Athens Journal of Technology and Engineering - Volume 9, Issue 4, December 2022 – Pages 267-280

Dependency Visualization Tool for Decision Support Systems with Preferential Dependencies

By Sobitha Samaranayake[*] & Shehan Senanayake[±]

Large-scale and long-term projects often consist of subtasks linked to each other with certain dependencies that need to be completed. Dependency evaluation and visualization tools can help identify project bottlenecks, prioritize tasks, and plan resource allocation. Hence, visualization tools can improve the efficiency of project interdependency management by dynamically displaying the project constraints and task dependencies. Many of the dependency visualization tools that are being used in the industry are designed to manage internal dependencies and task dependencies. In this work, we introduce an online tool for creating and maintaining visualizations of logical dependencies and preferential dependencies. The proposed online tool can be used with any process that is defined in terms of completing a well-defined logical sequence of activities that must be completed in order to reach a specific target, such as project planning, or with any process that can be completed by different sequences of activities, such as college degree planning (due to elective courses) or interest-aligned career planning.

Keywords: *academic decision making, decision support system, data visualization, dependency visualization, project management*

Introduction

There are different types of decision support systems (DSSs), such as data-driven systems, knowledge-driven systems, and model-driven systems. Both data-driven and knowledge-driven systems use either a file system or a data warehouse to store data, and these systems use charts, maps, graphs, or other graphic tools to visualize data. Model-driven DSSs are created for special purpose decision support needs and they can help analyze different scenarios.

Model-driven DSSs are used for analyzing project planning and scheduling problems. Large-scale and long-term projects often consist of subtasks linked to each other with certain dependencies that need to be completed. There are different types of task dependencies, such as logical, constraint-based, or preferential dependencies.

Logical dependencies are the tasks of a project that are necessary for a project's completion. They are the output for all of the preceding tasks and may not run parallel with other tasks. Tasks may have multiple preceding tasks and multiple succeeding tasks. Predecessors must complete before successors can start so completing logical dependencies can be considered as a scheduling problem.

[*]Associate Professor, University of Wisconsin-Whitewater, USA.
[±]Graduate Student, University of Wisconsin-Whitewater, USA.

https://doi.org/10.30958/ajte.9-4-1 doi=10.30958/ajte.9-4-1

Many visualization tools are available for creating charts, maps, and graphs for logical dependencies. Program Evaluation and Review Technique (PERT) (Kerzner 2003) is a project management tool that is widely used to visualize the timeline and the work that must be done to complete a project. The critical path is the longest sequence of tasks that must be completed to complete a project. Gantt charts (Clark 1922) are used for calculating the critical path and showing tasks displayed against time, and hence they are commonly used for tracking the completion of logical dependencies.

Preferential dependencies occur when there are many possibilities for completing a sequence of tasks. There may be many options for completing a particular task and the choices for completing preceding tasks may influence the choices for succeeding tasks. Hence, choosing a particular path to complete a sequence of such task dependencies may depend on user preferences, such as cost, time, quality, or resource allocation.

Data visualization tools help decision makers to match their creativity and background knowledge with the enormous storage and processing capabilities of advanced computing systems. Using advanced visual interfaces, humans can interact directly with data analysis (Alves and Cota 2018). There is a dearth of visualization tools for systems with preferential dependencies, especially when there are many choices for each task. This paper will focus on developing visualization tools for systems with preferential dependencies. We will consider a generic model that can represent a range of systems where the goal is to select a subset of a list of tasks and then complete each task in that subset. After reviewing the scope of the current visualization tools, we present a visualization tool that is capable of visualizing a variety of preferential dependencies.

Literature Review

Decision support systems have been used for a wide variety of decision-making activities. Power (2008) investigated historic foundation of data-driven DSSs and reported that data-driven DSSs have been used for many decision-making activities, such as air-defense command and control, geographic information systems, customer relationship management systems, static and real-time performance monitoring, and executive information systems. DSSs help businesses in many different ways, such as generating more accurate projections, better inventory management, data analysis, sales optimization, and optimizing industry-specific systems (Trieu 2017, Power 2013, Feng 2016, Rouhani et al. 2018).

Web based DSSs adds a new dimension to the evolution of DSSs, and many of the information systems people interact with are powered by decision support systems. Wright et al. (2009) analyzed three case studies to evaluate the potential of creating and sharing clinical DSSs with Web 2.0 technologies. Sugumaran and Sugumaran (2007) analyzed the evolution of web-based spatial DSSs. Krishnaiyer and Chen (2017) explained an implementation of a web-based visual DSS that helped an organization to successfully turn around the scheduling and capacity planning function along with extending the success to customer service, and

warehouse operations.

