Tech Know Learn (2018) 23:41–64
https://doi.org/10.1007/s10758-017-9337-9

ORIGINAL RESEARCH

Observing and Understanding an On-Line Learning Activity: A Model-Based Approach for Activity Indicator Engineering

Tarek Djouad[1] · **Alain Mille**[2]

Published online: 26 September 2017

Abstract Although learning indicators are now properly studied and published, it is still very difficult to manage them freely within most distance learning platforms. As all activity indicators need to collect and analyze properly traces of the learning activity, we propose to use these traces as a starting point for a platform independent Trace Based-Indicator Management System (TB-IMS). This approach allows learning indicators to be created and reused in such a way that there is no need to modify the computer code of the learning platform. This paper presents the underlying theory and how this theory is implemented in a first TB-IMS. This TB-IMS is illustrated through an actual learning situation based on a Moodle platform. This approach is compared with similar attempts to manage learning indicators properly and is available for use with any other learning platform, provided the TB-IMS can access the learning platform traces.

Keywords Technology enhanced learning system · Human learning indicator activity · Trace based system · Modeled trace

1 Introduction

Observing and understanding learners' behavior, performance and progress are difficult tasks. When the learning process is supported by a Technology Enhanced Learning (TEL) system, it is mandatory to know what to observe and how to understand and analyze it. In order to observe learners' behavior, learners' interactions must be collected from the TEL

✉ Tarek Djouad
tarek.djouad@gmail.com

Alain Mille
Alain.mille@liris.cnrs.fr

[1] Icosi laboratory, Khenchela University, El Hamma, Algeria

[2] Liris Laboratory, UMR5205, Claude Bernard Lyon1 University, 69622 Lyon, France

system. Classical TEL systems use log files for this purpose, while more innovative systems try to design the observation process during the learning activity. The result of the observation process is a trace of learner interactions[1] through the learning platform. In order to build useful knowledge from this observed interaction trace, it must be interpreted according to some semantics, based on explicit models. The most common way to represent this knowledge is to build indicators[2] (Soller et al. 2005).

This paper presents a new approach to create and reuse indicators using modeled traces. We use a model driven engineering approach to compute indicators using a trace based system (Settouti et al. 2009) able to manage interaction modeled traces and their interpretations. Our idea is to define an indicator as a *computer object* directly associated with corresponding modeled traces. The indicator is created, computed and reused through *operators* on this new kind of *computer composite object*: the *indicator formula* and the tree of modeled traces used to evaluate this formula. Consequently, we need a trace based-management system to build and manage such indicators.

One important 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. To guarantee this property, we consider indicators in a separate framework able to articulate fluently with any learning platform and to provide easy-to-use methods for creating, exploiting, visualizing and sharing indicators, which are adapted to a learning activity but not linked to a specific learning platform. Teachers/tutors should be able to create their own indicators without looking for programming details. Indicator descriptions can be managed by the researcher, the activity's designer, the trainer, the tutor or even by the learner him/herself during a learning activity.

It is very important to understand that our research does not propose any new interesting indicator but addresses the following research question: how to formalize the indicator computation process from the design step to the exploitation step in order to provide efficient tools for managing what could be called "indicator knowledge". We claim that clear access to the genesis of indicator computation helps its design, computation, usage and understanding by learners, teachers, designers, researchers and any other people involved in the learning process.

This formalization is neither a guarantee of the validity of an indicator nor a method for validating an indicator. However, using such formalization offers a possibility for the various validation protocols of a new indicator. This formalization has been successfully used in Ji et al. (2014) to validate new indicators. Moreover, when using an "on the shelf" indicator, the designer has to explain how he/she implements it in the learning environment. It is no easy task to demonstrate that implementation is correct and respects the indicator's semantics. We claim that if an "on the shelf" indicator could be formalized with our proposal, then it might be easier to check the value of such or such an implementation. Moreover, while the ability to describe an indicator by a set of formal transformations of observations and formal descriptions of computation guarantees discussion of indicator validity, it is by no means a demonstration of its validity in general or in a particular context.

[1] Even if it can be considered that a "*non interaction at a particular place and time*" can be observed. In this case, the system constructs an event as an observation of a non interaction. This is a tricky point that is discussed in (Champalle et al. 2013).

[2] Throughout the paper, we consider that the word "indicator" means: "a human learning indicator activity".

Consequently, what is new and radically changes the situation is the formalization of the entire indicator knowledge engineering process from the design step to the exploitation step.

Rather than just demonstrating the importance of formalization of indicator knowledge engineering, we decided to illustrate concretely our approach. We developed a generic assistant tool to cover the whole indicator life cycle, and used this tool on a real, concrete university learning situation based on the Moodle[3] environment to experiment this approach. This provided an opportunity to conduct a full-scale test on a real university education course. We have to emphasize that we are not using this learning situation to conduct research on new indicators, or to check some research hypothesis requiring observation of learning activities through indicator computations. Learning indicators can be found right *on the shelf* or specifically defined by a teacher, a tutor, a designer or a learner for their specific goals. There is no discussion of the intrinsic value of any indicator, but we evaluate how it is possible to manage learning activities efficiently with such a TB-IMS. The presented learning situation is used to make a *proof of concept* of our proposal to create indicators using a trace model driven engineering process. In this case, this work is a concrete contribution for the TEL community, which could consider this indicator management system to facilitate observation and management of on-line courses. Moodle[4] is a framework for designing, managing and supporting distance learning processes: it is what is usually called a Learning Management System (LMS). It offers a large number of tools for sharing documents (wiki), for interacting (chat), for discussing (forum), for providing pedagogical resources, for quiz purposes, and for many other learning activities. Moodle also offers hundreds of additional easy-to-install free extension tools, making it a very powerful LMS. These tools all enable easier interaction between learners and teachers. Moodle is used by millions of users in the world[5] and represents what could be found in a number of other LMS.

2 Indicators and Indicator Engineering in Technology Enhanced Learning Systems

As our goal is to propose a generic way to manage indicators, this section recalls how indicators are usually defined in the literature and why it is useful to formalize their design and computation to allow their management during a computer mediated learning activity. Indicators are widely studied in the technology enhanced human learning field. We present in this section some works concerning indicator computations, before focusing on existing works related to indicator engineering.

According to Dimitracopoulou et al. (2005) an indicator is a mathematical *variable*. The variable can be qualitative or quantitative and can be represented by symbolic, logical or alphanumerical values. The *value* can be used directly, calibrated or interpreted by researchers. In the event of a composite indicator, a list of variables can be represented through a graphical visualization gathering several indicators in the same frame and providing a kind of dashboard for a specific learning activity. An indicator has a list of attributes such as: *a name*, *a purpose* (cognitive, social or affective), *a validity field* (to

[3] Moodle stands for: Modular Object-Oriented Dynamic Learning Environment.

[4] https://moodle.org/?lang=en.

[5] https://Moodle.net/stats/.

define this field, the content of the activity, the learning participants' profile and the intended users should be considered), *dependencies* (time, content, etc.), and an associated *learning environment*.

According to this definition, several research works are related to indicator computation. For example, Santos et al. (2003) offer a tool to compute the degree of involvement of each learner in his/her course using interaction traces. This tool identifies criteria such as participation, non- collaboration, communication, etc., and builds a corresponding indicator. Martinez et al. (2003) compute social network density and use histograms to interpret it. Tedesco (2003) measures agreement and disagreement between learners. Reffay et al. (2011) compute cohesion and centrality in social networks using forums. May et al. (2011) provide a tool to compute and visualize messages read in a forum using interaction traces on both the client and the server side. Tools like TASCI (Laperrousaz 2007), Gismo (Mazza and Botturi 2007), MooDog (Zhang and Almeroth 2010) compute indicators in Moodle using its log files.

Merceron and Yacef (2004) propose a data mining approach to compute indicators from raw databases. Reading Tutor (Mostow et al. 2005) allows indicator computation using a specific query language applied to modeled traces. Butoianu et al. (2012) propose indicator models and tracking frameworks based on repositories able to share indicator values. None of these works offers an engineering approach to compute, share or reuse indicators.

Other research works tend to propose engineering and generic frameworks to design and implement indicators. For example, *EM-AGIIR*, a multi-agent architecture defined in Diagne (2009) provides a framework to compute and reuse indicators. This open architecture is structured in several agents:

- 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*.

