Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 17598–17603

COnCUR - COherence in CURricula: A tool to assess, analyze and visualize coherence in higher education curricula

Emil Wengle [*], **Steffi Knorn** [**,*], and **Damiano Varagnolo** [***]

[*] *Uppsala University, Sweden*

[**] *Otto-von-Guericke University Magdeburg, Germany*

[***] *Norwegian University of Science and Technology, Norway*

Abstract: We describe COnCUR, a tool to visualize the structure of a university program in terms of the courses and knowledge components included in the program, the connections between them as well as detecting, analyzing and visualizing inconsistencies within the program structure using graph theoretical concepts. The tool is especially designed for those who are involved in the planning of study programs, since enabling to create maps linking the various content and levels of abilities in the various courses (and thus to evaluate if the prerequisite of a student are compatible with a course, especially in the case of personalized programs).

Keywords: Knowledge Component, Knowledge Flow, Course Flow Matrix

1. INTRODUCTION

It is well known that obtaining consistent and transparent conceptualizations of the knowledge contents required and developed within a course or across courses in a program can provide a clear basis for creating structures and progressions that better supports student learning. For example, a well known strategy towards reaching such a goal is the so-called *black-box* approach within the Conceiving, Designing, Implementing, and Operating (CDIO) standard: this exercise, that aims at sequencing a curriculum, has been proposed as part of the CDIO standard for helping the management of university programs, and entails representing every course within a program as a set of inputs (e.g., prerequisite skills and knowledge) and outputs (e.g., contributions to the final learning outcomes). Coupling this information with all courses is expected to enable discussions, highlight connections (or lack thereof) among courses, provide an explorable and explainable overview of the program, and serve as a structured basis for both planning and improving curricula. However, this tool is still qualitative in nature, and does not provide quantitative indications that are free from personal interpretations. Further, most universities have not adapted the CDIO approach (and might not even plan to do so).

Continuous adaptations of the curricula are thus usually required during the entire lifespan of every university program. And even in the best designed and aligned programs, courses may be moved within the curriculum due to pragmatical needs (e.g., because of not fitting in the schedule for logistical reasons, or sudden unavailability of teachers). Further, changes in the curricula over time (at least gradually) may also happen unplanned, due for example to the potentially changing profiles or personal opinions of the individual teachers.

Traditionally, program management tasks (including the *black-box* approach) are more or less routinely performed at every university, and have the aim of ensuring high quality and coherence in the studies flow. The organization and execution of these tasks are almost invariably laying in the hands of the program boards; however, in our roles as university teachers, we have experienced that the common practises of executing these tasks are subject to several shortcomings: (i) they are typically called and executed centrally (by the program board), implying difficulties in collecting or using all the information that would be relevant to the tasks; (ii) they involve collecting, revising and reviewing information manually, which limits how much data can be processed, and (iii) they happen at discrete time instances or according to fixed cycles, and this may delay or prevent the detection of mismatches (especially if long temporal delays are present between the implementation of local changes in courses' contents and their reporting in the next revision of the program).

For these reasons, the standard approaches to curricula management activities thus tend to not: (i) guarantee coherency, alignment with standards and across learning contexts and practices, transparency in the studies; (ii) promote students' and other stakeholders' involvement, participation, and co-construction of curricula; (iii) enable a structured approach to adapting to the demands from industry and society.

This paper presents a tool that aims at that direction, and which allows to analyse and visualise the coherency of a higher education program through quantitatively capturing connections between the courses within the program, between the knowledge components included in the program, and the interrelations between courses and knowledge components. The here presented tool may hence be

⋆ The research leading to these results has received funding from pedagogical funds at Uppsala University, Norwegian University of Science and Technology, and the European Community through the Erasmus+ project 2019-1-NO01-KA203-060257 "Face It".

10.1016/j.ifacol.2020.12.2675

used to support and ease the program management task, and mitigate some of the shortcomings above. Importantly, the here proposed approach is: (i) democratic in nature, since collecting and processing data from individual teachers in the program; (ii) data driven and using algorithms to process and analyse large quantities of data, and (iii) asynchronous in the sense of allowing new analyses whenever new data are collected without being bound to be executed at specific time frames.

