THE DEVELOPMENT OF TUNTOOL
A Software Tool for the Tuning Methodology

Philippos Pouyioutas, Harald Gjermundrod and Ioanna Dionysiou
Department of Computer Science, University of Nicosia, Cyprus

Keywords: Tuning Methodology, Bologna Process, Software Tools.

Abstract: This paper presents the preliminary stages of the design of the TunTool, a software tool that supports the Tuning Methodology, a methodology used for re-engineering academic programmes of study using the European Credit Transfer System (ECTS). The paper starts by addressing the limitations of applying manually the aforementioned methodology vs. the advantages of having a tool, thus identifying the user requirements of the tool. It then presents its user interface design, giving screenshots of the tool that match the described functionality. The paper also discusses a relational database design that will underpin the development of the tool. Finally, the expected tool system architecture is briefly explained.

1 INTRODUCTION

The Tuning Methodology (Gonzalez and Wagenaar, 2008) is a methodology used for the design of academic programmes following the directives and suggestions of the Bologna Process (European Commission Education and Learning, 2008). It also provides a framework for the quality assurance of programmes. According to the Tuning Methodology, the first stage in designing a new programme is to build its profile, which includes among others, its aims and objectives, as well as the Learning Outcomes (LOs) (Kennedy et al., 2006) and the Competences (generic and subject specific) (European Commission Education and Learning, 2008; Gonzalez and Wagenaar, 2008). In order to make sure that the LOs of a programme and the expected Competences are achieved, the Tuning Methodology utilizes various matrices that relate the LOs with the various courses of the programme and the competences with the courses. Furthermore, every course of a programme has its own LOs and cultivates into students' competences, which they contribute to the LOs and the competences of the programme.

Currently, the Tuning Methodology has been adopted by many universities both in Europe and in Latin America (Beneitone et al., 2007 ; Keravnou-Papailiou, 2006). To the best of our knowledge, there is still no software tool to support the methodology and automate some of the tedious tasks that the users of the methodology have to perform. The informal design of such a tool was first presented in (Pouyioutas, 2009). In this paper, we discuss in detail the preliminary stages of the design of the TunTool. The rest of the paper is organized as follows: Section 2 addresses the problems and limitations of using the methodology without the support of a software tool. Doing so, we identify the need for the TunTool and its expected functionality, which is discussed in section 3 by providing the user interface design of the tool, accompanied by various screenshots to illustrate the TunTool operation. Section 4 proposes a relational database design that will underpin the development of the proposed tool. In section 5, we briefly expand on the expected TunTool system architecture. Finally, we conclude by discussing our current and future work.

2 THE NEED FOR TUNTOOL

When building the degree profile of an academic programme, one needs to define its Learning Outcomes (LOs) and Competences. Ideally, existing definitions could be utilized rather than reinventing the wheel. Thus, one could select as many LOs and Competences from a pool of such resources and then modify and add new ones accordingly. This not only would reduce the effort needed for building the programme profile, but also and more importantly perhaps, it would create programmes that are

Pouyioutas P., Gjermundrod H. and Dionysiou I. (2010).
THE DEVELOPMENT OF TUNTOOL - A Software Tool for the Tuning Methodology.
In *Proceedings of the 2nd International Conference on Computer Supported Education*, pages 401-406

compatible to a certain extent (of course one may argue that this compatibility would have a drawback such as reducing creativity and innovation). Even though the Tuning Methodology provides a list of generic and subject-specific competences in most subject areas, currently there is no database of LOs and Competences that would allow downloading of these resources. The creation of a database of such resources would allow one to select and use them as part of the programme profile under development, thus benefiting from the aforementioned advantages.

Another time-consuming and tedious task one faces is the verification that all programme's LOs and competences are met by at least one course of the programme. As mentioned in the Introduction, matrices could be constructed and checks could be made in order to accomplish this. In addition, a matrix entry does not necessarily have to be a Boolean value but instead a range of values could be specified that refer to the extent that a LO is met by a course. Furthermore, if one needs to find the LOs and competences achieved by a course or the courses that achieve a particular LOs and/or competence, s/he should consult the hard copy or electronic matrices and produce manually in both cases the required information. This happens because there is no database to store the relationships between LOs and courses and competences and courses. A software tool based on such database would produce automatically the required information. Such a database could even store the relationship between LOs and competences and produce some other useful information (e.g. LOs and related competences, competences and related LOs, etc.).

Furthermore, the database could store for each course its own LOs, its assessment methods, its learning methods and the expected student workload. This basically would automate the completion of the Tuning Methodology form which is used to calculate the student workload and thus the number of the ECTS of the course, reducing even more the time and effort needed for building further the programme components. The automation would also allow what-if analysis and perform workload and ECTS recalculations very fast and error-free. The system would also check the semester breakdown of the programme of study in terms of the 30/60 ECTS requirements per semester/year.

