The CITE App Manual

Beagle Boys[*]

Version 0.1

Contents

1 What is this? And why should I use it? 3
1.1 What if I just want to analyse a single course? 4

2 An overview of the structure of this manual 5

3 Inserting the data 5
3.1 Creating the database containing the files above 5
3.1.1 Internet-based case 7
3.1.2 Local repository case 7
3.2 Filling up the various Knowledge Components Matrices (KCMs) 8
3.2.1 The course summary sheet 8
3.2.2 The developed vs. prerequisite KCs sheet 9
3.2.3 The info on the developed KCs sheet 9
3.2.4 The remaining sheets 9
3.3 Summary of which steps shall be followed 10

4 Analysing the data 10
4.1 Launching Matlab 10
4.2 The initial prompt 10
4.3 The KCM analysis prompt 10
4.4 The CITE app 11

5 Interpreting the results 11
5.1 Interpreting the centrality indexes 11
5.2 Interpreting the Knowledge Components Graph (KCG) 12

A Lexicon 13

B How to define the knowledge components list 13

C How to interpret and assign taxonomy levels 13

D How to identify Intended Learning Outcomes (ILOs) 14

E How to identify Teaching and Learning Activities (TLAs) 14

F Debugging 14

G Bibliography 14

[*]The research leading to these results has received funding from the European Community through the Erasmus+ project 2019-1-NO01-KA203-060257 "Face It", and from Luleå University of Technology, Uppsala University and the Norwegian University of Science and Technology through a series of local pedagogical development projects.

Acronyms

CDIO	Conceiving, Designing, Implementing, and Operating
ILO	Intended Learning Outcome
ITS	Intelligent Tutoring System
KCG	Knowledge Components Graph
KCM	Knowledge Components Matrix
KC	Knowledge Component
LMS	Learning Management System
LTU	Luleå University of Technology
NTNU	Norwegian University of Science and Technology
PLO	Program Learning Outcome
SOLO	Structure of Observed Learning Outcomes
TLA	Teaching and Learning Activity
UU	Uppsala University

1 What is this? And why should I use it?

Among the needs of the persons involved in higher education, we identified the following:

for students: when studying, have a clear idea of how courses connect to each other from a contents-wise perspective, and of why it is important to memorize and understand what it is being asked to study (e.g., why shall I understand what is the geometrical interpretation of eigenvalues?);

for teachers: when modifying the contents of the own courses, have a holistic view on the effects of these contents modifications on the program (e.g., if I won't teach this concept anymore in my course, how will this affect the following courses?);

for program boards: when modifying the structure of a program, have instruments that help steering discussions and taking decisions based on evidence instead of opinions (e.g., why shall this course be taught before that other one?);

for administrators: when inspecting and assessing the program quality, have instruments that are quantitative oriented, explainable, and communicable, so to ease discussions and reporting.

This suite tries to meet these needs by representing the contents within university programs using graph-oriented quantitative descriptions and analyses, and suggesting these representations as tools to aid students, teachers, boards and administrators to gain evidence-based awareness and alignment.

The approach is the following:

1. consider courses and programs as opportune flows and transformations of prerequisite contents-wise knowledge (expressed in terms of prerequisite concepts) into developed contents-wise knowledge (expressed in terms of Knowledge Components (KCs). See Section A for an explanation of all the various terms);
2. analyze and visualize the structure of a program (or a part of it) in terms of these concepts development flows;
3. connect these concept development flows to the TLAs, ILOs, and Program Learning Outcomes (PLOs) of the various courses and the whole program;
4. increase the awareness of the stakeholders by visualizing and analysing these connections and flows in (hopefully) self-explanatory ways.

Example 1.1. To exemplify this process, assume that the contents of the fictitious "*Course X*" can be expressed in terms of which prerequisite concepts are required by the students plus which concepts are developed in the course itself. E.g., let the prerequisites and developed concepts be:

- prerequisites: "*vector spaces*", "*linearity*", and "*matrices-vectors multiplication*";
- developed: "*eigenvalues*", "*characteristic polynomials*", "*computation of Jordan forms*".

Ideally (and as an example), the teacher knows that the developed concepts are ideally reached by building on the prerequisite ones as summarized as in Figure 1. I.e.,

- to be able to learn about *eigenvalues* students should ideally have a prerequisite learning level (using Bloom's taxonomy [2] as an illustrative example) *2 - understand* for both prerequisite concepts *vector spaces* and *linearity*;
- to be able to learn about *characteristic polynomials* students should ideally have a prerequisite learning level *2 - understand* about *matrices and vectors multiplication*, and have reached - while studying for *Course X* - a learning level *1 - remember* about the developed concept *eigenvalues*;
- to be able to learn about *computation of Jordan forms* students should ideally have a prerequisite learning level *1 - remember* for the prerequisite *linearity* and level *2 - understand* about the developed *eigenvalues* and *characteristic polynomials.*

		prerequisite concepts			developed concepts			
	teaching time	vector spaces	linearity	matrix-vector multiplication	eigenvalues	characteristic polynomials	compute Jordan forms	intended final knowledge level
eigenvalues	45%	2	2					3
characteristic polynomials	20%			2	1			2
compute Jordan forms	35%		1		2	2		3

Figure 1: Tabular representation of how *Course X*'s developed concepts are ideally reached by building on its prerequisites. In this document we call this matrix a Knowledge Components Matrix (KCM).

