A multi-agents system to compute human learning indicators activities based on model-driven engineering approach

Tarek Djouad*
ICOSI Laboratory,
Department of Mathematics and Computer Science,
Faculty of Sciences and Technologies,
Abbes Lghourou Khendhela University, Algeria
Email: tarek.djouad@gmail.com
*Corresponding author

Mohammed Benmohammed
LIRE Laboratory,
Department of Software Technologies and Information Systems,
Faculty of New Technologies of Information and Communication,
Abdelhamid Mehri Constantine 2 University, Algeria
Email: ben_moh123@yahoo.com

Abstract: This paper presents a multi-agent architecture and its implementation for facilitating the evaluation of learners’ activities in real learning situations mediated by a technology enhanced learning systems. Our proposal is based on model-driven engineering using modelled activity traces to compute human learning indicators. We claim this approach facilitates the computation, the management and the reuse of learning indicators independently of any learning platform.

Keywords: activity traces; human learning indicators; model-driven engineering; multi-agents systems; technology enhanced learning systems.


Biographical notes: Tarek Djouad is a Lecturer at the Department of Mathematics and Computer Science, faculty of sciences and technologies, Abbes Lghourou Khendhela University, Algeria. His research interests include advanced human learning technologies, model-driven engineering, software engineering and human learning indicators computation.

Mohamed Benmohammed is a Professor at the Department of Software and Information Systems, faculty of new technologies of information and Communication, Abdelhamid Mehri Constantine 2 University, Algeria. He received the PhD from university of Sidi Bel-Abbes, Algeria, in 1997. His research interests include microprocessor, computer architecture, embedded system and multi-agents systems.
1 Introduction

Technology enhanced learning (TEL) systems offer a set of resources and tools allowing teachers and learners assessment and customisation of learning process. Most of these tools use activity traces in order to compute learning indicators. Computing such indicators is a key to observe and to understand learners’ behaviour, performance and progress.

Before explaining our claims, let us first recall what are multi-agents system, activity traces and learning indicators.

We recall that a Multi-Agents System (MAS) (Durfee et al., 1989) is “a loosely coupled network of problem-solving entities (agents) that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity (agent).” These systems have evolved over time and we will explain in this paper how we use this technology to facilitate indicator computation.

We recall also that an activity trace (Georgeon et al., 2012) is a “set of multiple streams of quantitative or symbolic data that record (at least partially) an activity performed by a subject”. These traces are generally saved in databases, Log files, etc.

And finally, we recall that a learning indicator (Dimitracopoulou et al., 2005) is a mathematical variable with a set of characteristics such as value, computation rule, role, validity field, etc.

Computing indicators includes generally three steps (Dimitracopoulou et al., 2005): selecting useful data, analysing and preparing selected data, and then indicator computation. We will detail these steps (ad hoc methods) in Section 2.2.

Unfortunately, indicators computation details are encapsulated into learning-platform’s source code. Computing and reusing any indicator is a hard task and requires computer designer experience for each of these tasks. It is still very difficult to manage them freely within most distance learning platforms.

Our claim is that it should be possible for anybody needing to design and/or to use learning indicators to do so without having to modify the computer code of the learning platform, and make the computation easier and faster than ad hoc methods. To guarantee this property, we propose to use model-driven engineering and multi-agents system to provide easy-to-use methods for creating, exploiting and visualising indicators, which are adapted to a learning activity but not linked to a specific learning platform.

We present in the next section some works related to traces, indicators computation and MAS in TEL systems. We explain in Section 3 the architecture we propose and we give in Section 4 our implementation with a case study. Finally, we conclude with our future works.
2 Activity traces, indicator computation and MAS in TEL systems

2.1 Activity traces in TEL systems

Most of learning platforms produce traces and several works propose to use and to exploit these traces in TEL systems. Betbeder et al. (2006), Guéraud et al. (2007), Dyke et al. (2010) define experiments to collect multimodal data in order to analyse traces.

Mazza and Dimitrova (2004) France et al. (2007), Cram et al. (2012) visualise traces in real time and provide a feedback to teachers and learners on their own activities.

Ferraris et al. (2005), Voisin and Vidal (2007) propose model-driven engineering approaches to understand learning scenarios using traces.

