Pedagogical agent models for massive online education

Matthew Yee-King and Mark d'Inverno
Department of Computing, Goldsmiths College
London, UK
m.yee-king@gold.ac.uk

Abstract

The effective implementation of Massively Open Online Courses poses fascinating challenges. We address two such challenges using an agent based approach, employing formal specifications to articulate an 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 referred to as the 'year of the MOOC', the massive, open, online course [Pappano, 2012]. Indeed one of the authors of this paper was part of a team which delivered a course to an enrolled student body of around 160,000 in 2013 and 2014. 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 [Koller and Ng,]. 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 if anyone 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 fulfil 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 150.000 students still active at the end of our MOOCs; Norvig and Thrun's famous Stanford AI CS211 course in 2011 went from 160,000 enrolments to 20,000 completions [Rodriguez, 2012]. 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, as opposed to on-campus learning aspect 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, and which can help the learner to explore the network of other learners to find help or to provide help and feedback to others.

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

Pedagogical agents There is a significant literature around pedagogical agents and there are many questions one might ask when considering the design of pedagogical agents.

What is the purpose of the agent and is it pro-active, reactive, conversational or argumentative? Skiar et al. present a review of work where agents are used to supporting learning [Sklar and Richards, 2006]. The researchers define three main trends in the field: pedagogical agents, peer learning agents and demonstrating agents. According to Soliman and Guetl, Intelligent Pedagogical Agents (IPAs) can help learners by 'providing narrations ... creating adaptive dialogues with the learner to improve learning situations, provide guidance, resolve difficulties, and improve motivation' [Soliman and Guetl, 2010]. Quirino et al. implemented a case based reasoning driven IPA for training medical students. They define the following important characteristics: domain-specific knowledge, autonomy, communicability, learning, reactivity and pro-activity, social skills, customisation, and learning abilities [Quirino *et al.*, 2009]. Magus et al. describe a math tutoring game which includes a conversational agent [Magnus *et al.*, 2010]. They have explored aspects of the visual embodiment of the agent as well as its conversational capabilities. The conversation can occur in a a focused, on topic mode mediated through multiple choice questions and a free, off topic mode. Agents capable of argumentation have appeared in the education technology literature. In 2009, Tao et al. presented a pilot study where agents and learners engaged in learning through argumentation around the topic of food chains (e.g. tiger eats sheep eats grass) [Tao *et al.*, 2009]. The user interacts with the agent through keyboard, mouse and text to speech conversion (agent talks to learner) and the agent is capable of engaging in various types of dialogue. The researchers found preliminary evidence that the learners enjoyed interacting with the arguing agent.

Is the agent an animated character? 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 [Lester *et al.*, 1997]. In a subsequent survey of animated pedagogical agents, Johnson et al. provide a list of technical issues for designers of animated pedagogical agents to consider: interface to the environment, behavioural building blocks, behaviour control. believability, emotion, platform and networking Issues [Johnson *et al.*, 2000].

How competent is the agent and what is its role? 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 [Xiao *et al.*, 2004]. Allowing users to choose the competency of their agent improved objective performance. Kim and Baylor present a study investigating the value of pedagogical agents as learning companions (PALs) with deliberately varying competency levels and interaction modes. They conclude that 'PALs should be designed as highly competent for learning contexts in which instructional goals focus on knowledge and skill acquisition...in contexts where learners' self-efficacy beliefs in the task are a major concern, less competent PALs could be more effective'[Kim and Baylor, 2006]. Baylor et al. present an initial study where agents take on different roles 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 [Baylor and Group, 2003].

What are the requirements for a pedagogical agent in a large scale, social learning context? Leading towards our interest in social learning, in [Spoelstra and Sklar, 2007], an agent based approach is used to simulate interactions between learners within a group. A parameterised learner model is presented which includes features such as ability, emotion, motivation (inc. competitiveness), learning rate, understanding, 'likeliness to help' and progress. Instances of the model are run in simulation and characteristics observed in real groups of learners are observed, such as the importance of group composition, team size and team rewards.