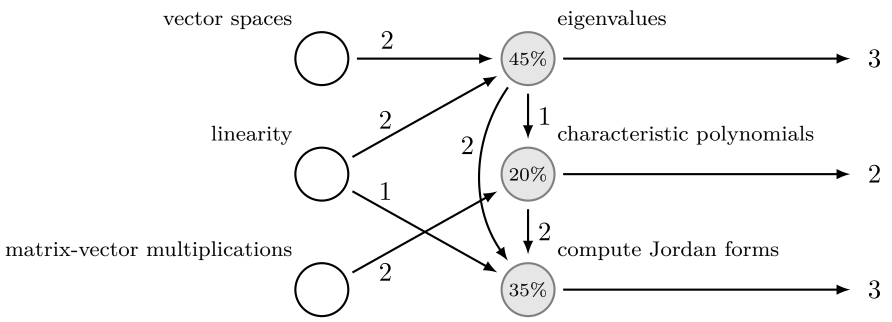


Figure 2: Graphical representation of the Knowledge Components Matrix (KCM) in Figure 1 as a (as called in this document) Knowledge Components Graph (KCG).

The KCM in Figure 1 can be converted into its graph representation defined in Figure 2. Within a program, the concepts developed in some courses may be prerequisites for other courses. This means that one can join the various individual KCMs and KCGs into a program-wide representation. This operation produces a directed graph representing the various "*concepts learning flows*" that students ideally follow during their studies.

KCG representations can help meeting the stakeholders' needs listed at the beginning as follows:

for teachers: changing the content of a course means changing the learning flows in the KCG. With the here proposed software one may then check if course modifications (i.e., KCM modifications) will lead to broken, redundant or non-pedagogically ideal paths in the KCG;

for program boards: the same software may be used to manage the discussions during boards meetings, and be a digital canvas where to test the meaningfulness of different program changes. This may help taking an evidence-based approach to designing and modifying programs;

for students: the software may be used by students to get intuitions about the purpose of studying specific concepts, what is the effect of having forgotten parts of the program, plus how central each part is within the program flow. The software may thus help single students perform self-assessment and monitoring, and help gaining holistic viewpoints on the expected learning process;

for quality assurance personnel: the software may be used also by quality assurance to perform automatic assessments of structural properties of the programs. We envision to use these representations also to help performing comparisons of different programs in different institutions - however we still did not implement these features.

1.1 What if I just want to analyse a single course?

The software may be used by a teacher or set of teachers also in the context of a *single course*. In other words, in the following one may build up a KCM for each *individual class* instead of for each *individual course*. This means that this software may be used as it is also for planning, visualizing, assessing and discussing the detailed contents of a course, instead of a program or part of it.

2 An overview of the structure of this manual

The first part of the manual is dedicated more to "how to use the software":

Section 1 describes the general purpose of the software, and frames the bigger picture;

Section 3 describes how to collect and insert the data that will be processed by the software;

Section 4 describes how to use the software to produce some information;

Section 5 describes how to interpret that information.

The appendix is more about "what do the various things mean":

Section A is a reference of meanings of the various terms that are used throughout the document;

Section B describes how to identify and define the KCs. This section is ancillary to Section 3;

Section C describes how to interpret (and thus assign) knowledge taxonomy levels. Also this section is ancillary to Section 3;

Section D describes how to identify and define the ILOs. Also this section is ancillary to Section 3;

Section E has the same purpose above, and is dedicated to defining the TLAs. Also this section is ancillary to Section 3;

Section F collects a series of known bugs / non-ideal features in the software that may make Matlab stop running. Please consider that this software is more in alpha-testing than in beta-testing.

3 Inserting the data

The purpose of the software is to create graphical representations of the programs. The software suite works by merging independent information from different courses. More specifically, one shall create:

1. a KCM (which is a specialized spreadsheet file) for each course;
2. a "*program definition file*" (which is a plain text file) that indicates which KCMs should be considered to be part of the program.

This section goes over the procedure for creating the files above, and for how to enter information in them.

3.1 Creating the database containing the files above

The first step is to create a database containing the KCM spreadsheets and the program definition file. The KCMs may either be on Google Drive / Dropbox / Onedrive / whatever-you-prefer, or on the computer that you will be using for your analyses. The program definition file must instead be on the computer that you will be using for your analyses, and is needed independently of whether you have the KCMs on the Internet or not (since it defines which courses shall be included in the analysis and which versions of the KCMs shall be considered).

As for the program definition file, it has to be a tabulated list structured as in Table 1. This means that te first row **must** be identical to that of the first row in Table 1, where each column is separated by **one** tab. The other rows should each have a course code, a version number (any positive rational number is valid) and a link to a KCM on Google Drive / Dropbox / whatever, in order. Also here, each column has to be separated by **one** tab.

This file, let's call it `my-program-definition-file.txt`, should be located in a very specific path. To understand where, consider the base folder shown and described in Figure 3.

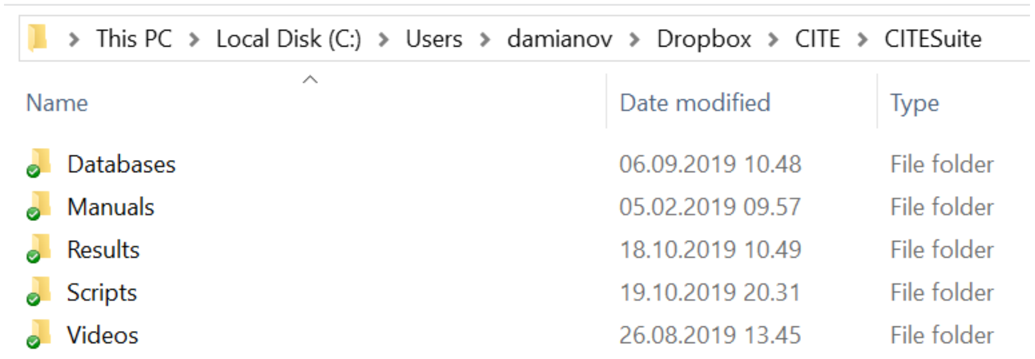
From the base folder in Figure 3 you should then enter the "Databases" folder, and obtain the folder structure as in Figure 4.