In a similar approach, Rollande [2015] discusses an approach to quantitatively analysz the structure of curricula by means of four distinct graphs to create individualised study plans for students. However, how to use such graphical representations to foster alignment within the curriculum is not discussed. The approach presented in Aldrich [2014, 2015] was used to analyze connections between courses according to curriculum structure in order to understand the coherence within a university program. The work focuses on defining and analyzing the network of prerequisites, with the purpose of understanding how their roles can differ within a curriculum. Moreover, the important questions of *why* courses are prerequisites for other courses and *what is learned* in the courses is not considered here. Another paper connecting learning goals, topics, and courses within a program is Pavlich-Mariscal et al. [2019], where the authors model learning goals, topics and courses as nodes, and model prerequisite dependencies as edges, so that the relation between courses and topics are represented as edges. It is important to note that the focus of this paper lies on the design of curricula, rather than on estimating the current situation. We also report Lightfoot [2010], that proposes to use graphs-oriented approaches to answer key questions on where to focus the assessments, data collection, and corrective actions within the curriculum. The focus is thus on how to analyze systematically and quantitatively the program contents so to help faculty and administrators that are tasked with creating quality assurance and assessment schemes.

In contrast to the works above, our project aims to develop methodologies to represent and visualize university program contents in order to increase awareness between the stakeholders and hence create opportunities to discuss curricula coherence and adjust the program accordingly. Moreover, we tailor our efforts towards promoting a *bottom-up* approach to the program management, by including mechanisms that enable asynchronous updates and analyses of the information. In this manuscript we thus present the IT tool, called COnCUR, and its functionalities. Our purpose is to overview our open source software, available in `https://github.com/damianovar/concur`, the most common ways to use this tool, and the most important results that can be obtained from it.

The paper starts this description thus with a short summary of which data shall be collected and in which format in Section 2, to then continue with discussing the tool's functionalities in more detail in Section 3 (Main Menu), Section 4 (Program View), Section 5 (Knowledge Component Flow View) and Section 6 (List Views). The manuscript ends with a section containing a list of intended development directions and the corresponding motivations behind these plans.

2. THE DATA COLLECTION STEP: FORMAT AND TYPES OF INFORMATION

The presented approach adopts a constructivist interpretation of knowledge and serves the greater aim of analysing and investigating the coherence of the cognitive learning outcomes (e.g., skills and competences) [Nusche, 2008] taught within a given university program. "Coherence" is here interpreted as a temporal and structural property, in the sense that the knowledge that is a prerequisite for a certain course has been developed in a course that has been offered previously in the same program and at the required level or higher. Our developed tool, COnCUR, which will be detailed from Section 3 on, has thus the primary goal of analysing and visualising how this developed and required knowledge flows in time. Doing so, we hence enable the analysis and visualisation of this type of coherence within the program.

This section describes which data is need to enable such analysis. We note that there may exist several different methods for collecting such information; the option tested here is to invite teachers of a curriculum to perform a workshop and collect the information for each course directly from the responsible teachers. This corresponds to taking a bottom-up approach of collecting information.

As a first step, the courses are described by a collection of Knowledge Components (KCs), where each KC describes an acquirable and testable unit of cognitive function, which may be facts, concepts, procedures, etc. [Koedinger et al., 1012]. Examples are "understanding evolution", "design a beam" or "programming a microcontroller". Describing a course by its KCs implies taking a constructivist interpretation of knowledge and focusing on cognitive learning outcomes. In our experience this is rather convenient for teachers in STEM disciplines, since in these contexts KCs are easily identifiable, being often corresponding to what knowledge can be tested on students or not.

The first piece of information that we ask for being collected for each course is then the list of KCs that correspond to the prerequisites and the developed knowledge of that course. Besides this information, we enable teachers to list other pedagogically important information, such as the Teaching and Learning Activities (TLAs) (e.g., lectures, tutorials, seminars, labs etc.) and Intended Learning Outcomes (ILOs) associated to their courses.