Research questions We have discussed our wider research goals in the introduction: better pathways to and through information for learners, better feedback to learners (on their work) and increasing learner motivation. In this paper, we will address the following questions which fall within this wider remit: How might one formally specify a human learner to allow operations upon that information by an autonomous agent? What kind of operations might be useful, given the wider research goals?

2 Agent requirements

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 persons 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.

3 Formal specification

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 [D'Inverno and Luck, 2003; Luck and D'Inverno, 1995].

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 formulae 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.

$$TaskType ::= Practice \mid Listen \mid Makemusic \mid Upload \mid Share \mid Annotate \mid Question \mid Answer \mid Visualise$$

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 and 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'.

$$\mid evaluateproficiency : Proficiency \to \mathbb{N}$$

In the system the community may evaluate many different aspects, such as feedback for example.

$$\mid evaluatefeedback : Feedback \to \mathbb{N}$$

Goals, Beliefs and Plans

As with the definition of the SMART Agent Framework [D'Inverno and Luck, 2003] 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 an agent must have before undertaking the plan) and a set of post conditions (the new set of skills and knowledge the agent will have after the plan). The predicate part of the schema states that the intersection of the pre and post conditions is 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 $optional[T]$.

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

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

$$Bool ::= True \mid False$$

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 user's 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. The predicate part states that the current task must have been defined in the workflow of the plan. By also defining a flag called *finished* we can specify a plan state as follows.

PlanInstance
$$Plan$$
$$current : optional[Task]$$
$$finished : Bool$$
$$thecurrent \in (\mathrm{ran}\, workflow)$$

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.

InitialPlanInstance
$$PlanInstance$$
$$undefined\, current$$
$$finished = False$$

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.

   ProposedPlan
   $$InitialPlanInstance$$

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

   ActivePlan
   $$PlanInstance$$
   $$defined\, current$$
   $$finished = False$$

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

   FailedPlan
   $$PlanInstance$$
   $$definedcurrent$$
   $$finished = True$$

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

   CompletedPlan
   $$PlanInstance$$
   $$undefined\, current$$
   $$finished = True$$

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:

FinishTask1
$$\Delta PlanInstance$$
$$current = \{last(workflow)\}$$
$$finished = False$$
$$undefined\, current'$$
$$finished' = False$$

In the second case like this:

FinishTask2
$$\Delta PlanInstance$$
$$current \neq \{last(workflow)\}$$
$$finished = False$$
$$current' = \{workflow((workflow^{\sim}(the\, current)) + 1)\}$$
$$finished' = False$$

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

$$instantiateplan : Plan \to InitialPlanInstance$$
$$\forall p : Plan;\ ps : InitialPlanInstance$$
$$\quad \mid ps = instantiateplan(p) \bullet$$
$$\quad ps.pre = p.pre \land ps.post$$
$$\quad = p.post \land ps.workflow = p.workflow$$

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

$$recoverplan : PlanInstance \to Plan$$
$$\forall p : Plan;\ ps : PlanInstance \mid p = recoverplan(ps) \bullet$$
$$\quad ps.pre = p.pre \land$$
$$\quad ps.post = p.post \land ps.workflow = p.workflow$$

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 an interpret function which takes a goal and returns a set of proficiencies (skills and knowledge). Note that the complexity of this function may vary as in some cases goals may be expressed as a set of proficiencies 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 of proficiencies. 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 agent's 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 accomplished 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 a user is offline, 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 to store the tasks that the agent is currently involved in ($currenttasks$) which can be calculated as the union of the tasks from the current plans.
13. We define a variable to store 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 of 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 iterating the current plans and accumulating the current tasks for each plan.
8. The current plans are calculated by taking the range of the intentions.

