CDIO CURRICULUM DESIGN FOR COMPUTING: A GRAPH-BASED APPROACH

Jaime A. Pavlich-Mariscal, Mariela Curiel, German Chavarro

Systems Engineering Department. Pontificia Universidad Javeriana.

ABSTRACT

An essential activity in curriculum design is to specify the topics of the curriculum and the courses where those topics will be taught. Disciplines, such as Computing present several challenges in this regard, since the topics that students must learn tend to be fine-grained and highly interconnected. First, one must ensure that the most important topics of the curriculum are taught in at least one course. Second, for every topic taught, their prerequisites must have been covered previously in the same course or in a previous one. Third, courses must include topics that are highly cohesive and with minimal dependencies to topics taught in previous courses. To address the above challenges, this paper proposes a graph-based approach to analyze and design a curriculum, which also includes some Backward Design elements. Learning goals (desired results), topics, and courses are modeled as nodes in a graph. Prerequisite dependencies are modeled as edges. The relation between courses and topics are also modeled as edges. Graph analysis techniques are utilized to measure several aspects of a curriculum. Edges between topics are utilized to verify consistency between topics and prerequisite and corequisite relations between courses. Course-topic edges are used to calculate topic coverage of the curriculum. Topological sorting and course-topic relations are utilized to automatically generate the draft of course syllabi. We also describe the results of a real-life application and argue that this approach is essential to make visible and verify the overall structure of a curriculum.

KEYWORDS

Curriculum design, graph, syllabus, computing, CDIO standards 2, 3.

INTRODUCTION

CDIO initiative definitions involve several tasks for proper curriculum design. One of them is the specification of the disciplinary topics to teach to students and the decision of which courses should include those topics in their syllabi. In some knowledge areas, such as Computing, this challenge can be difficult to solve, because they may comprise many fine-grained and strongly inter-dependent topics. For instance, the ACM Computing Classification System (ACM, 2012) defines more than 2000 topics that could be included in a computing curriculum. Moreover, as described in this paper, many Computing topics are highly interrelated. To properly learn such topics, the student may need to learn several other topics previously.

The above situation introduces several challenges. First, the most important topics in the curriculum should be taught at least in one course. While this is a common issue, topics may be hard to properly prioritize when connections between them are complex. Courses should teach highly coherent topics, with minimal dependencies to prerequisite topics and with a proper course ordering that ensures that none of those prerequisites is untaught before eachcourse.

To address the above challenges, this paper proposes an approach to support the definition of the body of knowledge of a program, to properly understand topics and their inter-dependencies, and to properly define course syllabi in disciplines such as Computing. The key element is to utilize a graph to represent the information to support the curriculum design process.

This graph specifies disciplinary topics, program courses, and program desired results as nodes. Prerequisite relations between topics are specified as edges. Similarly, associations between courses and their syllabi topics are modeled as edges. They are also utilized to specify the connections between topics and desired results.

Several metrics are utilized to identify curriculum issues related to the above challenges and also to guide in their resolution. Dependencies between topics and their relations to courses are used to verify consistency of current course prerequisites. Course-topic associations are utilized to calculate the topic coverage of the curriculum. Syllabi can be automatically generated from a topological sort of such topics.

This paper proposes an approach to integrate all of the above elements into the curriculum design process and also describes the real-world experience of implementing this process in the redesign of a Computing program.

The remainder of this document details all of the above elements: the overall approach and the graph utilized to model the knowledge base, the process to create and utilize this graph in curriculum design, the results of applying this approach in the curricular redesign at our university. There is a discussion of the results, a comparison with related work and the last section concludes and describes future work.

BACKGROUND

This section briefly describes the fundamental concepts to understand the proposed approach, and the nomenclature utilized in the remainder of this document.

Backwards Design

Aside from graph theory (Bondy & Murty, 1976), an important element in the proposed approach is Backwards Design. Backwards design is a component of Understanding by Design (Wiggins & McTighe, 2005), a framework to design curricula, assessment mechanisms, and teaching. Backwards Design comprises three main stages:

1. Identify desired results: This stage defines the expected results yielded by students at the end of their studies.

2. Determine assessment evidence: This stage defines the way to assess student

learning and the way this learning is evidenced.