Given our intended bottom-up approach, our approach is to collect this information by means of spreadsheets, one for each course, that in the following will be denoted with Course Flow Matrix (CFM). Summarizing, each CFM consists of several tables:

- **a course summary,** that lists the required and developed KCs, the ILOs and TLAs of the course as well as its course code and start and end date;
- **a Knowledge Component Matrix (KCM),** i.e., a table where each column y corresponds to a certain prerequisite or developed KC, each row x corresponds to a certain developed KC, and each element in row x, column y describes the minimum knowledge level [1]

[1] "Knowledge level" here refers to a taxonomy knowledge level. Loosely speaking, this is a description of the knowledge level that

that a student should have acquired about y in order to be able to learn x in a satisfactory way;

information on the developed KCs, i.e., specifications which knowledge level students should reach at the end of the course for each developed KC.

3. CONCUR MAIN MENU

In practice COnCUR consists in a collection of routines for processing, analyzing and visualizing the data collected through the spreadsheets discussed in Section 2. In the following we will assume that a set of spreadsheets is available for courses in a specific program

On launching COnCUR, available in

`https://github.com/damianovar/concur`,

the message in Listing 1 appears after a short initialization phase.

Listing 1: The main menu of COnCUR.

```
Main Menu: what would you like to do?
    Actions available:
1 - select a full program to analyze
    (currently loaded: NTNU-MIIK)
2 - plot the selected program (NTNU-MIIK)
3 - open the COnCUR app
4 - select which CFM you want to analyze
5 - show the currently selected CFM
    (now: TTK4115, TTK4130, TTK4135,
    TTK4225, TTK4230)
6 - analyze the currently selected CFM
    (now: TTK4115, TTK4130, TTK4135,
    TTK4225, TTK4230)
your choice:
```

Choosing Option 1 lets the user select which university program shall be analysed, while showing the currently loaded program. Option 2 opens the so-called Program View, which is described in detail in Section 4. Option 3 opens the KC Flow View in the COnCUR App, which displays the courses within the program, their temporal order and their connections as a graphical overview of the curriculum. Details will be described in Section 5. Option 4 lets the user choose and analyze one or more specific CFMs which contain information about how individual KCs are connected within a course, something that is instrumental to Option 6. Further, Option 5 prints information about the active CFMs and is described in more detail in Section 6, along with Option 6.

As for the actual data structure used to represent a program, the current implementation fetches the data from a set of CFMs and Program Definition Files (PDFs). More precisely, a PDF is a plaintext document that holds links to CFMs that are part of the university program it represents. This file is a tabulated list where each row after the first contains a course code, a version number, and a web link to the corresponding CFM, that may be shared among colleagues on (as an example) Google Drive. An example of PDF is given in Table 1; note that a course (identified by its course code) can have multiple versions, due to course revisions over time.

Table 1. Example of a PDF.

coursecode	version	link
1DT038	1	(a URL)
1RT495	2	(another URL)
1TE723	1	(a third URL)
1TE723	2	(a fourth URL)

4. PROGRAM VIEW

In this section, the Program View, which provides a graphical representation of the contents of the selected program, is described in detail. The administrators can use this view as an aid for suggesting revisions in curricula, and the program board can use it as a "debugging" tool for designing a curriculum. In this context, "contents" refers to the courses taught in the program, the order in which they are taught and their interconnections in terms of KCs they have in common. It also reveals temporal mismatches between courses.First, an overview of the Program View is given in Section 4.1, followed by a short description of the interactivity offered in this view in Section 4.2.

4.1 Overview of the View

Choosing Option 2 from the main menu opens the Program View as in Figure 1. For the purpose of this paper, a university program given at Luleå University of Technology (LTU) is used. The courses are sorted in chronological order, with the program prerequisites acting as an artificial first course and the program outcomes as an additional artificial last course. If a connection goes below and to the right of the courses rather than above and to the left of them, it means that the future course develops one or