Settouti et al. (2009), Zarka et al. (2013), Cordier et al. (2013) consider a trace as a structured model with a set of instances where each instance (or obsel) is located in time. They propose also to use a Trace Base System (TBS) to manage these so-called modelled traces (or m-traces).

We will use m-traces in our contribution. We will give more details about TBS in Section 3.1.

2.2 Indicator activities in TEL systems

Traces are used to compute indicators. We recall that in ad hoc methods (Dimitracopoulou et al., 2005), an indicator computation life cycle is as follows:

- Select data: identify and filter important data used to compute the indicator from raw data.
- Prepare data: modify, transform and prepare data prior to indicator computation.
- Compute the indicator: coding equations and visualising indicator values.

These steps need a computer designer familiar with all details concerning the selected data structure and indicator equation parameters, and are coded directly in the learning platform.

An indicator may indicate an individual contributions’ quality (Example: send a message in a chat), a collaboration (Example: division of labour, density, cohesion of a group, etc.), a quality of what was produced by learners (Example: discussion thread’s depth in a forum), etc.

Indicators are also used to build a feedback to different teachers/learners (Figure 1). According to the categories proposed in Soller et al. (2005), this feedback can be a direct visualisation of the indicator’s value (mirroring), or the value can be compared to a desired state (monitoring) or it can give more elaborated process providing guiding information to learners (guiding).
Several works propose indicators computation according to Dimitracopoulou et al. (2005). Santos et al. (2003) compute the involvement’s degree of students in an online learning course. Martinez et al. (2003) compute density in social networks and uses histograms to interpret it. Tedesco (2003) computes agreement/disagreement between learners. Reffay et al. (2011) compute the cohesion in a social network using forums. May et al. (2011) provide tool to compute and visualise the indicator “Read a message in a forum” using traces from server and client machines. As illustrated in Figure 2, the sphere size shows the time spent by a learner to read a message. The various spheres are located on the time axis, and their colour depends on the action’s type made by the learner (Example: blue for posted messages, green for read messages). This visualisation gives more accurate information on quality of readings messages and the time that learner spent on it. Tools such as TASCI in Laperrousaz (2007), Gismo (Mazza and Botturi, 2007), MooDog (Zhang and Almeroth, 2010) compute indicators from Moodle Log files.

**Figure 2**  Example of indicator in Travis (May et al., 2011) (see online version for colours)
Other research works propose engineering to compute indicators. For example, EM-AGIIR, an open multi-agents architecture defined in Diagne (2009) provides a framework to reuse indicators with (Figure 3):

- A database agent stores traces used to compute the indicator,
- A human/machine interface agent displays the indicator values,
- A query agent identifies important data used to compute the indicator values from log files,
- An indicator agent computes the indicator values. This agent asks the query agent to import important data from log files, and then asks the database agent to store these data. Finally, it computes values using a function $f$.

**Figure 3** EM-AGIIR architecture in Diagne (2009) (see online version for colours)

Iksal et al. (2010) propose to reuse and improve educational scenarios using a formal grammar to compute indicators *Usage Tracking Language UTL*. The representation indicator’s model is based on three parts: *defining part* which describes what data are used to compute indicators, *getting part* which describes how to get these data from raw data sources and *using part* which describes how to compute and use indicators.

Gendron (2010) propose a model to compute and reuse indicators with:

- Identity card to describe indicator’s name, variables and description
- Pattern to describe indicator’s type, structure and computation rule
- Interface view to describe indicator’s visualisation mode.

Gendron (2010) propose to use four modules: definition, design, contextualisation and visualisation: The *definition* module describes the identity card. The *design* module describes the indicator pattern. The *contextualisation* module defines the indicator’s
values. The visualisation module contains all information describing indicator’s representations when it is instantiated.

Ji et al. (2014) propose a dynamic dashboard for collection, analysis and visualisation of activity and reporting traces (DDART) to support meta-cognitive activities. DDART is based on the system proposed in Djouad et al. (2010) and is used to improve learners’ performances by monitoring their activities in Moodle. It is based on three steps: collecting data, integration data and computing indicator. Collecting data: use raw data from the Moodle database. Users can also introduce their non-instrumental activities such as Skype or Facebook activities. Integration data: build a primary m-trace combining Moodle raw data and non-instrumental data. Compute indicator: using a filter operator applied to the DDART primary m-trace.

