Dependency Evaluation and Visualization Tool for Systems Represented by a Directed Acyclic Graph

Sobitha Samaranayake[1], Athula Gunawardena[2]
Department of Computer Science, University of Wisconsin
Whitewater, WI 53190, USA

***Abstract*—There is a dearth of data visualization tools for displaying college degree-planning information, especially course prerequisite and complex academic requirement information. The existing methods for exploring degree plans involve a painstaking what-if analysis of static data presented in a convoluted format. In this paper, we present a data visualization tool, named as Dependency Evaluation and Visualization (DEV) chart, to visualize course prerequisite structure and a dynamic flowchart to guide students and advisors through all possible degree requirement completions. DEV chart uses an adjacency matrix of a directed acyclic graph to store a course structure for a degree into a database. Since DEV chart is created dynamically by updating data associated with each node of the directed graph, it provides a mechanism for adding an alert system when prerequisite conditions are not met, and hence the user can visualize the available courses at each step. Similarly, DEV chart can be used with project planning where nodes represent tasks and edges represent their dependencies.**

Keywords*—Data visualization; degree planning; dynamic flowchart; prerequisite structure; adjacency matrix

I. INTRODUCTION

Many universities employ direct communications between academic advisors and students as the primary advising system [1]. Academic advisors are either faculty or professional advisors employed by an academic unit, and they typically help students make decisions about class schedules, select an academic major or minor, plan for graduation, and many other academic related activities [2]. These important decisions are made based on information stored in academic planning tools and offered courses in the upcoming semester. Curriculum changes are typically made once or twice a year so advisors need to spend time understanding and updating their knowledge about degree requirements and academic policies as well as familiarizing themselves with students' progress toward academic degrees prior to any advising period [3].

The most common academic planning tool is the Academic Advising Report (AAR) or Degree Progress Report (DPR) that consists of a list of degree requirements, a list of courses credited towards satisfying each requirement, an indication of whether each requirement is satisfied, and the remaining number of courses/units needed to satisfy each requirement. Many existing academic planning tools utilize static documents or PDF files for displaying information pertaining to degree requirements and course prerequisites. Design and implementation of a Learning Analytics Dashboard for Advisers, LADA, to support the decision-making process of academic advisers through comparative and predictive analysis is presented in [4].

Degree completion process shares many characteristics with project management. Projects are defined in terms of a set of tasks that must be completed in order to achieve the desired outcome. Task dependencies are comparable to course prerequisites: tasks may have multiple preceding tasks (prerequisites) and multiple succeeding tasks. Predecessor must finish before successor can start. Program Evaluation and Review Technique (PERT) [5-6] is a project management tool that is widely used to visualize the timeline and the work that must be done to complete a project. In PERT, all predecessor tasks must be completed before a task is started. One main difference between PERT and degree planning is that the tasks needed to complete a project are predefined whereas a major/minor can be completed by completing different sets of courses. Graphical Evaluation Review Technique (GERT) [7-8] is a project management tool that allows looping of tasks to allow tasks that need to be performed more than once. In GERT, a choice may exist where one of several tasks may be selected based on the associated probabilities.

Degree requirements vary in structure from one academic institution to another, and some of the requirements can be considerably complex. Major/minor requirements are often defined in terms of a set of course requirements that covers specific subjects or areas of knowledge. Choosing a major/minor, planning degree completion, and maintaining the progress towards completing a degree is a complex planning and scheduling problem. Integer linear programing model for finding academic plans that would satisfy a given set of graduation requirements and other constraints in the shortest possible time is presented in [9] and [10]. A student advising system using artificial intelligence techniques is presented in [11].

Many courses specify prerequisites that are outlined using a list of courses, all of which or a subset of which must be completed successfully in order to satisfy the prerequisites. In addition, a few of the prerequisites may be tied to course grades to ensure students acquire the necessary knowledge for getting the maximum benefit from the next course. A directed acyclic graph can be used to represent prerequisite relationships where nodes represent courses lists and edges represent their dependencies. Prerequisite relationships are often defined using one of, all of, either or, and, or a combination of those logical relationships.

There has been an interest in developing visualization tools for academic curricula and advising [12–18]. Moreno et al. [7] presented an interactive visualization tool for exploring course dependencies between courses. Prerequisite visualization has been studied by Aldrich [13]. His work was focused on the overall topology of the courses at Benedictine University, and he proposed a directed acyclic graph for representing prerequisite relations where each edge represents a logical relationship such as all of or one of. Chen et al. [14] presented an interactive course selection scheme with prerequisite hierarchy. Their work includes visualization of all of, one of, or either or logical relationships of courses offered at University of British Columbia. Zucker [16] presented a curriculum visualization tool for developing and arranging the flow of courses for a particular program. In this work, we create a data structure that can process any compound logical relationships. To our knowledge, there is no previous published work in which complex prerequisites structures have been investigated.