3. Define instruction plan and learning experiences. The last stage creates a concrete plan to effectively teach all the concepts defined in the curriculum.

The proposed approach in this paper addresses stage 1, to define a set of desired results, and stage 3, to assist in the creation of the curriculum courses and their ordering

Nomenclature

For the purposes of this paper and to better understand our approach, it is necessary to define a basic nomenclature.

Desired Result: Something that the student will be able to do as a result of studying the proposed curriculum. This concept is directly based on Backwards Design (Wiggins & McTighe, 2005)

Topic: A relatively small unit of knowledge that the student will learn as part of the proposed curriculum. This is usually something that can be learned in a relatively small period, e.g., 1-2 weeks in a regular course with 2-4 credits, i.e., 2 to 4 hour of classroom work plus 4 to 8 hours of out-of-classroom work (see definition of credit below).

Credit: In our university 1 credit per semester is equivalent to 1 hour of classroom work per week, plus 2 hours of out-of-classroom work. A semester has 16 weeks.

Knowledge Area: A set of cohesive topics that form a well-known sub-discipline, e.g., Artificial Intelligence, Software Engineering, etc. A curriculum usually addresses one or more knowledge areas.

Course: A set of topics that the student will learn, usually during a semester.

Prerequisite: A relation between two courses or two topics that indicates the order in which courses/topics should be taught to the student. A prerequisite can also connect a topic and a desired result. In this case, it means that the student must learn that topic in order to achieve the desired result.
Course-topic association: A relation between a course and a topic that indicates that said topic is taught in that course, i.e., it is part of the course's syllabus.

GRAPH-BASED APPROACH FOR CURRICULUM DESIGN

This paper proposes the use of graphs to represent curriculum information and verify that the curriculum satisfies certain properties. Figure 1 shows the key components of the approach, and the information flows between them. The curriculum design process transitions from a current state to a desired state. The current state is the curriculum currently being taught at a university. Using this information, the process specifies the *desired results* of the new program. To achieve these results, the process also defines the *desired body of disciplinary knowledge* for the new program, i.e, the set of topics that every student should learn in the program. All of the above information is utilized to create the *desired curriculum*. The entire process is supported by a knowledge base in the form of a graph, which provides a structured way to store all of the information and to extract indicators to make decisions.