Unfortunately, in all these works presented in Santos et al. (2003), Martinez et al. (2003), Tedesco (2003), Laperrousaz (2007), Mazza and Botturi (2007), Zhang and Almeroth (2010), Reffay et al. (2011), May et al. (2011) indicators still computed directly in learning platforms which makes the indicators’ management a very difficult task.

Djouad (2009), Iksal et al. (2010), Gendron (2010), Ji et al. (2014) are the approaches most similar to our own. While these works propose to use models to manage indicators, they still have to code indicators in TEL systems and there is no possibility to easily to create and reuse indicators outside the context of the original TEL System. Table 1 compares our approach with existing indicators engineering methods.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Comparison of existing indicator engineering methods and our method based on MAS (see online version for colour)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our approach</strong></td>
<td><strong>Existing indicator engineering methods</strong></td>
</tr>
<tr>
<td>Collecting</td>
<td>Create primary m-trace. Smart collect according to indicator’s variables used to compute it.</td>
</tr>
<tr>
<td>Preparing</td>
<td>Create transformation sequence. Smart transformation according to existing m-traces (reuse existing m-traces to accelerate transformation)</td>
</tr>
</tbody>
</table>
1. **Table 1** Comparison of existing indicator engineering methods and our method based on MAS (see online version for colour) (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compute equation and visualise result</td>
<td>Use composer agent to compute values</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexible computation according to indicator’s equation defined by users.</td>
<td>Use ATL language to compute values</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reuse existing indicator to compute a new one from the same learning platform</td>
<td>Use equation to compute indicator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reuse existing indicator to compute a new one from another learning platform</td>
<td>Use equation to compute indicator</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

2.3 **MAS in TEL systems**

Multi-agents systems (MAS) are used in several research fields. An agent is defined in Jennings and Wooldridge (1998) as “Computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.” An intelligent agent must be capable of flexible autonomous such as responsive, proactive, social, etc.

Several works are published relating MAS to TEL systems, specially, to evaluate learning activity. SIGFAD (Mbala et al., 2003) provides to teachers some indicators to evaluate learners in their groups. SIGFAD proposes five agents to compute indicators (Figure 4): learner agent, tutor agent, coordinator agent, interaction supervisor agent and database agent.

**Figure 4** SIGFAD agents (Mbala et al., 2003)
Fougères and Ospina (2005) propose MAS to provide pedagogical relations between teachers and learners (Figure 5). The database agent and the GUI agent are used by the multi-assistant system (five assistant agents) to communicate with the pedagogical environment and manage pedagogical activities.

**Figure 5** Multi-assistance system for pedagogical activities (Fougères and Ospina, 2005) (see online version for colour)

France et al. (2007) visualise traces using three agents: Collecting agent used to collect traces from learners’ machines. Each learner’s machine has its related collecting agent. Visualisation agent used to visualise traces into teacher’s machine. Communication agent used to synchronise messages between collecting agent and visualisation agent.

SYSAT (Oumaira et al., 2011) supervise learners’ activities in Moodle. SYSAT use four agents: Collecting agent used to collect useful data and put it in SYSAT database. Request agents to execute requests sent by other agents. Analysis agents used to compute indicators. GUI agent used to manage communication between users and machines.

### 3 Our multi-agents system to compute indicators

Our proposal is based on the trace-based system (TBS) and the model-driven engineering (MDE). We recall in this section these two concepts before explaining our contribution.

#### 3.1 Trace-based system

The trace-based system or TBS is proposed and implemented by the TWEAK research group to manage m-traces (Settouti et al., 2009; Zarka et al., 2013; Cordier et al., 2013).
An m-trace in TBS is a structured object: the trace model and the corresponding trace instance in the form of a sequence of observed elements or obsels. Each instance’s obsel part of an m-trace is temporally situated by a time stamp and satisfies the trace model part of the m-trace.

Trace-based system proposes explicit transformation operators to be applied to a set of m-traces (transformation sources) in order to obtain other transformed m-traces (transformation targets). All m-trace obsels are represented by structured information resulting from a transformation operation using source m-trace obsels. Each m-trace is the result of some transformation of a lower level m-trace, except for the lower level, directly built from an observation process constructing the primary m-trace.

