Chapter 5
The PeerLearn Application

Dave de Jonge, Bruno Rosell, Patricia Gutierrez, Nardine Osman, and Carles Sierra

There are a number of available tools that support teachers in the management of lesson plans on the web. However, none of them is task-centred and support any form of lesson plan’s execution over the web. PeerLearn is an application that allows both the design and the execution of lesson plans, where lesson plans are designed with respect to a selected rubric. PeerLearn uses electronic institutions to coordinate interactions, ensuring the rules set by the lesson plan are followed, and it relies on a trust-based model to calculate automated marks. The automated marks provide tremendous support for teachers when their online classrooms have massive numbers of students. This chapter provides an overview of the PeerLearn application, describes its underlying electronic institution, and presents a brief introduction to its automated assessment technology.

5.1 Objectives

Online education has regained popularity in recent years due to the MOOC phenomenon. In this section, we present a pedagogical framework for online learning support in the context of communities of learners. The goal is to help tutors man-
aging (specifying and monitoring) pedagogical workflows that (1) easily allow the interaction within groups of students and (2) include automatic tools to support the task of assessing massive number of students as needed in MOOCs.

In terms of management, we provide a system, based on electronic institutions [3], that allows the tutor to specify pedagogical workflows. These are lesson plans that contain course steps and their interrelations (e.g. practice tasks, quiz solving, mutual assessments, etc). Once the pedagogical workflow is defined, a specific graphical user interface (GUI) is automatically generated to allow students navigate through the lesson. Every time the tutor modifies a workflow, a new GUI is generated accordingly without any programming effort. (The automatic GUI generation that is referred to in this chapter is described in detail in Chapter 6.) In terms of marking, we deal with the problem of massive number of students registered in online courses. Students may receive assessments from other peer students, increasing the level of interaction in the community and providing feedback about their performance. As the tutor is simply not able to mark everybody (since the number of students may reach tens of thousands), our infrastructure includes a trust-based peer-assessments service to help with the process of assessing large quantities of students. The service generates tentative marks for students by comparing peer assessments to tutor assessments. Additionally, we also make use of existing services providing automatic analysis of music performances, like the performance-analysis service, which analyses the onsets of the student’s performance with respect to the tutor’s uploaded reference performance, and the progress-analysis service, which compares a new performance with a previous performance and provide feedback on the progress that the student has made since his or her previous performance.

Our framework is based on electronic institutions [3], and it integrates a number of assessment and feedback services, including our trust-based peer-assessments, the performance-analysis service and the progress-analysis service. We use electronic institutions for the implementation of pedagogical workflows. Tutors are able to define a learning workflow with the Islander tool. This workflow is executed by the EIDE framework [8], meaning that it will provide a virtual environment for the students that automatically verifies that the students correctly upload homework assignments, and that grants them access to the next stage of the workflow only if they have successfully finished the current stage. For example, a student may not be allowed to start a level-2 course if he has not yet passed the exam of the level-1 course.

In existing online learning systems, these kinds of rules need to be re-implemented at the code-level for every new learning plan. With the EIDE framework, however, this is no longer necessary because one can define the rules in a visual manner, using Islander, and the application of those rules is then automatically taken care of by the EIDE framework. As mentioned earlier, and as we illustrate in Chapter 6, the GUI which allows people to interact in an EI through a web browser can automatically be generated too, meaning that one only needs to modify the EI specification

\[^{1}\] In audio analysis, onset detection compares two audio files, looking for a number of differences, such as changes in spectral energy, or changes in detected pitch.
5 The PeerLearn Application

when needed (i.e. whenever the lesson plan needs to be modified) without having to re-design the GUI for every such modification.

The remainder of this chapter is divided as follows. Section 5.2 presents the implementation of the PeerLearn application; Section 5.3 presents the electronic institution specification underlying the PeerLearn application; and Section 5.4 describes the integrated technologies that provide assessments and feedback of students’ performances.

5.2 Illustration

In this section we present the user interface of PeerLearn [2]. We remind the reader that the user interface is automatically generated from the EI specification. As such, every time the tutor modifies her pedagogical workflow, the views of the GUI are automatically modified accordingly, and this includes the actions that one is allowed to perform at each stage of the pedagogical workflow. We also note that while PeerLearn was designed for online learning in general, the pedagogical workflow illustrated in our examples focuses on music learning.

