Pedagogical agents for social music learning in Crowd-based Socio-Cognitive Systems

Matthew Yee-King and Mark d'Inverno

Goldsmiths, University of London, UK
dinverno@gold.ac.uk

Abstract. This paper considers some of the issues involved in building a crowd-based system for learning music socially in communities. The effective implementation of building such systems provides several fascinating challenges if they are to be sufficiently flexible and personal for effective social learning to take place when they are large number of users. Based on our experiences of building the infrastructure for a crowd-based music learning system in Goldsmiths called MusicCircle we address several some of the challenges using an agent based approach, employing formal specifications to articulate the agent design which can later be used for software development. The challenges addressed are: 1) How can a learner be provided with a personalised learning experience? 2) How can a learner make best use of the heterogenous community of humans and agents who co-habit the virtual learning environment? We present formal specifications for an open learner model, a learning environment, learning plans and a personal learning agent. The open learner model represents the learner as having current and desired skills and knowledge and past and present learning plans. The learning environment is an online platform affording learning tasks which can be carried out by individuals or communities of users and agents. Tasks are connected together into learning plans, with pre and post conditions. We demonstrate how the personal learning agent can find learning plans and propose social connections for its user within a system which affords a dynamic set of learning plans and a range of human/ agent social relationships, such as learner-teacher, learner-learner and producer-commentator.

1 Introduction

2012 has been called the 'year of the MOOC', the massive, open, online course [13]. Indeed one of the authors of this paper was part of a team which delivered a course to an enrolled student body of around 100,000 in 2013. The obvious problem with MOOCs is that there is a very high student to tutor ratio. This means it is not feasible to provide students with direct tutor support when they have problems with their learning and complex assessments which cannot be automated become impractical. The current solutions seem to be the use of forums and other social media wherein peer support can take place, and the use of peer assessment techniques such as calibrated peer assessment [7]. Running our MOOC, we noticed that the forum seemed to be an inefficient tool through which students could find information, where the same questions would be asked and answered repeatedly, and where the constant churn pushed old answers

away[1]. It was not clear who would bother to answer a given question, or who would be the ideal person to answer it. Regarding the assessment, there was a tendency to assess others' work superficially - to simply fulfill the most basic requirements of the peer assessment task. This was probably an instance of strategic learning, where the learner does the minimum to meet the apparent requirements. Another problem is a high drop out rate on courses. For example, we had around 10% of our 100.000 students still active at the end of our MOOC; Norvig and Thrun's famous Stanford AI CS211 course in 2011 went from 160,000 enrolments to 20,000 completions [14]. These figures improve if we instead consider the number of students actively accessing learning materials at the start of the course; in our case, 100.000 becomes 36,000. So motivation to complete the course is another area that needs work.

But how might one motivate a learner, given the particular characteristics of a MOOC, i.e. the high learner to teacher ratio, the presence of a large, heterogeneous peer group, the distance learning component and so on? Might motivation be amplified by leveraging the learner's peers - the social network? What might a 'networked learner' gain from being part of an active learning community? How can the learner be made aware of the structure and members of the community, and how that might help them achieve their learning goals?

In summary, guidance for learners, feedback to learners (on their work) and general learner motivation are areas for improvement for MOOCs. These are the key points we aim to address in our wider research work. In this paper we present our work on a representative component of this: the invention of a type of pedagogical agent called a *personal learning agent* which can provide a more intuitive and efficient route through the learning materials and information, which can hep the learner to network to find help or to provide help and feedback to others.

1.1 Pedagogical agents

Skiar et al. present a review of research where agents are used [15] and we propose the reader look at this for more details than we are able to present here. According to Soliman and Guetl, Intelligent Pedagogical Agents (IPAs) are agents which help learners by providing narrations and guidance in order to resolve difficulties and improve motivation' [16]. Magus et al. describe a math tutoring game which includes a conversational agent [11] which has some similar agent characteristics.

Animated pedagogical agents (APAs) operating in realtime virtual learning environments allow learners to request information that helps build an understanding of a student's thought processes and methods of knowledge acquisition [6]. Lester et al. trialled a 3D animated character with 100 middle school children. They discuss the *persona effect*, which encompasses the agent's encouragement (of learners), utility, credibility, and clarity, and which is much enhanced by the use of an animated character [8]. Johnson et al. provide a list of technical issues for designers of animated pedagogical agents to consider [5] and the interested reader is recommended to look at this paper for more details.

[1] this is somewhat alleviated by up-and down-voting of questions and answers but this is far from perfect

Xiao et al. empirically assessed the effect of pedagogical agent competency where learners were learning how to use a text editor supported by pedagogical agents with varying competency at the task [18]. Baylor et al. present an initial study where agents take on different roles (as in Electronic Institutions [4] when supporting learners: Motivator, Expert, or Mentor. More knowledgeable agents were more credible and seemed to transfer more knowledge but motivating agents were more engaging [1]. In [17], an agent based approach is used to simulate interactions between learners within a group.