$$
\begin{array}{|l}
\hline
[X, Y] \\
\hline
map : (X \to Y) \to (\operatorname{seq} X) \to (\operatorname{seq} Y) \\
mapset : (X \to Y) \to (\mathbb{P}\, X) \to (\mathbb{P}\, Y) \\
\hline
\forall f : X \to Y;\ x : X;\ xs, ys : \operatorname{seq} X \bullet \\
\quad map\, f\, \langle\rangle = \langle\rangle \land \\
\quad map\, f\, \langle x \rangle = \langle f\, x \rangle \land \\
\quad map\, f\, (xs \frown ys) = map\, f\, xs \frown map\, f\, ys \\
\forall f : X \to Y;\ xs : \mathbb{P}\, X \bullet \\
\quad mapset\, f\, xs = \{x : xs \bullet f\, x\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{PersonalLearningAgent} \\
\textit{desires} : \mathbb{P}\ \textit{Goal} \\
\textit{beliefs} : \mathbb{P}\ \textit{Belief} \\
\textit{interpretdes} : \textit{Goal} \pfun \mathbb{P}\ \textit{Proficiency} \\
\textit{interpretbel} : \textit{Belief} \pfun \mathbb{P}\ \textit{Proficiency} \\
\textit{intdesires} : \mathbb{P}\ \textit{Proficiency} \\
\textit{intbeliefs} : \mathbb{P}\ \textit{Proficiency} \\
\textit{availabletasks} : \mathbb{P}\ \textit{TaskType} \\
\textit{plandatabase} : \mathbb{P}\ \textit{Plan} \\
\textit{completedplans}, \textit{failedplans} : \mathbb{P}\ \textit{Plan} \\
\textit{intentions} : (\mathbb{P}\ \textit{Proficiency}) \pfun \textit{PlanInstance} \\
\textit{unplannedintdesires} : \mathbb{P}\ \textit{Proficiency} \\
\\
\textit{currenttasks} : \mathbb{P}\ \textit{Task} \\
\textit{currentplaninstances} : \mathbb{P}\ \textit{PlanInstance} \\
\hline
\textit{intdesires} = \bigcup(\textit{mapset interpretdes desires}) \\
\textit{intbeliefs} = \bigcup(\textit{mapset interpretbel beliefs}) \\
\textit{intdesires} \cap \textit{intbeliefs} = \emptyset \\
\bigcup(\mathrm{dom}\ \textit{intentions}) \subseteq \textit{intdesires} \\
\forall\, \textit{ps}1, \textit{ps}2 : \mathbb{P}\ \textit{Proficiency} \mid \\
\qquad (\textit{ps}1 \neq \textit{ps}2) \land (\{\textit{ps}1, \textit{ps}2\} \subseteq \\
\qquad (\mathrm{dom}\ \textit{intentions})) \bullet \\
\qquad\qquad \textit{ps}1 \cap \textit{ps}2 = \{\} \\
\textit{unplannedintdesires} = \bigcup(\mathrm{dom}\ \textit{intentions}) \setminus \textit{intdesires} \\
\\
\textit{currenttasks} = \{t : \textit{Task};\ \textit{ps} : \textit{PlanInstance} \mid \\
\textit{ps} \in (\mathrm{ran}\ \textit{intentions}) \bullet \textit{the ps.current}\} \\
\textit{currentplaninstances} = \mathrm{ran}\ \textit{intentions} \\
\hline
\end{array}
$$

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 come from the agent (i.e. the agent's database of plans), other agents in the community, from the user or from other users. In general, candidate plans with be a *synthesis* of the users and the agents of users working together.

For now we will suppose that suitable plans have all pre-conditions 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 satisfies this constraint and from these one plan *adoptedplan* is selected. The state of the agent is then updated such that its current plans include a mapping from the pre-conditions of the plan (which are necessarily interpreted desires for which no plan exists).

$$
\begin{array}{|l}
\hline
\textit{FindandAdoptPlan} \\
\textit{PossiblePlans}? : \mathbb{P}\ \textit{Plan} \\
\textit{SuitablePlans}! : \mathbb{P}\ \textit{Plan} \\
\textit{adoptedplan} : \textit{Plan} \\
\Delta \textit{PersonalLearningAgent} \\
\hline
\textit{SuitablePlans}! = \{\textit{ps} : \textit{PossiblePlans}? \mid \\
\qquad (\textit{ps.pre} \subseteq \textit{intbeliefs}) \land \\
\qquad (\textit{ps.post} \cap \textit{unplannedintdesires}) = \{\} \bullet \textit{ps}\} \\
\textit{adoptedplan} \in \textit{SuitablePlans}! \\
\textit{intentions}' = \textit{intentions} \cup \\
\qquad \{(\textit{adoptedplan.post}, \\
\qquad \textit{instantiateplan}(\textit{adoptedplan}))\} \\
\hline
\end{array}
$$

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.

