Int J Artif Intell Educ
DOI 10.1007/s40593-016-0113-0

ARTICLE

An Analysis of Student Model Portability

Benjamín Valdés Aguirre[1] ·
Jorge A. Ramírez Uresti[2] · **Benedict du Boulay**[3]

Abstract Sharing user information between systems is an area of interest for every field involving personalization. Recommender Systems are more advanced in this aspect than Intelligent Tutoring Systems (ITSs) and Intelligent Learning Environments (ILEs). A reason for this is that the user models of Intelligent Tutoring Systems and Intelligent Learning Environments, i.e. their student models, tend to be more heterogeneous and complex than traditional models used in Recommender Systems. To share and reuse student models we must first understand the restrictions for porting or reusing student models in new ITSs or ILEs. This paper proposes a classification of student models in terms of their portability. Portability is measured via each model's accessibility, complexity, architecture, popularity, and description. We use this classification to analyse and then grade student models that have been published in the AIED, EDM and ITS research communities in 2013 and 2014. The classification is intended to be used by researchers both as a methodology to measure the portability of a student model and as a guide to find existing reusable models.

Keywords Intelligent tutoring systems · Sharing student models · Student modelling · Portability

✉ Benjamín Valdés Aguirre
bvaldesa@itesm.mx

[1] ETIE - School of Computer Science and Electrical Engineering, Tecnologico de Monterrey Campus Querétaro, Querétaro 76130, Mexico

[2] Department of IT & Computing, Tecnologico de Monterrey Campus Estado de Mexico, Edo. Mex, Atizapan de Zaragoza 52926, Mexico

[3] Deparment of Computing, University of Sussex, Brighton BN1 9QJ, UK

Published online: 05 May 2016

Introduction

In 1990 Nwana described the classic architectures used for Intelligent Tutoring Systems (ITSs). The architecture consisted of a student model, a pedagogical module, an expert model, and an interface. The integration of these components created adaptive agents which could simulate some of the behaviours of human tutors. Since then, there have been new architectures for creating smart tools for learning, but the component which every approach shares is the student model. But what is a student model?

> Student models could be seen to perform two 'super' functions: acting as a source of information about the student, and serving as a representation of the student. In achieving these functions, they act in roles including corrective, elaborative, strategic, diagnostic, predictive and evaluative. Nwana (1990).

Nwana's statement reflects two main trends in student modeling. One trend focuses on describing the attributes of the student, e.g., storing grades, preferences, and decisions made. The other focuses in building representations that can reason about and simulate student behaviour, e.g., if the student has answered the last 3 questions right, the model will estimate the likelihood of the next answer being correct.

An example of a SM can be seen in Fig. 1 where a student uses a math tutor to practice arithmetic. The answers of the student (right or wrong) will be used to update what the tutor knows about the student, i.e., the SM. The model will then be used to determine whether the student has achieved mastery over a particular arithmetic skill, such as subtraction. If the student has achieved mastery, she or he is ready to move to the next lesson, otherwise she or he needs more practice to achieve mastery. Many models are used in a similar way, but the way in which each models calculates if the student is ready for the next lesson is what makes them different. To sum up we can say that a student model is a computational representation of a student which stores information and, in some cases, simulates student behaviour.

Sharing user information between systems is an area of interest for every field involving personalization. Recommender Systems (RSs) (Pu et al. 2012; Bobadilla et al. 2013) share factual information of their User Models (UMs) through centralized