2 The Music Circle System

The word presented in this paper is part of the efforts to build an online learning system for massively online learning of music within normative communities of practice. It is funded under the FP7 Technology-Enhanced Learning Program called Practice and Performance Analysis Inspiring Social Education (PRAISE). The first author is the Technical Project manager the project and the second author is the principal investigator. The major research questions of the project are as follows

1. How to evidence increase participation in musical learning activity?
2. Is giving and receiving feedback related to engagement with practice?
3. What is the right level of social coordination?
4. How to evidence musical learning?
5. How can automatic techniques be used to evidence feedback?
6. How to build a personal learning agent for personalised learning experiences?

In order to achieve this one of the key technologies we are developing is that of the *personal learning agents* that can enable pathways to and through information for learners, better feedback to learners (on their work) and general learner motivation, and as a resource for finding fellow students within which we can learnt together. In this paper, we will address the following questions which fall within this wider remit:

1. How might one formally specify a human learner that enables an autonomous personal learning agent to represent them in a crowd-based system for music learning?
2. What kind of operations might be useful, given the wider research goals articulated above?

The specific online system we are developing within the PRAISE project is called Music Circle and next we describe Music Circle and why it meets all the criteria for a Crowd-based Socio-technical System.

3 Music Circle as an example of a Crowd-based Socio-technical System

Before moving onto specify the personal learning architecture we first wish to state why this system fits the definition of a crowd-based socio-technical system that is described in a sister paper in this workshop by Pablo Noriega and Mark d'Inverno entitled "Crowd-Based Socio-Cognitive Systems" [12]. In this section we take their description of a *Crowd-Based Socio-cognitive System* and item by item explain why our Music Circle system matches the description given by Noriega and d'Inverno.

- Dimension 1. The Music Circle system contains *agents* which are both computational or human and are autonomous in that they can exhibit purposeful behaviour.
- Dimension 2. The population is a *mix* of human and software agents.
- Dimension 3. The human and computational agents have a *model* of the system in which they operate.
- Dimension 4. The agents within the Music Circle system are *rational* in that they are capable of choosing different courses of action based on their own models (however simple or complex these may be).
- Dimension 5. The human and computational agents in Music Circle are *social* in that they interact with other agents.
- Dimension 6. The human and computational agents in Music Circle have *social models* of the other agents in the system.
- Dimension 7. The human and computation agents are *socio-cognitive* in the sense that they base their decisions on some decision-making process which takes into account the models of the other agents in the Music Circle system.
- Dimension 8. The agents in Music Circle have *social capabilities* including awareness and models of others, and the ability to understand the norms of the learning community in which they are situated.
- Dimension 9. The Music Circle system is defined by the system of interacting agents which means that the state of the system at any stage can never been known by us as the designers and engineers of the systems (opacity).
- Dimension 10. Agents may enter and leave our system at any time and cannot be known in advance.
- Dimension 11. Music Circle communities are *regulated* in order to enable the social coordination of music learning in communities.
- Dimension 12. The human agents are *autonomous* and may not necessarily want to provide helpful feedback to the other music learners in the community and so we need to regulate communities in order to be sensitive to how such agents could destroy the trust within a learning community.
- Dimension 13. A Music Circle system is dialogical as all interactions are mediated by technological artefacts and may therefore be wrapped as communicative acts or messages typically in relation to audio media.
- Dimension 14. We currently have several hundred of students enrolled with the hope and expectation of getting this to thousands of music learners within the next 12 months
- Dimension 15. The system is being designed specifically so that norms can be determined by the community. Moreover, we are specifically including models of how trust and reputation arise and can be managed within such systems.
- Dimension 16. We are developing the Personal learning agent, and the Music Circle system in general (including norms, coordination, rest and reputation managements systems and so on) so that it will help the human users take a wide-variety of feedback on audio and suggested plans for practice from different agents in order to synthesise a particular plan of practice activity.

That is to say that that our system is central example of a *Crowd-based Socio-Cognitive System* and we are using norm-based MAS techniques in the specification,

design and implementation of our system. Next, we move to the specification of the personal learning agent which works with the human learning agent to facilitate music learning in our PRAISE system.

4 Requirements of the Personal Learning Agent

There are several issues about large online systems including (i) motivating the learner (high drop out rate, personalised learning pathways) (ii) connecting the learner (who can help me with this? and (iii) who is having the same problems, etc.) and giving the learner an individual pathway (how can I learn to do this?) One approach to combat these issues is to define a Personal Learning Agent we will use the specification techniques developed by Luck and d'Inverno over the last 20 years or so (e.g. [2, 9, 10]).