Figures 5.1 and 5.2 present the “map” view for the tutor and student, respectively. The map view provides the user with a general idea of the pedagogical workflow and its different stages, in addition to the current stage of the user (a white box with dashed – not dotted – yellow/light grey borders) and what they need to do next (a white box with solid green/dark grey borders). Dark/Red boxes represent pedagogical steps that are not relevant to the user role viewing the map. For instance, the tutor does not need to know the details of getting the song, performing it, and providing student feedback to other’ performances (Figure 5.1). In our specific implementation, one can see that the pedagogical workflow is specified differently for each role. The tutor first needs to upload the music sheet and audio file (“Select song” step of Figure 5.1) before moving to marking the students’ performances (“Song performance evaluation” step of Figure 5.1). As for the student, he first needs to get the song uploaded by the tutor (“Get song” step of Figure 5.1), then practice and upload its own performance (“Perform” step of Figure 5.1), before assessing their fellow students’ performances (“Give and receive feedback” step of Figure 5.1).

To present a sample of the interface for one of the pedagogical steps, Figure 5.3 illustrates the view where a student can upload her own performance. The actions that may be performed in this view are presented by the ‘gear’ icons on the left hand side, and these are: (1) Send message, where a student may send a chat message to fellow students; (2) Upload track, where a student uploads its performance; (3) Performance analysis, where an audio analysis server is requested to automatically analyse the student’s performance and compare it to that of the tutor; and (4) Progress analysis, where another audio analysis server is requested to automatically analyse the student’s history of performances and assess whether the student has been improving over time or not. Figure 5.4 provides an example of the performance analysis’ output, where the higher the line is, the farther the students performance
Fig. 5.1 The tutor’s view of the pedagogical workflow

Fig. 5.2 The student’s view of the pedagogical workflow
Fig. 5.3  The view for uploading one’s performance

Fig. 5.4  The student’s view of the pedagogical workflow
Fig. 5.5 The tutor’s view of the pedagogical workflow

Fig. 5.6 The pop up window for assessing one’s performance
is from that of the tutor’s. Figure 5.5 provides an example of the progress analysis’ output, where green/light grey boxes indicate improvement areas and red/dark grey boxes indicate retrogression areas. Note that both performance analysis and progress analysis are services external to the EI engine, but made use of by having the EI specification call them when needed.

After performances are uploaded, the student may move to the next pedagogical step, where he may assess other students’ performances. Figure 5.6 illustrates the interface for students to enter their assessments on each others’ performances. Every time a new assessment is introduced, whether by a student or a tutor, the automated assessment service (another service that the EI specification calls) recalculates the assessment of each student’s performance accordingly.

### 5.3 The EI Specification

As an example we have implemented a lesson plan, as an Electronic Institutions, that could be used for the students in the Music Circle platform, a platform where students can upload and discuss music performances. The idea of this lesson plan is that there is one teacher and a group of students. The task for the students is to learn to play a certain piece of music, chosen by the teacher. The students can record their own playing and upload it to the Music Circle web page. Once uploaded they can apply an online tool that analyses the quality of the student’s playing. Students can record, upload and analyse their music several times until they are satisfied. Then, the students share their recordings with each other and make peer assessments. Finally, the teacher also makes an assessment of a subset of the students, which will be their final mark. For the other students an automatic assessment service is called that calculates the students’ final marks based on their peers’ assessments. The performative structure of this institution is displayed in Figure 5.7.

The participants in the institution can play one of the following roles:

- student
- tutor

Apart from these, the institution defines two more roles that are to be played by software agents:

- database agent
- trust agent

The institution is launched by a tutor that wishes to set up a lesson. Once the institution has started the tutor enters the institution together with a database agent. The tutor and database agent both move to the ‘Upload Assignment’ scene where the tutor can upload the assignment (consisting of the score file of the piece of music to study and an example audio file) to the Music Circle database. When uploading has finished the tutor sends a message to the database agent with the url of the location

---

2 https://goldsmiths.musiccircleproject.com
where the file was uploaded. The database agent can then access the database to verify that the file is indeed there. In general, every database agent in the institution has free access to the database. In this way the database agent can keep the administration of the uploaded files, and can provide access to certain files to the students by sending messages containing the url of that file. When the assignment has been uploaded the tutor and database agent move on to the ‘Assess Students’ scene where they are accompanied by a trust agent. Here they will wait until the students have entered, and finished their assignments.