Iksal et al. (2010) propose to reuse and improve educational scenarios using a formal grammar to compute indicators. 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) proposes to use the indicator model with:

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

Gendron (2010) proposes to use four modules: definition, design, contextualization and visualization modules. The *definition* module describes the identity card. The *design* module describes the indicator pattern. The *contextualization* module defines the indicator's values. The *visualization* module contains all details about possible indicator representations.

In Santos et al. (2003), Martinez et al. (2003), Tedesco (2003), May et al. (2011), Reffay et al. (2011), Laperrousaz (2007), Mazza and Botturi (2007), Zhang and Almeroth (2010), Butoianu et al. (2012), Merceron and Yacef (2004) and Mostow et al. (2005) indicators are

directly programmed and coded as data structures in TEL systems or in related tools. None of these works offers an engineering approach to create, use or share indicators. Diagne (2009), Iksal et al. (2010) and Gendron (2010) are the approaches most similar to our own. While these three works propose to use models to manage indicators, they still have to code indicators in TEL systems and there is no possibility to easily reuse and share indicators outside the context of the original TEL System. At the end of this paper we shall compare these works to ours.

3 Building Indicators from Modeled Traces

We propose a knowledge oriented approach to manage indicators. We use a so-called trace based system (TBS) (Settouti et al. 2009) and a model driven engineering approach (MDE) (Laforcade et al. 2007) to manage the indicator life cycle. Before presenting our approach, we explain in the next section what TBS is, while Sect. 3.2 describes the MDE principle we adopt. Section 3.3 presents our indicator specification as a computer object and, finally, Sect. 3.4 describes our contribution.

3.1 Trace Based System

The Trace Based System or TBS is proposed and implemented[6] by the TWEAK[7] research group to manage interaction modeled traces (Settouti et al. 2009; Champin et al. 2013; Cordier et al. 2013; Zarka et al. 2013; Cordier et al. 2014). A modeled trace or *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.

TBS 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*. The TBS architecture includes (Fig. 1):

- An m-traces base (models and instances),
- A collector system that builds m-traces from raw data or tracing sources,
- A transformation system that transforms an m-trace into another m-trace.

TBS proposes three steps for using m-traces (Fig. 1):

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-traces 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-traces base. In turn, these *transformed* m-traces can be transformed again into

[6] https://kernel-for-trace-based-systems.readthedocs.org/en/latest/.

[7] https://liris.cnrs.fr/axes?id=71.

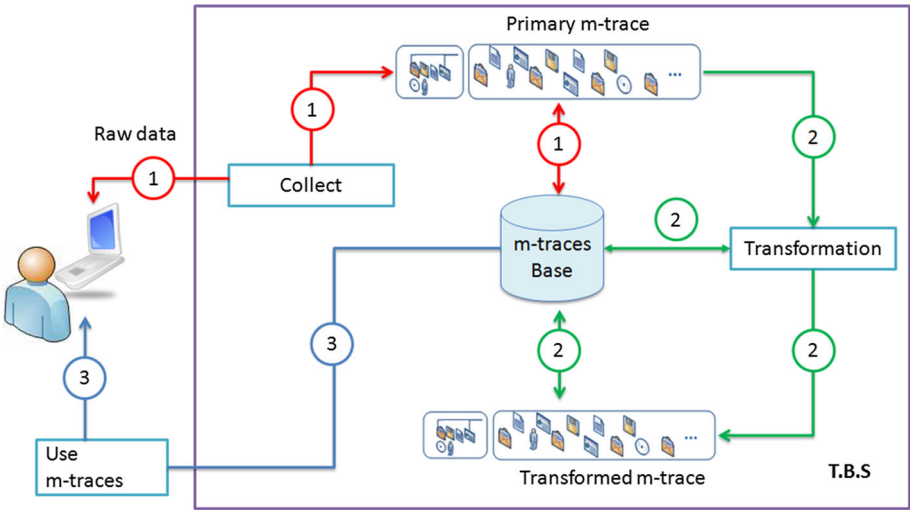

Fig. 1 Trace based system used by our research team to manage modeled traces

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-traces base.

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

3.2 Model Driven Architecture

As we claim to adopt a *model driven* architecture, we need here to recall the notion of model. A model in Seidwitz (2003) is defined as: *A set of statements about some system under study*.

Another definition in Bézivin and Gerbé (2001) defines a model as: *a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system*.

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 behavior 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 model worlds. The system *is represented by* its model, and a model *is conform* 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.

For this reason, we consider that our approach is directly related to model driven engineering. The indicator description method we propose requires m-traces and a

transformation sequence to lead from the primary m-trace to the indicator m-traces, and, finally, to allow computation of the actual indicator values. The TBS provides facilities to design new models and to produce m-traces from existing ones using the trace transformation process.

This is therefore the paper's main contribution. This approach is quite different from existing methods and provides original ways for describing, sharing and reusing indicators.

3.3 Indicator, Modeled Trace and Transformation Sequence

While the transformation sequence from the primary m-trace (what we can observe directly) to some transformed m-traces allows a learning indicator to be evaluated, we have to define what this indicator is, how it is connected to its corresponding (transformed) m-traces, and how it can be computed interactively.

3.3.1 Indicator Model and Instance Definition

We consider that an indicator is a structure allowing it to be computed and to be explained by its values when computed. To clarify this idea, we shall now provide a simple example of an indicator: *number of connections of a specific learner to a learning platform*. This indicator gives an idea of the learner's engagement in the learning activity, and has to be updated periodically (day, week, month, etc.). For example, *John* connects five times the first day, three times the second day, and so on. The indicator's structure stays the same, but the indicator's values need to be computed over time. We formalize an indicator as:

Definition 1 An indicator is a pair: {*indicator model, indicator instances*}.

Definition 2 An indicator model consists of the following attributes:

{*Name, Variables, Equation, Role, Category, Time interval, Shared*}.

where:

- Name: indicator name,
- Variables: variables used to compute indicator values. They are used to describe an indicator equation in order to compute its values,
- Equation: literal expression of the formula to compute the indicator,
- Role: what the indicator is used for (how to interpret the indicator values for some actions),
- Category: to index indicators in categories,
- Shared: to share or not the indicator with other people,
- Time interval: used to limit indicator computation in a specific time interval. Start and end time limits are provided.

Definition 3 An indicator is a computer object with a set of specific methods:{*Create indicator, Share indicator, Delete indicator, Compute indicator*}. where:

- Create indicator: instantiates indicator's attributes with possible default values,
- Delete indicator: removes the indicator from the indicator database,
- Share indicator: shares the indicator to be used by other users, systems, etc.,
- Compute indicator: computes the indicator according to the equation formula. The equation can define intermediate variables to compute indicator values.

Definition 4 Computing an indicator means computing its instances. Indicator instances are temporally situated by a time stamp and satisfy their indicator model.

Definition 5 The final value of an indicator instance is computed using its equation formula. An equation is computed using the m-trace transformation sequence: each indicator instance is related to a transformation sequence allowing its computation. The transformation sequence conformed to its transformation model.

Definition 6 Each variable used by the equation[8] is related to a transformed m-trace. Variable value is the number of obsels of its related m-trace.

3.3.2 Indicator Instances and m-Trace Transformation

We propose to compute a new indicator instance 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 for this instance. We associate a transformation sequence with each indicator instance. In this case, each indicator instance complies with its indicator model, and depends on the result of the transformation sequence which complies with its transformation model.

The transformation sequence *description*, starting from the primary m-trace through to the corresponding indicator m-trace, itself constitutes the *transformation model* of this indicator model.

Figure 2 illustrates the definitions presented in the above section with the relations between indicator model and transformation model, transformation sequence and transformation model, and between indicator instance and its transformation sequence.

3.4 Computing Indicator Instances

As stated above, indicator instances are consistent with their indicator model. The primary m-trace contains all good candidate data used to compute indicator instances. 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. An indicator is computed in three steps: *collecting* data in the primary m-trace, *transforming* this primary m-trace through a transformation sequence, and, finally, *computing* the indicator by valuating the indicator formula variables on the basis of the obsels of the transformed m-traces derived from the transformation sequence.

3.4.1 Collecting Data

Collecting data consists in selecting/filtering data from the learning platform's raw data. Just as for any analysis process, this is a preparation step and has to be carried out with the user. A *collect all* strategy can be adopted because it will be possible to select/filter items with some transformation within the m-traces base. The collecting step builds a primary m-trace. Information in the primary m-trace consists of what has been considered as useful and available in the learning platform for observing the learning process.