We will begin by framing the agent specification presented later with some requirements for the functionality of the agent. There are 4 key requirements: to store learner state, to report learner state, to find learning plans and to propose social connections. Each of these requirements has sub-requirements, as listed below:

1. Storing learner state:
   (a) Storing the goals of a person
   (b) Interpret the goals into desired skills and knowledge
   (c) Storing a person's current skills and knowledge
   (d) Storing a person's current and previous plans
2. Reporting learner state:
   (a) Reporting current state of goals and plans
   (b) Reporting current state of knowledge and skills
   (c) Reporting status of data/ content provided to and from the community i.e. plans, feedback, feedback agents, trust model
3. Plan finding:
   (a) Propose plans whose pre-conditions match current skills and knowledge
   (b) Propose plans whose post-conditions (goals) match a personal learning agent's goals
   (c) Generate evaluation data for plans based on users
   (d) Propose plans which are successful, i.e. verified post conditions
4. Agent finding:
   (a) Propose social relationships/ connections to people with similar goals/ skills/ knowledge (potential peers, potential as they must actively agree to connect to make a social relationship)
   (b) Propose connections to people with similar (musical/ geographical/ etc.) data
   (c) Propose connections to people who have related but superior skills and knowledge (potential tutors), or teaching goals. (I want to increase others' knowledge of scales on the guitar). These people might be able to assign plans, for example.

5 Formal specification of the Personal Learning Agent

In this section we will use the specification language Z to develop the models of our agents, following the methodology developed by Luck and d'Inverno [2, 9, 3].

Learner model

We begin our description by introducing our learner model. The purpose of the learner model is to represent various aspects of a person operating within our learning environment. There are two *types* which users of the system might want to learn about or teach about. The specification remains neutral about how they are encoded but this encoding might include free text descriptions or formulas in predicate calculous for example.

$$[Skill, Knowledge]$$

As an example a user might have the skill of playing the C major scale and the knowledge which includes being able to state which notes are in the scale of C major.

We then define *Proficiency* as the combination of skills and knowledge, representing all that a person would potentially wish to learn in music.

$$Proficiency ::= skills\langle\langle Skill\rangle\rangle \mid knowledge\langle\langle Knowledge\rangle\rangle$$

A particular person can be given a score which is an evaluation of their learning level regarding a particular skill or knowledge element:

$$Score == \mathbb{N}$$

Learning environment

We continue the description with some details about the learning environment which learners, teachers and agents will inhabit. For the purposes of our wider research, it is specialised for music education, and it is designed around a social, blended learning pedagogy wherein people upload recordings of themselves playing instruments and other media items. Discussion and feedback can occur around the uploaded items. Within the environment, people and agents can carry out tasks, where a task is something to be undertaken.

$$[Task]$$

We have identified 9 distinct tasks which can be carried out within our learning environment.

$$\begin{aligned} TaskType ::= {} & Practice \mid Listen \mid Makemusic \mid \\ & Upload \mid Share \mid Annotate \mid \\ & Question \mid Answer \mid Visualise \end{aligned}$$

Earlier, we mentioned that feedback might be provided about a media item. For the time being we define feedback as a given set. It is possible to define feedback in terms of constructive and evaluative praise and criticism. However, these are our first attempts at defining feedback we will remain neutral for the time being.

$$[Feedback]$$

We define evaluate to be a function which maps an proficiency to a natural number, e.g. I have evaluated the way you have played C major as scoring a 5.

$$evaluateproficiency : Proficiency \rightarrow \mathbb{N}$$

In the system the community may evaluate many different aspects. One of those is to evaluate feedback for example.

$$evaluatefeedback : Feedback \rightarrow \mathbb{N}$$

Goals, Beliefs and Plans

As with the definition of the SMART Agent Framework [2] we take a goal to be a state of affairs in the world that is to be achieved (by some agent).

$$[Goal]$$

The way that goals (or, equally, learning outcomes) are achieved is through a workflow of tasks: a sequence of tasks that have to be completed in order. We do not specify here who determines whether the tasks have been accomplished successfully or not because in general this could be a mixture of the system, the user themselves, the community and/or a teacher. Plans are typically specified in terms of what must be true before they can be adopted, what is true after they have been successfully completed, and the kinds of actions (or in our language tasks) that have to be completed in order Next we define a plan to be a set of preconditions (the skills and knowledge and agent must have before undertaking the plan) and a set of post conditions (which describe the new set of skills and knowledge the agent will have after the plan). The predicate part of the schema state that the intersection of the pre and post conditions are necessarily empty.