Dynamic data visualization tools directly influence the interpretability of visualizations [19-21]. There is a dearth of data visualization tools for displaying degree-planning information, especially course prerequisite information. None of the existing tools is capable of providing guidance on which of the available courses should be planned or when the available courses should be completed. Information pertaining to prerequisites are often scattered in various places, especially for hidden prerequisites. Therefore, planning and maintaining the progress toward completing a major/minor is a formidable challenge. The main objective of this paper is to introduce a novel data-visualization tool that is useful for academic advising as well as project planning where task dependencies play a major role.

Prerequisite visualization is challenging as defining an appropriate data structure for representing complex degree requirements and course dependencies is the most difficult part. Existing work is limited to most common types of degree requirements and prerequisite structures [13-14]. Since prerequisite structures and degree requirements vary from one academic program to another, it is important to identify an appropriate data structure that can process any complex degree requirement. Although we restrict this research to develop a data visualization tool for academic advising, the data structure introduced in this paper is useful for creating degree audit systems and other advising tools.

II. Data Structure for Degree Requirements

A. Degree Requirements

Most of the degree requirements are specified in terms of number of units, credits, or courses that must be taken to satisfy each requirement. There may be other requirements, such as GPA requirements, minimum number of credits/units needed to complete, internships, capstone projects, etc. First, we consider the degree requirements that are often expressed using one of the following terms:

- Complete a set of predefined courses.
- Select a subset from a set of eligible courses.
- Select a specific number of courses from each of several lists.
- Select a subset of lists and then select a specific number of courses from each of the selected lists (e.g., select two of three course lists and then select one course from each list).
- Select a specific number of courses from a selected subset of lists (e.g., select four courses from at least three different categories).
- Select courses with a specific total number of units from a list of courses.
- Select a specific number of units from a selected subset of lists (e.g. select at least five units from two different categories).

Requirements may refer to additional attributes such as course level (lower-division vs. upper-division) or student's minimum grade point average (GPA). In addition, some of the courses may not be taken until a minimum number of units has been earned. Courses may only count once in the major or minor, either as a required course or as an elective, but not as both. There may be hidden prerequisites (i.e. prerequisites of a prerequisite course that may not be explicitly listed as a part of any other requirements) and other requirements such as selecting major/minor emphasis areas.

First, we define a suitable data structure for evaluating degree requirements. A typical degree requirement belongs to one of the following categories:

- Type A: complete k courses from a set of p courses where $1 \leq k \leq p$
- Type B: complete at least m courses/units, but no more than n courses/units from a set of p courses where $0 \leq m \leq n \leq p$
- Type C: complete k units from a set of p courses where $1 \leq k \leq p$
- Type D: combination of Type A, Type B, and/or Type C requirements

Type A, Type B, and Type C degree requirements are relatively easy to implement but Type D requirements are often complex and difficult to implement. There may be other requirements, such as GPA requirements, minimum number of credits/units needed to complete, internships, capstone projects, etc. Those types of requirements can be treated separately by defining an appropriate data structure. Since degree requirements vary from one program to another, it is important to define a data structure that can represent any complex requirement. Such a data structure can be very valuable for introducing other useful advising tools.

B. Basic Requirements

In order to reduce the complexity of the model, we define a data structure to represent degree requirements.

Definition: A **basic requirement** is a 5-tuple (A, T, m, n, δ), where.

1. A is a set of objects,
2. T is the type of requirement (1:select number of objects, 2: select number of units),
3. m is the lower bound of courses or units,
4. n is the upper bound of courses or units, and
5. $\delta: A \rightarrow \{1, 0\}$ is a function such that $\delta(A) = 1$ if A is a credit-bearing set of objects and $\delta(A) = 0$ otherwise.

Type A, Type B, and Type C requirements defined in the previous section are basic requirements. Type D requirements can be represented using a set of basic requirements. Hence, requirements for any major/minor M_i are expressed as $M_i = \{R_{i1}, R_{i2}, \dots, R_{ir}\}$ where each degree requirement $R_{ij}(A, T, m, n, \delta)$ is either

a. a basic requirement where A is a set of courses or
b. a basic requirement where A is a set of basic requirements.