[8] This is not the case for the intermediate variables which can be used in the formula.

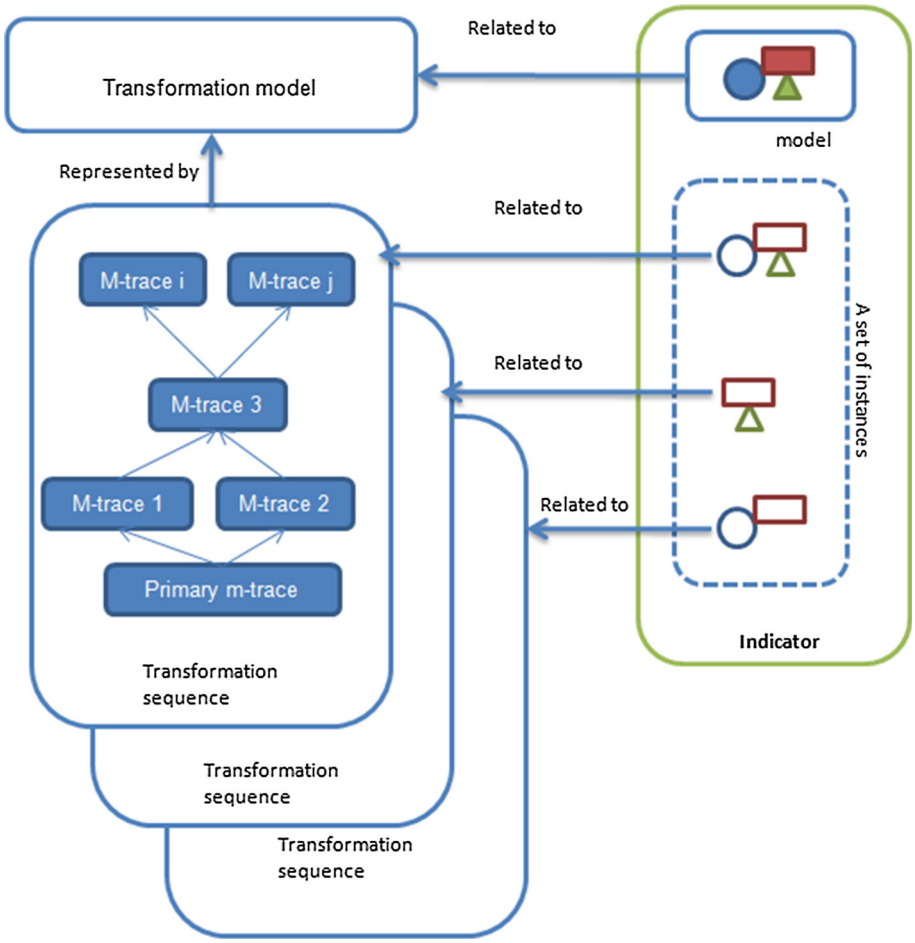

Fig. 2 Relation between indicator and transformation sequence

At this step, the primary m-trace contains all important information. We consider that all obsels of this primary m-trace are good candidates for providing information on the learning activity and indicator computation.

3.4.2 Transformation Sequence

Starting from the primary m-trace, we propose to use transformation sequences to create new m-traces. A transformation sequence uses operators to transform one m-trace into another one (examples are given in Sect. 4 to illustrate these operators). A transformation sequence is one important part of an indicator when considered as a computer object.

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

3.4.3 Indicator Engineering Driven by Trace Models

In order to demonstrate concretely how this can work, we developed an actual method to carry out the entire life cycle of an indicator.

We propose four steps to describe and compute an indicator instance using a TBS. **Each indicator instance** is computed using these four steps:

Step 1 To compute a new indicator (compute instances), we propose to associate it with a set of *empty*[9] m-traces. The ***m-trace*** instances will be built in the next steps. These m-traces will be used later for evaluating the indicator formula to compute indicator instances (Fig. 3).To simplify, we will explain in the next steps how *one* instance is computed. All other instances are computed in the same way.
Step 2 In this step, we associate a transformation sequence with the empty indicator m-traces. The transformation sequence related to these m-traces describes how to move from the primary m-trace model to the indicator m-traces. Figure 4 illustrates how to create a transformation sequence from the primary m-trace model to the corresponding indicator m-traces. Conversely, we can see this transformation sequence as a way of reformulating what we need to get to compute an indicator in a specific situation starting from its abstract description (the empty m-traces) through to the basic level (the available primary m-trace).
Step 3 This step collects data from the learning environment and builds the primary m-trace instance. The result of this step is a complete primary m-trace (model and collected obsels). Figure 5 shows an example of such a collection for the primary m-trace.
Step 4 Finally, we execute the transformation sequence (defined in step 2), from the primary m-trace instantiated in step 3 to the indicator m-traces defined in step 1. The result of this step 4 is a complete indicator m-trace with the indicator instance (Fig. 6).

4 Computing Indicators in a Real Learning Situation: Case Study

To illustrate the benefits offered by our approach based on model driven engineering, we propose to illustrate the indicator computation steps through three different scenarios:

- Creating a new indicator in the context of one learning platform,
- Reusing an existing indicator to compute a new indicator in the same learning platform,
- Reusing an existing indicator (from the shelves) in the context of another learning platform.

We will use the *proportion* indicator defined by Dimitracopoulou (2004) to illustrate scenario one, and the *division of labor* indicator defined by Jermann (2004) to illustrate scenarios two and three. We first present the context of the learning situation we choose to observe.

4.1 Describing the Learning Situation

We recall this paper does not study the *quality* of new indicators. We chose indicators as they are *on the shelf* and we illustrate how to use our approach and tools in different

[9] *Empty* means that the m-trace model is present but that there are no obsels in the instance part.

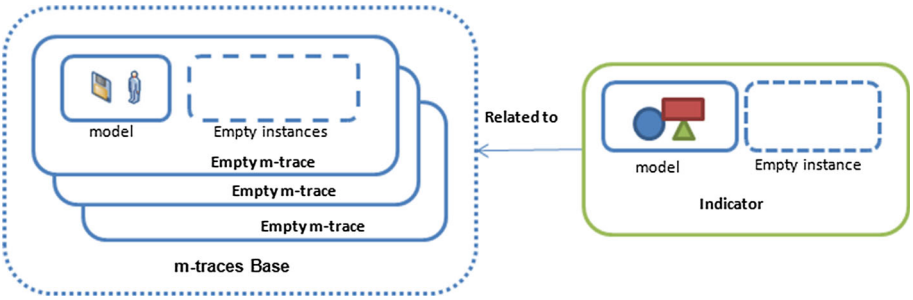

Fig. 3 Associating an indicator model with its set of m-traces (m-traces base)

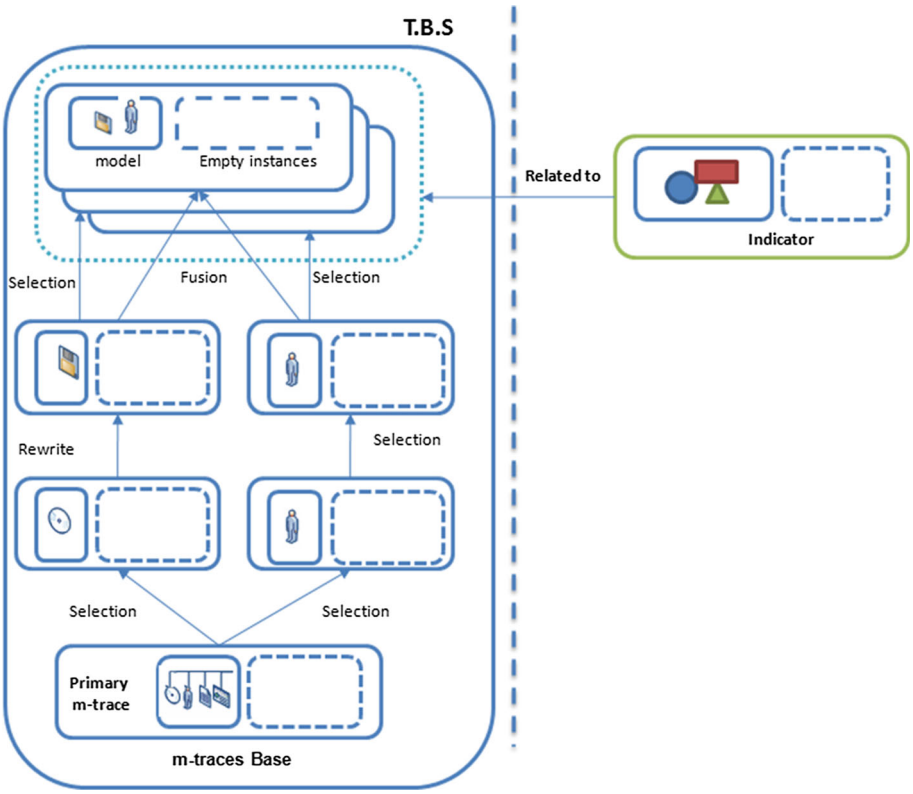

Fig. 4 Using a transformation sequence to relate the primary m-trace to the indicator *m-traces (parts of the indicator object)*