When it comes to the student calculations of their workload during a course and therefore the course ECTS, the system would allow the fast processing of all student filled tuning forms (and/or a different form to be adopted) and would produce average workloads for each course and each LO of a course and the average ECTS of the course, as estimated by the students.

All the aforementioned advantages of automating the application of the methodology clearly indicate the need for the proposed tool. TunTool (Pouyioutas, 2009) will provide a database of LOs and competences that will be accessed and shared by many users. Users will have the choice of using a LO and a competence created by another user but will not have write access on resources owned by other users. In this way, each LO and each competence is owned by its creator. Since however, each such resource after its creation will be used by other users and not just the owner, any change by the owner to such resource will have to trigger a flag to any user (including the owner) of the amended resource for accepting/adopting the new version or retaining the previous version in any part of the database that the resource is referred to. This means that the database should keep the various versions of the resources.

3 THE TUNTOOL INTERFACE

The TunTool interface has been provisionally designed based on the expected functionality of the system and its potential users. There will be three main user types, and thus three password-controlled authorised areas, namely programme coordinators, faculty members and students. The system will also support a system administrator area. The welcome screen interface will allow the user to login using his/her login name and password in one of the aforementioned areas.

3.1 System Administrator Area

The System Administrator area will provide the administrator the tools for managing (creating/editing) the end-users of the system and assigning them authorization privileges. Furthermore, the administrator will be responsible for the maintenance of the data pertaining to the programme of studies, courses and the assignment of courses into programmes of studies.

3.2 Programme Coordinator Area

The Programme Coordinator area will assist the academic faculty in charge of programmes to set up programme's LOs and competences. The first

interface screen will provide programme coordinators a list of programmes for which they are responsible. Once selecting one of the programmes, the program coordinator will be redirected to the specific programme's screen interface with a *Maintenance* menu choice that will allow him/her to:

- create/edit LOs and competences
- assign LOs and competences to the programme
- associate LOs and competences with the programme's courses

Furthermore the screen interface will support a *Reports* menu choice that will allow the coordinator to generate the following reports:

- LOs and Competences of a course
- Matrix showing the LOs and Competences of a programme vs. the programme's courses
- LOs and Competences of a programme that are not covered by any course
- Programme's total ECTS and Semester's total ECTS

Figure 1 illustrates the screen interface of the report *LOs and Competences vs. Courses* in a matrix format. The symbol √ indicates that the particular course achieves the particular LO/competence. The green colour is used in all √ cells of the matrix, whereas the red colour is used to colour any column without a √, highlighting the fact that a particular LO/Competence is not achieved by any course, thus giving a warning to the programme coordinator.

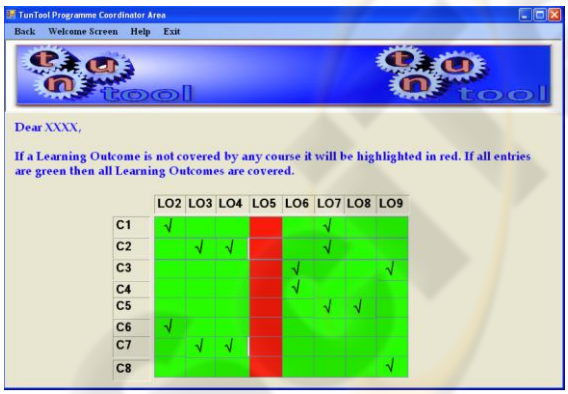

Figure 1: The Programme Matrix: "LOs and Competences vs. Courses".

3.3 Faculty Member Area

The Faculty Member area will provide a screen interface that will allow access to the courses that faculty is authorized to modify; those most likely will be the courses taught by the faculty member. Once the faculty member chooses a course, s/he will be redirected to the screen interface shown in Figure 2 that prompts the completion of the Tuning Methodology Teacher form. This form lists the LOs of the course, the associated educational activities (teaching/learning methods), the assessment methods and the estimated student workload (number of hours) that the students are expected to spend on each learning outcome.