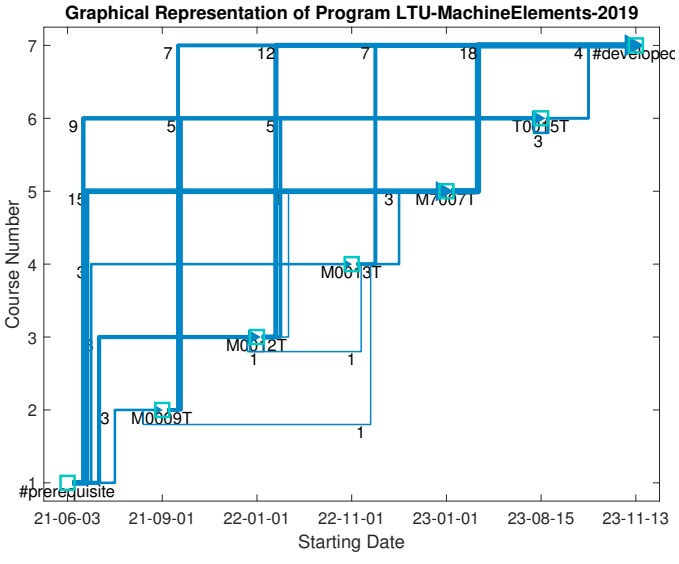

Fig. 1. The program view. Each node represents a course and each edge signifies a flow of KCs.

may be reached by a student. Examples of common taxonomies are the SOLO taxonomy [Biggs and Tang, 2011], which consists of levels 1-4, and Bloom's revised taxonomy [Anderson et al., 2001], which consists of six levels. To make an example, low knowledge levels may be "remember" or "recognise", while higher levels such as "analyse", "design" or "evaluate".

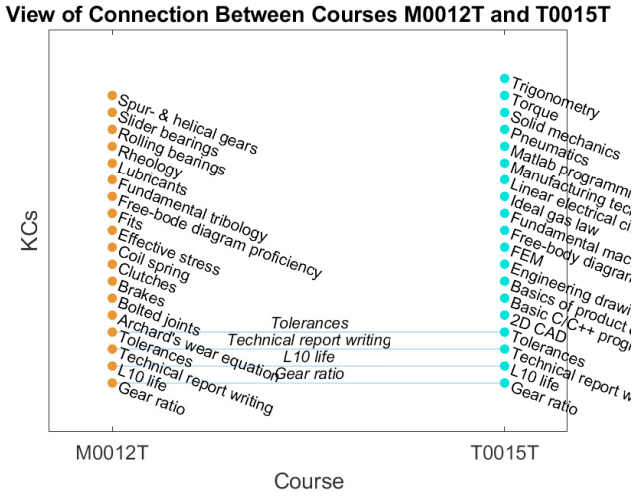

Fig. 2. The connection between M0012T and T0015T.

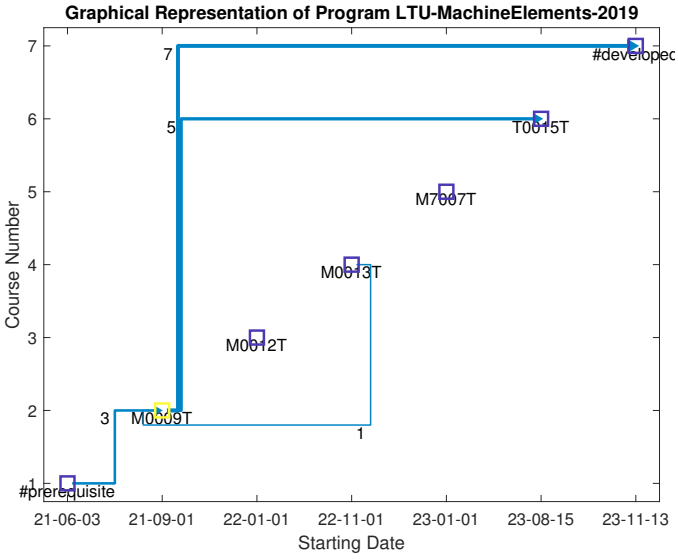

Fig. 3. The program view with one course highlighted.

more KCs that the past course have as prerequisites, which indicates a temporal discrepancy between the two courses.

4.2 Interaction with the View

The Program View has a few features, which will be described below.

The edges can be zoomed into, which opens the view in Figure 2. This view reveals which KCs are connected across the courses connected by the edge. Hence, through this view, it can be accessed in detail why, i.e., due to which KCs, a previous course is relevant or required for a later course.