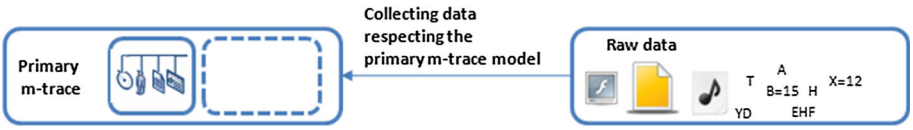

Fig. 5 Preparing data: building the primary m-trace obsels

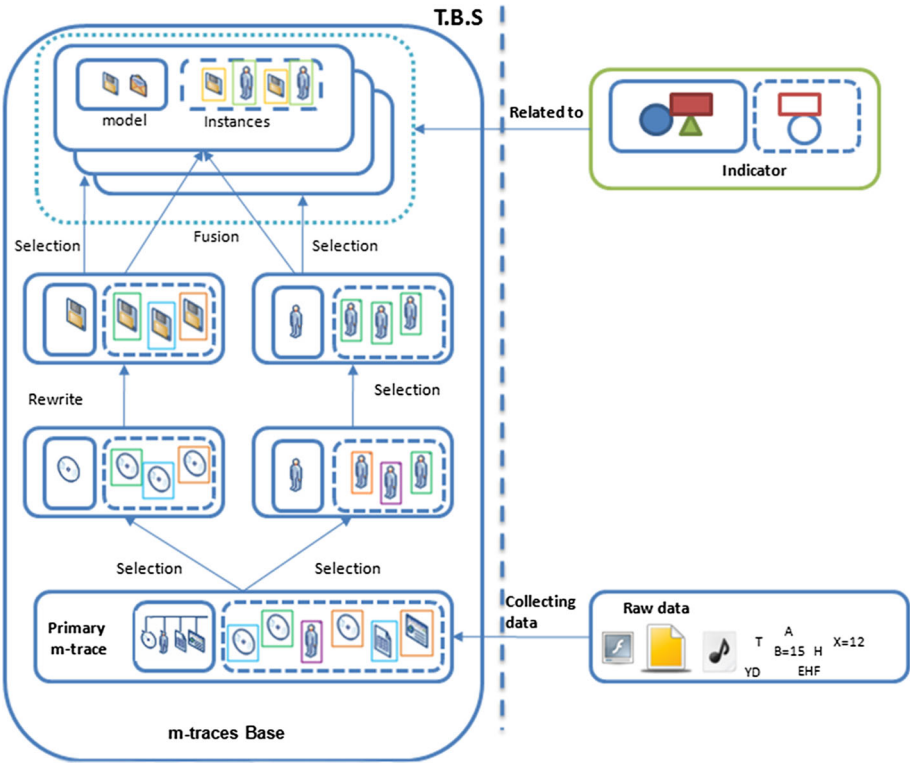

Fig. 6 Running the transformation sequence. In this case, we have grouped the four steps: from the collection step to indicator computation

situations. The situation is an ecological one, without any research-specific constraints for the teacher or learners. We focus on the proof of concept of the approach, i.e. on the efficiency of the approach and its ability to be used within a popular platform (Moodle). Offering modeled indicators, m-traces, and transformation sequences in TBS allows the creation, computation and reuse of indicators with a much lighter life cycle than the classical one.

In this illustration, the pedagogical activity is related to a master course named *Techniques and Applications of Artificial Intelligence*, at the computer science department of Claude Bernard Lyon 1 university.

38 students participated in the course over a period of 4 months. The teacher asked students of a real classroom to organize themselves into 9 groups of 3–5 students. Then he asked them to use the Moodle course. This course is divided into two main activities:

- Documentation with exercises about: the basics of case-based reasoning, the principles of case-based reasoning, the life cycle of case-based reasoning, etc.
- A collaborative project: we asked students to jointly design and prototype a case-based reasoning framework.

To organize their learning activities, students used Moodle tools such as forums, wikis, chats, web pages, links to on-line media, etc. These tools provide a large raw database that we can use to compute indicators.

4.2 Scenario 1: Computing a New Indicator

4.2.1 Our Goal

We propose to use our approach to compute a new indicator. We recall that to compute a new indicator in ad-hoc classical methods, it needs to be coded directly in the learning platform. This is a long and difficult task, requiring computer designer experience.

The goal of this scenario is to show how to describe and compute a new indicator through our method and with our tools. The scenario will be successful if it demonstrates the process is simpler than an ad hoc approach and above all that it proposes an explicit way to describe and compute a new indicator in a formal knowledge representation, for reuse and adaptation.

According to our method, we propose to create a new indicator and associate it with its transformation sequence (example in Fig. 7).

To illustrate how it works, we propose to compute *the proportion between a two learning activities* indicator, defined in Dimitracopoulou (2004). The computation rule of the proportion (1) between two activities or actions A and B for a specific user U and related to a time interval T is:

$$proportion(A, B, U, T) = \frac{UAT - UBT}{UAT + UBT} \quad (1)$$

where UAT is the number of actions A performed by user U according to a time interval T and UBT is the number of actions B performed by the same user U according to the same time interval T.

In our approach, each variable used in the equation is related to an m-trace. So, the value of each variable UAT and UBT can be obtained by selecting each obsel of type A and B for the user U according to a time interval T from the primary m-trace. We can create a transformation sequence to build two m-traces UAT and UBT related to User U according to a time interval T. With these m-traces, we just have to count their obsels to get the values of the variables UAT and UBT.

The transformation sequence from the primary m-trace to the indicator m-traces (*UAT and UBT*) is described in Fig. 7.

The process is the following:

- Select all instances related to user U from the primary m-trace. The m-trace result is m-trace U which contains only user *U's* obsels,
- Select and filter all m-trace U obsels related to a time interval T. The result is m-trace UT,
- Divide m-trace UT into two m-traces: m-trace UAT and m-trace UBT using two selections where we keep only m-trace obsels related to A and B obsel types, respectively,
- Create two new variables where their names are the m-trace names and their values are the number of obsels in the m-traces. The m-traces *UAT and UBT* become two variables UAT and UBT used in the proportion computation rule.

This generic transformation sequence is not the only solution available: we can always find other equivalent generic transformation sequences to compute this indicator.

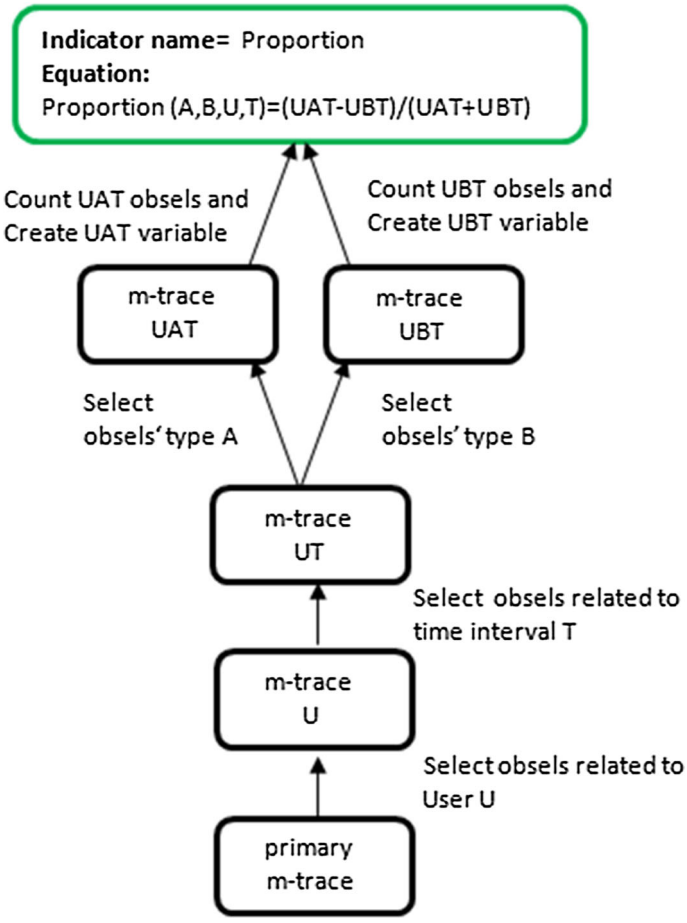

Fig. 7 Generic transformation sequence to compute *the proportion between two activities A and B* related to a specific user *U* according to a time interval *T*

4.2.2 Case Study

Indicator computation and m-trace transformation are managed by the prototype we developed: Trace Based-Indicator Management System TB-IMS.[10] Figure 8 shows the TB-IMS collector module, the transformation module, the equation editor and the visualization result. In this example we compute the proportion between *Chat messages* and *Private messages* related to *User15* according to a time interval *T*.