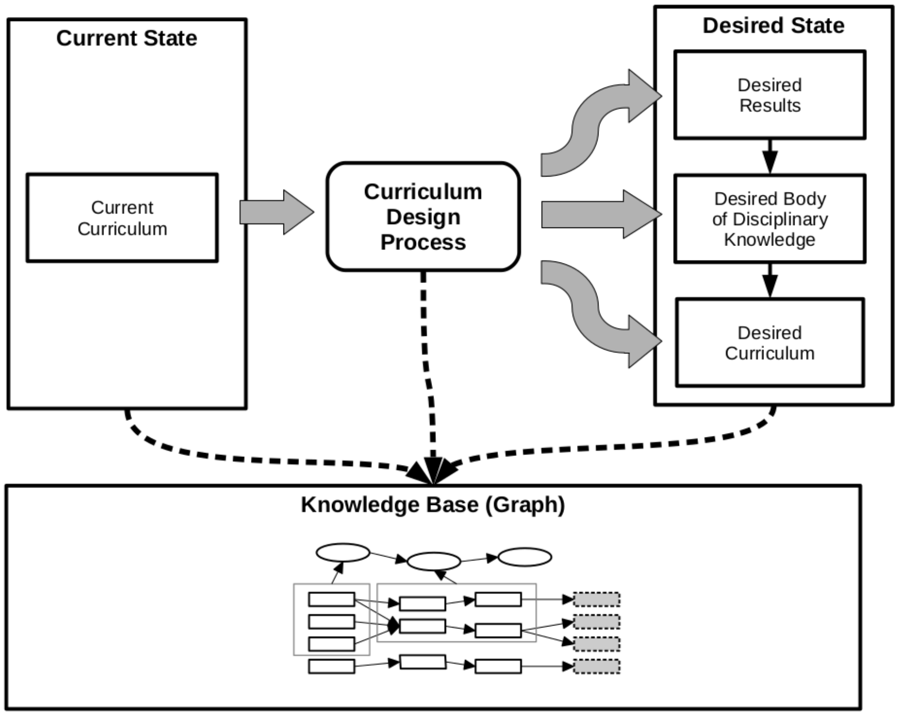

Figure 1: Overview of the Approach

The proposed approach relies on a directed graph to represent all of the above concepts. The graph represents desired results, courses, and topics as nodes, while prerequisite and course-topic associations are represented as edges. Figure 2 is an example of a curriculum graph. Courses are represented as ovals, topics are represented as rectangles, and desired results are represented as dashed rectangles. The course **Discrete Math** is prerequisite of **Data Structures** (the relation is depicted as a continuous arrow). The topics **Logic** and **Set theory** are part of the syllabus of the course **Discrete Math** (connected with dashed arrows). **Logic** and **Set Theory** are prerequisites of the desired result **Understand the basic concepts of discrete mathematics**.

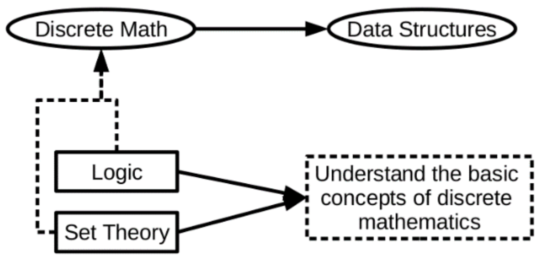

Figure 2: Example Graph

Formally, the knowledge base graph is defined as a tuple $G = (R, T, C, P, A, Q)$, where

- R is the set of desired result nodes
- T is the set of topic nodes
- C is the set of course nodes
- $P \subseteq (T \times (R \cup T)) \cup (C \times C)$ is the set of prerequisite edges
- $A \subseteq C \times T$ is the set of course-topic association edges
- $Q \subseteq C \times C$ is the set of corequisite edges

Curriculum Design Process

Figure 3 details the proposed process. The following sections explain each stage in turn.

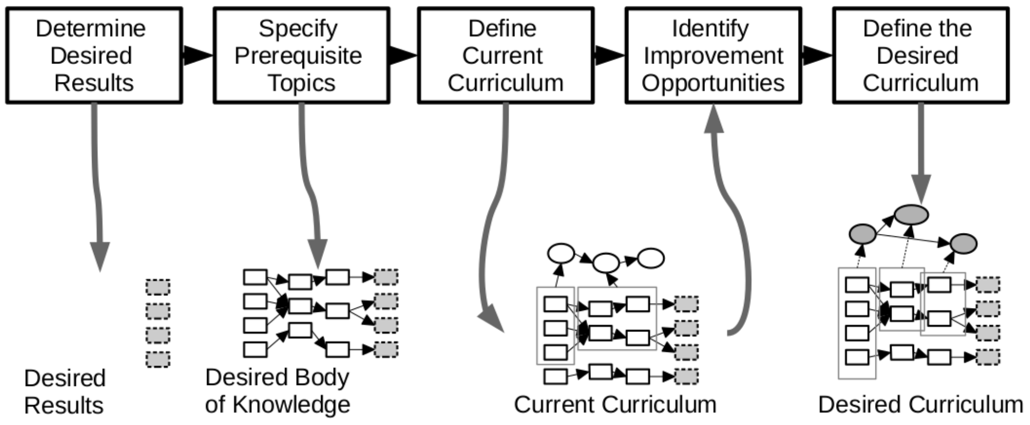
Determine Desired Results
Specify Prerequisite Topics
Define Current Curriculum
Identify Improvement Opportunities
Define the Desired Curriculum
Desired Results
Desired Body of Knowledge
Current Curriculum
Desired Curriculum

Figure 3: Proposed Process

Determine Desired Results

Similarly to Backwards Design Wiggins and McTighe (2005), the first stage is to determine the desired results the student will achieve after studying the courses in the curriculum. Desired results are written as short sentences that include the bloom level of attainment (Anderson et al., 2001) and the high-level subject of such result. For instance, the desired result **Understand the basic concepts of discrete mathematics** utilizes the verb *understand* to denote level 2 of attainment in the subject *discrete mathematics*. Each desired result is incorporated into the graph as nodes.