Let R_{ij}(A, T, m, n, δ) be a basic requirement. An object a_i (course or a basic requirement) satisfies a basic requirement R_{ij} if $a_i \in A$ belongs to A. We define a boolean function on A, $b_i: A \rightarrow \{1, 0\}$ such that $b_i(a) = 1$ if $a \in A$ and $b_i(a) = 0$ if $a \notin A$. A set $A = \{a_1, a_2, \dots, a_k\}$ satisfies a basic requirement R_{ij} if $m \leq \sum_{j=1}^{k} b_i(a_j) \leq n$.

Any set of degree requirements can be expressed using a set of basic requirements. Consider a set of requirements defined as follows:

R_1 : complete one of the courses $C1$ or $C2$

R_2 : complete all of the courses $C3, C4, C5$, and $C6$

R_3 : complete 6 units from the courses $C7, C8, C9,$

and $C10$

R_4 : Complete 6 – 12 units with at least two units in $A_1 = \{C11, C12, C13\}$, at least three units in $A_2 = \{C14, C15, C16\}$, and one unit in $A_3 = \{C17, C18\}$.

The requirements $R_1, R_2,$ and R_3 are basic requirements where A is a set of courses. The requirement R_4 may be expressed using the two basic requirements R_{41}(A, T, 6, 6, 0) and R_{42}(B, T, 6, 12, 1) where

$A = \{R_6, R_7, R_8\}$

$R_6 = R_6(A_1, 2, 2, 2, 0)$: complete two units in A_1

$R_7 = R_7(A_2, 2, 3, 3, 0)$: complete three units in A_2

$R_8 = R_8(A_3, 2, 1, 1, 0)$: complete one unit in A_3

$B = \{C11, C12, \dots, C18\}$

Suppose M is any major that is expressed using a set of basic requirements $M = \{R_1, R_2, \dots, R_r\}$.Let C be the set of all courses available to satisfy requirements of the major M and C_i be the set of courses available to satisfy requirement $R_i \in M$. Then the number of courses satisfying the requirement $R_i(A, T, m, n, \delta)$ is $\sum_{k=1}^{l} b_i(c_k)$; credits counted for a requirement $R_i(A, T, m, n, \delta)$ is $s_i = \sum_{k=1}^{l} b_i(c_k) * n(c_k) * \delta$ where $n(c_k)$ is the number of units of the course $c_k \in C_i$; and the total number of credits counted towards completing the major M *is* $S_M = \sum_{i=1}^{r} S_i$ where r is the total number of requirements of the major M.

C. Sample Major Requirements

In order to illustrate the effect of the data visualization tool, consider a sample major $M = \{R_1, R_2, \dots, R_5\}$ with five requirements. Let C= $\{C1, C2, \dots, C25\}$ be the set of all courses available to satisfy requirements $R_1, R_2, \dots, R_5$.

Requirements are defined as follows:

R_1 : complete one of the courses $C1$ or $C2$

R_2 : complete one of the courses $C3$ or $C4$

R_3 : complete the courses $C5, C6, C13, C15, C17,$ and $C25$

R_4 : complete 12 units from the courses $C8, C9, C10, C11,$

$C12, C14, C16, C18, C19, C20, C21, C22, C23, C24$

R_5 : Complete one of the courses $C7$ or $M11$

The set of courses available to satisfy each requirement is defined as

$C_1 = \{C1, C2\}$,

$C_2 = \{C3, C4\}$,

$C_3 = \{C5, C6, C13, C15, C17, C25\}$,

$C_4 = \{ C8, C9, C10, C11, C12, C14, C16, C18, C19, C20,$

$C21, C22, C23, C24\}$, and

$C_5 = \{ C7, M11\}$.

An appropriate subset of the set C = $\{ C1, C2, \dots, C25\}$ needs to be selected to complete the major M. There may be other requirements associated with a major, such as unique requirements or minor requirements. Let us assume that there are two other unique requirements, U_1 and U_2 defined as follows:

U_1 : complete one of the courses $M8$ or $M9$

U_2 : complete one of the courses $C7$ or $M11$

Table I shows the prerequisite course structure for major requirements and Table II shows the prerequisite course structure for unique requirements.

A few of the prerequisite conditions are very complex, and some of the prerequisites are tied to course grades and courses from other disciplines. In general, prerequisites are completed, waived, transferred courses, or test scores that must be completed before taking a specific course.