Computation is possible without coding in machine. The system saves in its databases the indicator, its transformation, intermediate m-traces and the primary m-trace, which allows the indicator to be reused to compute new indicators. We will explain this possibility in the next section.

[10] www.github.com/tb-ims/tb-ims/.

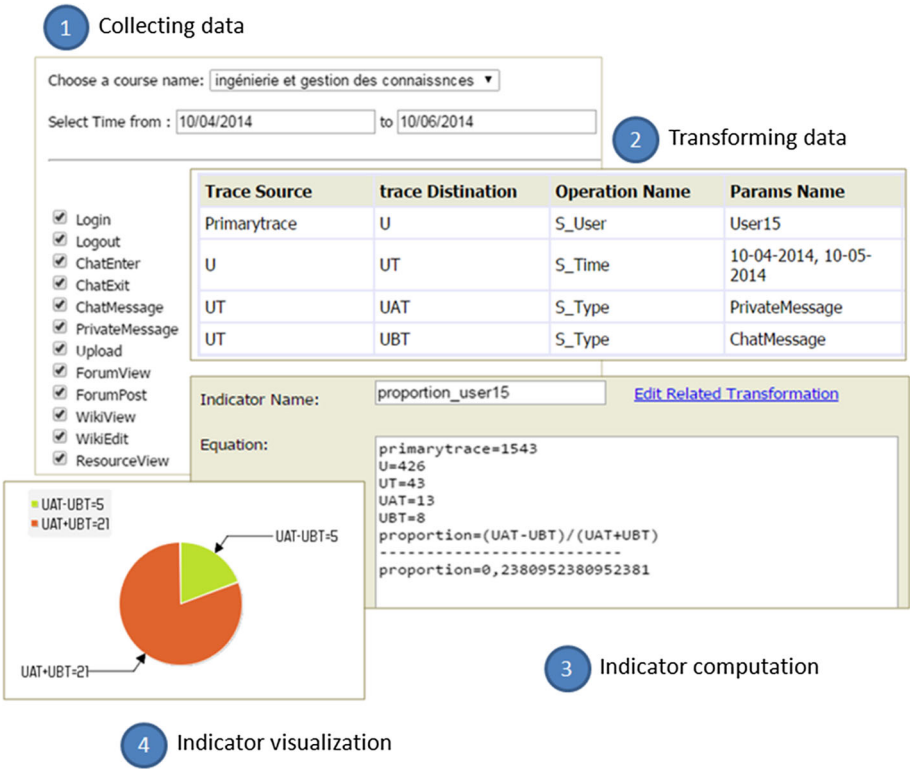

Fig. 8 Computing the proportion indicator value of the last example in TB-IMS

4.2.3 Synthesis

This scenario demonstrates that it is possible and easy to describe an indicator and how to compute it with an immediate ability to check the result. Explicit descriptions of the transformation, of the traces, of the final indicator formula do not need a big effort and this effort is rewarded by the automatic computation of the indicator.

4.3 Scenario 2: Computing a New Indicator from an Existing Indicator

4.3.1 Our Goal

The goal of scenario 2 is to show how to use our method in order to compute a new indicator *division of labor*, but using raw data issued from *the same* learning platform.

The system we built allows a new indicator to be created and all its related data to be saved (indicator instances, its primary m-trace, its transformation sequence, its intermediate m-traces generated by the transformation). Even it is not the same one, we propose to exploit our approach for reusing the *proportion* indicator computed in scenario 1, in order to compute a new indicator such as *division of labor.* Actually, it should be useful to re-use the *preparation transformations* of the existing indicator for the new one.

Several research works are related to collaborative learning measuring. Dillenbourg (1999) defines collaborative learning as *a situation in which two or more people learn or attempt to learn something together* where it has four meanings: situation, interaction, mechanism, and effects. Barros and Verdejo (2000) compute collaborative indicators using three elements: coordination, cooperation and argumentation. Each element is computed using other elements. For example argumentation is computed using initiative, interactivity, tree depth and work. Von Davier and Halpin (2013) measure collaboration using a framework for educational assessment of cognitive skills with collaborative problem-solving tasks. This framework proposes to: Measure cognitive skill; Identify the types of activities that count as evidence of the cognitive skill;- Use strategies for collaborative assessments; Non cognitive skills and variables that can affect any and all levels of the assessment elements.

In this example, our goal is not only to identify a better collaboration measurement, but is to offer a generic mechanism to compute it. The *division of labor* indicator is defined and implemented in Jermann (2004). It identifies the *division of labor* adopted by two users working on a set of shared resources. This indicator identifies the role assumed by each user in a collaborative learning process. The *division of labor* can be computed from the *sum of differences SD* (2):

$$SD(U1, U2, A, T) = \frac{\sum_i (U1AiT - U2AiT)}{U1AT + U2AT} \tag{2}$$

where *Ai* is an action type (example: chat, forum, etc.), *U1AiT* is a number of actions *Ai* performed by user *U1* according to a time interval *T* (respectively *U2AiT*), and *U1A* (respectively *U2A*) is the total number of actions performed by the User *U1* (respectively *U2*) according to a time interval *T*.

The sum of differences *SD* indicates the symmetry of actions between users. The value 0 means that both users have the same number of actions, while the value 1 means that all actions were performed by user 1.

In TB-IMS, *U1AiT* is the number of m-trace obsels related to a user *U1* and action type *Ai* according to a time interval *T*. The transformation sequence from the primary m-trace to the indicator m-traces is explained in Fig. 9. In this case, we *reuse* the transformation sequence defined in Fig. 7 to build this *new* transformation sequence. The blue parts in Fig. 9 are what we use from the last transformation sequence.

The process is:

- Select from the primary m-trace all instances related to a specific time interval *T*. The result is m-trace *All_T*,
- Extract, using two selection m-traces, instances related to Users *U1* and *U2*, respectively. The results are m-trace*U1T* and m-trace*U2T*,
- Extract, still using two selections, instances related to A and B instance types. We apply this to the two users *U1T* and *U2T*. The results are: m-trace*U1AT*, m-trace*U1BT*, m-trace *U2AT,* and m-trace*U2BT*,
- Count m-trace obsels using the count function and compute the indicator equation.