The `my-program-definition-file.txt` shall then go inside the folder "Programs". **As an example,** a person that has some few different program definition files may have a situation like in Figure 5.

As for the KCMs, they can be either on the Internet or on a local folder. Sections 3.1.1 and 3.1.2 explain how to structure these.

Table 1: Example of a program definition file. Each column has to be separated by **one** tab (this means that the columns may not look like "aligned" in the file).

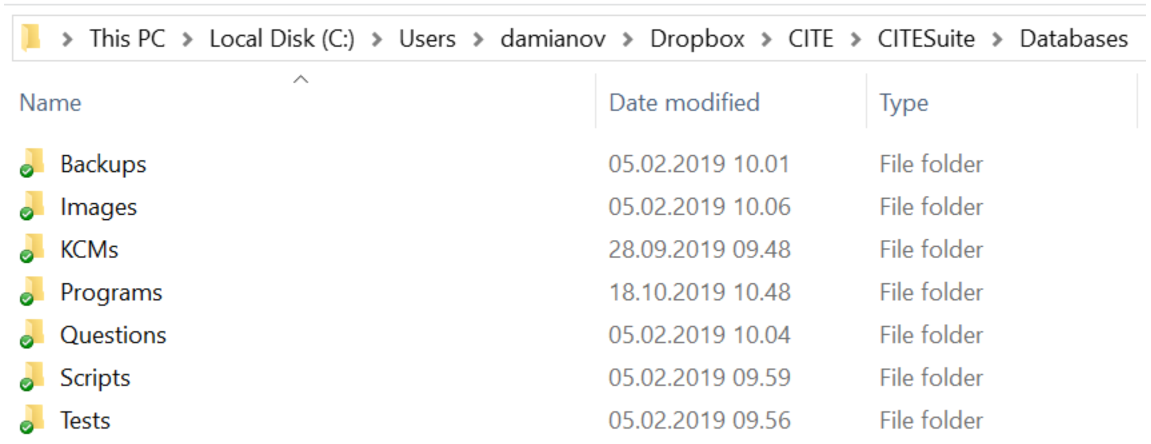
coursecode	version	link
TX8101	2	(URL link to the spreadsheet file)
AF9101	3	(URL link to the spreadsheet file)
XY0011	2	(URL link to the spreadsheet file)



This PC > Local Disk (C:) > Users > damianov > Dropbox > CITE > CITESuite

Name	Date modified	Type
Databases	06.09.2019 10.48	File folder
Manuals	05.02.2019 09.57	File folder
Results	18.10.2019 10.49	File folder
Scripts	19.10.2019 20.31	File folder
Videos	26.08.2019 13.45	File folder

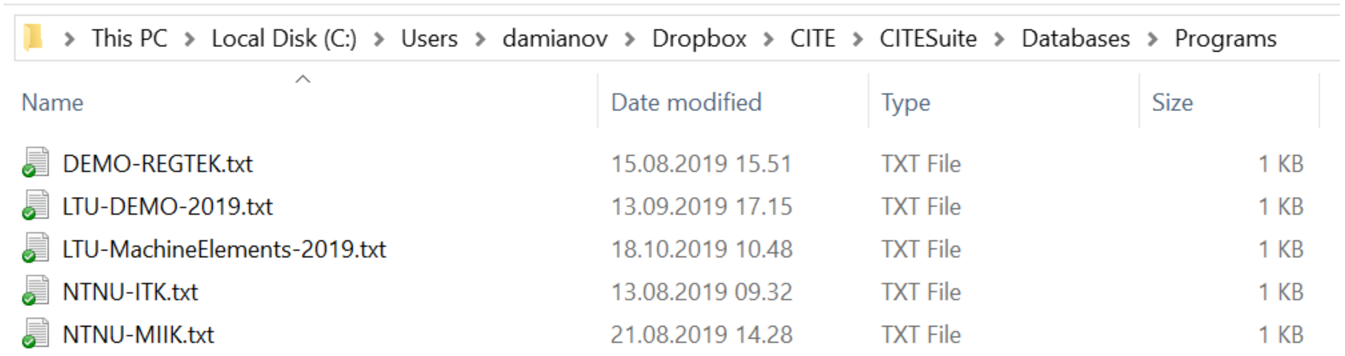
Figure 3: Structure of the base folder: when extracting the downloaded `CITESuite.zip` file, you obtain this.



This PC > Local Disk (C:) > Users > damianov > Dropbox > CITE > CITESuite > Databases

Name	Date modified	Type
Backups	05.02.2019 10.01	File folder
Images	05.02.2019 10.06	File folder
KCMs	28.09.2019 09.48	File folder
Programs	18.10.2019 10.48	File folder
Questions	05.02.2019 10.04	File folder
Scripts	05.02.2019 09.59	File folder
Tests	05.02.2019 09.56	File folder

Figure 4: Structure of the "databases" folder.



This PC > Local Disk (C:) > Users > damianov > Dropbox > CITE > CITESuite > Databases > Programs

Name	Date modified	Type	Size
DEMO-REGTEK.txt	15.08.2019 15.51	TXT File	1 KB
LTU-DEMO-2019.txt	13.09.2019 17.15	TXT File	1 KB
LTU-MachineElements-2019.txt	18.10.2019 10.48	TXT File	1 KB
NTNU-ITK.txt	13.08.2019 09.32	TXT File	1 KB
NTNU-MIIK.txt	21.08.2019 14.28	TXT File	1 KB

Figure 5: Example of how the "Programs" folder may look like. Note that when downloading the software suite you get an almost empty Programs folder (except a demo file).