When the students enter the institution, each of them goes to a separate instance of a ‘Download Assignment’ scene, together with a database agent. In this scene the database agent sends a message to the student containing a url to the files that have been uploaded by the tutor. Next, they move to the ‘Upload Recording’ scene. Now that the student has downloaded the score and example file he or she can start studying the piece and make a recording of him- or herself playing. Once such a recording has been made, the student can upload it in the ‘Upload Recording’ scene. Again, once the upload has finished successfully the student sends a message to the database agent containing the url.

The student and database agent can now move back and forth between the ‘Upload Recording’ scene, the ‘Analyse Progress’ scene and the ‘Analyse Performance’ scene, as often as the student wants. The ‘Analyse Performance’ scene allows the student to activate a web service that makes an onset analysis of the student’s recording compared with the example recording uploaded by the tutor. In the ‘Analyse
progress’ the student can activate a similar web service, only this time the uploaded track is also compared with a previous track uploaded by the student, so that the web service can give feedback on the progress that the student has made since his or her previous upload. In Section 5.4.2 we explain how these web services have been integrated with the EI infrastructure. Every time the student and the database agent move from one of these three scenes to another, a new instance of the target scene is created and the instance of the scene they came from is closed.

When the student feels that he has practised enough and is satisfied with his last recording he moves, together with his database agent, to an instance of the ‘Assess Peer’ scene. This scene has an instance for every student that has finished practising. Moving to this scene, one’s database agent will create a new scene instance corresponding to them. However, the point of this scene is that one enters an instance corresponding to another student. For example, if one enters the scene instance of student $x$, then they can download the final track of student $x$ and give a mark to that student.

It works as follows: if student $y$ enters the instance of student $x$ then $y$ sends a message to the database agent requesting to download student $x$’s track. The database agent will then reply with a message containing the url to that track. Student $y$ can then make a comment on the track by sending a ‘comment’ message to the database agent who will then store it in the database. Furthermore, student $y$ can make an assessment of the track by sending an ‘assess’ message to the database agent, containing a mark for each criteria (in this case the performance speed and maturity). Again, these marks are then stored in the database by the database agent. This protocol is displayed in Figure 5.8. After marking, student $y$ can leave the scene instance and move to another instance (corresponding to yet another student) of the ‘Assess Peer’ scene. A student can make as many assessments as he likes, but must make at least one assessment. Once the student feels he has made enough assessment he and his database agent move to the ‘View Assessment’ scene.

Meanwhile, the tutor is waiting in the ‘Assess Students’ scene. Whenever a student has uploaded his or her final recording, the tutor may download it and give a final mark to that student. Since there may be too many students active, it may not be feasible for the tutor to mark all present students. Therefore, he is aided by a trust agent that will calculate an assessment of all the other students, based on the peer assessments. The working of the trust agent is explained in Section 5.4.1. The protocol of the Assess Students scene is displayed in Figure 5.9. When all students have received a final mark, the tutor’s work is done, so he can move to the Exit scene and leave the institution.

When the student enters the ‘View Assessment’ scene, he can request his final mark. The protocol simply consists of the student sending a request message to the database agent, and the database agent responds with a message containing the final mark, if it is indeed available. If the final mark has not yet been written to the database (because the tutor or the trust agent have not yet given the mark) the database agent will respond with a ‘wait’ message. The student can then wait for some time and try again. When the student has received his final mark, he has completed the lesson and will therefore move to the Exit scene and leave the institution.
Fig. 5.8 The Protocol of the Peer Assessment scene

Fig. 5.9 The Protocol of the Assess Students scene
5.4 Integrated Technologies

5.4.1 Collaborative Assessment

We introduce the automated assessment service for online learning support. This service is thought for intelligent online learning applications that encourage student’s interactions benefiting from their feedback to build trust measures and provide automatic marks for students assignments.

