A Workbook for Distinctive Computer Science Curricula:

Designing Programs Aligned with Liberal Arts Institutional and Departmental Identity

{Authors anonymized during SIGCSE Review}

Introduction

Curricular guidelines, such as the CS2023: ACM/IEEE-CS/AAAI Computer Science Curricula that is currently under development, can be a valuable resource for those creating and revising curricula. The guidelines are intended, however, to address computing curricula in computer science and closely aligned fields at a wide variety of institutions and can be overwhelming to consider. There is no "one size fits all" for computing curricula, given the variety of contexts in which computing programs operate. In particular, liberal arts curricula often leverage their unique institutional context and mission to create innovative programs. Indeed, it is not practical, and most likely not useful, to blindly consider a curricular guidelines document like CS2023 and decide to adopt or modify a program at a particular institution without a clear understanding of the program's opportunities and constraints. It is important to develop, or at least clarify, the institutional and departmental identity and goals, and view the curricular standards through that perspective.

With a focus on computing departments that operate within a liberal arts context or otherwise reflect a liberal arts perspective, this workbook is designed to help faculty work through their program's current and desired identity and goals to develop (or improve) a set of design principles and program level learning outcomes before turning to CS2023 to inform curricular re-design. Institutions that adopt a liberal arts perspective "pursue a philosophy of higher education that emphasizes preparing students for the full range of thinking they will face throughout their lives; thinking in the service of a career, thinking in order to participate in civic affairs and society generally, thinking in order to have a fulfilling personal life, etc."[1]. While the approach outlined in this workbook can be applied to all computing programs, it emphasizes priorities that tend to be most important in a liberal arts context. These include strong ties to an institutional mission, an emphasis on a broad educational experience, and the constraints imposed and opportunities provided that result in unique and innovative curricula that exist

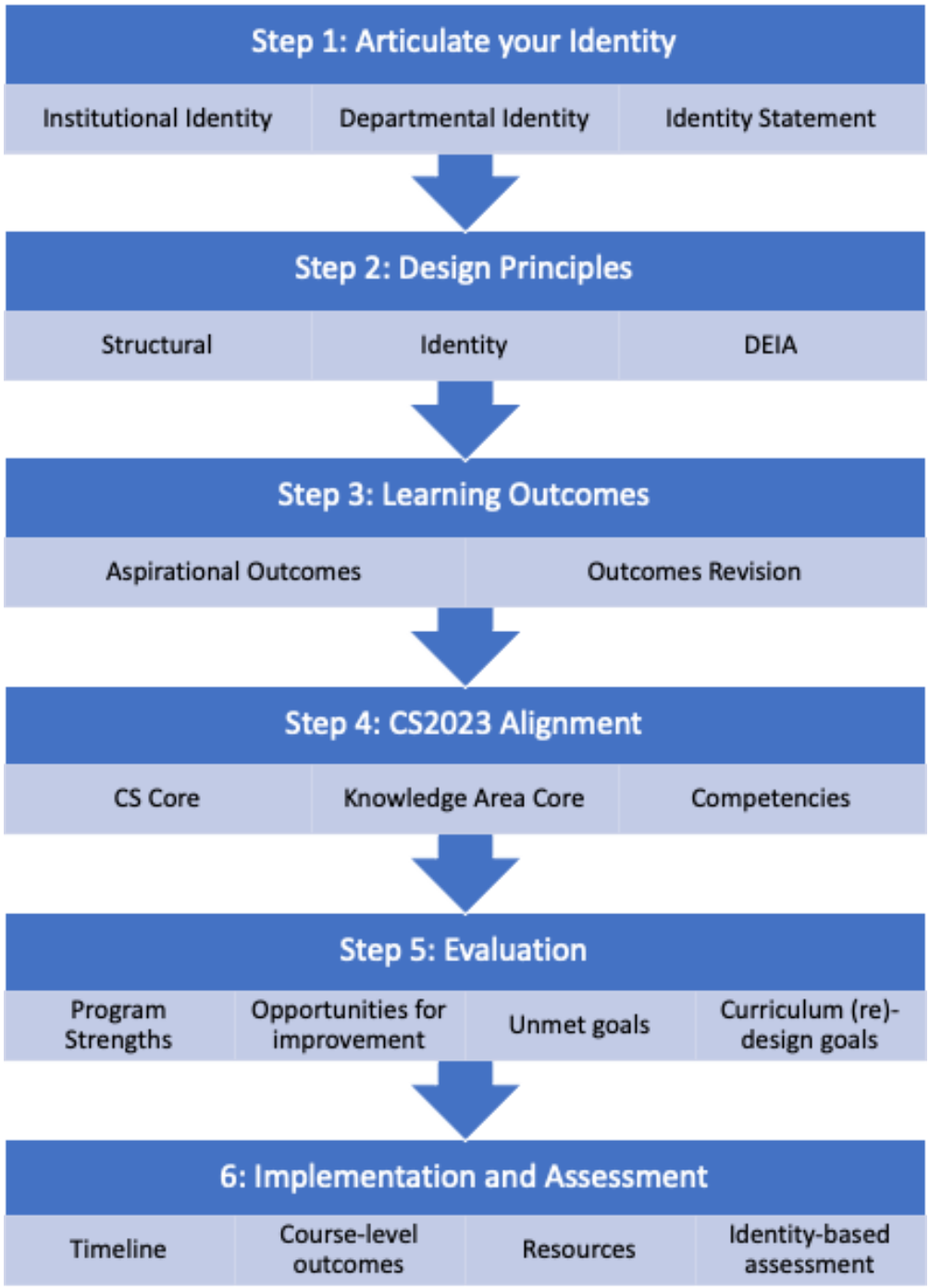
[1] Douglas Baldwin, Amanda Holland-Minkley, and Grant Braught. 2019. Report of the SIGCSE committee on computing education in liberal arts colleges. ACM Inroads 10, 2 (June 2019), 22–29. https://doi.org/10.1145/3314027

within that context. Further motivation for this process and the research behind its design is available in a companion article by the authors.[2]

Overview of the Process

This workbook separates the process of curriculum design into six steps:

[2] Amanda Holland-Minkley, Jakob Barnard, Valerie Barr, Grant Braught, Janet Davis, David Reed, Karl Schmitt, Andrea Tartaro, and James D. Teresco. 2023. Computer Science Curriculum Guidelines: A New Liberal Arts Perspective. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 617–623. https://doi.org/10.1145/3545945.3569793



Briefly, if you are using the workbook to revise an existing Computer Science curriculum, you can expect to do the following as you progress through these steps:

Step 1: Articulate your Identity

You will complete this step by articulating your identity at an institutional and program level. This will be informed by reviewing existing identity-focused documentation you

may have. Programs may wish to create or revise a Mission, Vision, and/or Core Values statement at this phase. Because the resulting Identity Statement will be used to inform all future steps in the workbook, strong departmental consensus will be important at this stage.

Step 2: Design Principles

Drawing on your identity statement from Step 1, you will state design principles that will shape the structure of your curriculum. You will focus first on identity-driven design principles that will ensure your curriculum reflects your unique programmatic identity. You will also reflect on any constraints you are operating under and related structural design-principles. This step will also ensure that you are reflecting on how your design principles will advance diversity, equity, inclusion and access efforts on your campus and in the discipline of computer science.

Step 3: Learning Outcomes

In this step you will generate a coherent set of identity-aligned program-level learning outcomes. You will be invited to reflect on or develop aspirational outcomes that would align with your identity and design principles. If you have existing program-level outcomes, you will similarly be asked to reflect on which ones should be removed or revised in order to better align with your identity and design principles.

Step 4: Interpreting and Adapting CS2023

Having reflected on your mission, identity, design principles, and program-level learning outcomes, you will now continue the top-down design process using the ACM/IEEE/AAAI Computer Science Curricula 2023..You will then use your work from Steps 1-3 to identify the areas of CS2023 that you want your curriculum to emphasize. You will then adjust and align your emphases using specific curriculum content from CS2023. You will reflect how your design deviates from CS0223 and on which deviations are well-justified by your mission, identity, design principles, and program level outcomes. and which deviations would be suitable to address in a curricular revision.

Step 5: Evaluation

Your goal in this step will be to identify your top priorities for your curriculum revision, based on an evaluation of your current state. You will make use of any institutional and departmental assessment data you have available to identify current strengths and weaknesses. You will also review your current curriculum with respect to the identity statement, design principles, and program learning outcomes you have developed and with respect to opportunities to address deviations from CS 2023 recommendations you have identified as priorities. Considering all of these evaluations, you will set three to give specific goals for your curriculum revision.