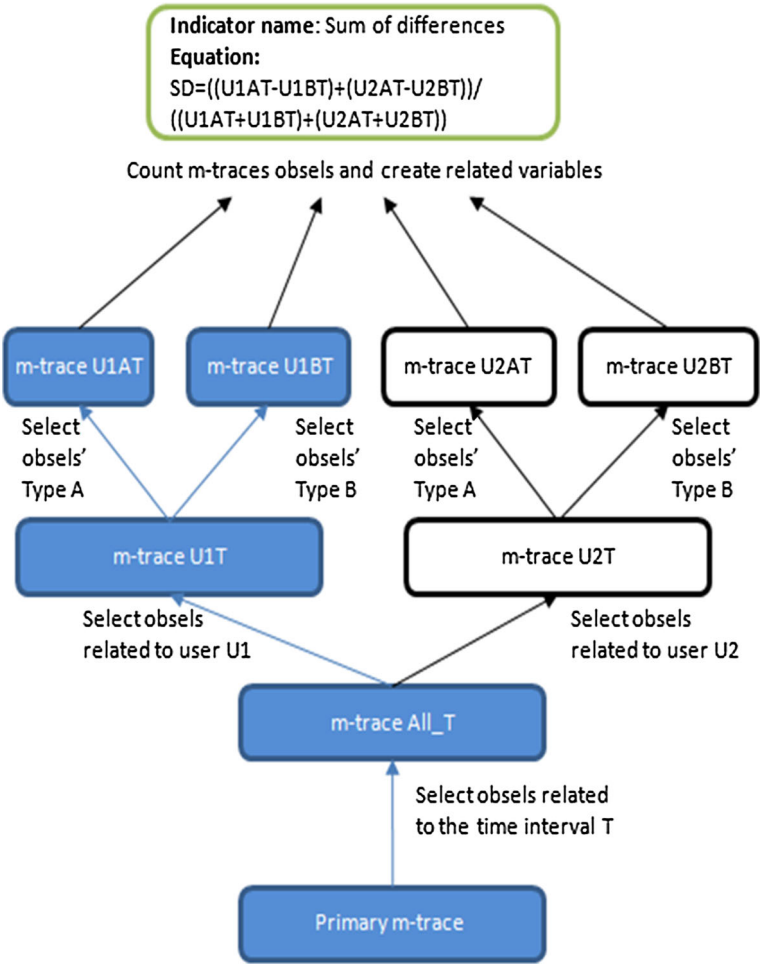

Fig. 9 Reusing the transformation in Fig. 7 to build a new indicator

4.3.2 Case Study

In TB-IMS, we compute for example the *division of labor* between *Private Message* and *Chat message* for two users *User1* and *User2* (Fig. 10). We can view the transformation module, the equation editor and the visualization result (Histogram).

4.3.3 Synthesis

Although it is not the same indicator, we demonstrated that it was useful and easy to do to re-use the knowledge of an existing indicator to design another one. In this case, we demonstrate that an important part of the process was re-usable, specially to transform the primary traces for getting well prepared traces for designing the new indicator. As a consequence, to design a new indicator for the same traces base is then easier to manage and the designer can check the result on the traces base.

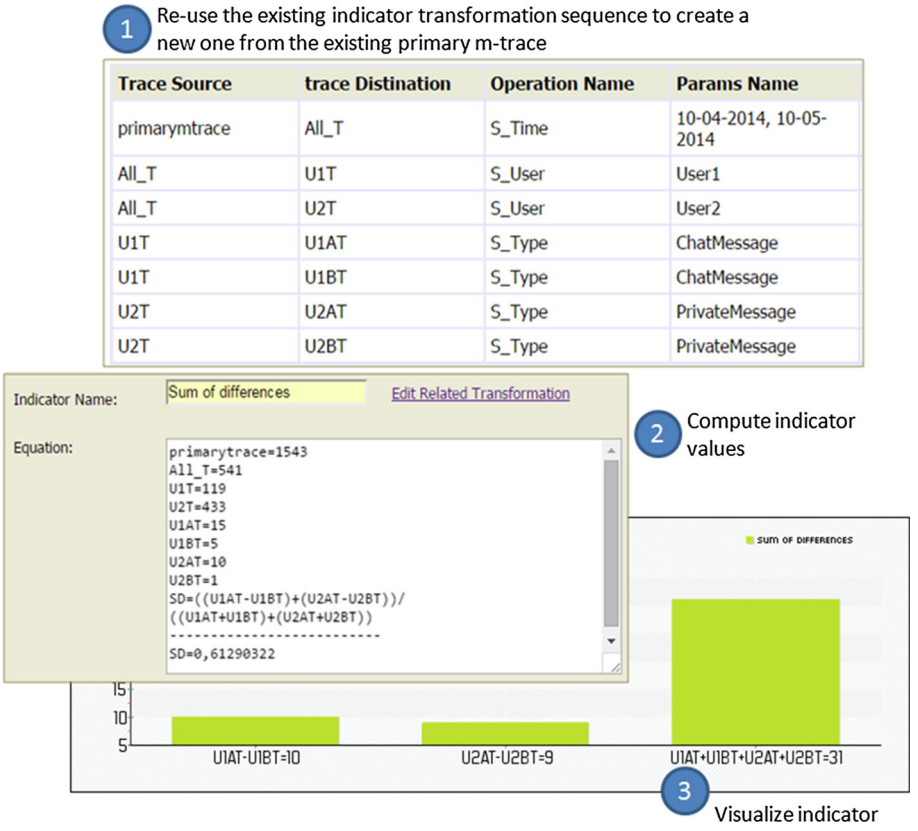

Fig. 10 Computing the *division of labor* indicator value of the last example in TB-IMS from the *sum of differences*

4.4 Scenario 3: Reusing the Same Indicator in Another Learning Platform

4.4.1 Our Goal

We recall that in ad-hoc methods (Dimitracopoulou et al. 2005), the indicator computation life cycle is:

- 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 visualizing 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. Teachers can then give feedback on the learning activity using the indicator values.

The goal of scenario 3 is to show how to use our method in order to compute the *same* indicator *division of labor* built in the previous section, but this time using raw data issued from *another* learning platform.

4.4.2 Case Study

We propose to use raw data derived from a real learning situation defined in Bratitsis and Dimitracopoulou (2009). We reuse, without modification, its transformation sequence to compute it with new raw data (Fig. 11). The only constraint is to update the collector module. This constraint is necessary to reuse transformation sequences when the raw data format from the two learning platforms is not the same.

4.4.3 Synthesis

In this scenario, the initial preparation transformations have to be adapted, but the final transformations for gathering the information about the formula variables and the formula itself were directly re-used. Time to check the existing indicator on new data is much shorter.

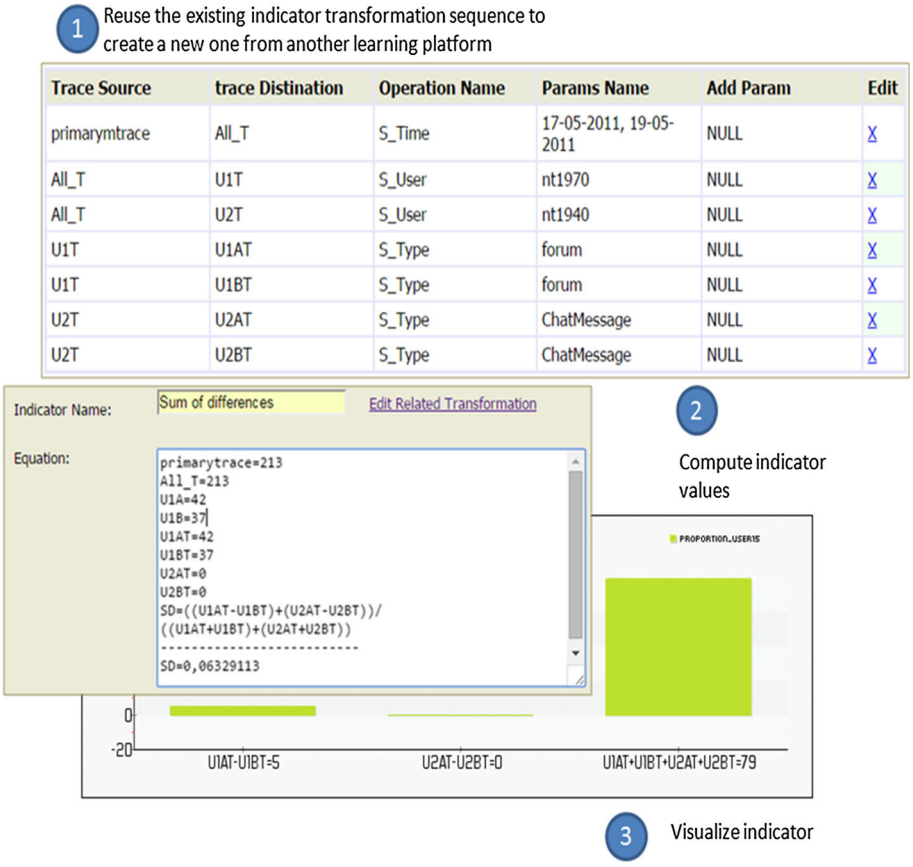

Fig. 11 Computing the *division of labor* indicator from another learning platform, using data issued from Bratitsis and Dimitracopoulou (2009). We reuse the same indicator built beforehand without modification to compute it with the new learning platform