The squares representing courses can be interacted with in three ways. First, a course can be highlighted. Doing so shows only the edges that connect directly to the highlighted course, as is the case for Course M0009T in Figure 3.

Second, one can open a simplified view of the KC flow of a course. There, the prerequisite KCs of the course are displayed to the left of the course and the developed KCs

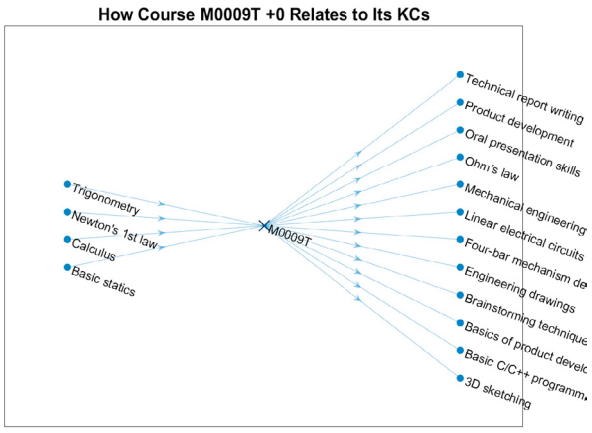

Fig. 4. A simplified view of a course.

are displayed to the right of it. The connections among the KCs themselves are not shown in this view. This gives students a comprehensive picture of what they are expected to know before taking a course as well as what they are expected to learn in the course. An example is shown in Figure 4, where Course M0009T has been chosen.

Third, a more detailed KC flow can be displayed. This view, known as the KC Flow View, is the topic of Section 5 below.

5. KC FLOW VIEW IN THE CONCUR APP

In this section, the COnCUR App and its KC Flow View is described. The purpose of the different elements is explained in Section 5.1. The plot window is detailed in Section 5.2. Structural problems in the plot and how to identify them is covered in Section 5.3 and interactivity with the plot window in Section 5.4.

5.1 The Elements of the KC Flow View

A screenshot of the KC Flow View, which contains mainly the large plot area, where the KC flow of the selected

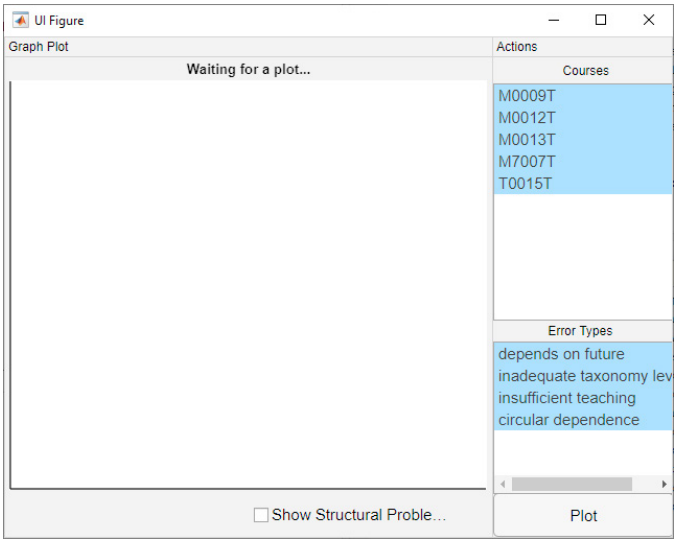

Fig. 5. Screenshot of the KC Flow View.

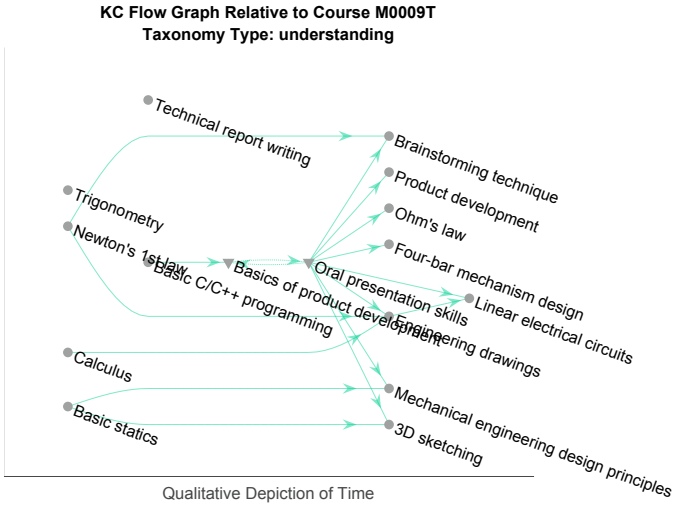

Fig. 6. The KC Flow Plot.