Plan

$$pre : \mathbb{P}\, Proficiency$$
$$post : \mathbb{P}\, Proficiency$$
$$workflow : \text{seq}\, Task$$

$$pre \cap post = \{\}$$

In specifying this system, it is useful to be able to assert that an element is optional. The following definitions provide for a new type, $optional[T]$, for any existing type, T, which consists of the empty set and singleton sets containing elements of T. The predicates, *defined* and *undefined* test whether an element of $optional[T]$ is defined (i.e. contains an element of type T) or not (i.e. is the empty set), and the function, *the*, extracts the element from a defined member of $optional[T]$.

$$optional[X] == \{xs : \mathbb{P}\, X \mid \# xs \leq 1\}$$

$$
\begin{array}{|l}
\hline
[X] \\
\textit{defined}\_, \textit{undefined}\_ : \mathbb{P}(\textit{optional}[X]) \\
\textit{the} : \textit{optional}[X] \pfun X \\
\hline
\forall\, \textit{xs} : \textit{optional}[X] \bullet \\
\quad \textit{defined xs} \Leftrightarrow \# \textit{xs} = 1 \land \\
\quad \textit{undefined xs} \Leftrightarrow \# \textit{xs} = 0 \\
\forall\, \textit{xs} : \textit{optional}[X] \mid \textit{defined xs} \bullet \\
\quad \textit{the xs} = (\mu\, x : X \mid x \in \textit{xs}) \\
\hline
\end{array}
$$

$$\textit{Bool} ::= \textit{True} \mid \textit{False}$$

Using this definition we can now specify the state of a plan. The state of a plan can be thought of as a running instance of a plan during the lifetime of a users activity. It means that the plan has been adopted to achieve a goal. In order to specify this we keep the information contained in the specification of a Plan using schema inclusion. We also state that if the plan has been started but not finished there will be a *current task* that the agent is currently undergoing. By also defining a flag called *finished* we can specify a plan state as follows. The predicate part states that the current task must have been defined in the workflow of the plan.

$$
\begin{array}{|l}
\hline
\textit{PlanInstance} \\
\textit{Plan} \\
\textit{current} : \textit{optional}[\textit{Task}] \\
\textit{finished} : \textit{Bool} \\
\hline
\textit{thecurrent} \in (\operatorname{ran} \textit{workflow}) \\
\hline
\end{array}
$$

The initial plan state (for any state schema the initial state should be specified in Z) is where the plan has just been proposed or adopted by a user.

$$
\begin{array}{|l}
\hline
\textit{InitialPlanInstance} \\
\textit{PlanInstance} \\
\hline
\textit{undefined current} \\
\textit{finished} = \textit{False} \\
\hline
\end{array}
$$

We are now in a position to define four specific sub-types of the plan state as follows.

1. Proposed Plan. A plan which has been selected to achieve a goal but which has not been started by the agent. As no task has been started the current task is set to undefined.

$$
\begin{array}{|l}
\hline
\textit{ProposedPlan} \\
\textit{InitialPlanInstance} \\
\hline
\end{array}
$$

2. Active Plan. A plan which is ongoing. It has not been completed and the current task is set to defined.

$$
\begin{array}{|l}
\hline
\textit{ActivePlan} \\
\textit{PlanInstance} \\
\hline
\textit{defined current} \\
\textit{finished} = \textit{False} \\
\hline
\end{array}
$$

3. FailedPlan. This is a plan which has a defined task but a flag set to finished. For example, this represents a situation one of the tasks in the workflow of a plan is too difficult for the user and the plan is discarded by the user.

$$
\begin{array}{|l}
\hline
\textit{FailedPlan} \\
\textit{PlanInstance} \\
\hline
\textit{definedcurrent} \\
\textit{finished} = \textit{True} \\
\hline
\end{array}
$$

4. Completed Plan. The flag *finished* is set to true and the current task becomes undefined.

$$
\begin{array}{|l}
\hline
\textit{CompletedPlan} \\
\textit{PlanInstance} \\
\hline
\textit{undefined current} \\
\textit{finished} = \textit{True} \\
\hline
\end{array}
$$

There are several operations that we could specify at the level of the plan but the key one is finish task. Either this leads to the plan being completed or the current place in the work flow moves to the next task.