Table 1 Comparison of indicator ad-hoc methods and our method

		Our method	Ad hoc method
Create new indicator from learning platform. Example: Compute proportion indicator from Moodle raw data-	Collecting	Coding collecting plug-in according to a specific learning platform. The result of this collecting is a primary m-trace needs a computer scientist	Coding and filtering raw data Needs a computer scientist
	Preparing	Use primary m-trace and create transformation sequence related to indicator model Teacher or researcher can transform data using transformation sequences, without needing a computer scientist for coding	Coding selected data transformation Needs a computer scientist
	Computing	Compute equation and visualize result Teacher or researcher can compute indicator without needing a computer scientist for coding The indicator model is available to all courses for the same learning platform	Coding indicator equation Computing needs a computer scientist
Reuse existing indicator to compute a new one from the same learning platform Example: Compute division of labor indicator from the same Moodle raw data	Collecting	Primary m-trace exists: no need for collecting data No need for coding	The same steps used to create new indicator Always coding
	Preparing	Reuse existing transformation sequence to compute indicator Does not need a computer scientist Teacher or researcher can reuse transformation sequences, without needing a computer scientist for coding	
	Computing	Reuse existing equation to create indicator values Does not need a computer scientist Teacher or researcher can reuse equation and compute indicator values without needing a computer scientist for coding	
Reuse existing indicator to compute a new one from another learning platform Example: Compute division of labor indicator using raw data of Bratitsis and Dimitracopoulou (2009)	Collecting	Coding and filtering raw data related to the new learning platform Needs a computer scientist	The same steps used to create new indicator Always coding
	Preparing	Reuse existing transformation sequence to compute indicator Does not need a computer scientist Teacher or researcher can reuse transformation sequences, without needing a computer scientist for coding	
	Computing	Reuse existing equation to create indicator values Teacher or researcher can reuse equation and compute indicator values without needing a computer scientist for coding	

Table 2 Comparison of existing indicator engineering methods and our method

		Our approach	Existing indicator engineering methods		
			Diagne (2009)	Iksal et al. (2010)	Gendron (2010)
Create new indicator from learning platform	Collecting	Create primary m-trace	Use database agent to collect data from database learning platform	Use ATL language to define useful data	Use trace based system to collect data
	Preparing	Create transformation sequence and use it to create indicator model	Use request agent to filter data	Use ATL language to get useful data	Use identity card to select data
	Computing	Compute equation and visualize result	Use composer agent to compute values	Use ATL language to compute values	Use equation to compute indicator
Reuse existing indicator to compute a new one from the same learning platform	Collecting	Reuse primary m-trace	No	No	No
	Preparing	Reuse existing transformation sequence to create a new one			
	Computing	Reuse existing equation to create a new one			
Reuse existing indicator to compute a new one from another learning platform	Collecting	Coding to create primary m-trace	No	No	No
	Preparing	Reuse existing transformation sequence without modification to create a new one			
	Computing	Reuse existing equation without modification to create a new one			

5 Conclusion

We presented in this paper a new approach for designing and computing human learning indicator activity in a framework separate from the learning platform used. This approach is based on a model driven engineering approach, and considers an indicator as a structured computer object (with model and instances parts) explicitly associated with modeled traces (m-traces) managed in a trace management system. Computation of the indicator instances is based on a transformation sequence of m-traces from the primary m-trace to the indicator m-traces.

Indicators can be easily created and reused. This is illustrated through three scenarios demonstrating the concept we propose. This method is thus an MDE approach for indicator engineering. A generic architecture and a tool to build and manage learning activity traces are used to compute indicators.

One other important contribution is that the proposed approach is able to support the modeling process by facilitating reuse of models for observable collecting as well as for indicator computation.

The developed system TB-IMS was used in Ji et al. (2014) to design a *Dynamic Dashboard for collection, analysis and visualization of Activity and Reporting Traces (DDART)* to support meta-cognitive activities. DDART is an extension used to improve learners' performances by monitoring their activities in Moodle. In DDART, learners can collect, analyze and visualize their traces, and then build indicators using an interactive dashboard. 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, using an activity report tool integrated into DDART; *Integration data:* build a primary m-trace combining Moodle raw data and non-instrumental data. The primary m-trace structure is based on the TB-IMS primary m-trace; *Compute indicator:* using a filter operator applied to the DDART primary m-trace.

The benefit of this method is demonstrated by comparing it to the ad-hoc indicator computation. The indicator description is available at an abstract level with transformation operators, leading to a trace specific to facilitating the final computation.

TB-IMS is widely independent from the learning platform except in data collection. Indicator computation is totally independent from the learning platform with the help of a TBS. Initial collection of user data remains specific to the learning environment in order to build a primary m-trace. After this step, all transformations can be independent to build indicator instances. Table 1 compares the indicator computation process with ad-hoc methods and with TB-IMS. Table 2 compares existing indicator engineering methods (from the literature) and our method.

We tried to make clearer the claim of the paper concerning the importance of formalizing explicitly what an indicator is. Although there are research frameworks to assist with analyzing learning processes in general, only a few research works or concrete frameworks formalize the complete indicator life cycle.

In future works, we will try to create a shared indicator library as an extension of our TB-IMS. This library will contain the most well-known indicators currently existing in human learning literature. Researchers and teachers/tutors can reuse these indicators directly, and then verify and enrich the library by providing their expertise. We offer them the possibility to create and reuse new indicators using our method without coding.

We will also try to improve TB-IMS using web component technology. Projects like Taaabs,[11] which is currently being developed, propose to use a trace based system such as a web component service.[12] We plan to integrate our TB-IMS through this technology. TB-IMS is a useful service that is completely independent from data collection and thus from learning platforms. TB-IMS directly uses the TBS web component services and can deploy new indicators in a shared library (work in progress).

[11] https://github.com/liris-tweak/taaabs.

[12] http://dsi-liris-silex.univ-lyon1.fr/taaabs/taaabs/dist/#!/taaabs/taaabs/dist/.

References

Barros, B., & Verdejo, F. M. (2000). Analyzing student interaction processes in order to improve collaboration. The DEGREE approach. *International Journal of Artificial Intelligence in Education, 11*(3), 221–241. http://ijaied.org/pub/1004/file/1004_paper.pdf.

Bézivin, J., & Gerbé, O. (2001). Towards a precise definition of the OMG/MDA framework. In *Proceedings of the 16th International Conference on Automated Software Engineering, IEEE* (pp. 273–280). San Diego, USA. https://pdfs.semanticscholar.org/6586/b19c6edf8850ac97fffb4ea6f65144200b1b.pdf.

Bratitsis, T., & Dimitracopoulou, A. (2009). Studying the effect of interaction analysis indicators on students' selfregulation during asynchronous discussion learning activities. In *Proceedings of the 9th International Conference on Computer Supported Collaborative Learning* (pp. 601–605). Greece. https://www.researchgate.net/publication/221033728_Studying_the_effect_of_Interaction_Analysis_indicators_on_student's_Selfregulation_during_asynchronous_discussion_learning_activities.

Butoianu, V., Vidal, P., & Broisin, J. (2012). A model-driven approach to actively manage TEL indicators. In T. Amiel, & B. Wilson (Eds.), *Proceedings of the EdMedia: World Conference on Educational Media and Technology* (pp. 1757–1765). Denver, Colorado, USA: Association for the Advancement of Computing in Education (AACE). https://www.researchgate.net/publication/230675845_A_Model-driven_Approach_to_Actively_Manage_TEL_Indicators.

Champalle, O., Sehaba, K., & Mille, A. (2013). Capitalize and share observation and analysis knowledge to assist trainers in professional training with simulation case of training and skills maintain of nuclear power plant control room staff. In *Proceedings of the 5th International Conference on Computer Supported Education* (pp. 627–632). Aachen, Germany. http://liris.cnrs.fr/Documents/Liris-6056.pdf.

Champin, P. A., Mille, A., & Prié, Y. (2013). Vers des traces numériques comme objets informatiques de premier niveau : Une approche par les traces modélisées. *Journal of Intellectica, 59*, 171–204. http://liris.cnrs.fr/Documents/Liris-5998.pdf.

Cordier, A., Lefevre, M., Champin, P. A., Georgeon, O., & Mille, A. (2013). Trace-based reasoning-modeling interaction traces for reasoning on experiences. In *Proceedings of the 26th International Florida Artificial Intelligence Research Society Conference* (pp. 363–368). Pete Beach, Florida, USA. http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS13/paper/download/5903/6100.