The five requirements R_1 , R_2 ,…, R_5 are basic requirements that are easy to implement, but the prerequisites are very complex, and there are many possible ways of choosing courses to satisfy prerequisites and major requirements.

TABLE I. PREREQUISITE COURSE STRUCTURE FOR MAJOR REQUIREMENT

		Requirements				
Courses	**Prerequisites**	R_1	R_2	R_3	R_4	R_5
$C1, C2$	$M2$ or $M3$, with a grade of C or better	x				
$C3, C4$	($C1$ or $C2$) and ($M4, M5, or\ M6$), all with a C or better		x			
$C5$	$C1$ or $C2$			x		
$C6$	$C3$ or $C4$, with a grade of C or better			x		
$C7$	$M5, M7$ or $M8$, with a grade of C or better					x
$C8, C9, C10$	$C1$ or $C2$				x	
$C11$	$C3$ or $C4$				x	
$C12$	$C8$ and $C9$				x	
$C13$	$C6$ and ($C7$ or $M11$)			x		
$C14$	$C6$ and ($C7$ or $M11$)				x	
$C15, C16$	$C6$			x		
$C17, C18$	$C6$				x	
$C19, C20, C21, C22, C23, C24$	$C5$ and $C6$				x	
$C25$	$C5$			x		

Many existing academic planning tools utilize static tables like Table I or Table II for displaying course prerequisites. It is very difficult to understand complex prerequisite structures without drawing a directed graph.

D. Dependency Evaluation and Visualization (DEV) Chart

In this paper, we present a data visualization tool, which is named as Dependency Evaluation and Visualization (DEV) chart, to visualize course prerequisite structure. DEV chart uses an adjacency matrix of a directed graph D(V, E) to represent course structure where nodes (V) represent courses and edges (E) represent prerequisite relationships. Similarly, DEV chart can be used with project planning where nodes represent tasks and edges represent their dependencies. Tables I and II contain information needed to define adjacency matrices of the directed graphs for major requirements and other courses, respectively.

We define a Boolean valued prerequisite function $p: V \rightarrow \{true, false\}$ associated with the directed graph $D(V, E)$ such that $p(V) = true$ if prerequisite relation is satisfied for the course list attached to the node V, $p(V) = false$ otherwise. We also define a Boolean valued rotation function, $rt: C \rightarrow \{true, false\}$ such that $rt(c_k) = true$ if the course c_k is offered in the planning semester. Fig. 1 shows the DEV chart for major requirements, prior to completing any of the courses in the set C.

Fig. 2 shows DEV chart for unique requirements, prior to completing any of the courses. In Fig. 1, nodes with a stack of courses represent prerequisite courses where only one of the courses is needed to be taken to satisfy the prerequisite. If two or more arrows are pointing to the same child node, then each of the prerequisite relationships must be satisfied for the course list attached to the child node to be available.

TABLE II. PREREQUISITE COURSE STRUCTURE FOR UNIQUE REQUIREMENTS

Courses	**Prerequisites**
$M2, M3$	$M1$ with a grade of C or better
$M4, M5$	$M2$ with a grade of C or better or *M3* with a grade of B or better
$M6$	$M3$ with a grade of C or better
$M7$	$M5$
$M8$	$M4$ or $M5$, with a grade of C or better
$M9$	$M6$ or ($M5$ and $M7$), with a grade of C or better
$M11$	$M8$ with a grade of B or $M9$ with a grade of C

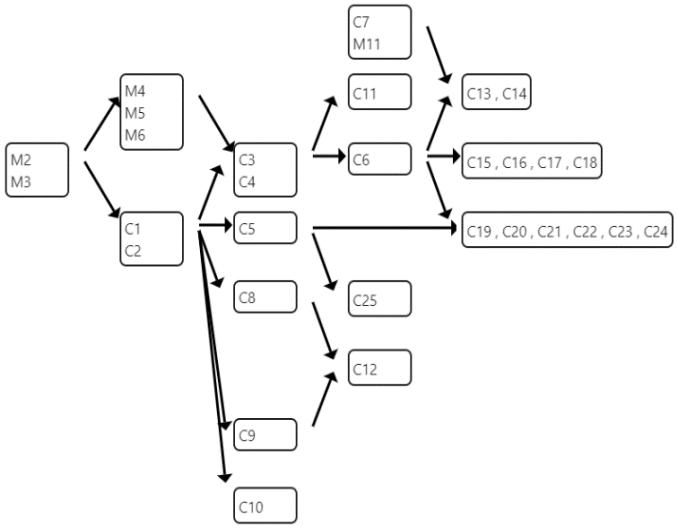

Fig. 1. DEV Chart for Major Requirements.