Step 6: Implementation and Assessment

With your goals for curriculum revision identified, you will now implement specific changes in your curriculum. This will include revising your curriculum map and

course-level learning outcomes, planning for assessment, and then putting your changes into practice. As you do this work, you will ensure that you are continuing to align with your identity statement and curricular design principles.

Within the workbook, each step is presented in more detail. This includes introductory material explaining its role in the overall curriculum design process followed by a series of questions designed to support you and your colleagues through information gathering, reflection, and decision-making. Steps are designed to build on each other so it is important to progress through the workbook in order. However, we provide guidance below on how different programs may wish to spend more or less time on different elements, depending on your priorities.

How to Use the Workbook

Just as a curricular guidelines document is not one size fits all, this workbook can be used in different ways. Some departments might have an outdated curriculum, developed by a group of faculty who are no longer there, or an outdated focus based on the origins of the program in either engineering or mathematics. Such departments might spend significant time on the first few steps to get a good understanding of the program's current identity and goals, articulating what the faculty would like their identity and goals to be (if different), and how current and future departmental identity and goals align with institutional identity and goals. Other departments might be coming off of a periodic program review and have just worked through many of the steps, and would be prepared to start looking at the final steps almost immediately. Some may be looking to make more incremental changes to their curriculum, while others may be designing a curriculum for a new program.

These different goals will affect how you use the workbook. A good way to start is to read through the entire workbook, and decide which parts would be most beneficial in the short term. In some cases, the process could be fairly quick, while in others it could require months or even a few years to work through. For a comprehensive program revision, it might not be desirable or possible to work through the entire process in a short period of time and starting with setting semesterly or yearly goals could be helpful. Programs just wishing to do a periodic review of their curriculum or who come to the process with documentation from previous curriculum design work may be able to work through the process more efficiently.

The workbook could be used as a framework for at least part of required periodic institutional program reviews for departments in institutions that require them, or as a standalone exercise to inform departmental planning. Even if not at the point where such a review is due, working through this process can make the next such review more straightforward. Departments planning to hire or wanting to make the case for additional faculty may find the process useful for identifying areas in most need of coverage and providing supporting material for a position request.[3]

---

[3] James D. Teresco, Andrea Tartaro, Amanda Holland-Minkley, Grant Braught, Jakob Barnard, and Douglas Baldwin. 2022. CS Curricular Innovations with a Liberal Arts Philosophy. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (Providence, RI, USA) (SIGCSE 2022).

A subset of faculty could be designated to lead the primary work for this process, but it is important to collect feedback and build consensus among the entire department at each step. It is also important to seek buy-in and support from other stakeholders on campus. Points in the process where this is especially important are noted throughout this document.

Prerequisites

Users of this workbook should have familiarity with the process of developing curriculum at least at the course level. This includes an understanding of developing learning goals, assessments, and activities. We recommend L. Dee Fink's *Self-Directed Guide to Creating Significant Learning Experiences* [4] for an overview of these concepts.

Licensing:

Association for Computing Machinery, New York, NY, USA, 537–543. https://doi.org/10.1145/3478431.3499329

[4] L. Dee Fink. 2013. Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses. John Wiley & Sons, Hoboken, NJ, USA. Google-Books-ID: cehvAAAAQBAJ. https://www.bu.edu/sph/files/2014/03/www.deefinkandassociates.com_GuidetoCourseDesignAug05.pdf

Workbook

Step One: Articulating Your Identity

As a first step you will reflect on the unique identity of your program as a computer science program within the liberal arts. This will include institutional elements as well as elements defined by your department and its faculty. This step will have you collect relevant information about your identity and help you come to a consensus statement of your view of the liberal arts and your identity as a liberal arts computer science program.

Institutional Level

Identify and review any of the following that exist at your institution, considering the institution as a whole, Academic Affairs, and/or the College/Division in which you are located. Institutional identity provides important context for articulating or developing your program identity in the later questions of this section. Although some of the documents listed below might be outdated or rarely used within your institution, others may provide valuable insights about your institutional identity. Similar items that are department/program-specific will be considered in the next step. Suggested documents:

- Mission Statement
- Vision Statement
- Values Statement
- Statement on Diversity, Inclusion, Equity, and Access (DEIA)
- Statement of Institutional Student Learning Outcomes
- Honor Code or Academic Integrity Policy
- General education, graduation requirements, and educational goals and outcomes: consider how both catalog language and internal documents articulate their purpose
- Strategic Plan Documentation
- Fundraising/Capital Campaign Objectives
- Admissions/Marketing Materials
- Faculty Review Criteria

1.1. Identify the resources that best indicate your institution's particular vision for or interpretation of liberal arts education. Optionally quote or list those resources in the space below.

1.2. Summarize your institution's particular vision for or interpretation of liberal arts education. Be specific about what is most important or distinctive.

Program or Department Level

Identify any of the following that exist for your department or program:

- Mission Statement
- Vision Statement
- Core Values Statement
- Diversity, Inclusion, Equity, and Access (DEIA) Statement
- Current programmatic student learning outcomes
- Departmental Strategic Plan
- Departmental External Review Documentation
- Catalog Description of Program
- Admissions and Marketing Description of Program
- Faculty Review Criteria
- Recent position descriptions/job ads
- Strengths and interests of current faculty
- Processes and practices with respect to DEIA

If you do not have existing Mission, Vision, and/or Core Values statements, consider creating these before proceeding. A visioning toolkit is available online at: https://atctools.org/toolkit_tool/visioning-toolkit/

If you do not have existing programmatic student learning outcomes, you can create them as part of a curriculum revision process as you progress through this workbook (See Step 3).

1.3. Identify the resources that best indicate your program's particular vision for, or interpretation of, computer science education. Optionally quote or list those resources in the space below.

1.4. Summarize your program's particular vision for or interpretation of computer science education. Describe carefully how it connects to or reflects its liberal arts, institutional, and departmental contexts.

Identity Statement

1.5. Compare your institution's vision and interpretation of the liberal arts (1.2) with your program's vision and interpretation of computer science (1.4). Optionally note any synergies or disconnects that you may want to address in the next step.

1.6. Putting together your institution's vision and interpretation of the liberal arts (1.2) and your program's vision and interpretation of computer science (1.4), develop a unified statement of identity for your program *as a liberal arts computer science program*. This may take the form of new or revised mission, vision, and/or core values statements for your program.

In the following steps, this statement will be used to articulate your curricular values and goals and drive decision making, so you should have strong departmental consensus around this statement and may wish to obtain administrative buy-in as well.

Step Two: Stating Your Design Principles

After articulating the mission for your program, identify design principles that will shape the structure of your curriculum. Here we draw on what Fink[5] calls "Situational Factors" in Step One of his process (identifying the teaching/learning context, nature of the material, and characteristics of the students and instructors; refer to the questions on page 7 for additional guidance). We also draw on our own disciplinary background in requirements analysis and the design process, applied now to curriculum rather than to software. We again emphasize an approach that considers your institutional and department identity. Below we divide these principles into two categories: identity-based principles focused on the identity you described in Step 1, and structural design principles based on institutional constraints. We note that there is some overlap between identity-driven and structural design principles. For example, your identity may motivate a desire to contribute broadly to institutional general education requirements, or you may have a structural need to do so either because it is required or a

[5] L. Dee Fink. 2013. Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses. John Wiley & Sons, Hoboken, NJ, USA. Google-Books-ID: cehvAAAAQBAJ. https://www.bu.edu/sph/files/2014/03/www.deefinkandassociates.com_GuidetoCourseDesignAug05.pdf

necessity for enrollment. We have included a number of examples of identity and structural design principles in Appendix A, but do not feel limited by these lists. We then draw your attention to reflecting how DEIA is represented in your design principles because of the importance of providing education that is inclusive, equitable, and accessible by a diverse population.

Identity-Driven Design Principles

Review your identity statement from Step 1.3. Ask yourself what structure or broad elements for a curriculum would be most important to ensure the student experience is strongly aligned with your identity.

For example, if your institution has a mission of recruiting and educating first-generation students, and your department similarly focuses on supporting this student population, you should have one or more design principles related to this goal. Depending on how well developed this portion of your identity is, it might be as simple as “supports the needs of first generation undergraduate students.” Or you may articulate your understanding of those needs within your student population in more detail, perhaps as relates to entry points into your major, flexibility, and mentoring.

While doing this, be careful not to describe what your curriculum *already does*. Instead, reflect on aspirational design principles driven by your identity. Subsequent steps will allow you to take into account your current state and operational constraints.