Knowledge-driven DSSs use facts, rules, and procedures to analyze data to help users make intelligent decisions. Hence, knowledge-driven educational DSS can provide valuable information to students, educators, and educational organizations, but developing an educational DSS is a complex process. Due to the flexibility of a credit system of education, degree requirements are different from organization to organization, hence, applying data mining techniques to educational data is challenging. Vo and Nguyen (2012) proposed a knowledge-driven educational decision support system for education with a semester credit system by taking advantage of educational data mining. Anardani et al. (2019) discussed the design of student performance monitoring information systems by applying the business intelligence concept. Moscoso-Zea et al. (2019) presented an infrastructure to analyze educational data and academic processes and for the creation of explicit knowledge using different algorithms and methods of educational data mining.

Model-driven DSSs are helpful for academic advisors to make more appropriate and reasonable decisions about student's studies and give further support to students for their course planning. There has been interest in designing a DSS for course planning. Siddiqui et al. (2018) introduced a web-based group DSS for academic term preparation at a business college of a large Middle Eastern university. Roushan et al. (2014) presented a DSS for course planning. Miranda et al. (2012) developed a web-based DSS for course and classroom scheduling. Oladokun and Oyewole (2015) presented a DSS for university admission seekers. Al-Qaheri et al. (2011) presented a DSS for a course scheduler. Meyer et al. (2021) presented an application of information technology in providing decision support in college planning.

Visual representation in DSSs has been discussed by many researchers. Basole et al. (2016) analyzed the effectiveness of using lists, matrices, and network visualization for business ecosystem analysis. Kuru et al. (2014) demonstrated the benefits of using machine learning algorithms and digital image processing techniques to build an intelligent diagnostic DSS in medical genetics. Hennocq et al. (2021) presented a literature review of all the methods for analyzing 2D pictures for diagnostic purposes. Interactive data visualization dashboards are being used for visualizing and analyzing trends in large volumes of data (Huber et al. 2018, West et al. 2015, Morgan et al. 2006, Nevo et al. 2015). Trase and Fink (2014) designed a model-driven visualization tool for use with model-based engineering projects. Burnay et al. (2019) analyzed some recent approaches of DSSs and information systems that present domain-specific and visualization-based user communication strategies and interfaces to support decision-making.

Overview of the Current Visualization Tools

Over the past couple of decades, there has been an interest in improving the efficiency of large projects that require the completion of multiple tasks in order to achieve the end goal. This has become its own process which is now done by people who work in positions such as project managers and academic advisors.

Although there are many different ways of presenting the information related to multiple tasks, such as using a text file to document the steps, the most efficient method of presenting such information is by using a visualization of the process.

We consider model-driven systems where the requirement is to complete a sequence of tasks. There are two main types of dependencies between tasks: logical dependencies and preferential dependencies. Having some sort of visualization makes the project planning or task scheduling more convenient as it is easier to read and understand certain dependencies between the tasks that make up the bigger project. Many visualization tools exist for systems where predecessors must complete before successors can start. Project management is such a system, and different visualizations such as flow charts, Unified Modeling Language (UML) diagrams, Program Evaluation and Review Technique (PERT) charts, and Gantt charts are used to visualize the required information. Flow charts and UML diagrams visualize the workflow and actions taken by the client of a specific project to help the team understand what features the product needs to support. Flow charts can display a process step by step. However, they focus on visualizing processes that require decision making based on how to do something instead of what to do. On the other hand, PERT charts visualize the step-by-step tasks that need to be completed by a team in order to complete the project. However, the PERT chart only displays sequential dependencies and concurrent dependencies. Also, the PERT chart does not display any sort of progress along the graph as it is a static graph once it is generated. This means the user has to keep track of where they are, or they will have to find their current task first and then look for the next step each time they refer to the visualization.

Similar to PERT charts, Gantt charts also visualize the tasks that need to be completed by a team. Gantt charts use a table format where the columns represent a time frame (days or months) and the rows represent the specific tasks. This allows the user to schedule tasks. An additional feature this provides is the ability to track the progress of each task as it shows a completed percentage. A disadvantage to using the Gantt chart is that it is not capable of showing dependencies between tasks as the tasks are usually displayed in each row, and the process is like a waterfall where you complete the top one and move on to the next. This makes processes such as academic advising very inconvenient as the path a student has to follow for a degree usually has classes with prerequisites.

Three main visualization systems that are currently used in the industry are Lucidchart (Faulkner 2018), Visio Data Visualizer (Parker 2016), and ProjectManager (tool provided by Projectmanager.com). Lucidchart is one of the most used tools in the industry right now as it allows a lot of functionality for all types of processes. However, the user has to create the visualizations using the drag and drop method and a list of built-in shapes. Using this technique, the user can create a static diagram with the information they want to display. Similarly, the Visio Data Visualizer allows the user to create a table in Excel and export it as a Visio diagram. This will generate a process diagram using the information from the table. Both Lucidchart and Visio do a great job at creating process diagrams similar to flowcharts, but they focus more on how to do something instead of what to do. Additionally, the visualization they create is static and the user cannot

interact with it to track their progress. The tool from Projectmanager.com on the other hand is able to track user progress. This tool allows the user to import data from a CSV file as well and the user can drag and drop the initial visualization. However, it usually tracks the progress of each individual task instead of the overall process.

