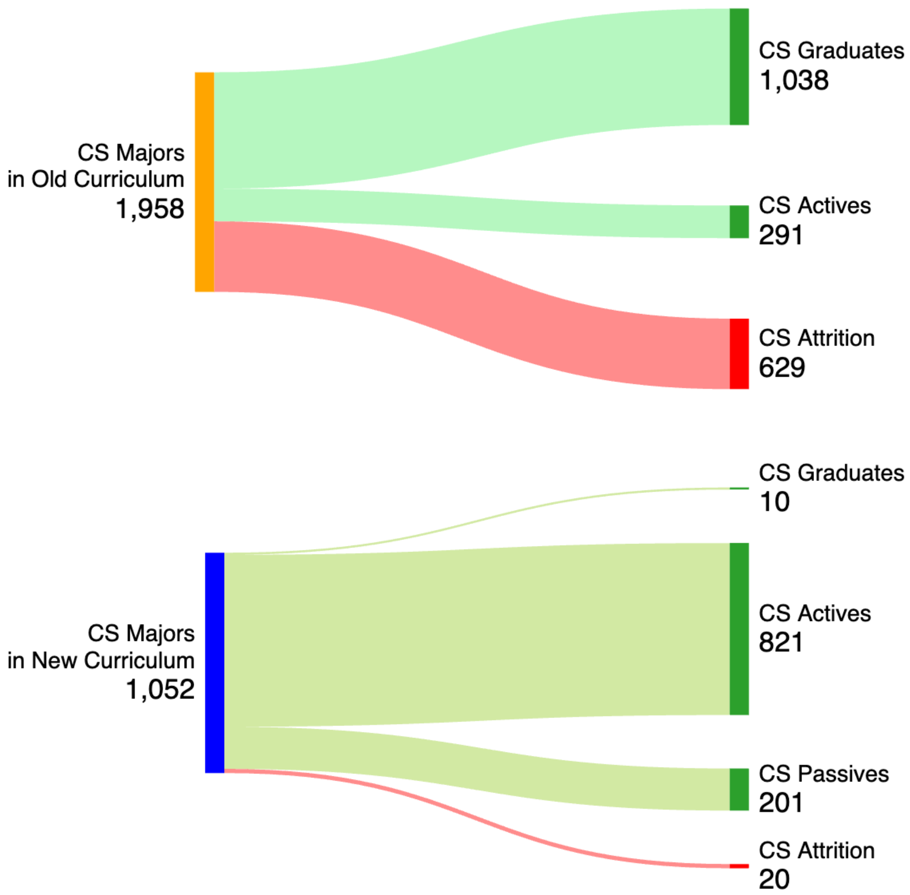


Terminal Courses – CS 0

Northeastern
Center for Inclusive Computing

Did retention/attraction increase?

- Program retention increased!
  - 67% Old degree
  - 98% New degree
- Undeclared seeking CS
  - Increased attraction!
  - 47% converted in old degree
  - 69% converted in new degree
- Overall numbers increasing
  - Fall 2015, 725,   12% women (88)
  - Fall 2023, 1078, 19% women (209)
- Students still reporting similar job placements



Northeastern
Center for Inclusive Computing

Summary/Recommendations

- A systematic review and reduction of prerequisites
  - <u>Does not reduce quality of degree content</u>
  - Gives students flexibility
  - Is attractive to students
  - Math still valuable, but is no longer an entry point
- Faculty
  - Still able to teach their preferred courses
  - Students still select courses
  - Tend to see more students in courses as program grows
- These changes are the start (Phase 3 ongoing)
  - Often snowball effects happen – meaning faculty get excited to make updates
  - Due to flexibility and picklists, possible to have A/B courses and measure differences
    - Picklists outside of minimal core allow for more dynamic changes to keep up with industry

Northeastern
Center for Inclusive Computing