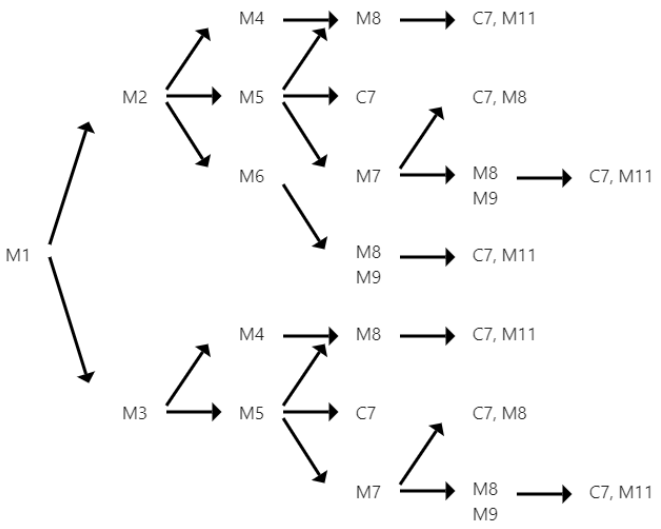

Fig. 2. DEV Chart for Unique Requirements.

TABLE III. MAJOR PROGRESS REPORT

Requirement	Satisfied?	Courses Taken	Courses Available
R_1	Yes	C1	
R_2	No	None	C3, C4
R_3	No	None	C5
R_4	No	None	C8, C9, C10
R_5	No	None	C7

When planning courses for a particular semester, students would normally have completed some of the courses required for the major and their prerequisites. It would be helpful to use a table similar to Table I to provide Academic Advising Report (AAR).

Table III shows the essential information that would be helpful for planning a major. It consists of a list of the major requirements, an indication of whether each requirement has been satisfied, and courses credited towards satisfying each requirement. The last column shows a list of courses available (prerequisites have already been satisfied) to satisfy the corresponding requirement, but not every AAR system has the capability to display such information.

The information in the last column of Table III is extremely valuable as it points to the courses that are available for planning the next semester. However, this information does not directly point to any bottleneck conditions that may prolong the graduation date. For example, students may plan the courses C8, C9, and C10 for the next semester and wait one more semester before taking either C3 or C4.

Note that the course C6 is a prerequisite for 12 of the 25 courses listed in Fig. 1. Hence, its prerequisites must be completed as soon as possible to minimize the time to complete the degree. Furthermore, courses C5 and C6 are prerequisites for six of the courses which are candidates for satisfying the requirement R_4. In this example, taking courses C5 and C6 would be the best choice for students seeking to minimize the degree completion time. The DEV chart is capable of conveying such useful information. Using degree progress report, the DEV charts in Fig. 1 and Fig. 2 can be updated dynamically to display the completed courses and the courses whose prerequisites have already been satisfied.

Fig. 3 and Fig. 4 represent an updated course structure, based on the completed courses and their grades. The color green is used to highlight completed courses whereas the color orange is used to highlight courses whose prerequisites are satisfied. Green arrows point to courses that are available to take in the next semester. Course grades are also displayed where * represents grades for the courses that are in progress and T represents transferred courses.

The course C13 is a required course for completing the requirement R_3, and its prerequisite is to complete C6 and either C7 or M11. Prerequisite for the course C7 is to complete either M5, M7, or M8, with a grade of C or better, whereas the prerequisite for M11 is to complete either M8 with a grade of B or M9 with a grade of C. Multiple paths exist for completing the prerequisite for the course C7 or M11. Based on the completed courses, the directed graph (Fig. 2) can be updated to narrow down the path choices.

Fig. 5 shows path choices after updating completed courses. DEV charts are created dynamically by updating data associated with each node of the directed graph. Hence, the DEV chart provides a mechanism for adding an alert system when prerequisite conditions are not met, as shown in Fig. 5.