2.1. What are your identity-driven design principles?

Structural Design Principles

In addition to the design principles that come out of your identity, it is important to understand the practical requirements and constraints that your program operates under. As described above, these may come from institutional requirements on majors or be shaped by department resources. Consider if your structural design principles can be phrased in neutral terms or as opportunities, rather than as limitations.

As you proceed with this step, think carefully about any requirements or constraints that might come out of department tradition or history but could now be revisited. For example, while a three-person department is unlikely to become a ten-person department, this is a good opportunity to consider whether you must offer a curriculum that can be staffed by no more than

three full-time faculty (which perhaps you do), or whether your institution would consider adding staffing in order to support an updated curriculum.

If you haven't done this recently, you should also confirm that you have an up-to-date understanding of institutional requirements for majors. You may want to review guidance or forms that your curriculum committee or academic affairs require when revising an academic program.

As you complete this step, be careful to be complete but not over-constrain your curriculum design.

2.2. Optionally list the historical constraints on your program design. Identify any constraints that could change or have already changed.

2.3. What structural design principles will inform your program design going forward?

DEIA Design Principles

Diversity, Inclusion, Equity, and Access are no longer optional considerations in curricular design. The computer science community has broadly adopted a commitment to providing equitable and accessible education for all students. This is reflected in the ACM/IEEE Computing Curricula 2020 (CC2020) Report as part of its vision for curricular guidelines in all areas of computing for the coming decade. In that report's introduction, they say:

> *"One underlying theme of the Report is the development of computing talent from all sectors and groups in our society. A lack of diversity limits creativity and productivity. It excludes many potentially qualified individuals and can be a significant concern for many employers. For example, the underrepresentation of women in computing in some countries has received much attention [Reg1]. This Report recognizes the importance of diversity and recommends that academic computing departments promote best practices to attract and retain greater student diversity."*[6]

[6] https://dl.acm.org/doi/book/10.1145/3467967

This commitment is also reflected in planned (academic year 2023-2024) changes to ABET/CSAB criterion, particularly with regard to Criterion 8, Institutional Support, and Criterion 5, Curriculum. The former will address creation and fostering of "a respectful environment among the program's students, faculty, staff, and administrators in which the student outcomes can be attained" while the latter has a new Outcome listing "Principles and practices for developing inclusive and accessible computing solutions that equitably address the needs of diverse user communities".

Reflect on your institutional context, including the particular DEIA challenges you face, goals that have been set, or initiatives that are underway. Think additionally about your program context and the ways it is similar to or different from your institutional context when it comes to DEIA. If you are new to this work, consider seeking out support from an Office or Director of DEIA on your campus. NCWIT provides helpful resources for higher education at https://ncwit.org/higher-ed/, while CRA and NSF also have support and resources through the BPCNet Resource Portal, including resources to help develop a departmental BPC plan. Another helpful resource is the Alliance for Identity Inclusive Computing Education (AiiCE).

Review your identity-driven and structural design principles and ask yourself if they encompass principles that will help you work towards greater diversity, inclusion, equity, and accessibility within your program and its curriculum. Consider if there are additional design principles you want to add to support this work.

2.4. Which of your design principles (2.1 and 2.3) will advance diversity, equity and inclusion efforts on your campus and in the discipline of computer science?

2.5. Do your design principles sufficiently address your DEIA goals? If not, consider revising your design principles or even your identity statement (1.6).

Step Three: Articulate Program-Level Learning Outcomes

Programmatic student learning outcomes actualize the vision set out by your identity and your design principles by describing what particular computer science knowledge and skills your

program considers most essential. A useful resource defining student learning outcomes is available at: https://www.colorado.edu/oda/sites/default/files/attached-files/program_learning_outcomes_v2.pdf. You can also see some sample program-level learning outcomes in Appendix D.

If you already have program-level student learning outcomes, you can use this step to review and possibly revise them. If you do not yet have program-level student learning outcomes, you should plan on spending significant time on this step. You may wish to consult with appropriate offices at your institution (e.g. a Center for Teaching and Learning, a Director of Assessment, a Curriculum Committee, or Academic Affairs) about any institutional guidelines about program outcomes since these are often also used for institution-wide accreditation review.

Aspirational Program-Level Outcomes

Use your work from the prior steps to identify program level outcomes that would align well with your identity statement (1.3) and design principles (2.1, 2.2, and 2.3). What does your identity tell you about the characteristics you want to see in your graduates? What outcomes align well with your design principles. If you already have program level outcomes, you'll turn to those in the next step. Here, your goal is to reflect on what new or different outcomes you might set if you were designing a program from scratch based on your identity.

3.1. Based on your reflections, what program-level student learning outcomes would align with your identity and design principles?

Program-Level Outcome Revision

Now, if you have existing program level outcomes, compare those to your list in 3.1. Are there any existing outcomes you would like to remove, replace, or revise? Are there new outcomes you would like to add? If you do not have existing program level outcomes, use this opportunity to reflect on what else might be missing. In both cases. iterate over your outcomes until you reach consensus within your department.

3.2. Give your new/revised set of program-level student learning outcomes.

Step Four: Interpreting and Adapting CS2023

The ACM/IEEE/AAAI Computer Science Curricula 2023 curricular guidelines (CS2023) provide important recommendations for the design and content of computer science major programs. The CS2023 recommendations in part aim to define what "*every* Computer Science graduate **must** know"[7] and what "programs must cover." However, they also allow for some flexibility by acknowledging "that often, the design of curricula in smaller programs is dictated by curricular emphasis based on regional needs, the local availability of instructional expertise, and historical evolution of computer science programs." Or rephrasing, liberal arts programs often have curricular emphases that are influenced by institutional and departmental missions and identities, design principles (identity-driven, structural, DEIA) and program-level aspirational learning goals.

In this step you will use a spreadsheet tool to continue (re)designing your curriculum by interpreting and adapting CS2023 in light of your outputs from steps 1-3.

CS2023 Structure and Terminology

CS2023 presents a "combined knowledge model and competency model (CKC)." The ***knowledge model*** is content focused, identifying essential and recommended topics that graduates will know. The ***competency model*** is task focused, identifying the types of tasks that graduates are able to complete. The Knowledge Model "facilitates the process of designing courses and curricula," while the Competency Model "is meant to be used to generate tasks," help "learners make associations among complementary concepts," and facilitate "evaluation of student learning."

Programs will have a ***competency area*** (or areas), that expresses the focus and the particular strengths of the program. A program may adopt one of the broad exemplar competency areas defined by CS2023, which include software, systems, or applications. Or as this step facilitates, they "may choose to design their [own] competency area(s) based on institutional mission and local needs" (i.e. your outputs from steps 1-3). As intended by CS2023, a programs' competency area is used with the knowledge model to guide the selection of content that is included or emphasized at the curriculum and course level. Then later the competency area is used with the competency model to generate specific assessments and facilitate evaluation, though that is currently beyond the scope of this workbook.

The CS2023 knowledge model divides the body of computer science into 17 **knowledge areas (KAs)**. The KAs focus on foundational knowledge (e.g. Mathematical and Statistical Foundations, Software Development Fundamentals), specific sub-area of computer science (e.g. Artificial Intelligence, Networking and Communications, Operatings Systems) or

[7] All quoted material in this section is from the Computer Science Curricula 2023, Version Gamma, August 2023. It will be updated and revised as appropriate when future versions are released.

cross-cutting themes that runs broadly across the field (e.g. Society, Ethics and Professionalism). Each knowledge area is subdivided into **knowledge units (KUs)**. The KUs subdivide the content of the knowledge areas. For example, the Algorithmic Foundations KA is subdivided into 5 KUs: Foundational Data Structures and Algorithms, Algorithmic Strategies, Complexity Analysis, Computational Models and Formal Languages, and Algorithms and Society. For each KU, CS2023 provides a recommendation for the number of ***instructional hours*** (a.k.a. core hours) to be "spent in the classroom imparting knowledge [associated with the KU]." Finally, a set of more specific ***topics*** are enumerated for each of the KUs. For example, the Foundational Data Structures and Algorithms KU contains topics that include arrays, linked lists, stacks, queues, heaps, and a range of searching, sorting and graph algorithms.