A dependency graph is a flow chart or a directed graph representing relationships between the nodes. There are several tools available for creating static dependency graphs. Sankey diagram is a graph that is used for visualizing a flow from one set of values to another. Lupton and Allwood (2017) presented a common data structure and a systematic method for generating different hybrid Sankey diagrams from a dataset. Oran et al. (2019) used Sankey diagrams to identify and visualize progress and mobility patterns in higher education. Horvath et al. (2018) used Sankey diagrams to visualize student flows to track retention and graduate rates. Designing a dependency visualization tool for systems with preferential dependencies is a complex process since the graph needs to be dynamically updated, based on the preferences. In this work, we introduce a visualization tool that is capable of visualizing a variety of preferential dependencies.

Method

Consider a model where the goal is to select a subset of a predefined list of tasks and then complete each task in that subset using the following requirements:

- Each task may consist of a set of subtasks such that a task can be completed by either type a: completing k subtasks from a set of p subtasks where $1 \leq k \leq p$, or type b: completing at least m subtasks, but no more than n subtasks from a set of p subtasks where $0 \leq m \leq n \leq p$, or completing a combination of type a and type b conditions.
- Predecessors must complete before successors
- Some of the subtasks are predecessors to multiple tasks.

There are many systems that can be modeled using the above requirements. None of the existing visualization tools are able to display the task dependencies described by the above model.

One of the goals of this work is to develop visualization tools to analyze the following questions for systems represented by the above model:

- What are the dependencies between the tasks?
- What is the current progress of the project?
- What are the next possible steps?

We answer these questions using a graph-based dependency visualization tool (DVT) which is implemented as a directed graph where the nodes represent a subset of tasks and the edges represent the dependencies.

We use a data model that incorporates a flexible grid system $G(N, D, O)$ to represent tasks and their dependencies as a directed graph where $N = \{N_1, N_2, \dots, N_{n\}}$ is a set of nodes, D is the dimension of the grid, and O is the orientation. Each node N_i is a task expressed using a tuple $N_i = (L, M, C, P, D, S, \delta)$ where L is the location of the node, M is a set of subtasks associated with the node, C is a logical condition on M that defines the rule for completion of a task, P is a set of predecessors, $, D$ is a logical condition that defines dependencies on preceding nodes, S is a set of successors, and $\delta: M \rightarrow \{1, 0\}$ is a function such that $\delta(M) = 1$ if M satisfies the logical condition C and $\delta(M) = 0$ otherwise. (This function is used to check if the task is complete). Nodes of the directed graph can be arranged vertically or horizontally. Our data model allows us to easily modify node locations and edges between the nodes.

To demonstrate the potential of the DVT, we will now look at a few example scenarios.

Implementation

College degree planning involves selecting a set of courses to fulfill degree requirements. Degree requirements are defined in terms of number of units or courses that needs to be completed. Many courses specify prerequisites or course dependencies.

There are more than 150 major emphasis areas offered at University of Wisconsin-Whitewater (UWW), and each of those emphasis areas is defined using a set of requirements. A typical 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

Figure 1 displays degree requirements for computer science major at UWW. Computer science students should complete three courses in software development fundamentals, five core courses, and twelve units of elective courses (four elective courses).

Many courses specify prerequisite relationships that are often defined using one of, all of, either or, and, or a combination of those logical relationships among a list of predefined courses. There may be prerequisites of prerequisite courses. Hence, a directed acyclic graph is the best structure to represent prerequisite relationships where nodes represent course lists and edges represent their dependencies.

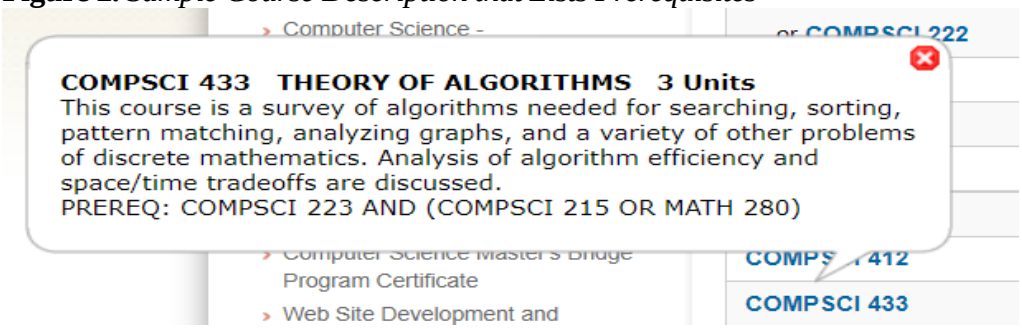
Figure 1. *Degree Requirements for Computer Science Major at UWW*