Self and peer assessment have clear pedagogical advantages [7, 11, 10, 5, 6]. Students increase their responsibility and autonomy, get a deeper understanding of the subject, become more active in the learning process, reflect on their role in group learning, and improve their judgement skills. Online learning communities encourage different types of peer-to-peer interactions along the learning process. These interactions permit students to get more feedback, to be more motivated to improve, and to compare their own work with other students accomplishments. Tutors, on the other hand, benefit from these interactions as they get a clearer perception of the student engagement and learning process.

Peer assessment may also have the positive side effect of reducing the marking load of tutors. This is specially critical when tutors face the challenge of marking large quantities of students as needed in the increasingly popular Massive Open Online Courses (MOOC). Previous works have proposed different methods of peer assessment as part of the learning process [9, 1]. Differently from these works, we want to study the reliability of student assessments when compared with tutor assessments. We place more trust in students that have similar opinions to those of the tutor and give more weight to their assessments when calculating automated marks. Although part of the learning process is that students participate in the definition of the evaluation criteria, tutors want to be certain that the scoring of the students’ works is fair and as close as possible to his/her expert opinion.

5.4.1.1 Notation and preliminaries

We say an online course has a tutor \( \tau \), a set of peer students \( \mathcal{S} \), and a set of assignments \( \mathcal{A} \) that need to be marked by the tutor and/or students with respect to a given set of criteria \( \mathcal{C} \).

The automated assessment state \( S \) is then defined as the tuple:

\[
S = (R, \mathcal{A}, \mathcal{C}, \mathcal{L})
\]

\( R = \{ \tau \} \cup \mathcal{S} \) defines the set of possible referees (or markers), where a referee could either be the tutor \( \tau \) or some student \( s \in S \). \( \mathcal{A} \) is the set of submitted assignments that need to be marked and \( \mathcal{C} = \langle c_1, \ldots, c_n \rangle \) is the set of criteria that assignments are marked upon. \( \mathcal{L} \) is the set of marks (or assessments) made by referees, such that \( \mathcal{L} : R \times \mathcal{A} \rightarrow [0, \lambda] \) (we assume marks to be real numbers between 0 and some
maximum value $\lambda$). In other words, we define a single assessment as: $\mu_a^\rho = M$, where $\alpha \in \mathcal{C}$, $\rho \in R$, and $M = \langle m_1, \ldots, m_n \rangle$ describes the marks provided by the referee on the $n$ criteria of $\mathcal{C}$, $m_i \in [0, \lambda]$.

5.4.1.2 Similarity between marks

We define a similarity function $sim : [0, \lambda]^n \times [0, \lambda]^n \to [0, 1]$ to determine how close two assessments $\mu_a^\rho$ and $\mu_a^\eta$ are. We calculate the similarity between assessments $\mu_a^\rho = \{m_1, \ldots, m_n\}$ and $\mu_a^\eta = \{m'_1, \ldots, m'_n\}$ as follows:

$$sim(\mu_a^\rho, \mu_a^\eta) = 1 - \frac{\sum_{i=1}^n |m_i - m'_i|}{\sum_{i=1}^n \lambda}$$

This measure satisfies the basic properties of a fuzzy similarity [4]. Other similarity measures could be used.

5.4.1.3 Trust relations between referees

Tutors need to decide how much to trust assessments made by peers. We use two different intuitions for this. First, if the tutor and the student have both assessed some common assignments, then the more similar their marks are the more trusted is the student, and vice versa. Similarly, we can define the similarity of marks of any two students by looking into the assignments evaluated by both of them. In the case that there are no assignments evaluated by the tutor and a particular student, one approach would be to simply not take that student’s opinion into account because the tutor would not know how much to trust the assessment of that student. However, in our approach, we approximate that unknown trust by looking into the chain of trust between the tutor and the student through other students. To model this we define two different types of trust relations:

- **Direct trust**: This is the trust between referees $\rho, \eta \in R$ that have at least one assignment assessed in common. The trust value is the average of similarities on the assessments over the same peers. Let the set $A_{\rho, \eta}$ be the set of all assignments that have been assessed by both referees. That is, $A_{\rho, \eta} = \{ \alpha \mid \mu_a^\rho \in \mathcal{L} \text{ and } \mu_a^\eta \in \mathcal{L} \}$. Then,

$$T_D(\rho, \eta) = \frac{\sum_{\alpha \in A_{\rho, \eta}} sim(\mu_a^\rho, \mu_a^\eta)}{|A_{\rho, \eta}|}$$

We could also define direct trust as the conjunction of the similarities for all common assignments as:
\[ T_D(\rho, \eta) = \bigwedge_{\alpha \in A, \eta} \text{sim}(\mu^\alpha_\rho, \mu^\eta_\rho) \]

However, this would not be practical, as a significant difference in just one assessment of those assessed by two referees would make their mutual trust very low.

- **Indirect trust:** This is the trust between referees \( \rho, \eta \in R \) without any assignment assessed by both of them. We compute this trust as a transitive measure over chains of referees for which we have pair-wise direct trust values. We define a trust chain as a sequence of referees \( q_j = h_{r_1} \rightarrow \cdots \rightarrow h_{r_{m_j}} \) where \( r_i \in R \), \( r_1 = \rho \) and \( r_{m_j} = \eta \) and \( T_D(p_i, p_{i+1}) \) is defined for all pairs \((p_i, p_{i+1})\) with \( i \in [1, m_j - 1] \). We denote by \( \mathcal{Q}(\rho, \eta) \) the set of all trust chains between \( \rho \) and \( \eta \). Thus, indirect trust is defined as an aggregation of the direct trust values over these chains as follows:

\[ T_I(\rho, \eta) = \max_{q_j \in \mathcal{Q}(\rho, \eta)} \prod_{i \in [1, m_j - 1]} T_D(p_i, p_{i+1}) \]

Hence, indirect trust is based on the notion of transitivity.\(^3\)

Ideally, we would like to not overrate the trust of a tutor on a student, that is, we would like that \( T_D(a, b) \geq T_I(a, b) \) in all cases. Guaranteeing this in all cases is impossible, but we can decrease the number of over-trusted students by selecting an operator that gives low values to \( T_I \). In particular, we prefer to use the product \( \prod \) operator, because this is the t-norm that gives the smallest possible values. Other operators could be used, for instance the \( \min \) function.

### 5.4.1.4 Trust Graph

To provide automated assessments, our proposed method aggregates the assessments on a given assignment taking into consideration how much trusted is each marker/referee from the point of view of the tutor (i.e. taking into consideration the trust of the tutor on the referee in marking assignments). The algorithm that computes the students’ final assessment is based on a graph defined as follows:

\[ G = (R, E, w) \]

where the set of nodes \( R \) is the set of referees in \( S \), \( E \subseteq R \times R \) are edges between referees with direct or indirect trust relations, and \( w : E \rightarrow [0, 1] \) provides the trust value. We note by \( D \subseteq E \) the set of edges that link referees with direct trust. That is, \( D = \{ e \in E | T_D(e) \neq \bot \} \). An similarly, \( I \subseteq E \) for indirect trust, \( I = \{ e \in E | T_I(e) \neq \bot \} \). The \( w \) values will be used as weights to combine peer assessments and are defined as:

\(^3 T_I \) is based on a fuzzy-based similarity relation \( \text{sim} \) presented before and fulfilling the \( \otimes \)-Transitivity property: \( \text{sim}(u, v) \otimes \text{sim}(v, w) \leq \text{sim}(u, w) \), \( \forall u, v, w \in V \), where \( \otimes \) is a t-norm [4].
Figure 5.10 shows examples of trust graphs with $e \in D$ (in black) and $e \in I$ (in red/light gray) for different sets of assessments $\mathcal{Z}$.

### 5.4.1.5 Computing collaborative assessments

The first step is to build a trust graph from $\mathcal{L}$. Then, the final assessments are computed as follows. If the tutor marks an assignment, then the tutor mark is considered the final mark. Otherwise, a weighted average ($\mu_a$) of the marks of student peers is calculated for this assignment, where the weight of each peer is the trust value between the tutor and that peer. Other forms of aggregation could be considered to calculate $\mu_a$, for instance a peer assessment may be discarded if it is very far from the rest of assessments, or if the referee’s trust falls below a certain threshold.