CS2023 then organizes the topics in each KU into three categories, CS Core, KA Core and Non-Core. The ***CS Core*** for a given KU is the collection topics in that KU that CS2023 requires "*every* Computer Science graduate **must** know." The ***KA Core*** and ***Non-Core*** categories then include topics that are recommended for curricula that emphasize the associated KU (i.e. that KU is within its competency area). For example, the Foundational Data Structures and Algorithms KU places topics for most standard sorting algorithms in the CS Core category, indicating that they are important for all students. However, pseudo-O(n) sorting algorithms such as radix, counting and bucket sorts appear in the KA Core category, and parallel and quantum computing algorithms appear in the Non-Core category. The placement of these topics in the KA Core and Non-Core categories indicates that they should be considered for inclusion in programs that place a respectively higher emphasis on the KU.

The CS2023 Design Tool

The spreadsheet tool introduced in this step is an aid to top-down curriculum-level (re)design using CS2023. You will use this tool with your outputs from steps 1-3 to indicate the emphasis that you wish for your curriculum to place on each of the CS2023 knowledge areas (KAs). Based on these emphases you will draft a statement of your program's competency area(s). Given the size of your curriculum and using the KA emphases you specified the tool will generate a rough estimate of the amount of instructional time to allocate to each of the KAs (and KUs within them) that reflects your emphases. You will then use your design principles (step 2) and learning goals (step 3) to guide the selection (and omission) of specific CS Core and KA Core Knowledge Units (KUs) that support your program's competency area(s), while remaining consistent with the estimated time constraints of your curriculum. The end result of this step is a statement of your curriculum's competency area(s) and a set of time allocations for each of the CS2023 KAs and KUs that align with the program's mission, design constraints, learning goals and its competency area(s).

A few high-level notes about the tool will be helpful to keep in mind throughout its use.

- The spreadsheet is an aid to help you be intentional about designing a curriculum that interprets and adapts CS2023. It is designed to call attention to areas for closer

inspection, but it is intended to neither generate a curriculum for you nor to be prescriptive about time allocations. It should be used flexibly in conjunction with the outputs of steps 1-3 to guide decisions. While the following tasks take one path through the use of the tool, it can be used in any way that is helpful in thinking about your curriculum.

- Using the tool will very likely be an iterative process. You may specify your KA emphases, then find that the estimated time available for a KA is insufficient for the CS Core and KA Core KUs you wish to include. As a result you may revisit your KA emphases to reallocate time, make different tradeoffs regarding the CS Core and KA Core KUs that are included, or make other smaller adjustments that are supported by the tool.
- The estimates of instructional hours that are generated and any areas highlighted by the tool should be interpreted as guides, not as prescriptive targets. They should serve as checks that the allocation of time in your curriculum *approximates* the relative emphasis that you wish to place on the KA. To this end, the tool provides mechanisms for you to make intentional decisions about your deviations from the estimates.

Getting the CS2023 Design Tool

The CS2023 design tool is a collection of Google Sheets spreadsheets. Make a copy of these sheets to your a Google drive space where you have write permission so that you will be able to edit them:

https://docs.google.com/spreadsheets/d/1mP23TWUiiXw_isimifhu72Z88d_6HGpKwoAPCWpgmPQ/edit?usp=sharing

Identifying Competency Area(s) via KA Emphases

The “KA Emphasis” sheet lists the 16 CS2023 KAs (column B) with their abbreviations (column A). Columns D and E provide different metrics for comparing the emphasis that CS2023 places on each of the KAs. The “Percent of CS2023 CS Core” shows the percentage of the instructional hours in the CS Core that are made up from each KA. For example, 20.45% of the instructional hours in the CS Core are in MSF. The “CS2023 Relative Emphasis” is a unitless expression of the relative emphasis that the CS Core places on the KA. For example, based on the instructional hours recommended by CS2023, MSF is emphasized almost twice as much as AL and five times as much as AI[8]. The “shape” of the CS2023 CS Core based on these relative emphases is illustrated by the black dotted series in the radar chart to the right.

Columns G-Q allow you to specify the relative emphasis that you would like each of the KA to receive in your curriculum. As a default starting point all KA’s are given equal emphasis. Note

[8] This is a bit of a simplification. When a KA in CS2023 includes topics that relate to more than one KA those topics can appear in both. However, their instructional hours are only counted in one. For example, if MSF and AL (or AI) have a number of topics in common, CS2023 only counts the associated instructional hours in one KA or the other.

that Google Sheets does not provide a radio button widget. Thus, only one checkbox should be checked for each KA. If multiple checkboxes are selected, the leftmost one will be used.

The “My Relative Emphasis” (column S) shows the relative emphasis of each KA with respect to the others based on the checkboxes that have been selected for the KAs. The “Percent of My Curriculum” (column T) estimates the percentage of instructional hours that should be allocated to each KA to approximate the expressed emphases. The shape of your curriculum is shown by the solid red series in the radar chart.

4.1. Use the checkboxes on the KA Emphasis sheet to express the relative emphasis (or equivalently the % of instructional hours) that your curriculum will place on each KA. Try to consider only content that will be required of all students.

You may find it helpful to review the “Preamble” to each KA as published in the CS2023 Recommendations to better understand what content each KA contains. The CS2023 Recommendations can be found here:

https://csed.acm.org/

Paste a copy of the radar chart showing the shape of your curriculum as compared to the CS2023 CS Core here.

4.2. Reflect on the differences between the “shape” of your curriculum and the shape of the CS2023 CS Core. Note the KAs where your curriculum has a significantly higher or lower emphasis than the CS Core and briefly justify these differences in terms of the outputs from:

- **Step 1 - Institutional and departmental mission and identity.**
- **Step 2 - Design principles (identity-drive, structural, DEIA).**
- **Step 3 - Learning outcomes.**

4.3. Draft a name and description for your curriculum’s competency area(s) based on your responses to 4.1 and 4.2. Note that CS2023 names competency areas of software, systems and applications and gives “computing for the social good”, “scientific computing”, and “secure computing” as examples of competency areas that a program might define “based on institutional mission and local needs.”

Selecting CS2023 Content

The “KA/KU Hours” worksheet assists in choosing CS2023 CS Core and KA Core content to be included in (or excluded from) your required curriculum while ensuring that the selected content:

A. aligns with the relative emphases expressed on your “KA Emphases” worksheet and thus supports the competency area of your program.
B. fits within the time constraints of your curriculum.

The “KA/KU Hours” worksheet is more complex and will take more effort to understand and use than the “KA Emphasis” sheet. To help, the following subsections describe the sheet and walk you through a few tasks that illustrate its use. In addition, hovering over many of the cells in the worksheet will display a note containing notes explaining their purpose.

As you work through the remainder of this step keep in mind that the use of this tool is an iterative process. The overall goal of this process is to align the KA emphases that you have expressed with the CS content that you would like to be included in your curriculum. So tThe initial estimates and choices that you make can be rough as they will likely need to be revisited and refined multiple times as you add more information. So, try not to get bogged down, at least initially, trying to resolve small differences in hours as they are likely to change multiple times as you iterate.

Estimating the Total Instructional Hours required of all Majors

The CS2023 CS Core KUs plus the KA Core KUs that a program includes in its competency area are intended to be required of all students in the program. Thus the number of instructional hours that are required to be taken by all students in a program are the hours that are available to cover this content. Cells E4-E6 in the “Instructional Hours Summary” section of the sheet are used to estimate the total number of instructional hours that are required of all majors.

4.4. Estimate the number of courses that are (or will be) required of all students who complete the major. Use fractional courses to:

- **exclude parts of courses that are not dedicated to CS Core + KA Core content (e.g. content not specifice by CS2023, elective content, exam periods, review sessions, etc).**
- **include CS Core + KA Core content covered outside of the standard course meetings (e.g. required laboratory or recitation sections).**
- **include CS Core + KA Core content shared across a set of required courses (e.g. students take either CS250 or CS260 to see the content).**

Document how you made your estimation here and enter the resulting value cell E4 on the KA/KU Hours sheet.

4.5. Estimate the average number of instructional hours in a course. Note that CS2023 defines this to be 40 hours. However, a typical course that meets for 50 minutes, 3 times per week for 15 weeks will contain a maximum of 37.5 hours of in class instructional time.

Document how you made your estimation here and enter the resulting value cell E5 on the KA/KU Hours sheet.

The values that you entered in E4 and E5 are used compute the "Total hours required of all majors" (E6 = E4 * E5). The final two cells (E7, E8) in this section will be described later.

Allocated Hours vs Included Hours

The "KA Hours" sheet uses the notions of allocated hours and included hours to help align the desired emphases of a curriculum with the content that it contains. This section explains these terms and where they appear in the "KA/KU Hours" worksheet.

Allocated Hours

Allocated hours indicate the hours that will actually be available for classroom instruction in your program when it is delivered to students.

The "**Instructional Hour Allocations"** section (columns Q-U[9]) of the "KA Hours" sheet is used to generate an estimate of the number of hours that your program should allocate to each KA.