Specify Prerequisite Topics

The previous phase determines what the student should become at the end of his/her studies. The next step is to define all of the topics that the student should know at the end of the program. Each topic is specified as a node in the graph and their prerequisite relations are specified as edges. Some topics may also be prerequisites of desired results, which indicate that they are required to achieve those results.

Define the Current State of the Curriculum

At this point, the graph represents the desired *body of knowledge* that the student should learn and the goals of that are satisfied by learning that knowledge. It is necessary to understand the relation between the current curriculum and this body of knowledge. This stage adds to the graph the courses in the current curriculum, their prerequisites, and the connection to the topics that are part of their syllabi.

Identify Improvement Opportunities

The above data can be processed to find information of interest to improve the current curriculum. The results are the following indicators:

Curriculum Coverage (CC): This is the percentage of topics that are assigned to at least one course in the curriculum. It is recommended to also calculate this indicator aggregated by knowledge areas to have sufficiently detailed information about coverage. With this information, curriculum designers may decide to create new courses to cover unassigned topics, expand existing course syllabi, remove courses with unwanted topics, etc.

Course Interdependence (CI): This is a number that indicates the consistency between prerequisites of the topics belonging to courses. Given a graph $G = (R, T, C, P, A)$ and course nodes $a, b \in C$, course interdependence CI is calculated as follows:

$$CI(a,b) = \frac{dep(a,b)}{dep(a,b) + dep(b,a)}$$

Where $dep(a, b)$ is the amount of prerequisite dependencies between topics assigned to a and b.

$$dep(a, b) = |\{(t_a, t_b) \in P : (a, t_a) \in A \wedge (b, t_b) \in A\}|$$

A value of $CI(a, b)$ close to 1 means that most prerequisite relations are from topics in a course a to topics in course b, while a value close to 0.5 means that there is a similar amount of prerequisite relations from a to b and from b to a.

The CI indicator can be interpreted in two ways. First, if $CI(a, b)$ is close or equal to one, it may be interpreted as a being a clear prerequisite of b and this should be contrasted against the explicit course prerequisite relations in the graph, i.e., a, b $P.$(If such relation does not exist, a potential discussion among the curriculum designers would be to decide whether or not to denote a as explicit prerequisite of b.

Another way to interpret CI is when it has a value close to 0.5, which means that may be a strong mutual dependency between both courses. Curriculum designers may decide whether to combine both courses into a bigger one, to define a corequisite relation between them or to redistribute topics between courses to reduce the mutual dependency.

Define the Desired Curriculum

A new curriculum is designed that should address the improvement opportunities identified in the previous stage (among other goals outside the scope of this paper). In practice, this means to create, eliminate or modify existing courses, and assign the topics to be taught in each of them. This new curriculum can be analyzed similarly as in the previous phase, to verify that the new curriculum has no inconsistencies.

At this point, curriculum designers can utilize the graph to automatically generate syllabi drafts.

These syllabi are constructed with two pieces of information:

- **Course-topic associations** to determine syllabi items.
- **Topic prerequisite relations** to topologically sort those topics and provide a recommended sequence to teach those topics in a course.

CASE STUDY

The proposed approach has been applied to the curriculum redesign of the Computing program at our university. The knowledge base graph includes 150 desired results (nodes surrounded with a green circle), 1232 topics and 2295 prerequisite edges. The current curriculum includes 61 courses, of which 42 are required courses and 19 are elective courses.