In the first case the specification looks like this:

$$
\begin{array}{|l}
\hline
\textit{FinishTask}1 \\
\Delta \textit{PlanInstance} \\
\hline
\textit{current} = \{\textit{last}(\textit{workflow})\} \\
\textit{finished} = \textit{False} \\
\textit{undefined current}' \\
\textit{finished}' = \textit{False} \\
\hline
\end{array}
$$

In the second case like this:

$$
\begin{array}{|l}
\hline
\textit{FinishTask}2 \\
\Delta \textit{PlanInstance} \\
\hline
\textit{current} \neq \{\textit{last}(\textit{workflow})\} \\
\textit{finished} = \textit{False} \\
\textit{current}' = \{\textit{workflow}((\textit{workflow}^{\sim}(\textit{the current})) + 1)\} \\
\textit{finished}' = \textit{False} \\
\hline
\end{array}
$$

The other is to instantiate a plan which essentially means creating a PlanInstance in it is initial state from a Plan.

$$
\begin{array}{|l}
instantiateplan : Plan \rightarrow InitialPlanInstance \\
\hline
\forall p : Plan;\ ps : InitialPlanInstance \\
\quad \mid ps = instantiateplan(p) \bullet \\
\quad ps.pre = p.pre \land ps.post \\
\quad = p.post \land ps.workflow = p.workflow
\end{array}
$$

The (almost) inverse function of this is a function which takes any PlanInstance and returns the plan.

$$
\begin{array}{|l}
recoverplan : PlanInstance \rightarrow Plan \\
\hline
\forall p : Plan;\ ps : PlanInstance \mid p = recoverplan(ps) \bullet \\
\quad ps.pre = p.pre \land \\
\quad ps.post = p.post \land ps.workflow = p.workflow
\end{array}
$$

Beliefs

This is a representation of what the agent knows and what it can do. Again we remain neutral on the representation.

$$[Belief]$$

The Personal Learning Agent In the schema below we have the following definitions.

1. An agent has a set of goals at any stage which we call desires (typically these are associated with learning outcomes as described earlier in the document.)
2. An agent has a set of beliefs. These refer to the information which is stored about what the user knows or what the user can do (skills).
3. An agent has some interpret function which takes a goal and returns a set of proficiency (skills and knowledge). Note that the complexity of this function may vary as in some cases goals may be expressed as a set of proficiency directly and so this function becomes a simple identity function. However, in other situations this function has to take a free text description and turn it into a set proficiency. Clearly, in general no automatic process can do this and such an operation will often be left to the community. In which case we specify the agents interpret function as a *partial* function.
4. An agent has a similar interpret function for beliefs which maps its beliefs to a set of machine readable (skills and knowledge).
5. *intdesires* is a set of proficiencies which can then be used by the agent and the community to plan. Note then, that *interpreteddesires* is made up of the automatic function *interpret* of the agent, possibly the automatic interpretation of other agents, but also from human users in the music learning community.
6. *intbeliefs* is the analagous set of proficiencies which the agent has recorded as known or accompished by the agent.

7. It is not unreasonable to suggest that all tasks are not available to a user at all times and so the agent can record which tasks are currently available to a user. (If the internet is down, upload is not an available task. If a newcomer joins a community then possibly they do not feel like giving any feedback and so the agent can record that the user is currently not offering this task.).
8. Then we define the set of plans which the agent knows about (possibility learned from other agents). This is where the agent contains its *procedural knowledge* about what plans work in what situations to achieve which desired proficiency.
9. The agent maintains a record of all of the plans that have been completed and all of those which have failed.
10. There is a record of the intentions. This is a mapping from a set of proficiencies (this set may only have one proficiency in it of course) to the plan instance which the agent has adopted to attain those proficiencies.
11. Finally, we record all those interpreted desires for which the agent has no active plan.

    There are also two dummy variables that we can use (which can be calculated from the variables described so far but which aid us in the readability of the specification)
12. We define a variable record the tasks that the agent is current involved in (*currenttasks*) which can be calculated as the union of the tasks from the current plans.
13. We define a variable to records the current plan instances of the agent

Next we consider the constraints on the state of a personal learning agent

1. The interpreted desires are the result of applying the interpret desire function to the desires.
2. The interpreted beliefs are the result of applying the interpret desire function to the beliefs.
3. The intersection between interpreted desires and interpreted beliefs is an empty set, (in other words you can't desire a proficiency you already have).
4. If there is a plan for a subset of proficiencies then those proficiencies must be contained in the the interpreted desires.
5. If there is a plan for one subset of proficiencies and a plan for another distinct set pif proficiencies then their intersection is empty.
6. The unplanned desires are those interpreted desires for which there is no intention
7. The current tasks are calculated from taken the current plans and the current task from each.
8. The current plans are calculated from taking the range of the intentions.