courses is visualized can be found in Figure 5. There are also two list boxes on the right of the plot. The upper list contains all courses in the currently selected program and the lower list contains all types of structural problems that can be analysed and visualised with the tool. Any number of courses or problems can be selected in these lists. There is also a checkbox that determines whether or not to show exclusively the structural problems. Finally, the "Plot" button updates the KC Flow View with the current settings.

5.2 The KC Flow Plot

The plot window is the main feature of the KC Flow View. It shows how the KCs in the selected courses depend on each other. See the example in Figure 6, where only one course has been selected for readability.

Each KC is represented by a node and dependencies between KCs are visualised by edges going from the dependency to the dependent. The program-wide prerequisite KCs are placed on the left-hand side and the KCs that are developed in at least one of the courses are distributed to the right of the program-wide prerequisites in a qualitatively chronological order.

The KC Flow View makes use of different colours for the KC and dependencies ranging from mint green (lowest level) to violet (highest level). Some edges may be coloured or styled differently, which is explained in Section 5.3.

5.3 Structural Problems View

The KC Flow View has the ability to illustrate and highlight structural problems in the KC flow. A structural problem is a logical, temporal or structural flaw in the graph. All structural problems, that can be displayed are listed in Table 2. The case "Future Dependence" describes that a KC A depends on a prerequisite KC B that is taught after A, which leads to a problem in terms of temporal consistency. "Too High Taxonomy Needed" is a structural problem that can occur both within a course and across courses. This problem means that KC B demands that KC A is known at a higher taxonomy level than A is

Table 2. Legend for the structural problems.

Problem	Style
Future Dependence	Dashed line
Too High Taxonomy Needed	Different colour
Too High Expectation of KC	Coloured label
Circular Dependence	Dotted lines, triangle markers

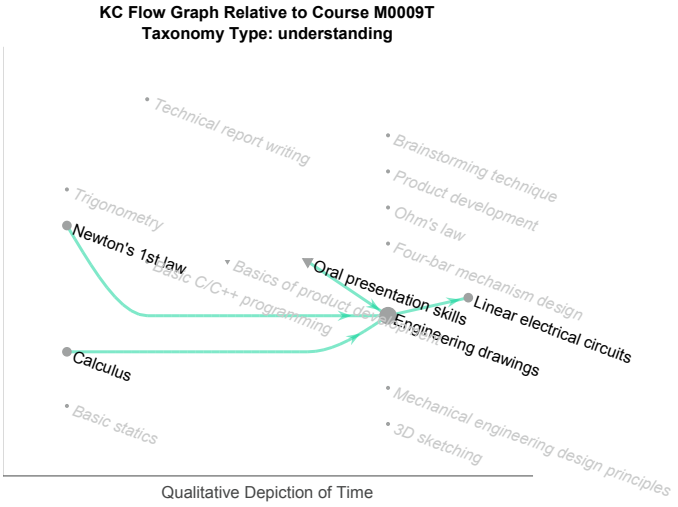

Fig. 7. Figure 6 with "Engineering Drawings" highlighted.

ever taught at in the same course, or any of the earlier courses. Such edges are coloured yellow, orange or red, depending on the taxonomy level they demand. "Too High Expectation of KC" is a problem where a developed KC A is taught up to a given level within a course, but the taxonomy level that should be reached for A at the end of a course is higher. "Circular Dependence" is a problem where two or more KCs depend on each other. For instance, a case where KC A is needed to learn KC B *and* A is learned from B is a circular dependence. If an edge has both circular dependence and future dependence, the edge is dash-dotted.