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 $\textit{CompletedPlan}$.
2. The post conditions are added to the interpreted beliefs (these may in turn be reverse ineterprered 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 interprered 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.

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

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 evaluating the plan considers each of the post conditions (or another member of the community does) to work out whether they are now proficiencies (interpreted 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 snap shot of the agent's state for future comparison by the community.

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.

$$[User]$$

$$\begin{array}{|l} \textit{Community} \\ \hline community : \mathbb{P}\, User \\ agents : User \pfun PersonalLearningAgent \\ \hline community = \mathrm{dom}\, agents \end{array}$$

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 done 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.

$$\begin{array}{|l} \textit{SocialRelationships} \\ \hline peer, teacher : User \rel User \\ fans : User \rel User \\ \hline \forall\, u1, u2 : User \bullet (u1, u2) \in \\ peer \Rightarrow (u2, u1) \in peer \\ \forall\, u1, u2 : User \bullet (u1, u2) \in \\ teacher \Rightarrow (u2, u1) \notin teacher \end{array}$$

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.

4 Concluding remarks

No one could have predicted the rise in technologies for facilitating different kinds of online social behaviour. Despite a sometimes limited scope of interact possibilities (such as liking, or rating content), huge numbers of us choose to socialise in this way. More and more technology platforms are being released, aiming to encourage us to spend our social time on them. Not only that, but we are now seeing a whole range of such systems that encourage us to spend our learning time on them, making use of a range of techniques to allow the learning experience to be less isolated and more social, particularly around the idea of peer feedback and assessment.

Given this explosion of systems for social experiences including social learning experiences, it is perhaps a little surprising that the multi-agent systems (MAS) community, with all its rich work on agency, coordination, norms and regulated social behaviour has not been more involved in taking up the challenge of trying to understand the science of such systems and in turn bringing that understanding into well-defined methodologies for designing compelling systems.

In this paper, we have shown that it is possible to use a standard agent-based formal specification methodology to model various aspects of a social learning environment. Building on that we have shown how such an architecture can be used to solve problems in these environments, such as selecting learning plans and selecting other users of interest. In parallel to this formal modelling work, we are building real social learning environments and trialling them at scale in our own institution and beyond, as part of a research programme investigating social learning. Now we have systems with users and data, we are investigating how our agent concepts can be operationalised to solve real problems within our systems. Our work is significant because we are bridging the theory/ practice divide.

Relating the theory and practice of sociological agent systems to the design of socio-technical systems more generally also enables us in future work to consider a range of questions about how the scientific social multi-agent approach that the MAS has developed for 25 years or more can be applied to the analysis and design of systems such as ours. Questions that quickly present themselves are: could we start to map out the space of such systems relating technology to sociality in a useful way? Could we start to provide platforms and design methodologies for building such systems in the future using a regulated MAS approach? Indeed these are some of the questions we are investigating with partners on our research project.

This paper is our first foray into these woods in describing an agent-based approach to the design of a community of human and learning agents working in the common interests of learning how to play musical instruments together. We hope that we will increasingly see the huge body of work that has been developed in our community over the last 25 years or so become mainstream in the analysis, design and specification of future instances of such systems.

Acknowledgments

The work reported in this paper is part of the PRAISE (Practice and Performance Analysis Inspiring Social Education) project which is funded under the EU FP7 Technology Enhanced Learning programme, grant agreement number 318770.