$[X, Y]$

$$map : (X \to Y) \to (\operatorname{seq} X) \to (\operatorname{seq} Y)$$
$$mapset : (X \to Y) \to (\mathbb{P} X) \to (\mathbb{P} Y)$$

$$\forall f : X \to Y;\ x : X;\ xs, ys : \operatorname{seq} X \bullet$$
$$map\ f\ \langle\rangle = \langle\rangle \land$$
$$map\ f\ \langle x \rangle = \langle f\ x \rangle \land$$
$$map\ f\ (xs \frown ys) = map\ f\ xs \frown map\ f\ ys$$
$$\forall f : X \to Y;\ xs : \mathbb{P} X \bullet$$
$$mapset\ f\ xs = \{x : xs \bullet f\ x\}$$

PersonalLearningAgent

$$desires : \mathbb{P}\ Goal$$
$$beliefs : \mathbb{P}\ Belief$$
$$interpretdes : Goal \pfun \mathbb{P}\ Proficiency$$
$$interpretbel : Belief \pfun \mathbb{P}\ Proficiency$$
$$intdesires : \mathbb{P}\ Proficiency$$
$$intbeliefs : \mathbb{P}\ Proficiency$$
$$availabletasks : \mathbb{P}\ TaskType$$
$$plandatabase : \mathbb{P}\ Plan$$
$$completedplans, failedplans : \mathbb{P}\ Plan$$
$$intentions : (\mathbb{P}\ Proficiency) \pfun PlanInstance$$
$$unplannedintdesires : \mathbb{P}\ Proficiency$$
$$currenttasks : \mathbb{P}\ Task$$
$$currentplaninstances : \mathbb{P}\ PlanInstance$$

$$intdesires = \bigcup(mapset\ interpretdes\ desires)$$
$$intbeliefs = \bigcup(mapset\ interpretbel\ beliefs)$$
$$intdesires \cap intbeliefs = \varnothing$$
$$\bigcup(\operatorname{dom} intentions) \subseteq intdesires$$
$$\forall ps1, ps2 : \mathbb{P}\ Proficiency \mid$$
$$(ps1 \neq ps2) \land (\{ps1, ps2\} \subseteq$$
$$(\operatorname{dom} intentions)) \bullet$$
$$ps1 \cap ps2 = \{\}$$
$$unplannedintdesires = \bigcup(\operatorname{dom} intentions) \setminus intdesires$$
$$currenttasks = \{t : Task;\ ps : PlanInstance \mid$$
$$ps \in (\operatorname{ran} intentions) \bullet the\ ps.current\}$$
$$currentplaninstances = \operatorname{ran} intentions$$

Plan Finding Plan finding is the process of taking a set of candidate plans and selecting those whose preconditions are met and where at least some subset of the postconditions are desired. For this operation we assume the input of a set of candidate plans. Again we do not specify whether these comes from the agent (i.e. the agent's database of plans),

other agents in the community, or from the user or from other users and in general with be a *synthesis* of the users and the agents of users working together.

For now we will suppose that suitable plans have all preconditions satisfied and it is the case that both: (a) none of the postconditions are things which the user is already proficient in (b) all of the postconditions are current interpreted desires of the user. In the schema below *SuitablePlans* is generated which satisfy this constraint and from these one plan *adoptedplan* is selected. The state of the agent is then updated to include that the current plans now includes a mapping from the pre-conditions of the plan (which are necessarily interpreteddesires for which no plan exists.

Plan Completion The very simplest way this could happen is as follows:

1. Because of a successfully completed task a plan instance becomes an element of *CompletedPlan*
2. The post conditions are added to the interpreted beliefs (these may in turn be reverse interpreted into beliefs which can then be seen by the community)
3. Any post conditions that were formerly desires are now removed from interpreted desires (these may in turn be reverse inetreprered into beliefs which can then be seen by the community)
4. The completed plans function is updated with the plan that has just successfully completed.

CompletePlan

$$
\begin{array}{l}
completedplan? : CompletedPlan \\
\Delta PersonalLearningAgent \\
\hline
completedplan? \in (\mathrm{ran}\, intentions) \\
intentions' = intentions \rhd \{completedplan?\} \\
intdesires' = intdesires \setminus completedplan?.post \\
intbeliefs' = intbeliefs \cup completedplan?.post \\
completedplans' = completedplans \cup \\
\qquad \{recoverplan\ completedplan?\}
\end{array}
$$

However, this process will not be automatic in general within the system. In general, the user (or other users in the community) will be asked to evaluate the plan. There may be several ways in which this can happen. For example, a simple score could be given but in general each user who is evaluaring the plan considers each of the post conditions (or another member of the community does) to work out whether they are now proficiencies (intepreted beliefs), whether they have not been met and so are still interpreted desires, or whether they have not been met but are not desires. Indeed the evaluating user could rank each of the postconditions with a score and the agent may also wish to keep a snapshopt of the agent's state for future comparison by the community.