Computer Science Requirements (BA/BS)

Major Requirements: [1]		
Software Development Fundamentals:		
COMPSCI 172 or COMPSCI 174	INTRODUCTION TO JAVA INTRODUCTION TO C++	3
COMPSCI 220 or COMPSCI 222	INTERMEDIATE JAVA INTERMEDIATE C++	3
COMPSCI 223	DATA STRUCTURES	3
Core Courses:		
COMPSCI 271	COMPUTER ORGANIZATION AND ASSEMBLY PROGRAMMING	3
COMPSCI 366	DATABASE MANAGEMENT SYSTEMS	3
COMPSCI 412	COMPUTER ORGANIZATION AND SYSTEM PROGRAMMING	3
COMPSCI 433	THEORY OF ALGORITHMS	3
COMPSCI 476	SOFTWARE ENGINEERING	3
Electives: [2]		**12**
Any COMPSCI courses numbered 300 or above (no limit)		
Up to 6 total units in this category may come from the following courses:		
MATH 355	MATRICES AND LINEAR ALGEBRA	
MATH 450	GRAPH THEORY	
MATH 471	NUMERICAL ANALYSIS	
STAT 342	APPLIED STATISTICS	
Total Units		**36**

Figure 2 shows a sample course description that lists course prerequisites. Going through a text file to look for classes that need to be completed is time consuming and the advisor has to check course perquisites to make sure that the student is eligible to take each of the planned courses.

Figure 2. *Sample Course Description that Lists Prerequisites*

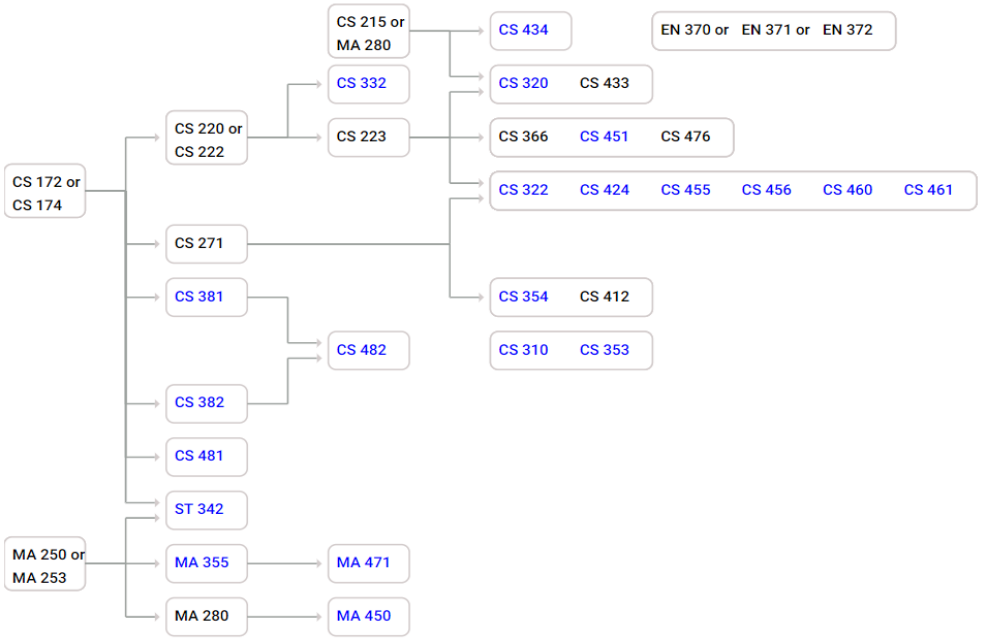


We use a predefined pattern to store a logical condition that defines dependencies on preceding nodes. For example, COMPSCI 223 and either COMPSCI 215 or MATH 280 must be completed before taking COMPSCI 433. We consider such a condition as completing two tasks *A* and *B* where *B* is also another task of completing one of the two tasks, task *C* or task *D*. We created an online graph visualization tool to evaluate course dependency structure, based on

the requirements for each major. Since the courses that are completed by a student are stored in the database, this information can be used to display the current progress for each student using a directed graph. Students meet with their academic advisor to create a semester plan prior to enrolling for courses each semester. Using this visualization tool, the student and the advisor can easily explore the possible courses to take, based on the prerequisites and completed courses.

Figure 3 shows an implementation of our DVT which is based on the work by Samaranayake and Gunawardena (2020) to allow students to visualize degree paths. The objective is to display course dependencies to help computer science majors create a degree plan. In this implementation, we use a single node to represent a list of courses from which a subset of those courses needs to be completed. If two or more arrows are pointing to the same child node, then all the parent nodes are prerequisites to the child node. Courses shown in blue are the elective courses.