Figure 5.10 illustrates how the trust graph evolves with new assessments. The criteria $\mathcal{C}$ in this example are speed and maturity and the maximum mark value is $\lambda = 10$. For simplicity we only represent referees that made assessments in $\mathcal{L}$. In Figure 5.10(a) there is one node, the tutor, who has made the first assessment over assignment $e_{x_1}$. There are no links to other nodes as no one else has assessed anything yet. In (b) student Dave assesses the same exercise as the tutor and thus a link is created between them. The trust value $w(tutor, Dave) = T_D(tutor, Dave)$ is high since their marks were similar. In (c) a new assessment by Dave is added with no consequences on the trust graph. In (d) student Patricia adds an assessment on $e_{x_2}$ that allows building direct trust between Dave and Patricia and indirect trust between the tutor and Patricia, through Dave. The automated assessments generated in case (d) are: $(5, 5)$ for exercise 1 (which preserves the tutor’s assessment) and $(3.7, 3.7)$ for exercise 2 (which uses a weighted aggregation of the peers’ assessments).

### 5.4.2 Integrating Web Services

When learning to play a musical instrument online one would typically require access to certain resources, such as example music files and score files. Also, a student should be able to record his or her performance and share it with his teacher, or fellow students. More generally, we want the users in an EI-based application to be able to access web services. The problem however is that the existing EI-infrastructure only provides a mechanism for sending short text messages. It does not support the exchange of bulk data, such as audio files.

Of course, one could still allow users to exchange their files and to access web services outside the EI-framework, but that would violate the purpose of the Electronic Institution. The point of using Electronic Institution is that one can control the actions of the users, by allowing users to perform certain actions only under
certain conditions. Therefore we need a mechanism that on one hand allows users to access web services and exchange files, but on the other hand makes sure that the Electronic Institution stays in control and is able to control users’ access of those tools.

One can imagine for example a web service that analyses uploaded audio files and gives feedback to the user about the quality of certain technical aspects. Such a tool would require a lot of CPU time on the server and therefore we may want to restrict users to upload a file to this service, say, once a day. Also we could provide more access to the service to premium users.

We describe here how we have solved this problem with the existing Electronic Institutions framework. As an example we take a web service in which a user sends an audio file, containing music played by the user with a musical instrument, together with a file that represents the score of the music, to a web service which then analyses the ‘onset’ of the musical notes. The results of this analysis are then sent back to the user in HTML, JSON or XML format. Using this tool works as follows (see also Figure 5.11):

1. The user clicks ‘open webservice’, which causes an http request to be sent to the EI-server, which passes the request on to the GuiAgent.
2. The GuiAgent sends an EI-message within the institution to a WebServiceAgent.
3. The web service agent sends an ‘access granted’ or ‘access denied’ message, with a url to the web service, back to the GuiAgent.
4. The GuiAgent passes this message back to the browser in an http response.
5. The browser then opens the url in a separate window.
6. The new window enables the user to send data to the web service. For example, it allows the user to choose an audio file to upload. The user clicks submit to send the data. This will cause the data to indeed be sent to the web service, but also sends a notification to the EI-server.
Interaction between an user and a web service, mediated by an EI. Solid arrows represent exchange of messages, dashed arrows represent the creation of a resource. The arrows are numbered in chronological order.

7. When the web service has analysed the audio, the results are stored on the server.
8. The web service notifies the WebServiceAgent that the data is ready.
9. The WebServiceAgent sends an EI-Message to the GuiAgent including a url to the file with the results.
10. The GuiAgent sends the url to the browser, which can then use this url to load the data.

Note that indeed the EI is in control of the interactions between the user and the web service. The users cannot access the service without permission from the EI, since the user needs the url of the web service. The EI could send, together with the url, a password that can be used only once, to make sure that the user needs to request permission from the EI every time.

We have assumed in the above that the web service provides its own GUI, that needs to be opened in a separate browser window. Of course, this is not necessary, and the GUI to the web service may as well be fully integrated with the EI GUI. In that case step 5 can be skipped.

References

5 The PeerLearn Application