- The "Estimated Allocation" (column Q) gives an automatic estimate by distributing (i.e. allocating) your estimated "Total hours required of all majors" (E6) across the KAs based on the relative emphases that you assigned to each of the KAs on the "KA Emphases" sheet.[10]

[9] Not all column letters in this range are visible. Hidden columns in this range are used for the computation of intermediate values and information used for cell highlighting.

[10] The hour estimates for the KUs within each KA are initially distributed among the KUs in the same ratios as they are in the CS Core (column E). However, this distribution will adapt as you begin to adjust the content that is included in your curriculum.

- "Allocation Adjustment(s)" (column T) allows you to fine tune the allocations by subtracting hours from one KU and adding them to another. The "Hours Available for Manual Adjustments" (T7) indicates any difference between hours that have been subtracted and hours that have been added. Negative values here indicate that more hours have been allocated than are required for all majors (E6). Positive values indicate hours that can be added to KUs without exceeding the total required hours.
- **The "My Allocation" (column U) gives your refined estimate of the hours to be allocated to each KA (and KU) and should reflect the desired emphases of the KAs within the curriculum**.

Included Hours

Included hours indicate the number of instructional hours that CS2023 recommends for teaching the KUs that you indicate will be included in your program.

The "**Included CS2023 Instructional Hours"** section (columns E-N[11]) of the "KA Hours" sheet allows you to select and adapt the CS2023 CS Core and KA Core KUs that will be included in your program.

- The "CS2023 CS Core" values (column E) indicate the number of instructional hours that CS2023 requires for all students for each KA (and KU).
- "CS Core Adjustments" (column G) can be entered to indicate that you plan to increase or decrease the number of hours spent on the CS Core topics in a KU. You should refer to the CS2023 Recommendations for the specific topics that are included in each CS Core KU. **Emphasizing or deemphasizing CS Core KUs is one mechanism by which your program can tailor the CS2023 content to its competency area(s).**[12]
- The "CS2023 KA Core" values (column I) indicate the number of instructional hours that CS2023 recommends for teaching the topics in a KA Core KU if they are included in a program.
- The checkboxes under "Include KA Core KU" (column J) allow you to indicate if a KA Core KU will be included in your program. **Selecting KA Core KUs to include is the primary mechanism by which your program tailors the CS2023 content to its competency area(s).**
- "KA Core Adjustments" (column K) can be entered to indicate that you plan to increase or decrease the number of hours spent on an included KA Core topics in a KU. You should refer to the CS2023 Recommendations for the specific topics that are included in each KA Core KU.
- The "Included CS+KA Core" (column N) sums the "CS2023 CS Core" hours plus or minus any "CS Core Adjustments" and the selected "CS2023 KA Core" hours plus or minus any "KA Core Adjustments." **Thus, the "Included CS+KA Core" for a KA (or KU) indicates the time that will be required to teach the included content.**

---

[11] See previous footnote.

[12] While CS2023 requires that programs ensure that "*every* Computer Science graduate **must** know" the topics in the CS Core, we advocate a more flexible approach where CS Core topics may be emphasized or deemphasized based on program context including mission and identity, design principles, learning goal and competency area(s). That said, the specific topics contained in a CS Core KU should be examined closely before adjusting these hours.

Aligning Included Hours and Allocated Hours

Much of the work that you do with the “KA/KU Hours” sheet will be in selecting (and adjusting) the CS Core and KA Core KUs to be included in the curriculum and adjusting the KA emphases (or tweaking allocated hours) such that the included hours and the allocated hours align.

When the included hours for a KA align with the hours that are allocated to it (within some epsilon[13]) it suggests that the amount of CS2023 content that is included aligns with the desired emphasis of the KA. However, if the included hours are less than the allocated hours for a KA, it suggests that either too little content has been included or that the desired emphasis of KA should be decreased. Conversely, the included hours exceeding the allocated hours is indicative that either the emphasis of the KA should be increased or that content should be removed.

Ultimately, when the included and allocated hours match for all of the KAs it suggests that the curriculum aligns with its defined competency area(s).[14]

Cell Highlighting

The “KA/KU Hours” sheet uses cell highlighting to call attention to mismatches between included and allocated hours.

The “Hour Tolerances” cells (N3-N7) specify the percent differences that are indicated by each highlight color that is used. For example, a cell highlighted yellow will indicate that its value is at least 20% more than the cell to which it is being compared.

These highlights are applied to the following cells:

- “My Allocation” (e.g. U12 for MSF) is compared to the “Included CS+KA Core” hours (e.g. N12 for MSF).
- “Included CS+KA Core” (e.g. N12 for MSF) is compared to the “My Allocation” (e.g. U12 for MSF).
- “CS2023 CS Core” (e.g. E12 for MSF) is compared to the “My Allocation” (e.g. U12 for MSF).
- “Total hours included in CS+KA Core” (E7) is compared to the “Total hours required of all Majors” (E6).
- “Total hours in My Allocation” (E8) is compared to the “Total hours required of all majors” (E6).

Note that highlighted cells do not necessarily imply an issue. Rather they are intended to draw attention to areas worthy of closer inspection. For example, cells that are highlighted may indicate that:

---

[13] The hours allocated and included here are not intended to be hard targets, though with enough diligence they could become so. Instead, they should typically be used as a coherence check on the curriculum design to identify and address areas where there are significant mismatches between emphasis and content.

[14] See previous footnote.

- additional or fewer CS Core or KA Core hours should be included to support the desired emphasis for a KA.
- the emphasis for a KA should be increased or decreased (on the “KA Emphases” sheet) or an “Allocation Adjustment” (column T) should be made to align with the CS Core and KA Core hours that are included.
- a difference exists but is justifiable based on mission and identity, design principles, learning goals and/or competency area(s) and thus is acceptable. In these cases Notes (column W) may be added to document the justification.

Designing Your Curriculum

Use the “KA/KU Hours” worksheet to specify the CS Core and KA Core content that you wish to include in your curriculum. Refer to the CS2023 Recommendations for additional details about the specific topics that are included in the CS Core and KA Core for each KA. The CS2023 Recommendations can be found here:

https://csed.acm.org/

Use the cell highlighting to identify and address mismatches between the emphasis that you have placed on KAs and the CS Core and KA Core content that you have included.

4.6. Identify any KAs for which the “CS2023 CS Core” (column E) has a red highlight. For each KA identified, use your outputs from steps 1-3 and your program’s competency area(s) to document the rationale for allocating fewer instructional hours to the KA than are required by the CS2023 CS Core.

4.7. Identify any KAs for which the “Included CS+KA Core” (column N) is mismatched with “My Allocation” (column U). For each KA identified, use your outputs from steps 1-3 and your program’s competency area(s) to document the rationale for the differences.

Step Five: Evaluating Your Current State

This step provides the opportunity for you to use existing data to determine if a curriculum revision is needed and, if so, set goals for that revision. At a minimum, you should identify areas of strength and areas for improvement with respect to your identity (Step 1), design principles (Step 2), programmatic outcomes (Step 3), and the CS2023 curricula (Step 4). You will also use existing departmental and institutional data to guide your revisions, or collect data that will help you identify strengths and weaknesses. Different programs may spend more or less time on this section depending on how much data is available and personal preference for focusing on the data. In particular, too many data streams may be overwhelming to those new to disciplinary-level curriculum development and revision.

Start by discussing how you are achieving the identity, design principles, and programmatic outcomes identified in steps 1-3:

- How does your current curriculum reflect the identity statement you described in 1.3?
- How does your current curriculum address the design principles you described in 2.1, 2.2, and 2.3?
- How does your current curriculum achieve the learning outcomes identified in 3.2?
- How does your current curriculum provide opportunities, or not, for addressing the priorities for curriculum revision you set in 4.7 and 4.8?

Next, identify and gather institutional and program assessment data. Examples of the types of data you may already have available are listed below. Consider consulting with Academic Affairs, Office for Institutional Research, Office of Institutional Effectiveness, or other similar groups to learn what specific data sources are available at your institution. You may also have an Institutional Fact Book that will indicate what data sources are available and provide summaries of the data.