Figure 3. *Course Prerequisite Structure*



Computer science majors at UWW should complete 12 courses in order to satisfy the major requirements. There are more than 15,000 ways of completing those 12 courses: three courses in software development fundamentals, five core courses, and four of the elective courses. Many of the core courses and elective courses have common course dependencies. Hence, planning courses to create a degree path without using any visualization tools is a tedious and time-consuming process.

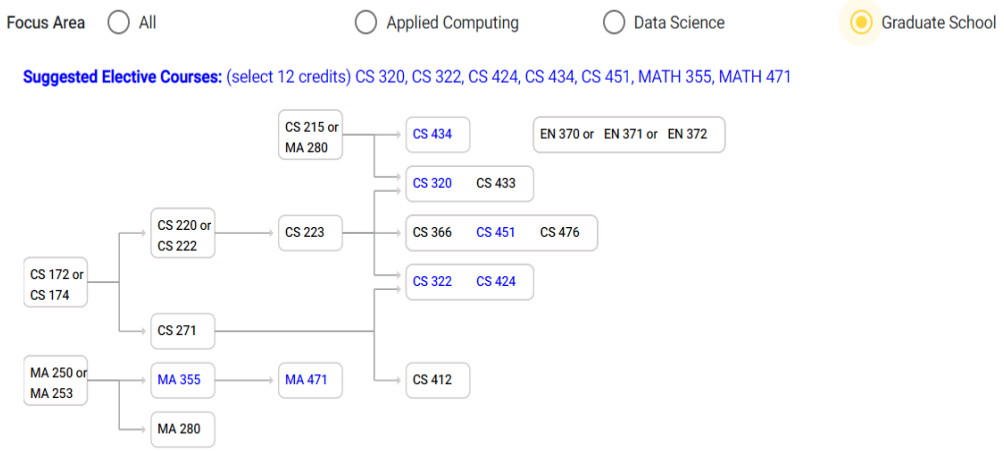
In general, graph representation is not unique. Hence, dependencies can be represented by many different directed graphs, based on user preferences or the type of dependencies. Suppose a particular task is defined as completing a subset of many predefined subtasks. We may have to use a separate path for each of those

subtasks that have different predecessors. We will have to use a separate path for each subtask if we prefer to display all possible paths. Hence, the graph representation may depend on the user requirements, task dependencies, and/or type of the information represented by the system. For example, condition for fulfilling course dependencies of COMPSCI 433 (CS 433), as shown in Figure 2, can be defined using three subtasks: completing CS 223, CS 215, and MA 280. In Figure 3, those dependencies are displayed using different paths for each subtask that has a different predecessor. The prerequisite condition for CS 482 is to complete both CS 381 and CS 382, and those dependencies are displayed in Figure 3 using a separate path for each subtask even though they are two subtasks with the same predecessor. In this example, the choice of the path depends on whether subtasks are predecessors to multiple tasks or not.

Students are required to select 4 of the 22 elective courses for the computer science major. Students may select those 4 elective courses based on their course preferences, career goals, schedule preferences, or to optimize the time to graduate.

The data model is capable of dynamically updating the directed graph by adding new nodes to the graph or removing existing nodes from the directed graph. Hence, our tool provides a mechanism to filter elective courses based on their career goals or other preferences. Figure 4 shows a list of suggested elective courses from which students may complete four courses if they plan on pursuing graduate studies.