3.1.1 Internet-based case

One may want to keep the KCM files on the Internet, so that different persons may access/edit the information in an asynchronous way. The program definition file says where these are, and the software works by downloading the KCMs when one runs the analysis. The structure of the Internet-based database of the KCMs is as in Figure 6.

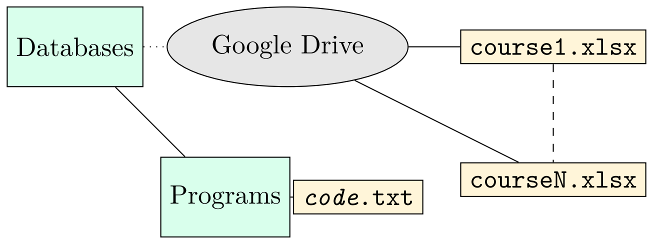


Figure 6: Diagram of the internet-based program database.

This means that the program definition file should be placed in the `Programs` folder of the `Databases` folder, as said before, and the links that it contains should point to the corresponding KCMs.

3.1.2 Local repository case

The other option is to have the KCM files placed in suitable folders. The structure of such a database is given in Figure 7.

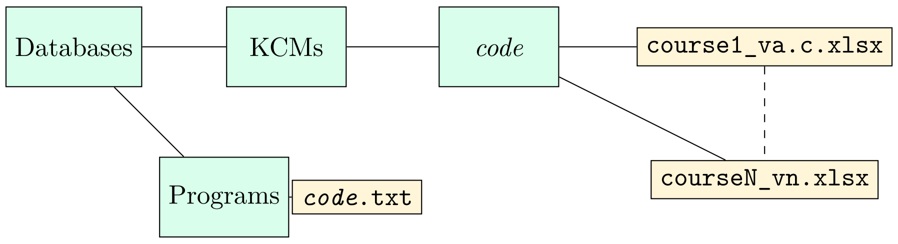
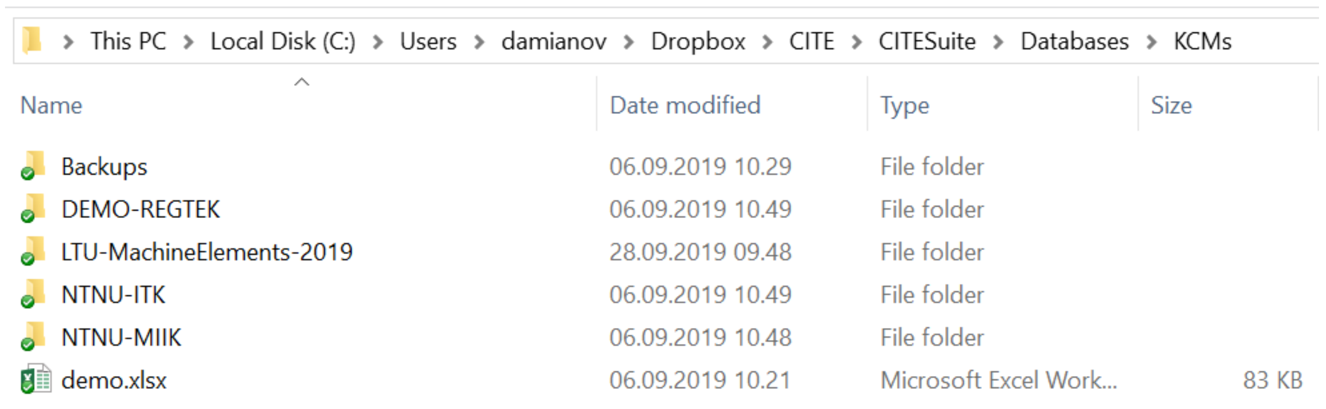


Figure 7: Diagram of the local database of the KCM spreadsheet files.

In more details, if one considers the base folder shown in Figure 3, one should then enter the Databases folder (as in Figure 4), and then enter the folder “KCMs”. For a person that has been analysing several programs this “KCMs” folder would look like as in Figure 8 (again, this is an example - you get an almost empty “KCMs” folder when you download the software suite).