1. Institution-Wide Data
    a. Standardized surveys such as the National Survey of Student Engagement (NSSE; https://nsse.indiana.edu/nsse/about-nsse/index.html) or Higher Education Data Sharing Consortium (HEDS) Surveys (https://www.hedsconsortium.org/heds-surveys/)
    b. Customized internal institutional assessment surveys,
    c. General education curriculum assessment data,
    d. Performance and retention (enrollment) data,
    e. Reports to and responses from Regional Accrediting Organizations
2. Departmental or Program Data
    a. External review documents (reports and responses)
    b. Current programmatic assessment goals, tools, and data
    c. Current curriculum map (a curriculum map associates student learning outcomes with specific courses; see

https://www.colorado.edu/oda/sites/default/files/attached-files/curriculum_mapping_v2.pdf for information and examples)

d. Departmental contributions to institution-wide assessment data (e.g.your department's assessment for institutional accreditation, enrollment data for your courses, etc.)
e. Data for DIEA assessment, such as current and historical demographic data, disaggregated versions of these data sources, or specialized surveys/focus groups/etc. This may involve institutional degree and demographic data (as reported to Federal agencies) as well as department demographic data. Cohort based demographic data analysis can be generated at https://aiice.shinyapps.io/AiiCE/, though it is highly dependent on how your institution reports data for CS.
f. Annual reports to the Dean/College/etc.
g. Surveys or focus groups with current students
h. Feedback from graduates/alumni, including senior exit interviews, alumni "first destination" data, or alumni interviews
i. Feedback from an industry advisory board or other advisory panel

If your program does not already have an assessment cycle, particularly a process that includes senior exit interviews and engagement with standardized surveys or institutional data, you may want to plan a separate time to examine and implement something like the National Center for Women and Information Technology's (NCWIT) Student-Evaluation of the Major (https://ncwit.org/resource/sem/) or their more general Evaluation tools (https://ncwit.org/resource/evaluation/)

Review these data for strengths and weaknesses, as well as for "opportunity gaps" that may become evident when the demographic data is reviewed.

5.1. Based on your discussion of your identity, design principles, learning outcomes, and desired alignment with CS2023, combined with your data review, what are your program's current strengths?

5.2. Based on your discussion of your identity, design principles, learning outcomes, and desired alignment with CS2023, combined with your data review, what are places for improvement or opportunities for change in your current program?

5.3 Based on your discussion of your identity, design principles, learning outcomes, and desired alignment with CS2023, combined with your data review, what program goals are currently not met by your curriculum?

Finally, set three to five specific goals for your curriculum revision. These should respond to the strength and weakness identified above, may incorporate high priority design principles, and help inform decisions. For example, McGill and colleagues[15] identified five goals for their Chemistry curriculum revision, including: “help students make connections between courses,” “articulate and incorporate scientific practices,” “maximize flexibility at the advanced level,” “increase student satisfaction with coursework” for both majors and non-majors, and earlier exposure to subdisciplines (p. 36).

5.4. What specific goals do you want to achieve in your curriculum revision?

Step Six: Implementation and Assessment

At this point, you should have four types of guiding documents prepared for your curriculum: your statement of program identity, your list of curriculum design principles, your program-level student learning outcomes, and your goals for your curriculum revision. You have also considered how your curriculum aligns with CS2023 at a high-level and discussed what you would like to change and maintain about that alignment. Essentially, you have worked through the “data gathering” stage of a traditional curriculum review[16], with a focus on liberal arts identity

[15] McGill, Tracy & Williams, Leah & Mulford, Douglas & Blakey, Simon & Harris, Robert & Kindt, James & Lynn, David & Marsteller, Patricia & McDonald, Frank & Powell, Nichole. (2018). Chemistry Unbound: Designing a New Four-Year Undergraduate Curriculum. Journal of Chemical Education. 96. 10.1021/acs.jchemed.8b00585.

[16] Dyjur, P. & Kalu, F. (2018). Introduction to curriculum review. Taylor Institute for Teaching and Learning. Calgary: University of Calgary. https://taylorinstitute.ucalgary.ca/sites/default/files/Introduction%20to%20CR%20UPDATED%202019.pdf

and CS curriculum recommendations. We now combine implementation and assessment into one step - albeit a very large step. The process now aligns with traditional curriculum review[17]:

1. Develop course-level learning outcomes.
2. Map the curriculum: associate course-level learning outcomes with specific courses, and determine “routes” through the major, i.e., sequence the courses.
3. Plan assessment.
4. Action: plan and implement changes (e.g., create new courses, revise courses, etc).

Because this process is well-documented elsewhere, we provide some general guidance in this section but also include references to other resources you may want to begin using.

We also note that at this stage, curriculum (re-)design could be applied at different scales; for example, do you want to focus on a specific knowledge area, a sequence of core courses, or target the full curriculum? In this step, we guide you through implementation and assessment by providing you with some guiding questions with connections to CS2023, and pointing you to existing resources. You should also consider reaching out to your Center for Teaching and Learning or other support you might have for curriculum and instructional design on campus. The documentation you have produced up to now will be a good starting point for a conversation with them about the resources they may have to help with things like developing curriculum maps.

Implementation of your curriculum (re-)design will necessitate iteration over Steps 1-5, and perhaps even a multi-year plan that sequentially addresses different program learning outcomes or course creation/revision. Consider creating a timeline for implementing curriculum changes. Be sure new and revised aspects of your curriculum always consider your identity (1.3), design principles (2.1, 2.2 & 2.3), and program outcome goals (3.2). Revisit the goals of your curriculum (5.4) (re-)design frequently.

6.1. What is your timeline for implementing your curriculum (re-)design? Your timeline could be based on curriculum review steps (above- course-level outcomes, map, assessment, action), program learning outcomes, course creation/revision, or other steps.

Guiding Questions

Some high-level questions you will want to answer during your curriculum revision include:

[17] Ibid

6.2. What is the set of course-level student learning outcomes that you expect all of your majors to achieve? Where possible, organize or group these outcomes so that any sequencing of outcomes is reflected.

It is tempting to answer this question by thinking about the courses that make up your major and then articulating the learning outcomes for those courses. That approach, however, risks reinforcing your current curricular structures and can make it harder to identify opportunities for innovation and improvement. While some or many of your courses may remain the same after a curriculum revision, you may also identify places where content can shift, different pathways through the majors can be created, or new combinations can be considered.

Start with your program-level outcomes and turn to CS2023 to consider what specific computer science skills or competencies are required to achieve that at the course level. Each Knowledge Area includes not just topics, but learning outcomes for that topic. Discussing these learning outcomes and considering which ones align most strongly with your program-level outcomes, as well as your programmatic identity, can provide helpful input to this process.

6.3. What is the organization of your course-level outcomes into courses? What courses exist (including in other departments or programs)? What courses will need to be modified? What courses need to be created?

6.4. What pathways will students take to complete these courses?

Resources

We encourage you to seek guidance from appropriate offices at your institution (e.g. a Center for Teaching and Learning, a Director of Assessment, a Curriculum Committee, or Academic Affairs), At this step, we additionally recommend the following resources (also mentioned elsewhere in this document):

- Patti Dyjur, Kim Grant, and Frances Kalu, 2019, Introduction to Curriculum Review: https://docs.google.com/document/d/15LRmo2j8L-w21ZIVZTCAN22W795s4c-ZIUD72qfEz6o/edit#
- CU Boulder, 2019, Aligning Program Outcomes and Curricula through Curriculum Mapping: https://www.colorado.edu/oda/sites/default/files/attached-files/curriculum_mapping_v2.pdf
- L. Dee Fink, n.d., A Self-Directed Guide to Designing Courses for Significant Learning: https://www.bu.edu/sph/files/2014/03/www.deefinkandassociates.com_GuidetoCourseDesignAug05.pdf
- L. Dee Fink, 2003, Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses (San Francisco: Jossey-Bass)

We have also included a set of questions to consider at the curriculum mapping step in Appendix B.

(Re-)Consider Assessment

Programs require an assessment plan to inform how well they are aligning with their identity, adhering to design principles, and achieving program outcome goals as well as for external evaluation and accreditation. A curriculum revision is not complete without a plan in place for assessment. Existing program assessment may require revision in light of your progress, and new programs will need a plan in place. Stakeholders to consider in your assessment include: introductory students, continuing students, graduating students, alumni, and faculty. Assessment methods include: standardized or customized surveys, focus groups, and standardized or customized skills assessment (such as ETS's Major Field Test: Computer Science). Consider what outcomes suggest success or opportunities for improvement. You will likely want to discuss your plans with your institution's assessment and research office.

6.5. How will you assess alignment with your identity? What outcomes suggest success and opportunities for improvement?

6.6. How will you assess adherence to design principles? What outcomes suggest success and opportunities for improvement?

6.4. How will you assess program learning outcomes? What outcomes suggest success and opportunities for improvement? Consider each learning outcome separately.