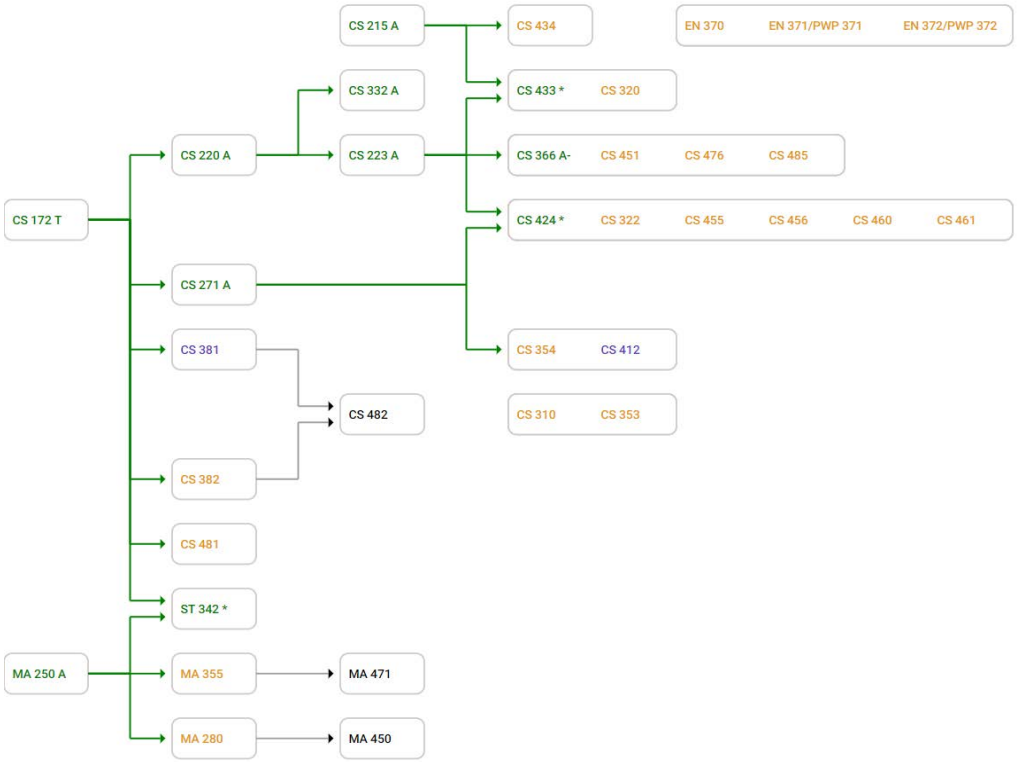
Figure 4. *Dynamic Filtering of Suggested Elective Courses*



The data model that is used to generate these DVTs allows us to store the directed graph so that we can dynamically modify the node structure. Hence, the course prerequisite structure can be updated at the end of each semester to help students select courses based on the updated dependencies. Figure 5 shows a dynamically updated course prerequisite structure which is being used for academic advising. The course structure is updated dynamically to narrow down the course choices, based on the completed and planned courses. 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, and the

courses shown in purple are the courses planned for the next semester. Green arrows point to courses that are available to take in the next semester. Two arrows pointing to the same block means both predecessors must be completed before starting the successor.

Figure 5. *Dynamically Updated Course Prerequisite Structure*



Visualization tools are extremely useful for identifying any bottleneck conditions that may prolong the completion of the process. For example, CS 223 and CS 271 are prerequisite courses for many of the 300-level or higher computer science courses. Hence, those two courses and their prerequisites must be completed as soon as possible to minimize the time to complete the degree.

The data model allows us to link each node to another directed graph. This feature is very useful for complex systems where each task is a subsystem that can be represented by our model. For example, Figure 3 includes a list of courses that can be used for planning the computer science major but there may be other hidden prerequisite courses that may not count for the computer science major. Figure 6 shows the implementation of course prerequisite structure for CS 172 and CS 174 as a directed graph linked to those two courses.

Figure 7 shows a directed graph of proficiency courses for different subject areas. This online application is being used to educate new students about the proficiency requirements for different subject areas. It is a very efficient method of sharing information that is usually scattered among many static documents.

Figure 6. *Course Prerequisite Structure for CS 172 and CS 174*

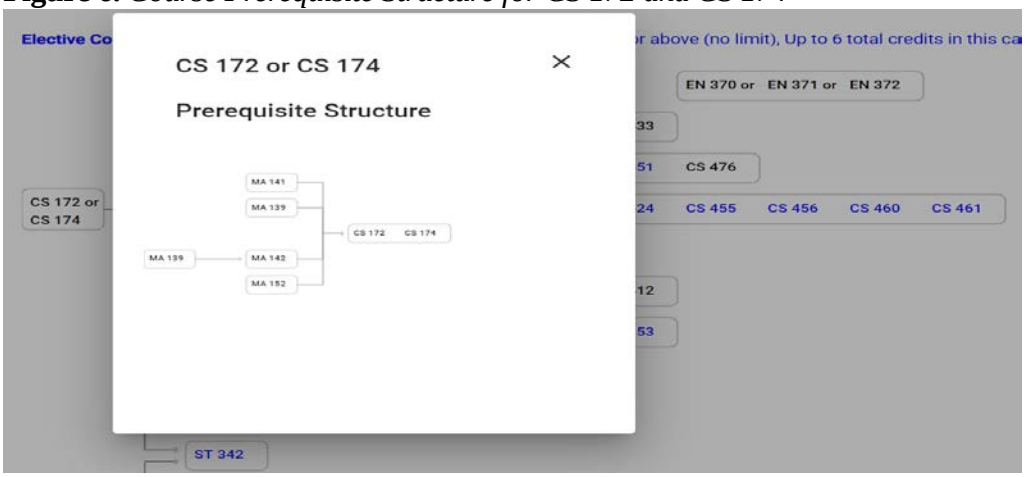
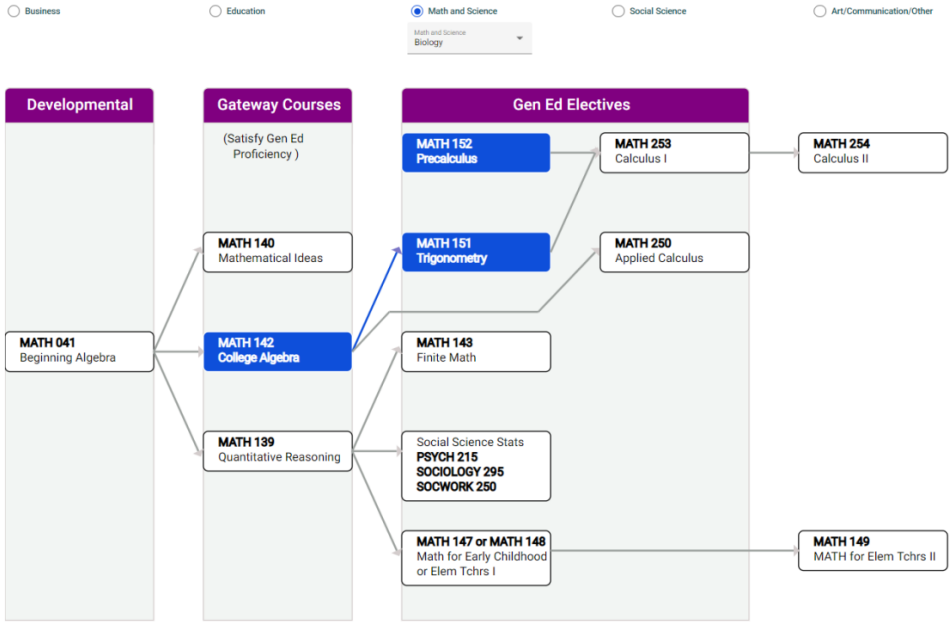