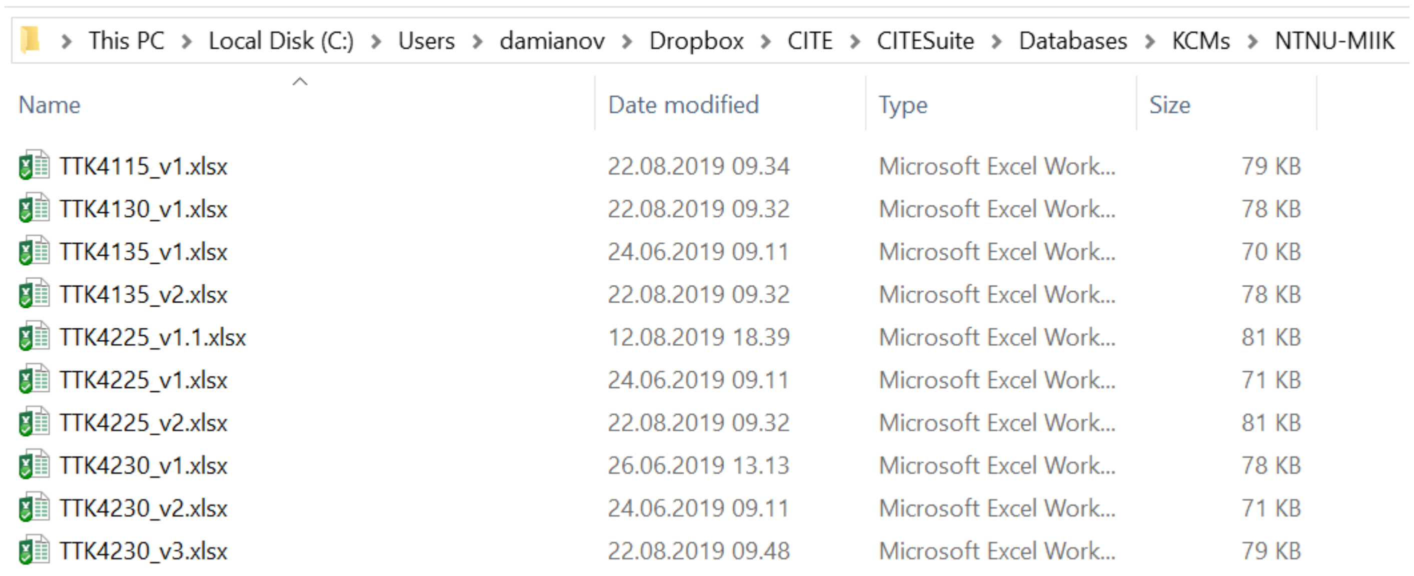


› This PC › Local Disk (C:) › Users › damianov › Dropbox › CITE › CITESuite › Databases › KCMs

Name	Date modified	Type	Size
Backups	06.09.2019 10.29	File folder	
DEMO-REGTEK	06.09.2019 10.49	File folder	
LTU-MachineElements-2019	28.09.2019 09.48	File folder	
NTNU-ITK	06.09.2019 10.49	File folder	
NTNU-MIIK	06.09.2019 10.48	File folder	
demo.xlsx	06.09.2019 10.21	Microsoft Excel Work...	83 KB

Figure 8: Structure of the “KCMs” folder for a person that has been analysing several programs.

Each folder in the “KCMs” folder is then a separate program: in this sub-folder you shall place your `.xlsx` files. As an example, Figure 9 shows the content of such a sub-folder relative to a program in NTNU. Note that one may have different versions for the same course.



This PC > Local Disk (C:) > Users > damianov > Dropbox > CITE > CITESuite > Databases > KCMs > NTNU-MIIK

Name	Date modified	Type	Size
TTK4115_v1.xlsx	22.08.2019 09.34	Microsoft Excel Work...	79 KB
TTK4130_v1.xlsx	22.08.2019 09.32	Microsoft Excel Work...	78 KB
TTK4135_v1.xlsx	24.06.2019 09.11	Microsoft Excel Work...	70 KB
TTK4135_v2.xlsx	22.08.2019 09.32	Microsoft Excel Work...	78 KB
TTK4225_v1.1.xlsx	12.08.2019 18.39	Microsoft Excel Work...	81 KB
TTK4225_v1.xlsx	24.06.2019 09.11	Microsoft Excel Work...	71 KB
TTK4225_v2.xlsx	22.08.2019 09.32	Microsoft Excel Work...	81 KB
TTK4230_v1.xlsx	26.06.2019 13.13	Microsoft Excel Work...	78 KB
TTK4230_v2.xlsx	24.06.2019 09.11	Microsoft Excel Work...	71 KB
TTK4230_v3.xlsx	22.08.2019 09.48	Microsoft Excel Work...	79 KB

Figure 9: Contents of a folder that contains the `.xlsx` files describing a given program.

Note the name formats: the name of each KCM file **must** be on the form `coursecode_v`*`version`* and have the extension `.xlsx`. The `coursecode` and *version* information must be the same one that is in the program definition file. The name of the folder containing the KCMs of the program `must` be identical to the name of the program definition file. The folder itself must be placed in the `KCMs` folder.

3.2 Filling up the various KCMs

Each `.xlsx` spreadsheet capturing a KCM has six sheets, described in the following subsections.

ESSENTIAL: do NOT add or remove rows and do NOT use "control-copy, control-paste" anywhere in these files. Otherwise Excel and Google spreadsheets will mess around with some macros that are encoded in these files and you will get GARBAGE, not analyses of your program.

The template `.xlsx` has the following colors-scheme:

gray: parts that are comments or indications (i.e., cells that you should not modify);

pink: parts that you have to fill (i.e., if you don't fill them then you won't get results);

yellow: parts that you should fill (i.e., if you don't fill them then you will get only partial results, and not everything that the tool may give you);

green: parts that you should fill if you want to get all the results that the suite can compute.

Thus: one should fill up at least all the pink parts. Doing the yellow and green ones will produce more informative results (but of course will also require more compilation time).

3.2.1 The course summary sheet

The first sheet, called `course summary`, is a summary of the course and has the following information:

- **Prerequisite KCs**, which is a list of the KCs that are prerequisites for the course, i.e. KCs that students should be familiar with at the beginning of the course. See Section B for a description on how to identify these KCs. Note that, given that you should not add rows to the tabs, you may insert only up to 20 KCs;
- **Developed KCs**, which is a list of the KCs that the students should get familiar with as they advance through the course.

- **TLAs**, which is a list of teaching and learning activities (lectures, classes, labs etc.) that occur in the course. See Section E for a description on how to identify these TLAs;
- **ILOs**, which is a list of KCs that the students should have acquired after passing the course. See Section D for a description on how to identify these ILOs;
- **Starting and ending dates**, which are the dates when the course begins and ends, respectively. **Respect the format that is indicated in the `.xlsx` file**;
- **Taxonomy types**, which is a list of which taxonomy types, e.g., "SOLO" or "Bloom", that are included in this KCM. If left blank, it defaults to "SOLO". See Section C for a description of how to interpret and assign taxonomy levels;
- **Course code**, which is self-explaining. If using a local repository, the file name should be equal to this entry.