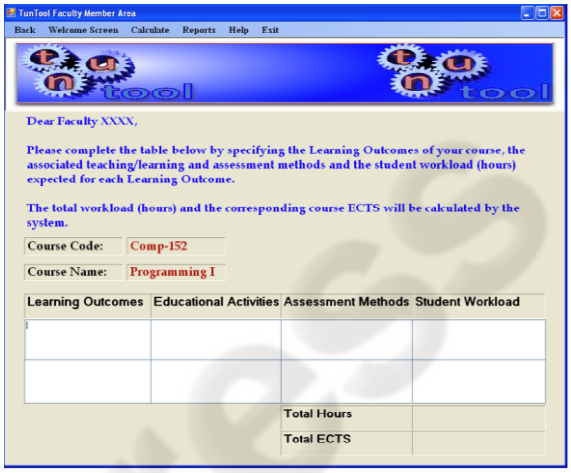

Figure 2: The Faculty Member Course Workload/ECTS Form.

The *Calculate* menu choice will allow a faculty member to calculate the total student workload in hours and thus the total ECTS of the course, whereas the *Reports* menu choice allows one to access and compare with the student estimated workload and ECTS and hence make any amendments if needed.

3.4 Student Area

The Student Area will mainly provide a screen interface, as shown in Figure 3, that will allow students to record the number of hours they spend every week in a course. The total number of hours will be automatically calculated by the system and displayed on the form. The system will also calculate the average total number of hours spent by all students in the course and thus calculate the average student workload which will then be translated into the course ECTS as estimated by the students.

We have to point out however that this method we use herein for the calculation of the course ECTS by the students (and thus the proposed interface) differs from the one suggested in the Tuning Methodology. The Tuning Methodology suggests

that the same process that is used by faculty members should also be used by the students. We strongly believe that the methodology's proposed approach will not be fully understood by the students during the semester of their studies but probably only at the end of the semester, when at that point of time it will be too late for them to estimate the course workload.

Figure 3: The Student Course WorkLoad/ECTS Form.

Our approach forces the students to record the time they spend on the course every week, without breaking down this time in terms of LOs and teaching/learning activities. At a later stage, once students get used to this approach, we may revise the form to allow students record their time according to the learning/teaching activities, but definitely not according to the LOs, since we believe that the latter may be understood only by the end of the semester (and not during the semester); at that point of time the student will not be able to reflect and estimate the time spent on the learning outcomes.

4 THE RELATIONAL DATABASE DESIGN

In this section we provide a relational database design that accommodates the system functionality as explained in the previous section. First, the ER model diagram of the application is described, followed by the corresponding decomposed ER diagram. The relational tables of the database are presented next.

4.1 The ER Model

Figure 4 illustrates the ER model that describes the the database entities and relationships of the proposed database.

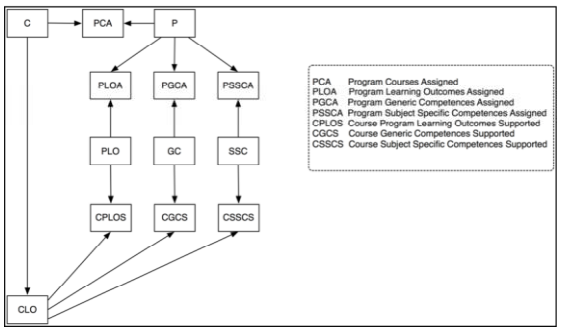

Figure 4: The ER Model of the Proposed Database.

The decomposed ER model is illustrated in Figure 5.

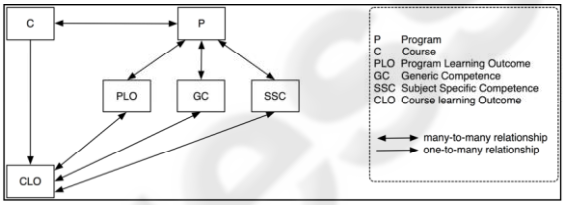

Figure 5: The Decomposed ER Model of the Proposed Database.

4.2 The Relational Tables of the Proposed Database

Based on the decomposed ER model of Figure 5, the following relational tables are created. The abbreviated entity names in the ER model are also used herein as the names of the tables.

P (Pid, Ptitle, Paims, Pobjectives)

C (Cid, Ctitle, Caims, Cobjectives)

PCA (Cid, Pid)

PLO (PLOid, PLOversionnumber, PLOtitle, PLOdescription)

GC (GCid, GCversionnumber, GCtitle, GCdescription)

SSC (SSCid, SSCversionnumber, SSCtitle, SSCdescription)

PLOA (Pid, PLOid, PLOversionnumber)

PGCA (Pid, GCid, GCversionnumber)

PSSCA (Pid, SSCid, SSCversionnumber)

CLO (CLOid, CLOtitle, CLOlearningmethod,

CLOassessmentmethod, CLOstudentworkload, Cid)

CPLOS (CLOid, PLOid, PLOvesrionnumber)

CGCS (CLOid, GCid, GCversionnumber)

CSSCS (CLOid, SSCid, SSCvesrionnumber)

5 THE TUNTOOL SYSTEM ARCHITECTURE

In order to build the database presented in the previous section, we propose to use MySQL as the back-end database management system. The client application will connect to the database through ODBC connectors of the .Net framework. The application will be written in the C# programming language, which provides for rapid prototype development and reuse of the .Net components like WYSIWYG (What You See Is What You Get) GUI building, database connectors, and LINQ (Brooks, 2008) for specifying queries. The proposed high-level architecture is depicted in Figure 6.