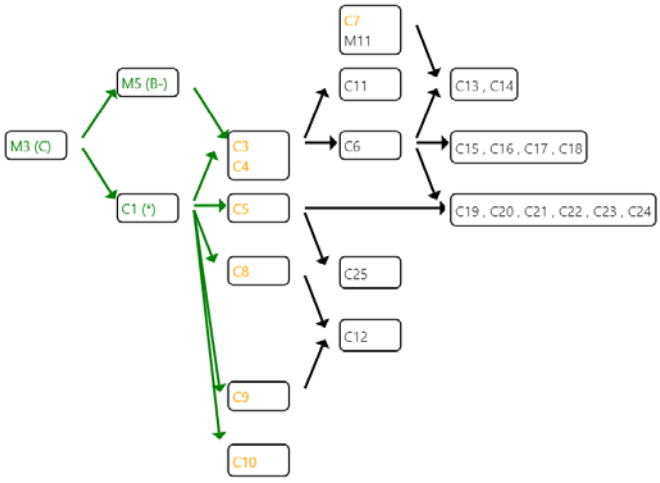

Fig. 3. Updated DEV Chart for Major Requirements.

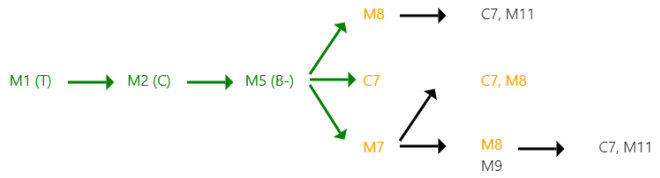

Fig. 4. Updated DEV Chart for Unique Requirements.

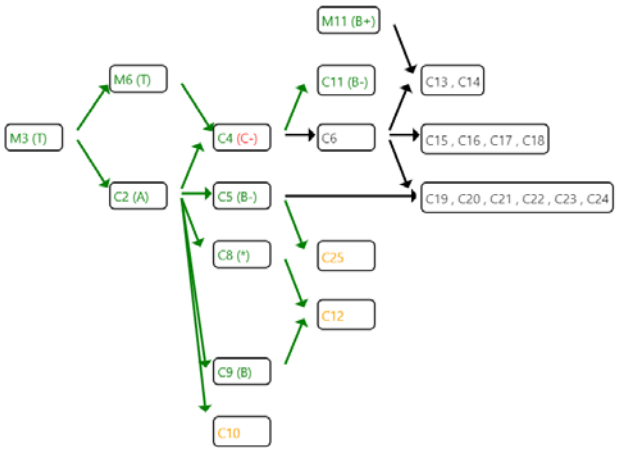

Fig. 5. Updated DEV Chart with Alerts for Major Requirements

The prerequisite for the course C6 is to complete either C3 or C4 with a grade of C or better, but the grade earned in this case is a C-, shown in color red, for the course C4. Subsequently, student has taken the course C11 that only requires a passing grade for C4. Instead of taking C11, the student should have repeated C6 for a better grade since C6 is a prerequisite for many other courses required for the major.

Most of the existing tools do not possess the ability to automatically alert a student as soon as the degree progress data or semester grades are updated. Therefore, many of the degree offering institutions rely on manual inspection to generate such alerts. The information displayed using a DEV chart can help students minimize degree completion time.

III. Implementation

DEV chart uses an adjacency matrix of a directed acyclic graph. We use custom-made tools to extract degree requirements and use basic requirement structure to store each requirement into a database. Similarly, we use custom-made tools to extract course descriptions and prerequisite relationships, and store the data using a format that is easier to process using any server-side scripting language. Course structure for a specific major is stored into a database using the corresponding adjacency matrix.

Fig. 6 shows the DEV chart that includes the major and unique requirements for Computer Science general emphasis major offered at University of Wisconsin-Whitewater (UWW). The DEV chart depicts the completed courses for an incoming freshman. In this example, student has earned credits for only one of the math courses (MATH 041) and eligible to take either MATH 139 or MATH 141.

Fig. 6. Course Structure for Computer Science Major General Emphasis at UW-Whitewater.

Fig. 7 shows the progress of the major requirements after completing some of the courses required for the major. Course grades are displayed where * represents grades for the courses that are in progress. The courses shown in orange are the courses whose prerequisites are satisfied. Green arrows point to courses that are available to take in the next semester. Note that COMPSCI 223 is a prerequisite course for most of the higher-level computer science courses. The prerequisite for 223 is a grade of C or better in either COMPSCI 220 or COMPSCI 222. In our test case, the student has earned a grade of C- for COMPSCI 220, as shown in color red. The red arrow associated with the course COMPSCI 220 is an indication that the student cannot take the COMPSCI 223 course until the prerequisite condition is satisfied. Such alerts can help students and advisors identify prerequisite issues. The Academic planning tool implemented at our institution does not have the ability to generate such alerts. Hence, these alerts can be very useful for academic advising.