3.2.2 The developed vs. prerequisite KCs sheet

The second sheet, called "developed vs. prerequisite KCs", is a dependency matrix for the developed KCs versus the prerequisite KCs and the developed KCs. In this matrix, each row is associated with one of the KCs that are being developed in the course, and each column is associated to either a prerequisite KC or a developed KC. Assume that `X` is the row KC, and `Y` is the column KC. Then the content of the cell `X-Y` shall indicate the minimum taxonomical knowledge level that the students should have reached about `Y` before starting learning `X` so to be able to learn `X` in a satisfactory way. In other words, each entry says how much (in a taxonomical knowledge way) the column KC is instrumental to learn the row KC.

Note that if you did not fill up all 20 slots for the KCs in the `course summary` tab then some rows and/or columns will show a "0" in the corresponding cell. That rows and/or columns should not be filled, obviously.

3.2.3 The info on the developed KCs sheet

The third sheet is a table of the developed KCs and has the following information:

- **Developed KCs**, which is the same as in the course summary;
- **Target taxonomy level**, which is the taxonomy level that the associated KC should be known at by the end of the course;
- **Time spent teaching**, which is the number of hours the course invests in teaching the associated KC;
- **First lesson**, which is the number of the first teaching event when the associated KC is taught.

3.2.4 The remaining sheets

The fourth sheet, called "KCs vs. TLAs", is similar to the second sheet in the sense that it also is a dependency matrix. However, the meaning of the rows and columns indexes is slightly different than before, and indicates only a *relation*: assume that `X` is the row TLA, and `Y` is the column KC. Then a nonzero content in the cell `X-Y` indicates that `Y` is either rehearsed or learned during `X`. Moreover the indicated taxonomical knowledge level indicates at which complexity that rehearsal or learning happens. In other words, each entry says how much (in a taxonomical knowledge way) the column KC is rehearsed or learned during the row TLA.

The fifth sheet, called "info on the TLAs", is similar to the third sheet, but has information on the TLAs rather than KCs. This sheet holds the following information:

- **TLA**, which is the same as in the course summary;
- **Time spent on activity**, which is the number of hours this course spends on the associated TLA;
- **When the TLA starts**, which is the number of the teaching event when the course starts hosting the associated TLA;
- **When the TLA ends**, which is the number of the teaching event when the course stops hosting the associated TLA.

The sixth sheet, called "developed KCs vs. ILOs", is equivalent to the fourth sheet. This time, though, each entry says how much (in a taxonomical knowledge way) the column KC is needed to reach the row ILO.

3.3 Summary of which steps shall be followed

To summarize, to create a database that can be analysed through the CITE Suite one should follow these macro-steps:

1. **Create the program definition file**. This text document holds course codes, version numbers and (if having the KCMs in Internet), links to these KCMs.

2. **Place the KCMs in the correct location**. If downloading the files over Internet, ensure that the links in the program definition file point to the correct URLs. If using a local repository, ensure that the KCMs are placed in the correct folder and named correctly.

4 Analysing the data

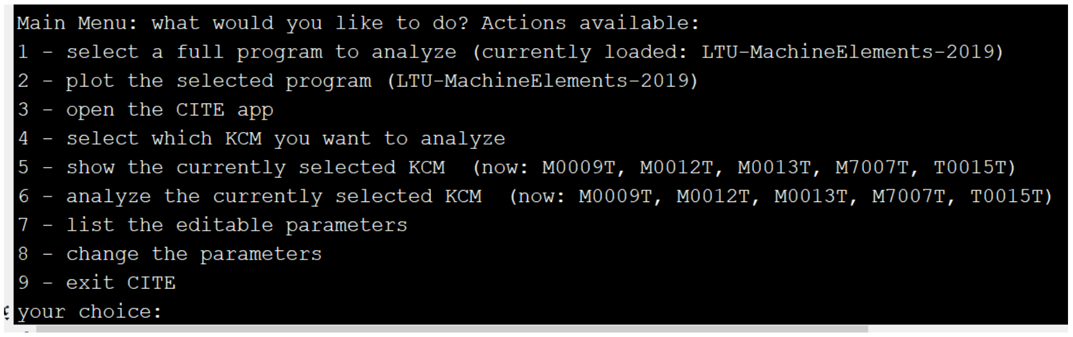
4.1 Launching Matlab

Note that to launch the CITE app below one needs Matlab 2019a or a higher version. The KCM analysis part, instead, requires Matlab 2018a or a higher version.

Once Matlab is launched, one shall change the directory in the command window to the folder "Scripts" (cf. Figure 3). Once there, execute `main.m`.

4.2 The initial prompt

Launching `main.m` will initially produce a series of debug messages, and then a prompt like the one in Figure 10.



```
Main Menu: what would you like to do? Actions available:
1 - select a full program to analyze (currently loaded: LTU-MachineElements-2019)
2 - plot the selected program (LTU-MachineElements-2019)
3 - open the CITE app
4 - select which KCM you want to analyze
5 - show the currently selected KCM  (now: M0009T, M0012T, M0013T, M7007T, T0015T)
6 - analyze the currently selected KCM  (now: M0009T, M0012T, M0013T, M7007T, T0015T)
7 - list the editable parameters
8 - change the parameters
9 - exit CITE
your choice:
```

Figure 10: Main menu that one obtains by launching `main.m`.

The most interesting (and not self-explaining) items are:

- “plot the selected program”, that will produce a figure that summarizes which course connects with which other in terms of KCs. Note that the suite will assign all the prerequisite KCs that are not taught by any course to an artificial “prerequisites” course. Note also that the plot is explorable - in the sense that clicking on the various arrows will expand which KCs form the connections;

- “open the CITE app” - for this item see Section 4.4;

- “analyse the currently selected KCM” - for this item see Section 4.3.