Figure 7. *Proficiency Course Structure*



There are many such online applications where the DVTs are very useful for helping users make decisions. DVTs are also useful for managing long-term projects with multiple different ways to get to the end goal where the use of the flow chart can help project managers view current progress and preferential paths for project completion.

The data model we incorporate to generate these DVTs is very powerful and flexible. It allows us to dynamically modify the nodes and edges, format arrows connecting the nodes, and apply themes to format the appearance. Additionally, the orientation of the graph can be modified by applying rotations.

Conclusions

We present an online DVT which is useful for analyzing interactive systems with preferential dependencies, such as interest-aligned degree planning, project management, or career planning. Our implementation of the online DVT system to visualize the course prerequisite structure proved to be very effective and significantly faster than the traditional method of looking up indirect course dependencies and doing reverse prerequisite lookups. DVT system was able to help students quickly and easily find answers to questions like "If I have taken these courses, then what courses can I take next?" or "Why cannot I take this course yet?". Furthermore, the DVT helped students to identify the bottleneck scenarios in course planning. The DVT system that displays the proficiency course structure has been well received by academic advisors across several universities.

Data visualization tools are shown to be extremely useful for analyzing different scenarios to help make decisions about such systems. Also, online DVTs can help content management systems organize and display related information using a compact and efficient format. Furthermore, each node can represent a directed graph so the visualization tool can be used to analyze complex systems.

References

Al-Qaheri H, Hasan MK, Al-Husain R (2011) A decision support system for a three-stage university course scheduler with an application to College of Business Administration, Kuwait University. *International Journal of Data Analysis and Information Systems* 3(2): 95–110.

Alves CMO, Cota MP (2018) Visualization on decision support systems models: literature overview. *Advances in Intelligent Systems and Computing* 745: 730–740.

Anardani S, Sofyana LS, Maghfur A (2019) Analysis of business intelligence system design for student performance monitoring. *Journal of Physics Conference Series* 1381(1): 012015.

Basole RC, Huhtamäki J, Still K, Russell MG (2016) Visual decision support for business ecosystem analysis. *Expert Systems with Applications* 65(Dec): 271–282.

Burnay C, Dargam F, Zarate P (2019) Special issue: data visualization for decision-making: an important issue. *Operational Research* 19(Nov): 853–855.

Clark W (1922) *The Gantt chart: a working tool of management*. Ronald Press Company.

Faulkner A, (2018) Lucidchart for easy workflow mapping. *Serials Review* 44(2): 157–162.

Feng L (2016) Intelligent logistics and distribution system based on Internet of Things. In *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 228–231.

Hennocq Q, Khonsari RH, Benoît V, Rio M, Garcelon N (2021) Computational diagnostic methods on 2D photographs: a review of the literature. *Journal of Stomatology, Oral and Maxillofacial Surgery* 122(4): 71–75.

Horvath DM, Molontay R, Szabo M (2018) Visualizing student flows to track retention and graduation rates. In *2018 22nd International Conference Information Visualisation (IV)*, 338–343.