References

[Baylor and Group, 2003] Amy L Baylor and PALS (Pedagogical Agent Learning Systems) Research Group. The impact of three pedagogical agent roles. In *Proceedings of the second international joint conference on Autonomous*

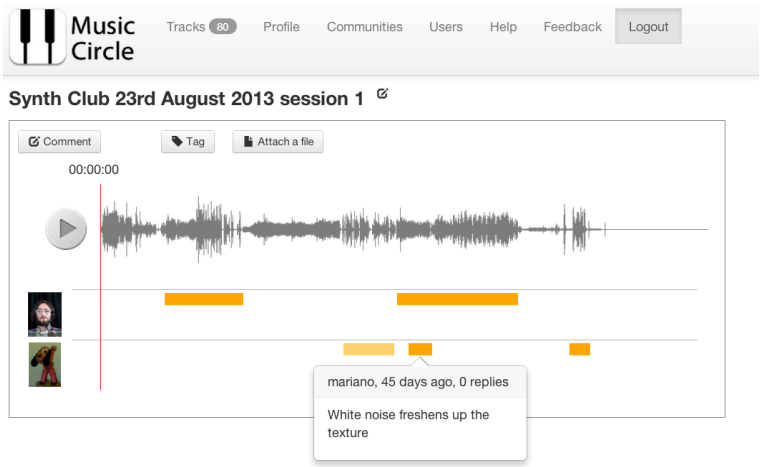

Figure 1: The music discussion user interface

agents and multiagent systems, AAMAS '03, pages 928–929, New York, NY, USA, 2003. ACM.

[D'Inverno and Luck, 2003] Mark D'Inverno and Michael Luck. *Understanding agent systems*. Springer, 2003.

[Johnson *et al.*, 2000] W. Lewis Johnson, Jeff W. Rickel, and James C. Lester. Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of . . .*, pages 47–78, 2000.

[Kim and Baylor, 2006] Yanghee Kim and AL Baylor. Pedagogical agents as learning companions: The role of agent competency and type of interaction. *Educational Technology Research and Development*, 54(3):223–243, 2006.

[Koller and Ng,] Daphne Koller and Andrew Ng. The Online Revolution : Education at Scale. Technical report, Stanford University.

[Lester *et al.*, 1997] JC Lester, SA Converse, and SE Kahler. The persona effect: affective impact of animated pedagogical agents. In *CHI 97 Conference on Human Factors in Computing Systems*, Atlanta, 1997.

[Luck and D'Inverno, 1995] Michael Luck and Mark D'Inverno. A formal framework for agency and autonomy. *Proceedings of the first international conference on Multi-Agent Systems*, 254260, 1995.

[Magnus *et al.*, 2010] H Magnus, S Annika, and S Björn. Building a Social Conversational Pedagogical Agent-Design Challenges and Methodological Approaches. In Diana Perez-Marin and Ismael Pascual-Nieto, editors, *Diana Perez-Marin (Editor), Ismael Pascual-Nieto (Editor)*, pages 128–155. IGI Global, 2010.

[Pappano, 2012] Laura Pappano. The year of the MOOC. *The New York Times*, 2(12):2012, 2012.

[Quirino *et al.*, 2009] E Quirino, F Paraguaçu, and B Jacinto. SSDCVA: Support System to the Diagnostic of Cerebral Vascular Accident For Physiotherapy Students. In *22nd IEEE International Symposium on Computer-Based Medical Systems, CBMS*, pages 2–5, 2009.

[Rodriguez, 2012] 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.

[Sklar and Richards, 2006] Elizabeth Sklar and Debbie 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.

[Soliman and Guetl, 2010] M Soliman and Christian 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.

[Spoelstra and Sklar, 2007] Maartje Spoelstra and Elizabeth Sklar. Using simulation to model and understand group learning. In *Proc. AAMAS'07 Workshop on Agent Based Systems for Human Learning and Entertainment*, 2007.

[Tao *et al.*, 2009] Xuehong Tao, YL Theng, and Nicola Yelland. Learning through argumentation with cognitive virtual companions. In C Fulford and George Siemens, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications3179-3194*, pages 3179–3194, 2009.

[Xiao *et al.*, 2004] Jun Xiao, John Stasko, and Richard 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.