Note that the COMPSCI 223 course is a prerequisite for many other COMPSCI courses that are either core courses or elective courses for the major. Therefore, students should make plans to take this course as soon as possible in order to graduate on time. It is difficult to identify such bottleneck courses without using a data visualization tool.

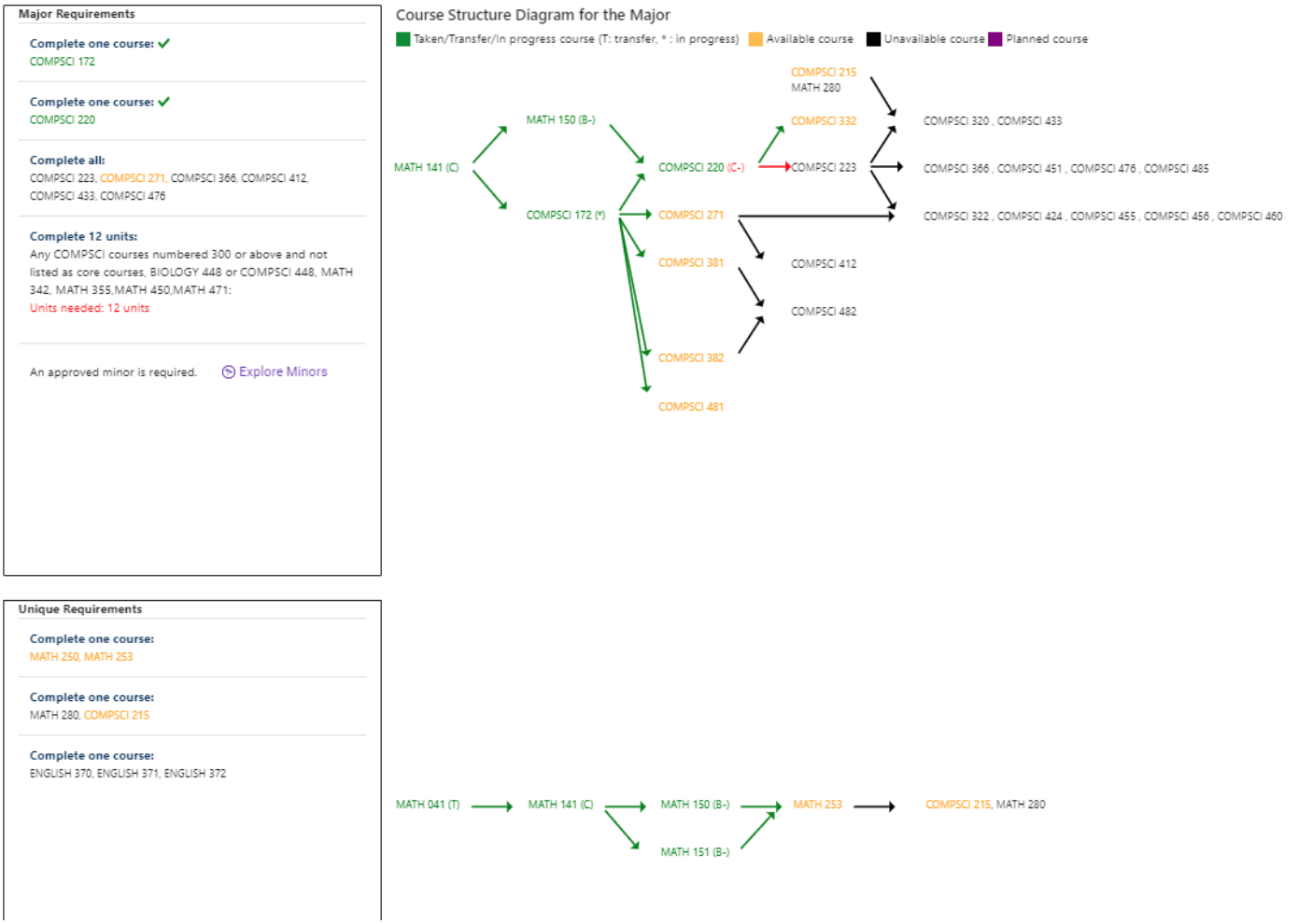

Fig. 7. Updated Course Structure for Computer Science Major General Emphasis at UW-Whitewater.

V. Conclusion and Future Work

We present a data visualization tool, DEV, for course prerequisite relationships. Our system is capable of interpreting one of, all of, either or, and, or any combination of those logical relationships. The implementation includes an interactive layout of major requirements and course relations.