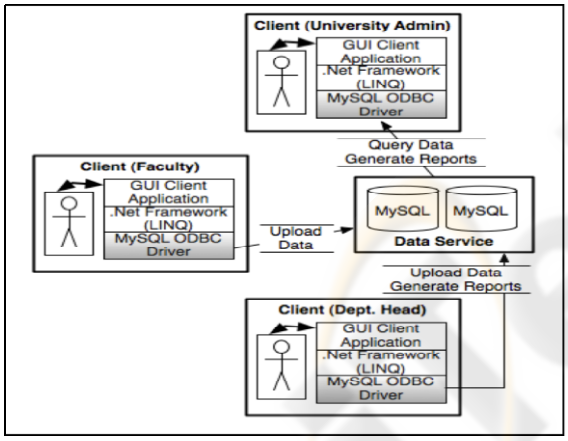

Figure 6: The Proposed TunTool System Architecture.

The dataservice component will consist of a MySQL database service. The choice of using MYSQL as the back-end database system is due to the fact that it is an open-source service that supports a wide variety of platforms with respect to both programming languages and operating systems. Future client applications could be developed in Java and targeted to run on Linux or Mac OS X. The currently proposed architecture will consist of only one database service, but a distributed solution could be provided in the future if scalability should be an issue.

The client component of the proposed architecture has a three-layer design. The lowest level consists of the ODBC connection driver which is provided by MySQL for remotely access to the MySQL database service. The second level is the .Net framework including the LNIQ query language that is provided by Microsoft. The choice of using the .Net platform for the client application is due to the possibility of rapid prototype development. Although the implementation for the framework is provided by Microsoft for the Windows platform, there exists an open source project, Mono (Avery and Holmes, 2006) that provides implementation for the .Net framework on other platforms. The top level is the client application, which is implemented in C#. The top level provides an intuitive user-friendly GUI that the user interacts with in order to populate the database as well as retrieve data from the database. The end-users will not directly specify the LINQ queries, but instead they will choose from options when generating the forms. The client application will generate the appropriate queries based on what the users specify.

6 CONCLUSIONS

This paper has presented the preliminary stages of the design of the TunTool to support the application of the Tuning Methodology for designing academic programmes using the ECTS. The need for the tool was discussed and justified through the addressing of various current limitations of applying manually the Tuning Methodology. TunTool's user interface has also been presented illustrating the tool's functionality. In order to develop the tool, we provided a relational database design. Finally the paper has briefly presented the tool's provisional system architecture.

We are currently in the process of building the database based on the proposed design and then proceeding with the development of the full functionality of the tool. We expect that the tool will be a very useful asset to all academic personnel in charge of designing academic programmes.

REFERENCES

Avery, J. and Holmes, J. (2006). *Windows Developer Power Tools.* O'Reilly Media, Inc.

Beneitone, P., Esquetini, C., Gonzalez, J., Marty, M., Siufi, G., and Wagenaar, R. (2007). *Tuning Latin America Reflections on and outlook for Higher Educatio in Latin America*. Bilbao. ISBN: 978-84-

9830-097-0.
Brooks, T. (2008). Watch this: Linq shifts the paradigm of query. *Information Research*, 13(2). Available at http://InformationR.net/ir/13- 2/ TB0806.html.
European Commission Education and Learning (2008). The bologna process. Available at http://ec.europa.eu/education/policies/educ/bologna/bologna_en.html.
Gonzalez, J. and Wagenaar, R. (2008). *Tuning Educational Structures in Europe Universities Contribution To The Bologna Process: An Introduction*. ISBN: 978-84-9830-132-8.
Kennedy, D., Hyland, A., and Ryan, N. (2006). Writing and using learning outcomes, a practical guide. *EUA Bologna Handbook*. Available at http://www.bologna-handbook.com/docs/ frames/content_c.html.
Keravnou-Papailiou, E. (2006). Implementing ects at the university of cyprus. *EUA Bologna Handbook.* Available at http://www.bologna- handbook.com/ docs/frames/content_c.html.
Pouyioutas, P. (2009). The design of the tuntool, a software tool for the tuning methodology. *In IADIS International Conference on e-Society*, volume 2, pages 75–79, Spain.