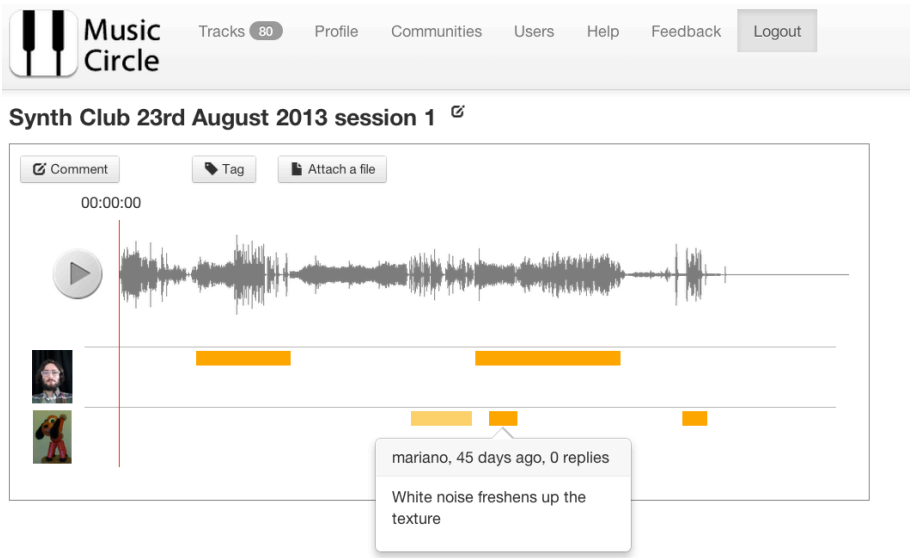

Fig. 1. The music discussion user interface

Finding and adopting a plan Plan finding is the process of taking a set of candidate plans and selecting those whose preconditions are met and where at least some subset of the postconditions are desired. For this operation we assume the input of a set of candidate plans. Again we do not specify whether these comes from the agent (i.e. the agent's database of plans), other agents in the community, or from the user or from other users and in general with be a *synthesis* of the users and the agents of users working together. For now we will suppose that suitable plans have all preconditions satisfied and it is the case that both: (a) none of the postconditions are things which the user is already proficient in (b) all of the postconditions are current interpreted desires of the user. In the schema below *SuitablePlans* is generated which satisfy this constraint and from these one plan *adoptedplan* is selected. The state of the agent is then updated to include that the current plans now includes a mapping from the pre-conditions of the plan (which are necessarily interpreteddesires for which no plan exists.

FindandAdoptPlan

$$PossiblePlans?, SuitablePlans! : \mathbb{P}\, Plan$$
$$adoptedplan : Plan$$
$$\Delta PersonalLearningAgent$$

$$SuitablePlans! = \{ps : PossiblePlans? \mid (ps.pre \subseteq intbeliefs) \wedge (ps.post \cap unplannedintdesires) = \{\} \bullet ps\}$$
$$adoptedplan \in SuitablePlans!$$
$$intentions' = intentions \cup \{(adoptedplan.post, instantiateplan(adoptedplan))\}$$

It would be a simple matter to add more detail to this schema including choosing the plan with the highest rating for example, or a plan which has completed successfully in the community the most number of times, or making sure the plan has not failed in the users history, or that the plan has not failed in the community with users which have similar profiles as defined by the personal learning agent. In general, the plan finding system requirements, and this specification alongside it, will develop as we gain experience of how the system is used.

Community of Music Learning

Agent finding Now we move to defining a community of learners each of which has one and only one personal learning agent. First we define the set of all users.

$$[User]$$

Community

$$community : \mathbb{P}\, User$$
$$agents : User \rightarrowtail PersonalLearningAgent$$

$$community = \mathrm{dom}\, agents$$

To this we can define all kinds of social relationships. For example, peer and teacher and others as they become useful. It is up to the designer of the system to state what the constraints are on any such relationships. To provide examples (not necessarily ones we would subscribe to) of how this is none we state that if user1 is a peer of user2 then user2 is a peer of user1 and, in addition, if user2 is a teacher of user1 then user1 cannot be a teacher of user2. Another example would be the idea of a fan who would always adopt the advice of another.

SocialRelationships

$$peer, teacher : User \leftrightarrow User$$
$$fans : User \leftrightarrow User$$

$$\forall u1, u2 : User \bullet (u1, u2) \in peer \Rightarrow (u2, u1) \in peer$$
$$\forall u1, u2 : User \bullet (u1, u2) \in teacher \Rightarrow (u2, u1) \notin teacher$$

Using these schemas it then becomes possible to ask agents to start to look for users who have similar profiles as stated in the requirements detailed earlier in this document. In order to refine the search to include (for example) looking for agents who have a motivation to teach, we will need to develop the specification to define ways in which agents can broadcast that they are able to teach certain plans. This will come in later versions of this specification.

6 Concluding remarks

We are developing a system for social music learning called MusicCircle that we hope will be populated by large numbers of learners. As part of the design of our system we will be incorporating personal learning agents to provide a more personalised, social and effective learning experience. In this paper have use a standard agent-based formal specification methodology for modelling social agent systems to specify the design of these agents. The website can be found at `musiccircleproject.com` and a screen shot of the system is given above.