As stated in Section 5.1, the COnCUR App can be configured to show only a subset of the structural problems in Table 2. It can also hide any edges or nodes that do not have any of the selected structural problems. This allows the user to isolate and identify where the structural problems are such that the problematic course can be revised.

5.4 Interaction with the View

The KC Flow View is, like the Program View in Section 4, interactive. In this section, the two different interactions in the plot area are presented.

First, a KC can be highlighted. Like the highlighting feature in Program View, only the edges that connect to any highlighted KC are shown. The KC Flow View also enlarges the highlighted nodes and puts those that neither precede nor succeed any highlighted KC in the background. An example is given in Figure 7. Isolating a KC in this way is useful as it helps to illustrate and understand why the isolated KC is important or not for the course or the program, which other KCs it connects

to and in which way. Second, the relations of a KC to the courses can be inspected. Doing so opens a view like that of the simplified KC view mentioned in Section 4. This view shows which courses a given KC is taught in or required for, whereas Figure 4 shows vice versa.

6. LIST VIEWS

In this section, the views that result from choosing Options 5 or 6 are described in further detail. Option 5 is covered in Section 6.1 and Option 6 in Section 6.2.

6.1 The Information View

Choosing Option 5 in the Main Menu prints information about the currently selected (set of) CFM(s), which can be selected with Option 4. The CFMs in the current program are selected by default.

The Information View lists all KCs, TLAs and ILOs in the selected CFM.It also lists the connections between KCs, TLAs and ILOs in the form of tables, and additional information about the developed KCs and TLAs as well as a list of all structural problems that were found.

6.2 The Analysis View

Choosing Option 6 in the Main Menu opens the CFM Analysis View, which appears as in Listing 2.

Listing 2: The CFM Analysis Menu.

```
CFM Analysis Menu: what would you like
    to do? Actions available:
1 - list the available centrality indexes
2 - select the current centrality index
    (now: betweenness)
3 - plot the centrality indexes
4 - list the most central
    prerequisite KCs
5 - list the most central developed KCs
your choice:
```

Option 3 displays a bar diagram of the centrality indexes per KC. Options 4 and 5 list the prerequisite and developed KCs, respectively. The list is sorted in descending order by the currently selected centrality index. With these lists, it becomes easier to understand which KCs are the most important, i.e., most central, in the selected CFMs, Knorn et al. [2019]. The indexes are normalized before listing.

7. CONCLUSIONS

This paper presented COnCUR, a tool to analyze and visualize the structure of a university program. Based on information which knowledge is required and taught, at which level and when in each course of a program, the structure and coherence of the program consisting of the individual courses can be analyzed and visualized. Further, structural, logical and temporal inconsistencies within the program can be detected, analyzed and visualized in order to facilitate course and curriculum revision. For these reasons the tool is expected to foster high-quality Higher Education, strengthening quality assurance in the design, implementation, execution and evaluation of the associated programs.

REFERENCES

Aldrich, P. R. (2014). The curriculum prerequisite network: a tool for visualizing and analyzing academic curricula. In *CoRR*.

Aldrich, P. R. (2015). The curriculum prerequisite network: Modeling the curriculum as a complex system. *Biochemistry and Molecular Biology Education*, 43(3).

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., and 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*.

Biggs, J. and Tang, C. (2011). *Teaching for Quality Learning at University*. Maidenhead, UK: Open University Press.

Knorn, S., Varagnolo, D., Staffas, K., Wrigstad, T., and Fjällström, E. (2019). Quantitative analysis of curricula coherence using directed graphs. In *IFAC Symposium on Advances in Control Education*.

Koedinger, K. R., Corbett, A. T., and Perfetti, C. (1012). The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science*, 36(5).

Lightfoot, J. M. (2010). A graph-theoretic approach to improved curriculum structure and assessment placement. *Communications of the IIMA*, 10(2).

Nusche, D. (2008). Assessment of learning outcomes in higher education.

Pavlich-Mariscal, J. A., Curiel, M., and Chavarro, G. (2019). Cdio curriculum design for computing: a graph-based approach. In *Proceedings of the 15th International CDIO Conference*.

Rollande, R. (2015). *The Research and Implementation of Personalized Study Planning as a Component of Pedagogical Module*. PhD thesis, Riga Technical University.