Trace-based system proposes three steps for using m-traces (Figure 6):

1. Users, as teachers/tutors or learners, use learning platforms. These platforms provide raw data as a source of observation. TBS connects to learning platforms, collects raw data and uses these data to build a primary m-trace (model and instances). This primary m-trace is then saved in an m-trace base,

2. TBS uses this primary m-trace and transforms it into other transformed m-traces according to the semantics of these transformations. The transformed m-trace is saved in the m-trace base. In turn, these transformed m-traces can be transformed again into other transformed m-traces. Starting from one primary m-trace, a transformation graph is progressively built and saved for providing explicit explanations of any transformation, i.e. providing the semantics of any m-trace in the m-trace base.

3. Moreover, the m-trace base can be used by any assistant to manage indicators, allow indicator computation, provide smart visualisation, etc.

Figure 6 Trace-based system used by our research team to manage m-traces (see online version for colour)
3.2 Model-driven architecture

As we claim to adopt a model-driven architecture (MDE), we need here to recall the notion of model.

In Laforcade et al. (2007), a model is a description or a prescription of all or part of a system using a defined language. In the case of a description, the model is correct if its characteristics and behaviour evolve in the same way over time as the real system. However, in the case of prescription, the system is considered valid if model characteristics do not contradict the obtained system. Meta-models are also used to describe models: they define languages to express models.

Model-driven engineering is based on model-driven architecture. It considers two worlds: the real world or the system, and models world. The system is represented by its model, and a model conforms to its meta-model.

Model-driven engineering in TEL systems is inspired from software engineering and focuses on model changes rather than system coding. This considerably reduces the efforts of designers, teachers, researchers, etc., where all efforts are focused on model definition and transformation rather than on system coding. We will explain this point in next section.

3.3 Our proposal

3.3.1 Use MDE to create and reuse indicator

We consider that an indicator is a structure allowing it to be computed and to be explained by its values when computed. We define an indicator as:

\[
\text{Indicator} = \{\text{Name}, \text{Variables}, \text{Time interval}, \text{Computation rule}, \text{Category}, \text{Transformation sequence}\}.
\]

Where:

- **Name**: the indicator name.
- **Variables**: list of variables used by the computation rule to compute indicator’s values.
- **Time interval**: we compute indicator according to a time interval.
- **Computation rule**: equation used to compute indicator’s values.
- **Category**: we use this variable to classify indicator.
- **Transformation sequence**: Indicators’ variables are empty. We propose to associate for each indicator its transformation sequence used to transform m-traces and give for each indicator’s variable its value. We use here MDE to transform these m-traces.

We propose to compute a new indicator by re-applying the corresponding m-trace transformation sequence to the current primary m-trace and by computing the new current value of the indicator. We associate a transformation sequence with each indicator (Figure 7). In this case, each indicator depends on the result of the transformation sequence. The primary m-trace contains all good candidate data used to compute indicator.
A multi-agents system to compute human learning

Figure 7 Use MDE to compute indicator values using transformed m-traces (see online version for colour)

Starting from this primary m-trace, new indicators are computed through transformation sequences and according to an explicit equation describing indicator computation with some variables, where the values are derived from the directly associated m-traces. A transformation sequence uses operators to transform one m-trace into another one. A transformation sequence is one important part of an indicator when considered as a computer object.

Even if some indicator has already been built, and a user wants to modify the indicator, it is easy to modify the related transformation sequences to update all variables.

3.3.2 Use MAS to manage indicators

We propose also to use a set of agents around this indicator’s definition. These agents allow creating, updating, deleting, reusing and sharing indicator. These agents are important in order to:

- *Minimise the indicator’s computation complexity*. If the indicator computation is difficult (Example: the case of a high-level indicator defined by sub-indicators), then our system reuses existing indicators to compute new one. We use information such as transformation sequence, computation rule and category to perform this computation.

- *The need to compute a same indicator in different contexts is very high*. Several users attempt to compute the same indicator with closest time intervals. The system creates for each new computation a list of agents that will minimise the indicator computation time.