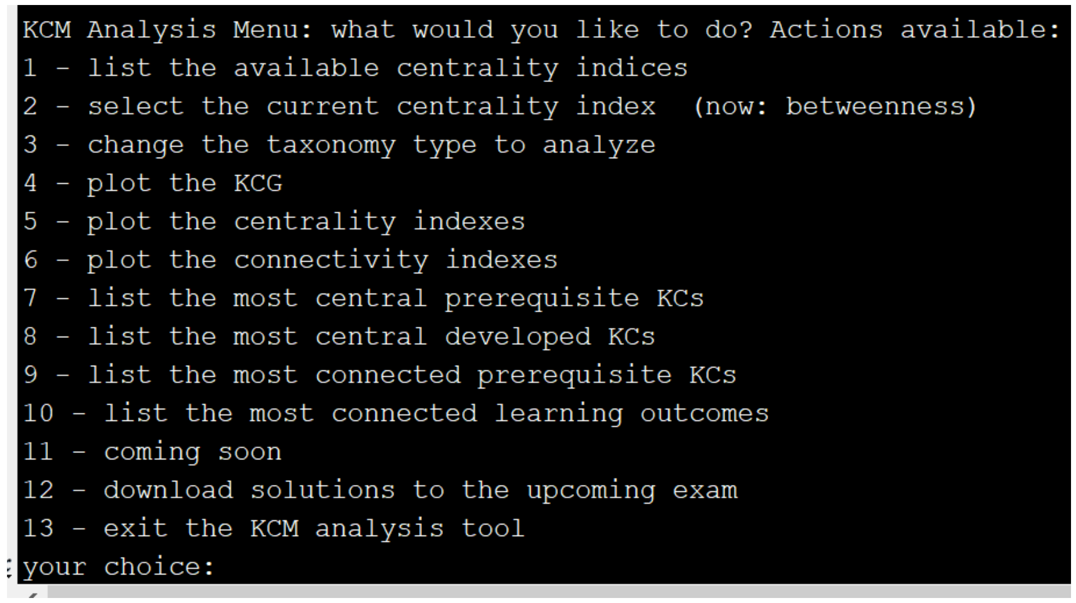
4.3 The KCM analysis prompt

Selecting to analyse the currently selected KCM will produce a prompt like the one in Figure 11.

The most interesting (and not self-explaining) items are:

- “plot the KCG”, that will open the CITE app (thus check Section 4.4 for this item);

- “plot the centrality indexes”, that will produce a figure that should be interpreted as suggested in Section 5.



```
KCM Analysis Menu: what would you like to do? Actions available:
1 - list the available centrality indices
2 - select the current centrality index  (now: betweenness)
3 - change the taxonomy type to analyze
4 - plot the KCG
5 - plot the centrality indexes
6 - plot the connectivity indexes
7 - list the most central prerequisite KCs
8 - list the most central developed KCs
9 - list the most connected prerequisite KCs
10 - list the most connected learning outcomes
11 - coming soon
12 - download solutions to the upcoming exam
13 - exit the KCM analysis tool
your choice:
```

Figure 11: Menu that one obtains by launching an analysis of the currently selected KCM.

4.4 The CITE app

Once the app is launched, you will get a blank graph plot, and a list of courses on the right. Selecting the desired courses with the mouse (shift+click) and then pressing the “Plot” button will produce an other explorable plot. Here clicking on the various nodes will highlight connections. Different colors are also placeholders for different taxonomical levels. Clicking the “Show Structural Problems” item will moreover highlight structural issues in the KCG, that may be either of a temporal structure (i.e., a course assumes that a prerequisite KC has been taught, but it has actually been taught at a later course) or taxonomical level structure (i.e., one assumes that a prerequisite has been previously taught at a certain level, but actually it has been taught at a lower level).

5 Interpreting the results

5.1 Interpreting the centrality indexes

out degree: when considering a course, a high out-degree indicates that this course exposes students to KCs that are likely being introductory or instrumental to build a framework later on. This means that assessments in courses that have high out degrees correspond to "baseline assessments", i.e., evaluations of students’ initial knowledge. Similar concepts apply to a KC: a high out degree would imply that this KCs likely represents baseline knowledge;

in degree: when considering a course, a high in-degree indicates that this course is the destination of the learning efforts. From a pedagogical perspective this indirectly indicates courses where it is important to engage students with active and higher-level learning activities. Assessments on these courses are a sort of "final assessments" for the program. Similar concepts apply to a KC;

betweenness: when considering a course, a high betweenness indicates that this course serves as a key link between different clusters of courses or KCs in a program. This indicates that they serve as a bridge; for this reason assessments on courses with high betweenness serve both the purposes described above, i.e., as evaluations of students’ initial knowledge for the next "cluster" of courses, and of students final knowledge for the previous "cluster" of courses. Similar concepts apply to a KC;

eigenvector centrality: in graph theory, the eigenvector centrality measures the influence of a node in a network in a similar way that the Google pagerank algorithm works: if a node is pointed to by many nodes which also have high eigenvector centrality then that node will have high eigenvector centrality. In a sense, the links of a node with higher eigenvector centrality are more influential than the links of a node with a lower centrality index. When considering a course, an interpretation of this index is in terms of "how much this course serves as a support and reinforcement for other courses". Assessments on courses with high eigenvector centrality indexes serve as complementary assessments for other courses;

incloseness: yet to be interpreted

outcloseness: yet to be interpreted

authorities: yet to be interpreted

hubs: yet to be interpreted

A Lexicon

Bloom's taxonomy: set of hierarchical models useful to classify learning objectives into levels of complexity and specificity. The models capture these different levels in different domains - namely cognitive, affective and sensory. For engineering and natural sciences purposes the most important is likely the cognitive one, whose levels are reported in Table 2;

1	*Remember*	be able to recognize or remember facts, terms, basic concepts, or answers without necessarily understanding what they mean
2	*Understand*	be able to organize, compare, translate, interpret, give descriptions, and state the main ideas
3	*Apply*	be able to use prior knowledge to solve problems, identify connections and relationships and how they apply in new situations
4	*Analyze*	be able to break information into component parts, determine how the parts relate to one another, identify motives or causes, make inferences, and find evidence to support generalizations
5	*Evaluate*	be able to present and defend opinions by making judgments about information, the validity of ideas, or quality of work based on a set of criteria
6	*Create*	be able to build a structure or pattern from diverse elements and to put parts together to form a whole

Table 2: Summary of the knowledge levels defined in Bloom's taxonomy [2].