The analysis of the graph indicated curriculum coverage of 58.6% by required courses and 25.7% by elective courses with 15.7% of the topics not addressed by any course. The course interdependence analysis yielded 103 pairs of courses with a Course Interdependence (CI) of less than 1. After further analysis, 19 of them were deemed to require further examination and were distributed among teacher teams to determine potential improvement actions. Among the pairs of courses with $CI = 1$, seven of them were also assigned to teacher teams for examination, since they did not correspond with existing prerequisites.

The teams are currently designing the new curriculum, having at their disposal the following information: Curriculum coverage, decomposed by knowledge areas, course syllabi expressed in terms of topic nodes assigned to courses, interdependences between the 19 + 7 pairs of courses identified previously, and a searchable spreadsheet that comprises all of the information in the graph. We expect to apply a similar analysis to the new curriculum to further verify its consistency.

DISCUSSION

Overall, the approach provided useful information to understand the state of the current curriculum, and to identify potential improvements.

The key element of this approach is the prerequisite relation between topics and courses. This may be particularly useful for programs with several, strongly interrelated topics. For disciplines with less coupled, more independent topics, the course interdependence indicator (CI) may not be useful, although the curriculum coverage (CC) can still be useful.

The most demanding parts of the approach are the identification of the prerequisite topics. The team required approximately 4 months, 2 hours per week, to complete this stage. In our experience, this stage should be approached with caution, since it may generate a degree of rejection from the teachers. However, future curricular reflections may reuse the same graph with relatively small changes, thus reducing further efforts.

The current approach does not explicitly address other curriculum elements, such as assessments, scheduling, or resource assignment. These aspects are part of our ongoing work.

RELATED WORK

There are several attempts to use graphs to model curricula. Kabicher and Motschnig-Pitrik (2009) created a collaborative wiki space that utilizes graphs to store curricular information. Gestwicki (2008) and Zucker (2009) developed curriculum visualization applications that represent courses as vertices and prerequisites and corequisites as edges.

The work of Auvinen, Paavola, and Hartikainen (2014) has some similarities to ours. They use graphs to model both the courses, learning outcomes and their prerequisites and utilize this information to provide custom learning plans to students (suggested sequence of courses to take). Svetlik et al. (2017) has a similar goal but utilizes artificial intelligence techniques to automatically generate a curriculum graph. Both approaches only model an existing curriculum and not the desired body of knowledge, thus their approach cannot be directly used to verify the consistency of a curriculum. Similarly, Lie, Brennan, and Nygren (2018) use graphs to model courses, learning outcomes, assessments, and stakeholders. In contrast, our approach focuses on finer-grained topics, which facilitates interdependency analysis.

Other related applications (Gupta, Ludäscher & Moore, 2002; Ugljanin & Kajan, 2012) utilize ontologies to represent curriculum information. Their aim is to compare different curricula to find similarities. In contrast, our work is focused on curriculum design.

Lightfoot (2014) explores different graph metrics, such as in-degree, out-degree, centrality, clustering coefficients, to extract information of course graphs. This work complements our approach since it provides additional ways to analyze a desired curriculum. Further work is necessary to explore the usefulness of those same metrics in our graph, which provides much more detailed information about the components of those courses. Willcox and Huang (2017) utilize graphs to model courses and CDIO skills. As such it is also a useful complement to our work, which in contrast focuses on disciplinary topics.

CONCLUSIONS AND FUTURE WORK

This paper proposes an approach to curriculum design that utilizes graphs to specify a topic, courses, desired results, and their inter-dependencies. This paper also describes the experience of applying this approach in the curriculum design of a Computing program. These graphs provided useful information to understand the state of the previous curriculum, identify improvement opportunities, define the new curriculum, verify its course prerequisites, and adequately define course syllabi.

Although this approach requires an important amount of upfront work, it provides more precise means to support the curriculum design decisions and to verify any proposed curricula.

Future work is to improve some aspects in the current approach: to better tools to create the graph, capture more information about the association between topics and their courses and evaluate new graph-based metrics. In addition, we are currently applying an improved version of the approach to the design of the Master in Cybersecurity program that is being currently developed at the University.