6.5. How will your assessment plan be used for external evaluation and accreditation?

Conclusion

This workbook has guided you through a process for designing or revising a liberal arts computer science curriculum that centers program identity and uses it as a lens through which curricular guidelines are applied. If you have completed this entire workbook, you and your colleagues will have reached a shared understanding of the following:

- Your unique programmatic identity as a liberal arts computer science program,
- Your identity-driven, structural, and DEIA-driven curricular design principles,
- Your programmatic student learning outcomes,
- Your level of desired alignment with the CS2023 curricular guidelines,
- Your current strengths as a program and most important places for improvement,
- Your plans for curricular change to achieve that improvement,
- Your plans for re-assessment and re-design in the future.

This process is an inherently iterative process, so we hope you will return to the steps in this workbook as you continue to review and revise your courses and curriculum. As noted in the introduction, depending on your goals, you can focus more on some steps than others. However, we do recommend that you always spend at least some time on the first three steps prior to jumping into steps four through six to ensure that you are proceeding with your curriculum work with a strong grounding in the identity, design principles, and curricular goals for your program.

The authors of this document welcome feedback on this process and suggestions for how to improve it. We would also welcome any examples of artifacts from this process that you would like to share with the broader liberal arts community. *Contact information will be shared in the de-anonymized version of this workbook.*

Appendix A: Design Principle Suggestions

Identity-based Design Principles

Design principles can also include institutional or departmental priorities for curricula, such as:

- Provide time for major exploration; allow students to begin the major as sophomores
- Connect to students' additional programs of study (double majors, minors, etc)
- Support internships
- Support study abroad
- Account for prior learning credit
- Provide service-learning or community work opportunities
- Provide paths for non-majors
- Prepare graduates for a wide variety of computing and computing-aligned careers (students may become developers, testers, documentation/training specialists, and or take on whole host of other technical and less technical roles based on their degree in CS)
- Support the needs of first generation undergraduate students
- Support the needs of transfer students
- Offer "quantitative reasoning" general education classes (separate from or same as major intro course)
- Offer general education classes besides quantitative reasoning
- Develop courses that contribute to other programs of studies on campus (majors, minors, list specifics)
- Utilize courses offered by other programs on campus
- Develop a "spiral" curriculum where students are introduced to topics in one course and revisit them frequently in other courses

They may also include some broad, content-specific design principles:

- Students develop skills in at least two programming languages
- Students participate in at least one larger-scale group software development project
- Students develop empathy for users, including understanding accessibility for different user abilities
- Students understand social, legal, and ethical concerns of technology
- Students understand their own role and the role of technology in a diverse, equitable, inclusive, and just society
- Prepare students for a particular career path (e.g. software development, cybersecurity, research)
- Incorporate experiential learning at the introductory level
- Incorporate supervised lab time into most core and elective courses

Structural Design Principles

Your institution may have rules in place for programs of study, such as:

- Size of the major (number of courses)
- Composition of the major (number of upper-level electives)
- Inclusion of a thesis or capstone
- Contributions to the general education curriculum (e.g., quantitative reasoning or writing across the curriculum)

You students may have particular characteristics you must take into account:

- Students tend to double major and seek additional credentials limiting the size of your program OR students tend to have a single concentration and you need to incorporate applications of computing.
- Students come in with variable experience in CS
- A large percentage of students come in on a pre-health or other pre-professional track
- Students have jobs/work study/athletics time constraints
- Students tend to work on their own OR students tends to collaborate/form study groups
- Students do/do not seek help in office hours
- Students have financial constraints that may limit the materials you can use
- Balance of residential and commuter students and ease of access to on-campus labs versus reliance on personal technology

You may have constraints based on the size (number of faculty or students) of your program or the University schedule that may affect the design of pre-requisite chains, paths through your program, variety of electives, or reliance on courses in other departments:

- Courses that can only be offered occasionally (e.g. every 3rd or 4th semester)
- Specific expertise of your faculty members
- Number of students needing elective options each year
- Student preferences regarding electives

You may have courses that are required for students in other majors are that have been committed to support college-wide requirements:

- List courses, requirements, and frequency of offering

In addition, consider physical space and personnel resources:

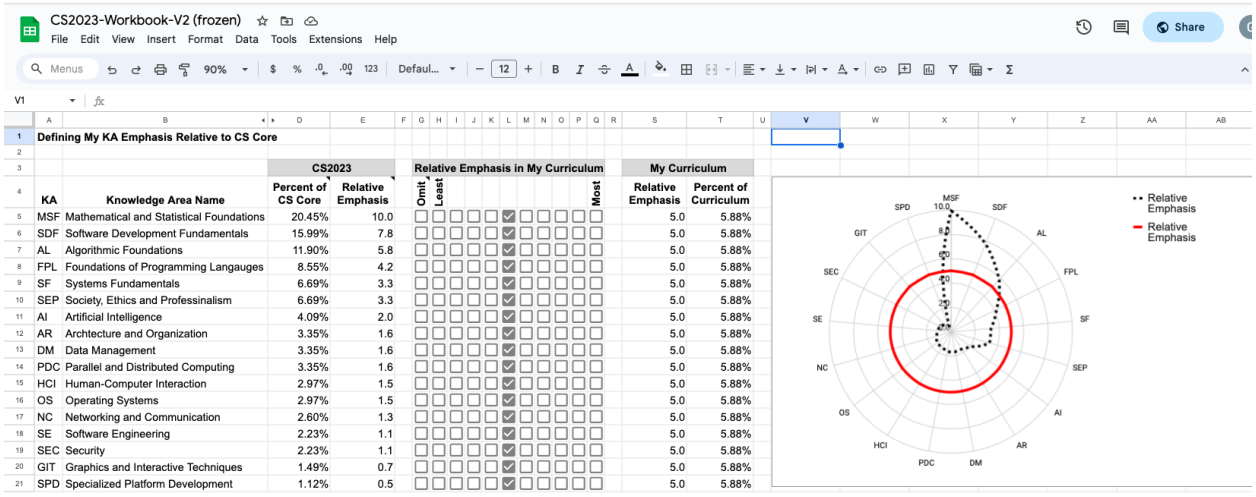
- Number of lab spaces and seats within labs
- Size and layout of classroom spaces
- Professional lab assistance or Information Technology support
- Student/peer teaching assistant support
- Office hours expectations

Appendix B: Suggestions for questions to consider during curriculum mapping (hint: refer to your design principles)

- Which course-level student learning outcomes are essential to cover early and how is that accomplished in your curriculum?
- How do you want to support students entering the major with a range of prior experience?
- Do you want all students to take a shared set of core courses, or do you want multiple options/pathways for achieving some of your course-level outcomes?
- How important is a shared introductory course versus multiple introductory options?
- How important is a shared capstone course or senior experience?
- Are any of your course-level outcomes covered by courses offered outside your program? (E.g. related to math, writing, communication, cultural competency, etc.)
- Which of your course-level outcomes complement each other well and could be effectively covered in the same course?
- What is the maximum length chain of prerequisites you want to permit for upper level courses in your program?
- Given your staffing, what is the maximum number of regular catalog courses you can reasonably offer?

Appendix C: The CS2023-Workbook Spreadsheet - Screen Shots for Anonymous Review.

The CS2023-Workbook Spreadsheet showing the KA Emphases worksheet.



CS2023-Workbook-V2 (frozen)