At the heart of this is the design of a social system of personal learning agents that can enable users to have a stronger sense of their place in the social community of music learners. We have used a long-standing agent-based specification methodology for building models of these agent systems which we are using in the principled development of our system. In addition to using formal agent-based methodology for designing agent systems and on the other hand we are testing versions of our systems with users across a range of music learning cites in the UK (from school, to pre-consevertoire to the HE sector) so that our work clearly spans agent-based theory and the practice of building systems for large numbers of users. Furthermore, we have demonstrated that Music Circle is an example crowd-based socio-technical system as described by Noriega and d'Inverno in the paper.

In relating the theory and practice of sociological agent systems within the design of socio-technical system more generally also enables us in future work to consider a range of questions about how the scientific social multi-agent approach that our community has developed for 20 years or more can be applied to the analysis and design of crowd-based socio-cognitive systems. We need to understand to what extent a MAS approach to analysing and designing systems such as MusicCircle helps? Could we, for example, start to map out the space of such systems relating technology to sociality in a useful way using the multi-agent tradition? Then could we start to provide platforms and design methodologies for building such systems in the future using a regulated MAS approach?

One hope is that we will see a greater influence from the MAS community, applying the work developed over recent years to become mainstream in the analysis, design and specification of crowd-based socio-technical systems in the future.

Acknowledgements

This research was supported by the FP7 project in the Technology-Enhanced Learning Program called Practice and Performance Analysis Inspiring Social Education (PRAISE). The authors would like to thank Harry Brenton, Andreu Grimalt-Reynes, Maria Krivenski, Jonathan James and Francois Pachet who are colleagues on the PRAISE project at Goldsmiths.

References

1. A. L. Baylor and P. P. A. L. S. R. Group. The impact of three pedagogical agent roles. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, pages 928–929, New York, NY, USA, 2003. ACM.
2. M. d'Inverno and M. Luck. *Understanding agent systems*. Springer, 2003.
3. M. d'Inverno and M. Luck. Creativity through autonomy and interaction. *Cognitive Computation*, 4(3):332–346, 2012.
4. M. d'Inverno, M. Luck, P. Noriega, J. A. Rodriguez-Aguilar, and C. Sierra. Communicating open systems. *Artificial Intelligence*, 186(Complete):38–94, 2012.
5. W. L. Johnson, J. W. Rickel, and James C. Lester. Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of ...*, pages 47–78, 2000.
6. A. Josephson. New Technology-based Models for Postsecondary Learning: Conceptual Frameworks and Research Agendas. Technical report, Computing Research Association, 2013.
7. D. Koller and A. Ng. The Online Revolution : Education at Scale. Technical report, Stanford University.
8. J. Lester, S. Converse, and S. Kahler. The persona effect: affective impact of animated pedagogical agents. In *CHI 97 Conference on Human Factors in Computing Systems*, Atlanta, 1997.
9. M. Luck and M. d'Inverno. A formal framework for agency and autonomy. *Proceedings of the first international conference on Multi-Agent Systems*, 254260, 1995.
10. M. Luck and M. d'Inverno. Structuring a Z specification to provide a formal framework for autonomous agent systems. *ZUM'95: The Z Formal Specification Notation*, pages 46–62, 1995.
11. H. Magnus, S. Annika, and S. Björn. Building a Social Conversational Pedagogical Agent-Design Challenges and Methodological Approaches. In D. Perez-Marin and I. Pascual-Nieto, editors, *Diana Perez-Marin (Editor), Ismael Pascual-Nieto (Editor)*, pages 128–155. IGI Global, 2010.
12. P. Noriega and M. d'Inverno. Crowd-based socio-cognitive systems. In *Crowd Intelligence: Foundations, Methods and Practices*. European Network for Social Intelligence, Barcelona, January 2104.
13. L. Pappano. The year of the MOOC. *The New York Times*, 2(12):2012, 2012.
14. O. Rodriguez. MOOCs and the AI-Stanford like Courses: two successful and distinct course formats for massive open online courses. *European Journal of Open, Distance, and E-Learning*, 2012.
15. E. Sklar and D. Richards. The use of agents in human learning systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 767–774, New York, NY, USA, 2006. ACM.
16. M. Soliman and C. Guetl. Intelligent pedagogical agents in immersive virtual learning environments: A review. In *MIPRO, 2010 Proceedings of the 33rd International Convention*. IEEE Computer Society Press, 2010.
17. M. Spoelstra and E. Sklar. Using simulation to model and understand group learning. In *Proc. AAMAS'07 Workshop on Agent Based Systems for Human Learning and Entertainment*, 2007.
18. J. Xiao, J. Stasko, and R. Catrambone. An Empirical Study of the Effect of Agent Competence on User Performance and Perception. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 178–185, 2004.