REFERENCES

ACM. (2012). The 2012 ACM Computing Classification System. Retrieved 2018-11-02, from https://www.acm.org/publications/class-2012

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Wittrock, M. C. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman*.

Auvinen, T., Paavola, J. & Hartikainen, J. (2014). STOPS: A Graph-based Study Planning and Curriculum Development Tool. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* (pp. 25–34). New York, NY, USA: ACM.

Bondy, J. A. & Murty, U. S. R. (1976). *Graph theory with applications* (Vol. 290).

Gestwicki, P. (2008, October). Work in progress - curriculum visualization. In *2008 38th Annual Frontiers in Education Conference* (pp. T3E–13–T3E–14). doi: 10.1109/FIE.2008.4720392

Gupta, A., Ludäscher, B. & Moore, R. W. (2002, July). Ontology services for curriculum development in NSDL. In (pp. 219–220). ACM. doi: 10.1145/544220.544266

Kabicher, S. & Motschnig-Pitrik, R. (2009, July). Coordinating Curriculum Implementation Using Wiki-supported Graph Visualization. In *2009 Ninth IEEE International Conference on Advanced Learning Technologies* (pp. 742–743). doi: 10.1109/ICALT.2009.54

Lie, S., Brennan, R. W. & Nygren, A. (2018, December). Graph-based approach to model the dependency information of graduate attributes for supporting the accreditation process. *Proceedings of the Canadian Engineering Education Association (CEEA)*. doi: 10.24908/pceea.v0i0.13004

Lightfoot, J. (2014, June). A Graph-Theoretic Approach to Improved Curriculum Structure and Assessment Placement. *Communications of the IIMA*, *10*(2).

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., & Stone, P. (2017, February). Automatic Curriculum Graph Generation for Reinforcement Learning Agents. In *Thirty-First AAAI Conference on Artificial Intelligence.*

Ugljanin, E., & Kajan, E. (2012, November). Use of ontology in creating and monitoring of curriculum.In *2012 20th Telecommunications Forum (TELFOR)* (pp. 1397–1400). doi: 10.1109/TELFOR.2012.6419479

Wiggins, G. & McTighe, J. (2005). *Understanding By Design* (2nd Expanded edition ed.). Alexandria, VA: Assn. for Supervision & Curriculum Development.

Willcox, K., & Huang, L. (2017). Mapping the CDIO Curriculum with Network Models Worldwide CDIO Initiative. In *Proceedings of the 13th International CDIO Conference.*

Zucker, R. (2009, December). ViCurriAS: a curriculum visualization tool for faculty, advisors, and students. *Journal of Computing Sciences in Colleges*, *25*(2), 138–145.

BIOGRAPHICAL INFORMATION

Jaime A. Pavlich-Mariscal is an Associate Professor at the Systems Engineering Department of the Pontificia Universidad Javeriana (Colombia). His research interests are: model driven development, code generation, adaptation, access control, CASE tools, and curriculum development.

Mariela Curiel is the Director of the Systems Engineering program and Associate Professor of the Pontificia Universidad Javeriana (Colombia). She was Assistant, Associate, and Full Professor at the Simón Bolívar University (Venezuela) during 20 years, where she was Head of the Computer Science Programs (Bachelor, MSc, and PhD), and Director of the Unit responsible for Academic Staff Development. Her research interests include: distributed systems, computer performance evaluation, mobile computing, and operating systems.

German Chavarro is an Assistant Professor at the Systems Engineering Department of the Pontificia Universidad Javeriana (Colombia). He has been coordinator of the national and international accreditation processes and of the curricular reflection process at his university.

Corresponding author

Jaime A. Pavlich-Mariscal
Associate Professor
Systems Engineering Department
Pontificia Universidad Javeriana
Calle 40 No 5-50, Ed. José Gabriel Maldonado S.J. Piso 3
Bogotá, Colombia
Tel: (57-1) 3208320 Ext 5308 - 5314
jpavlich@javeriana.edu.co