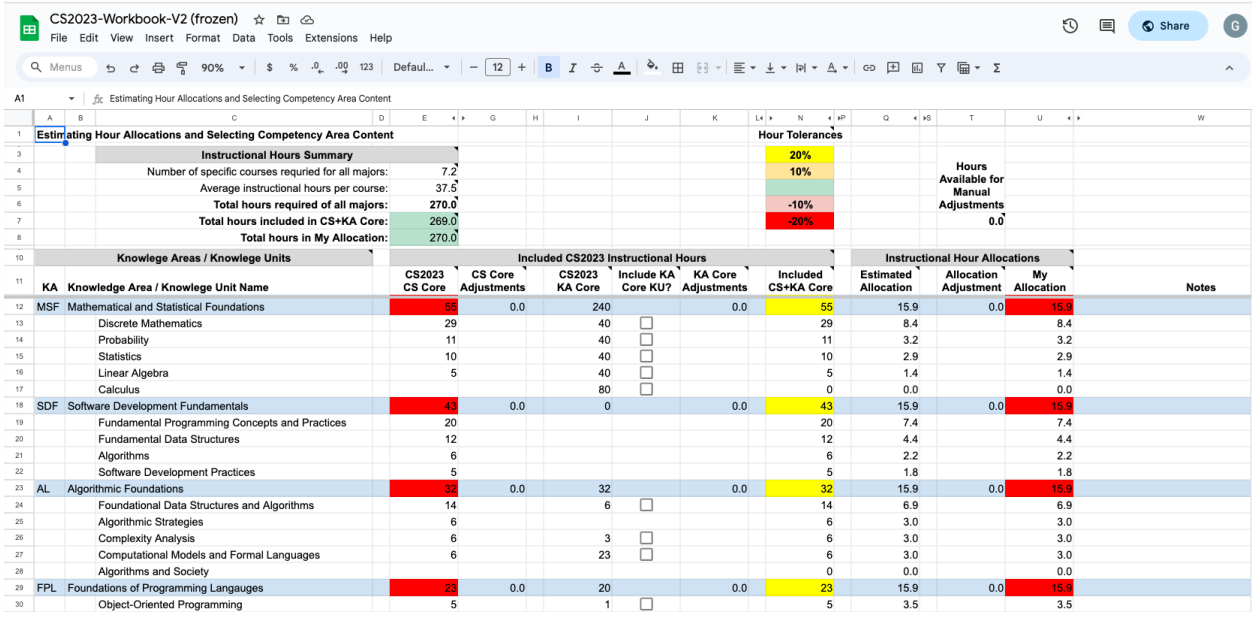
File Edit View Insert Format Data Tools Extensions Help

Share

Defining My KA Emphasis Relative to CS Core

KA	Knowledge Area Name	CS2023 Percent of CS Core	CS2023 Relative Emphasis	Relative Emphasis in My Curriculum: Omit	Least	Most	My Curriculum Relative Emphasis	My Curriculum Percent of Curriculum
MSF	Mathematical and Statistical Foundations	20.45%	10.0			☑	5.0	5.88%
SDF	Software Development Fundamentals	15.99%	7.8			☑	5.0	5.88%
AL	Algorithmic Foundations	11.90%	5.8			☑	5.0	5.88%
FPL	Foundations of Programming Langauges	8.55%	4.2			☑	5.0	5.88%
SF	Systems Fundamentals	6.69%	3.3			☑	5.0	5.88%
SEP	Society, Ethics and Professinalism	6.69%	3.3			☑	5.0	5.88%
AI	Artificial Intelligence	4.09%	2.0			☑	5.0	5.88%
AR	Archtecture and Organization	3.35%	1.6			☑	5.0	5.88%
DM	Data Management	3.35%	1.6			☑	5.0	5.88%
PDC	Parallel and Distributed Computing	3.35%	1.6			☑	5.0	5.88%
HCI	Human-Computer Interaction	2.97%	1.5			☑	5.0	5.88%
OS	Operating Systems	2.97%	1.5			☑	5.0	5.88%
NC	Networking and Communication	2.60%	1.3			☑	5.0	5.88%
SE	Software Engineering	2.23%	1.1			☑	5.0	5.88%
SEC	Security	2.23%	1.1			☑	5.0	5.88%
GIT	Graphics and Interactive Techniques	1.49%	0.7			☑	5.0	5.88%
SPD	Specialized Platform Development	1.12%	0.5			☑	5.0	5.88%

The CS2023-Workbook Spreadsheet showing the KA/KU Hours worksheet.



CS2023-Workbook-V2 (frozen)

Estimating Hour Allocations and Selecting Competency Area Content

Instructional Hours Summary	
Number of specific courses requried for all majors:	7.2
Average instructional hours per course:	37.5
Total hours required of all majors:	270.0
Total hours included in CS+KA Core:	269.0
Total hours in My Allocation:	270.0

Hour Tolerances
20%
10%
-10%
-20%

Hours Available for Manual Adjustments: 0.0

Knowlege Areas / Knowlege Units			Included CS2023 Instructional Hours						Instructional Hour Allocations			
KA	Knowledge Area / Knowlege Unit Name		CS2023 CS Core	CS Core Adjustments	CS2023 KA Core	Include KA Core KU?	KA Core Adjustments	Included CS+KA Core	Estimated Allocation	Allocation Adjustment	My Allocation	Notes
MSF	Mathematical and Statistical Foundations		55	0.0	240		0.0	55	15.9	0.0	15.9	
		Discrete Mathematics	29		40	☐		29	8.4		8.4	
		Probability	11		40	☐		11	3.2		3.2	
		Statistics	10		40	☐		10	2.9		2.9	
		Linear Algebra	5		40	☐		5	1.4		1.4	
		Calculus			80	☐		0	0.0		0.0	
SDF	Software Development Fundamentals		43	0.0	0		0.0	43	15.9	0.0	15.9	
		Fundamental Programming Concepts and Practices	20					20	7.4		7.4	
		Fundamental Data Structures	12					12	4.4		4.4	
		Algorithms	6					6	2.2		2.2	
		Software Development Practices	5					5	1.8		1.8	
AL	Algorithmic Foundations		32	0.0	32		0.0	32	15.9	0.0	15.9	
		Foundational Data Structures and Algorithms	14		6	☐		14	6.9		6.9	
		Algorithmic Strategies	6					6	3.0		3.0	
		Complexity Analysis	6		3	☐		6	3.0		3.0	
		Computational Models and Formal Languages	6		23	☐		6	3.0		3.0	
		Algorithms and Society						0	0.0		0.0	
FPL	Foundations of Programming Langauges		23	0.0	20		0.0	23	15.9	0.0	15.9	
		Object-Oriented Programming	5		1	☐		5	3.5		3.5	

Appendix D: Sample Program-Level Outcomes

The following are examples of program-level outcomes for a variety of computer science programs.

Trinity Christian College, Computing and Data Analytics (joint outcomes):

Program-Level Student Learning Objectives:

1. Students will solve substantive problems in computing and data through an interdisciplinary and collaborative approach.
2. Students will creatively engage complex problems with ethical and practical consequences that reflect inclusive viewpoints.
3. Students will evaluate how technology and data-related practices can support or resist God's vision for all people, societies, and creation.
4. Students will effectively communicate about computing and data analytics to audiences of varying technical expertise through written, oral, and visual means.
5. Students will demonstrate willingness and proficiency to thrive in new technological contexts as life-long learners.

Trinity's Reflection on Mapping to Identity and Design Principles

Key Concepts in our identity statement:

A. Translational computer and data skills (i.e. applicable to many problems)
B. Skills that grow as technology evolves
C. Recognition of the inherent value in diverse viewpoints

D. Participation in God's vision for the world
E. Wrestling with hard problems
F. Flexibility in thinking
G. A foundation in interdisciplinarity

PLO #1: This is meant to capture identity components: A, C, E, G.
PLO #2: This is meant to capture identity components: C, D, E, F.
PLO #3: This is meant to capture identity components: C, D, (F).
PLO #4: This is meant to capture identity components: A, B, G.
PLO #5: This is meant to capture identity components: A, B, F, G.

University of Jamestown
Computer Science & Technology Program Level Outcomes

- Design effective and usable technology-based solutions and integrate them into the user environment. This involves analyzing, identifying, and defining the requirements that must be satisfied to address problems or opportunities faced by organizations or individuals.
- Use independent critical thinking and problem-solving skills to explore and generate possible solutions to new technology-related problems/situations.
- Communicate effectively and efficiently with clients and peers both verbally and in writing.
- Collaborate in teams to accomplish a common goal by integrating personal initiative and group cooperation.
- Demonstrate independent learning through research, preparation, and presentation of a technology project.
- Describe the impact of technology on individuals, organizations, and society, including ethical, legal, and policy issues.

Whitman College
Computer Science Major Learning Goals
Upon graduation, a student majoring in Computer Science will be able to:

- Understand and apply fundamental algorithms and data structures;
- Understand the abstractions supporting modern software systems, and how the construction of those mechanisms affects the supported systems;
- Apply mathematical techniques to justify computational solutions and explore the limitations of computers;
- Communicate computational ideas through speech, writing, diagrams, and programs;
- Work with a team to design and implement a substantial, integrative project;
- Propose and compare multiple solutions to computational challenges, with consideration for the context and impact of each solution on the creators, maintainers, and users of that solution.

Washington & Jefferson College
Computing and Information Studies Program Level Outcomes

- Research and analyze an I.T. challenge and make sound recommendations regarding its solution

- Implement robust and well-documented I.T. solutions that respond to specific user requirements and that anticipate future needs
- Work as a productive member of a team to accomplish project goals; to plan and schedule project components effectively; to communicate clearly to a variety of audiences in both written and oral forms about the progress, status, and results of an I.T. project.
- Act ethically in the execution of all these objectives.