Huber TC, Krishnaraj A, Monaghan D, Gaskin CM (2018) Developing an interactive data visualization tool to assess the impact of decision support on clinical operations. *Journal of Digital Imaging* 31(5): 640–645.

Kerzner H (2003) *Project management: a systems approach to planning, scheduling, and controlling*. Hoboken, NJ: Wiley.

Krishnaiyer K, Chen FF (2017) Web-based visual decision support system for letter shop. *Robotics and Computer-Integrated Manufacturing* 43(Feb): 148–154.

Kuru K, Niranjan M, Tunca Y, Osvank E, Azim T (2014) Biomedical visual data analysis to build an intelligent diagnostic decision support system in medical genetics. *Artificial Intelligence in Medicine* 62(2): 105–118.

Lupton RC, Allwood JM (2017) Hybrid Sankey diagrams: visual analysis of multidimensional data for understanding resource use. *Resources, Conservation and Recycling* 124(Sep): 141–151.

Meyer RM, Gunawardena ADA, Samaranayake S, Deshpande V, Premadasa K (2021) Chapter 14: information technology for student decision support in College planning. In *Computing Technologies and Applications: Paving Path Towards Society 5.0*. Taylor & Francis Group.

Miranda J, Ray PA, Robles JM (2012) udpSkeduler: a web architecture based decision support system for course and classroom scheduling. *Decision Support Systems* 52(2): 505–513.

Morgan MB, Branstetter BF, Mates J, Chang P (2006) Flying blind: using a digital dashboard to navigate a complex PACS environment. *Journal of Digital Imaging* 19(1): 69–75.

Moscoso-Zea O, Castro J, Paredes-Gualtor J, Lujan-Mora S, (2019) A Hybrid infrastructure of enterprise architecture and business intelligence analytics for knowledge management in education. *IEEE Access* 7(Mar): 38778–38788.

Nevo D, Nevo S, Kumar N, Braasch J, Mathews K (2015) Enhancing the visualization of big data to support collaborative decision-making. In *2015 48th Hawaii International Conference on System Sciences*, 121–130.

Oladokun V, Oyewole DI (2015) A fuzzy inference based decision support system for solving the university-course admission choice problem. *International Journal of Computer Applications* 112(3): 10.5120/19643-1229.

Oran A, Martin A, Klymkowsky M, Stubbs R (2019) *Identifying students' progress and mobility patterns in higher education through open-source visualization*. Working Paper. University Administration Faculty and Staff Contributions, 1.

Parker DJ (2016) *Mastering data visualization with Microsoft Visio professional 2*016. Packt Publishing Ltd.

Power DJ (2008) Decision support systems: a historical overview. In *Handbook on Decision Support Systems 1*. *International Handbooks Information System*, 121–140. Berlin, Heidelberg: Springer.

Power DJ (2013) Mobile decision support and business intelligence: an overview. *Journal of Decision Systems* 22(1): 4–9.

Rouhani S, Ashrafi A, Ravasan AZ, Afshari S (2018) Business intelligence systems adoption model: an empirical investigation. *Journal of Organizational and End User Computing (JOEUC)* 30(2): 43–70.

Roushan T, Chaki D, Hasdak O, Chowdhury M, Annajiat A, Rahman M, et al. (2014) University course advising: Overcoming the challenges using decision support system. In *16th International Conference on Computer and Information Technology (ICCIT)*, 13–18.

Samaranayake S, Gunawardena ADA (2020) Dependency evaluation and visualization

tool for systems represented by a directed acyclic graph. *International Journal of Advanced Computer Science and Applications* 11(7): 1–7.

Siddiqui AW, Raza SA, Tariq ZM (2018) A web-based group decision support system for academic term preparation. *Decision Support Systems* 114(Oct): 1–17.

Sugumaran, V, Sugumaran R (2007) Web-based spatial decision support systems (WebSDSS): evolution, architecture, examples and challenges. *Communications of the Association for Information Systems* 19(1).

Trase K, Fink E (2014) A model-driven visualization tool for use with model-based systems engineering projects. In *2014 IEEE Aerospace Conference*, 1–10.

Trieu V (2017) Getting value from business intelligence systems: a review and research agenda. *Decision Support Systems* 93(Jan): 111–124.

West VL, Borland D, Hammond WE (2015) Innovative information visualization of electronic health record data: a systematic review. *Journal of the American Medical Informatics Association* 22(2): 330–339.

Wright A, Bates DW, Middleton B, Hongsermeier T, Kashyap V, Thomas SM, et al. (2009) Creating and sharing clinical decision support content with Web 2.0: Issues and examples. *J Biomed Inform* 42(2): 334–346.

Vo TNC, Nguyen HP (2012) A knowledge-driven educational decision support system. In *2012 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future*, 1–6.