Course prerequisites are very similar to task dependencies in project management. Completing the course prerequisites is similar to completing task dependencies: a task cannot be completed until the dependencies are completed. Hence, DEV chart can be extremely useful for project management where task dependencies play a major role. Furthermore, Dev charts can be useful for displaying information about graphs, such as cycles or specific branches/nodes satisfying a given criteria.

A pilot system has been successfully implemented for the Computer Science major offered at University of Wisconsin-Whitewater (https://cs.uww.edu/advising). This pilot system allows computer science majors to access an online advising system and interactively plan courses for their major, in consultation with their academic advisors. We are in the process of obtaining intellectual property rights for DEV charts.

The new data structure introduced in this research is extremely useful for analyzing student's academic progress toward a degree. We are exploring the possibility of developing a mechanism that would enable an undeclared student to explore requirements for all possible majors and explore the shortest path for graduation.

Acknowledgment

This work was supported by an NSF SBIR grant.

References

[1] J. F. Marques, "Hitting and missing the jackpot: The NACADA 2005 National Conference", The Mentor: An Academic Advising Journal, 2006.

[2] S. Ohrablo, S. Ohrablo, "High-Impact Advising: A Guide for Academic Advisors", United States: Academic Impressions, 2018.

[3] V.N Gordon, W.R. Habley, T. J. Grites, "Academic advising: A comprehensive handbook", John Wiley & Sons, 2011.

[4] F. Gutiérrez, K. Seipp, X. Ochoa, K. Chiluiza, T. De Laet, and K. Verbert, "LADA: A learning analytics dashboard for academic advising", 2018.

[5] H. Kerzner, Project Management: A Systems Approach to Planning, Scheduling, and Controlling, John Wiely and Sons, 2003.

[6] R. F. Aziz, "RPERT: Repetitive-Projects Evaluation and Review Technique", vol 53, pp. 81-93, 2014.

[7] L. Zhou, J. Xie, X. Gu, Y. Lin, P. Leromonachou, X. Zhang, "Forecasting return of used products for remanufacturing using Graphical Evaluation and Review Technique (GERT)", International Journal of Production Economics, vol 181, pp. 315-324, 2016.

[8] R. G. Nelson, A. Azaron, S. Aref, "The use of GERT based method to model concurrent product development processes", vol 250, pp. 566-578, 2016.

[9] A. Dechter, "Model based student academic planning", International Journal of Applied Management and Technology, vol 5, pp. 87-104, 2007.

[10] A. Dechter, "Facilitating timely completion of a college degree with optimization technology", Association for the Advancement of computing in Educational Journal, vol 17, pp.215-229, 2009.

[11] K. Kowalski and D. Ealy, "Schedule advisement expert system", Computers in Human Behavior, vol 17, pp. 259-265, 1991.

[12] G. A. Moreno, W. F. Bischof, H. J. Hoover, "Interactive visualization of dependencies", Computers and Education, vol 58, pp. 1296-1307, 2012.

[13] P. R. Aldrich, "The curriculum prerequisite network: a tool for visualizing and analyzing academic curricula", arXiv preprint arXiv:1408.5340, 2014.

[14] J. Chen and H. Siyuan, UBCourse Vis, 2017.

[15] K. E. wilcox, L. Huang, "Network models for mapping educational data", vol. 3, doi:10.1017/dsj.2017.18, 2017.

[16] R. Zucker, "ViCurriAS: a curriculum visualization tool for faculty, advisors, and students" Journal of Computing Scienes in Colleges, vol 25, pp. 138-145, 2009.

[17] H. Siirtola, K-J Raiha, V.Surakka, "Interactive curriculum visualization", Proceedings of the 17th International Conference on Information Visualization, vol IV, pp. 108-117, 2013.

[18] A. S. Phadke, S. S. Kulkarni, "Use of Network Model for Analysis of Circulum and its Mapping to Program Outcomes", Journal of Engineering Education Transformations, vol 31, pp. 30-34, 2018.

[19] P. Simonetto, D. Archambault, S. Kobourov, "Event-Based Dynamic Graph Visualization", IEEE Transactions on visualization and computer graphics, vol 26, 2020.

[20] J. A. Cottam, A. Lumsdaine, C. Weaver, "Watch this: A taxonomy for dynamic data visualization", 2012 IEEE Conference on Visual Analytics Science and Technology (VAST), doi: 10.1109/VAST.2012.6400552, 2012.

[21] F. Beck, M. Burch, S. Diehl, D. Weiskopf, "A taxonomy and survey of dynamic graph visualization", Wiley Online Library, https://doi.org/10.1111/cgf.12791 , 2016.