Intended Learning Outcomes (ILOs): what students should know or be able to do at the end of the course that they couldn't do before. They should be about students performance. Typical verbs used to describe ILOs are: "Memorize, identify, recognize, define, find, classify, describe, list, discuss, select, compute, analyse, explain, compare, construct, review, solve, prove," etc;

Knowledge Components (KCs): acquirable units of cognitive functions. They can be facts, concepts, procedures, etc. Importantly, a KC must have the quality that it is possible to design tests that can show whether a student has acquired that KC or not. *If one defines something for which one cannot test whether a student acquired that thing or not, then that thing is not a KC.* Specially important cases of KCs in engineering and natural sciences education are *facts*, *procedures* and *concepts*. These correspond to mental representations, abstract objects or abilities that make up the fundamental building blocks of thoughts and beliefs, plus instructions, recipes, or sets of commands that show how to achieve some result, such as to prepare or make something;

Structure of Observed Learning Outcomes (SOLO) taxonomy: a taxonomy that describes levels of increasing complexity in students' understanding of subjects. In practice, a way of classifying educational learning objectives into levels of complexity and specificity. A graphical illustration of the SOLO taxonomy is given in Figure 12. A video explaining what the levels mean using LEGOs as an example can be found searching "SOLO taxonomy using LEGO" in youtube.

taxonomy: (i.e., a set of ordered labels)

Teaching and Learning Activities (TLAs): the work that is done in and out of the class in relation to the course, e.g., seminars, lectures, labs, etc. See `https://www.uwc.ac.za/TandL/Pages/TandL-Activities.aspx` for some more examples;

B How to define the knowledge components list

To be done.

C How to interpret and assign taxonomy levels

To be done.

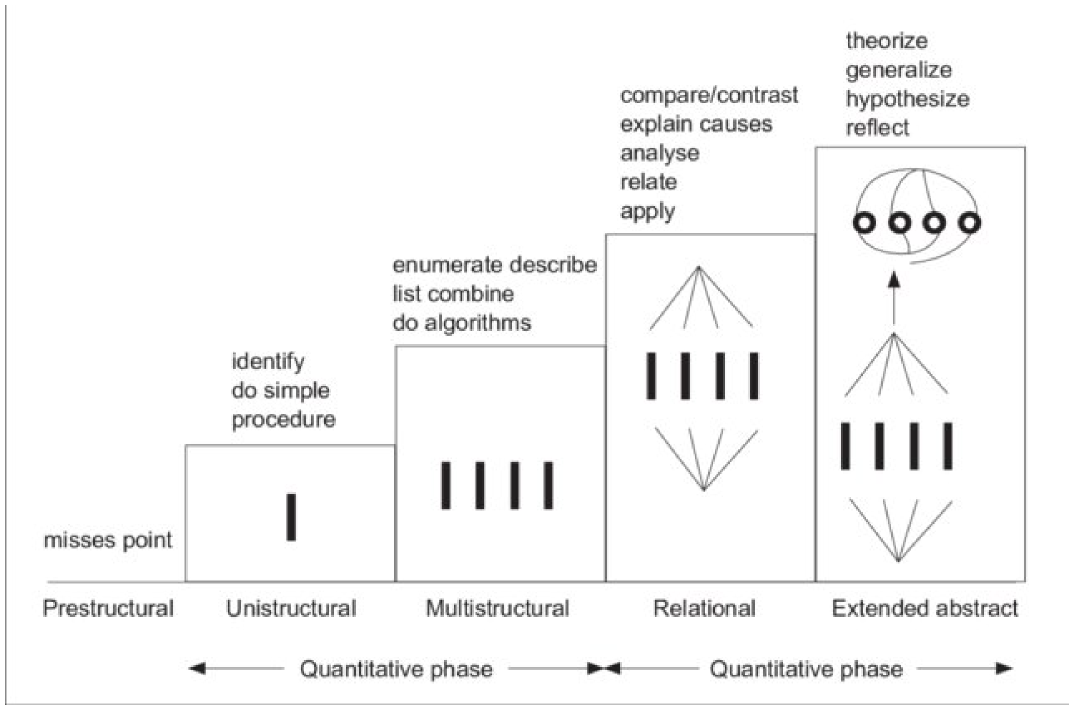


Figure 12: Graphical representation of the SOLO taxonomy as proposed in [1].

D How to identify ILOs

To be done.

E How to identify TLAs

To be done.

F Debugging

- Be sure that each `.xlsx` has 24 rows in the 'course summary tab', and that the last two rows are 'course starting date' and 'course ending date';
- put a zero in the A1 cell of the non-first-sheet of the various KCM files. This serves as an "anchor" for Matlab;
- each column in the program definition file has to be separated by **one** tab (see Section 3.1).

G Bibliography

References

[1] John B Biggs and Kevin F Collis. *Evaluation the quality of learning: the SOLO taxonomy (structure of the observed learning outcome)*. Academic Press, 1982.

[2] Benjamin S Bloom et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, pages 20–24, 1956.