We propose these following agents to build our system (Figure 8):

- *The collecting agent* uses raw data issue from learning platforms (Example: database, Log files, etc.) to select useful data used to compute indicator. The collecting agent is guided by variables which are listed in the computation rule to optimise the collecting step. Once completed, it stores the collecting results in an m-traces base as a primary m-trace and keeps the collecting history to reuse it later.
This agent aims to prepare things for having a better performance in a future collecting step.

- **The transformation agent** transforms a primary m-trace created by the collecting agent. Each transformed trace becomes a variable named according to the name of the transformed trace and takes its value from the number of instances present in the transformed m-trace. This agent uses transformation operators such as filtering, merging, matching, etc. to transform m-traces.

- **The computation agent** executes the computation rule using variables created by the transformation agent and saves it in the indicator base.

- **The interface agent** displays indicators using graphical views (Pie charts, Bar graphs, Histograms, etc.). It also retrieve information provided by teachers to define indicators.

- **The coordinator agent** coordinates and synchronises the different agents to provide a consistent functioning of the system.

Figure 8  Class diagram of our agents using AUML notations
We propose these steps to compute a new indicator using our system (Figures 9 and 10):

- The teacher defines a new indicator using the interface agent.
- The interface agent stores this indicator into a base and informs the coordinator agent to start computation.
- The coordinator agent requests the collecting agent to start collecting data used to compute indicator.
- The coordinator agent requests the transformation agent to prepare and to start m-traces’ transformation from existing m-trace base, or as soon as the collecting agent finishes collecting data.
- The coordinator agent requests also the computation agent to start computation using existing m-traces, or as soon as the transformation agent finishes m-traces’ transformation.
- The computation agent executes the computation rule, updates the indicators base and asks the interface agent via the coordinator agent to display results to the teacher/learners.

**Figure 9** The architecture of our system relating agents and databases we use to maintain the system functionalities (see online version for colour)
Figure 10  Sequence diagram of our agents using AUML notations

4 A case study in Moodle learning platform

We implemented our architecture with the different agents we have defined. Our implemented Indicator Computation Multi-Agents System IC-MAS\textsuperscript{6} facilitates indicators computation regardless of any learning platform. IC-MAS connects to learning platforms and manages indicators according to users requests.

As a case study, we propose to compute the division of labour indicator between two learners according to a specific time interval in Moodle.

We recall that the division of labour indicator is defined and implemented in Jermann (2004). It identifies the division of labour adopted by two learners working on a set of shared resources. Specifically designed for researchers, this indicator identifies the role taken by each participant in the collaborative learning process. There are three types of division of labour: Task-based division where each learner acts on separate resources; Role-based division where one learner edits all resources; Concurrent division where
both learners act more or less equally on all resources. The characterisation of these three types of division of labour can be computed from the sum of differences (SD) and the sum of absolute differences (SAD):

$$SD = \frac{\sum (S1Ai - S2Ai)}{S1A + S2A}$$

$$SAD = \frac{\sum |S1Ai - S2Ai|}{S1A + S2A}$$

where S1Ai (respectively S2Ai) is the number of actions made by learner S1 (respectively S2) on the resource Ai, and S1A (respectively S2A) is the number of actions made by learner S1 (respectively S2) on all the resources. The SAD indicates the symmetry of actions. The value 0 means that both learners have the same number of actions, whereas the value 1 means that all actions were made by the same learner.

We used real traces from the Moodle learning platform of Khenechela University. We propose to compute the division of labour between two learners User3 and User15 and related to a Moodle chat activity. Figure 11 shows the screenshot of this indicator where teacher choose to display it as a histogram. The teacher can also choose different display forms, and he can easily manage transformed m-traces, computation rules and indicators’ base.

Figure 11 Example of indicator computation with our system: division of labor indicator between two learners: User3 and User15 related to a chat activity (see online version for colour)
5 Conclusion