Cordier, A., Lefevre, M., Champin, P. A., Mille, A., Georgeon, O., & Mathern, B. (2014). Connaissances et raisonnement sur les traces d'interaction. *Journal of Intelligence Artificielle, 28*(2–3), 375–396. https://pdfs.semanticscholar.org/92c3/fb8d4cb4d2a4f45b1320afb4ed99af271ad3.pdf.

Diagne, F. (2009). *Instrumentation de la supervision par la réutilisation d'indicateurs: Modèles et Architecture*. Doctoral dissertation, Joseph Fourier University, Grenoble, France. https://hal.inria.fr/tel-00366368/document.

Dillenbourg, P. (1999). What do you mean by collaborative learning?. *Collaborative learning: Cognitive and computational approaches, 1*, 1–19. http://tecfa.unige.ch/tecfa/publicat/dil-papers-2/Dil.7.1.14.pdf.

Dimitracopoulou, A. (2004). State of the art of interaction analysis: Interaction analysis indicators interaction and collaboration analaysis supporting teachers and students selfregulation (ICALTS). In *Researsh report. JEIRP Deliverable D.26.1.1. Kaleidoscope NoE* (pp. 153). https://hal.archives-ouvertes.fr/hal-00190145/document.

Dimitracopoulou, A., Petrou, A., Martinez, A., Marcos, J. A., Kollias, V., Jermann, P., et al. (2005). State of the art on interaction analysis for metacognitive support and diagnosis. In *Researsh report JEIRP. D.31.1.1, EU 6th Framework programme priority 2, Information society technology, Kaleidoscope Network of Excellence* (pp. 2–62). https://hal.archives-ouvertes.fr/hal-00190146/document.

Gendron, E. (2010). *Cadre conceptuel pour l'élaboration d'indicateurs de collaboration à partir des traces d'activité*. Doctoral dissertation, Claude Bernard Lyon1 University, France. https://tel.archives-ouvertes.fr/tel-00708083/document.

Iksal, S., Choquet, C., & Pham Thi Ngoc, D. (2010). Generic modeling of indicator with UTL-the collaborative action function example. In *Proceedings of the 2nd International Conference on Computer Supported Education* (pp. 114–119). Valencia, Spain. https://www.researchgate.net/publication/221130379_A_Generic_Modeling_of_Indicator_with_UTL_-_The_Collaborative_Action_Function_Example.

Jermann, P. R. (2004). *Computer support for interaction regulation in collaborative problem-solving*. Doctoral dissertation, Geneva University, Switzerland. http://tecfa.unige.ch/tecfa/research/theses/jermann2004.pdf.

Ji, M., Michel, C., George, S., & Lavoué, E. (2014). DDART: A dynamic dashboard for collection, analysis and visualization of activity and reporting traces. In *Proceedings of the 9th European Conference on Technology Enhanced Learning* (pp. 440–445). Graaz, Austria. https://hal-univ-lyon3.archives-ouvertes.fr/hal-01130922/document.

Laforcade, P., Nodenot, T., Choquet, C., & Caron, P. A. (2007). Model-driven engineering (MDE) and model-driven architecture (MDA) applied to the modeling and deployment of technology enhanced learning (TEL) systems: Promises, challenges and issues. In *Architecture Solutions for ELearning Systems* (pp. 116–136). https://www.researchgate.net/publication/281327763_Model-Driven_Engineering_MDE_and_Model-Driven_Architecture_MDA_applied_to_the_Modeling_and_Deployment_of_Technology_Enhanced_Learning_TEL_Systems_promises_challenges_and_issues.

Laperrousaz, C. (2007). Question de la réutilisation d'outils de suivi d'activités d'apprenants dans des plates-formes de formation en ligne. In *Proceedings of the 3rd Conference on EIAH: Environnements Informatiques pour l'Apprentissage Humain* (pp. 485–496). Lausanne, Switzerland. https://hal.archives-ouvertes.fr/hal-00161482/document.

Martinez, A., Dimitriadis, Y., Rubia, B., & Fuente, P. (2003). Combining qualitative evaluation and social network analysis for the study of classroom social interactions. *Computers and Education, 41*(4), 353–368. https://telearn.archives-ouvertes.fr/hal-00190427/document.

May, M., George, S., & Prévôt, P. (2011). TrAVis to enhance online tutoring and learning activities: Real time visualization of students tracking data. *Journal of Interactive Technology and Smart Education, 8*(1), 52–69. https://www.researchgate.net/publication/220373199_TrAVis_to_Enhance_Online_Tutoring_and_Learning_Activities_Real_Time_Visualization_of_Students_Tracking_Data.

Mazza, R., & Botturi, L. (2007). Monitoring an online course with the GISMO tool: A case study. *Journal of Interactive Learning Research, 18*(2), 251–265. https://www.researchgate.net/publication/252554029_Monitoring_an_online_course_with_the_GISMO_tool_A_case_study.

Merceron, A., & Yacef, K. (2004). Mining student data captured from a web-based tutoring tool: Initial exploration and results. *Journal of Interactive Learning Research, 15*(4), 319–346. https://www.researchgate.net/publication/292023411_Mining_student_data_captured_from_a_web-based_tutoring_tool_Initial_exploration_and_results.

Mostow, J., Beck, J., Cuneo, A., Gouvea, E., & Heiner, C. (2005). A generic tool to browse tutor-student interactions: Time will tell!. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education* (pp. 29–32). Amsterdam, The Netherlands. https://pdfs.semanticscholar.org/eeb8/7c154e2dca61141354ba6208210908509c8c.pdf.

Reffay, C., Teplovs, C., & Blondel, F. M. (2011). Productive re-use of CSCL data and analytic tools to provide a new perspective on group cohesion. In *Proceedings of the 9th International Computer-Supported Collaborative Learning Conference: Connecting Computer-Supported Collaborative Learning to Policy and Practice* (pp. 846–850). Hong Kong, China. https://halshs.archives-ouvertes.fr/edutice-00616547/document.

Santos, O. C., Gaudioso, E., & Boticario, J. G. (2003). Helping the tutor to manage a collaborative task in a web-based learning environment. In *Proceedings of the AIED 2003 Workshop Towards Intelligent Learning Management Systems* (pp. 72–81). Sydney, Australia. https://pdfs.semanticscholar.org/40e6/ab1079a0848a5ebb260082c09a226c7d98a6.pdf.

Seidwitz, E. (2003). What models mean. *IEEE Software, 20*(5), 26–32. http://www.ie.inf.uc3m.es/grupo/docencia/reglada/ASDM/Seidewitz03.pdf.

Settouti, L. S., Marty, J. C., Mille, A., & Prié, Y. (2009). A trace-based system for technology-enhanced learning systems personalisation. In *Proceedings of the 9th International Conference on Advanced Learning Technologies ICALT* (pp. 93–97). Riga, Latvia. https://www.researchgate.net/publication/221423776_A_Trace-Based_System_for_Technology-Enhanced_Learning_Systems_Personalisation.

Soller, A., Martinez, A., Jermann, P., & Muehlenbrock, M. (2005). From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. *International Journal of Artificial Intelligence in Education, 15*, 261–290. http://ijaied.org/pub/1016/file/1016_Soller05.pdf.

Tedesco, P. A. (2003). MArCo: building an artificial conflict mediator to support group planning interactions. *International Journal of Artificial Intelligence in Education, 13*, 117–155. http://ijaied.org/pub/979/file/979_paper.pdf.

Von Davier, A. A., & Halpin, P. F. (2013). *Collaborative problem solving and the assessment of cognitive skills: Psychometric considerations*. Research report no. 13–41. P. Educational testing service, NJ (Ed.) (p. 36). https://www.ets.org/Media/Research/pdf/RR-13-41.pdf.

Zarka, R., Champin, P. A., Cordier, A., Egyed-Zsigmond, E., Lamontagne, L., & Mille, A. (2013). TStore: A trace-base management system using finite-state transducer approach for trace transformation. In *Proceeding of the 1st International Conference on Model-Driven Engineering and Software Development* (pp. 117–122). Barcelona, Spain. http://liris.cnrs.fr/Documents/Liris-5880.pdf.

Zhang, H., & Almeroth, K. (2010). Moodog: Tracking student activity in online course management systems. *Journal of Interactive Learning Research, 21*(3), 407–429. https://www.learntechlib.org/p/32307.