We presented in this paper an architecture and an implementation of a multi-agents system to compute human learning indicators activities. The system we propose computes indicators in real time using agents, managing the different computation steps from the collecting data to the indicator computation. The different agents we use can make a smart data collecting, a smart m-trace transformation and a smart indicators computation according to the indicator’s definition. Agents can then minimise computation time if computation demand increases. Agents also minimise the degree of computation complexity using existing transformation sequences to compute new indicators. Then, we compute real indicators with real learners and in real learning situations in Moodle. Comparing to ad hoc methods, our approach allows indicators computation without coding in learning platforms, making it easier to manage. This is possible using MDE where the m-traces transformation sequence allows creating variables used by computation rule without coding them in machine. Table 2 compares our approach to ad hoc methods and explains what news in our method.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Comparison between our approach and ad hoc methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our method</strong></td>
<td><strong>Ad hoc methods</strong></td>
</tr>
<tr>
<td>Create new indicator from learning platform. Example: Compute proportion indicator from Moodle raw data.</td>
<td>Coding and filtering raw data.</td>
</tr>
<tr>
<td><strong>Collecting data</strong></td>
<td></td>
</tr>
<tr>
<td>Create primary m-trace.</td>
<td>Coding selected data transformation.</td>
</tr>
<tr>
<td>Smart collect according to indicator’s variables used to compute it.</td>
<td>Needs a computer scientist</td>
</tr>
<tr>
<td><strong>Preparing data</strong></td>
<td></td>
</tr>
<tr>
<td>Create transformation sequence.</td>
<td>Coding indicator equation.</td>
</tr>
<tr>
<td>Smart transformation according to the trace-base stat (reuse existing traces to accelerate transformation).</td>
<td>Computing</td>
</tr>
<tr>
<td>Teacher or researcher can transform data using transformation sequences, without needing a computer scientist for coding.</td>
<td>Needs a computer scientist.</td>
</tr>
<tr>
<td><strong>Indicator computation</strong></td>
<td></td>
</tr>
<tr>
<td>Compute equation and visualise result.</td>
<td></td>
</tr>
<tr>
<td>Flexible computation according to indicator’s equation defined by users.</td>
<td></td>
</tr>
<tr>
<td>Teacher or researcher can compute indicator without needing a computer scientist for coding.</td>
<td></td>
</tr>
<tr>
<td>The indicator model is available to all courses for the same learning platform.</td>
<td></td>
</tr>
</tbody>
</table>
Table 2  Comparison between our approach and ad hoc methods (continued)

<table>
<thead>
<tr>
<th>Our method</th>
<th>Ad hoc methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuse existing indicator to compute a new one</td>
<td>The same steps used to create new indicator.</td>
</tr>
<tr>
<td>Collecting data</td>
<td>Always coding.</td>
</tr>
<tr>
<td>Primary m-trace exists: no need for collecting data.</td>
<td></td>
</tr>
<tr>
<td>No need for coding.</td>
<td></td>
</tr>
<tr>
<td><strong>Preparing data</strong></td>
<td></td>
</tr>
<tr>
<td>Reuse existing transformation sequence to compute indicator.</td>
<td></td>
</tr>
<tr>
<td>Does not need a computer scientist</td>
<td></td>
</tr>
<tr>
<td>Teacher or researcher can reuse transformation sequences, without needing a computer scientist for coding.</td>
<td></td>
</tr>
<tr>
<td><strong>Indicator computation</strong></td>
<td></td>
</tr>
<tr>
<td>Reuse existing equation to create indicator values.</td>
<td></td>
</tr>
<tr>
<td>Does not need a computer scientist</td>
<td></td>
</tr>
<tr>
<td>Teacher or researcher can reuse equation and compute indicator values without needing a computer scientist for coding.</td>
<td></td>
</tr>
</tbody>
</table>

As future works, we hope extend indicators computation to other learning platforms such as Claroline and WebCT and test our system in other real learning situations.

We hope also improve our system. We still have some difficulties like latency and synchronisation in communication between agents. For this reason, we hope use specialised MAS platforms like JADE to have better performances.

We hope also propose methods to share indicators. We assume that indicators computation is an important knowledge and needs to be shared (computation rule, traces transformation sequence, etc.) between TEL systems researchers. Define new mechanism and methods to share indicators libraries between research areas (Example: Computer science, Sociology, Economy, etc.) is one of the goals we claim in our future works.

References


A multi-agents system to compute human learning


Notes
1 Throughout this paper, we mean by trace: human learning activity trace.
2 Throughout this paper, we mean by indicator: human learning indicator activity.
3 www.moodle.org.
5 liris.cnrs.fr/equipes/?id=75.
6 https://github.com/Activity-Traces/IC-MAS.
7 http://tele-ens.univ-khenchela.dz/moodle/.
9 http://www.mdc.edu/ctd/webct/.