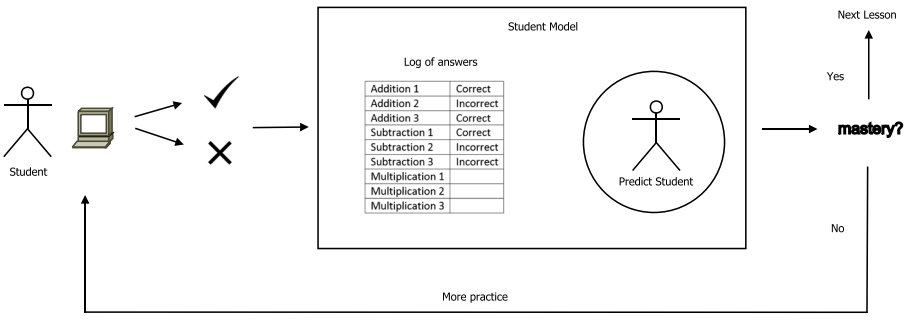

Fig. 1 Example of how a student model is used in an ITS

structures or via translation interface languages (Carmagnola et al. 2011). UMs used in ITSs (Nwana 1990; Woolf 2009) and Intelligent Learning Environments (ILEs) (Brusilovsky 1994) are what we know as SM. Student models are heterogeneous and complex, this makes them harder to share than the rest of UMs. For example a UM in a RS typically represents factual information about the user's preferences in a web page, their location, click through rate, and time spent looking at items. The SM in an ITS or ILE represents not only factual information, but also process information such as simulated behaviours (Chou et al. 2003), patterns of knowledge change, emotional states, facial expressions (Kort et al. 2001; Chen and Liu 2008; Li et al. 2011; San-Pedro and Baker 2011; Hussain et al. 2011; Baker et al. 2012), and may make predictions for user performance during learning (Sison and Shimura 1998; Stern et al. 1999; Beck 2000; Khajah et al. 2014; Jaques et al. 2014). Much of the functionality in a SM involves pattern detection, prediction, and simulation. To achieve this many SMs use machine learning (ML) and data mining. Sharing information stored in machine learning components is not a trivial problem.

The differences between a UM and a SM are not explicitly defined, rather they represent the result of focusing on different aspects of the interaction, i.e. user preferences vs student learning processes. Recording the surface behaviour of users browsing web pages differs significantly from modelling the cognitive and affective processes of students engaged in learning activities. So even though SMs can be regarded as a subset of UMs, in practice they behave differently because they are designed with different goals in mind.

The use of machine learning and data mining, and the wide variations between models makes it difficult for ITSs and ILEs to share SMs in the same way that RSs share UMs. However, this does not mean that SMs cannot be reused. In fact some SMs are already reused, modified, and extended (Baker et al. 2012; Paquette et al. 2014; Dowell et al. 2014), but there is no simple way to know whether a model can be re-implemented in another system without having a deep understanding of the model and the system. This limits a model's re-usability. It is often the case that only the creators of the original system are able to reuse its SM, and it is common that doing so means re-implementing the model for each new system. This is even more difficult for third party researchers who are trying to reuse a SM created by other researchers from different labs.

Knowing ahead of time which SMs are easier to reuse would save time for researchers who wish to build an intelligent system without having to build all of its components from scratch. By reuse we mean the utilization of previous implementations in other applications, as well as their modification and tuning.

This work aims to measure the amount of effort required to integrate recent SMs into other ITSs, ILEs, or UM sharing systems (Kobsa 2001; Vassileva et al. 2003; Niu et al. 2004; Lorenz 2005; Schwartz et al. 2006; Zhang et al. 2006; Carmagnola and Dimitrova 2008; Cena and Furnari 2009). The classification of the models can help researchers find models with fewer restrictions, and potentially more akin to their needs instead of developing new ones. Another reason for using previously developed SMs is that it generates continuity which benefits the developers of the models and research community. This work does not describe each of the models in detail and it is not a survey of SMs.

The paper is organised as follows. First we describe previous approaches for classifying user models. Then we explain how we used the ideas from previous classifications to calculate portability. We then present the methodology we followed to: choose the models to work with, measure the portability of each model, and create a graded categorization of models based on their portability. After explaining the criteria for grading we examine the categorization of portability through different dimensions. We also estimate the reimplementation difficulty for each model. We then analyse and discuss our findings. In the final part of the paper we make recommendations as to how to quickly identify the portability of a model derived from a machine learning classifier.

Previous Work and Ideas for the Classification of Student Models

According to Washizaki et al. (2003) portability is the ease with which software can be transferred from one environment to another. There is, as far as we know, no previous work that attempts to classify student models in terms of their portability restrictions. Surveys such as Desmarais and Baker (2011), Kobsa (2001), Kobsa et al. (2007), and Carmagnola et al. (2011) analyse factors that are related to student model portability. So we turn to these surveys as previous work.

Desmarais and Baker (2011) made a survey of learner and skill modelling. In this survey they provide descriptions of different types of student model in ITSs and ILEs. Desmarais and Baker group models by their type: skill models, affect models, meta-cognition, open learner, collaborative, and long term models among others. To establish a common ground with most SM literature we use the definitions for types of model from Desmarais and Baker (2011). We use these types of model to help us segment the vast literature into more approachable subsets and to find portability traits across similar models. Due to the increase in the use of Natural Language Processing (NLP) as components of models to detect and predict learning (Dowell et al. 2014), we have extended the original types of student model to include a language category. Language models as a category describe those models which map features of student learning and performance detected via NLP. We also include an "other models" category in which we place those SMs that were implemented less than three times in the SMs we checked. Table 1 is integrated from the types of model reviewed in Desmarais and Baker (2011) and our inclusion of the language models category.

Table 1 presents the types of model which are relevant to our study; the criteria we used to determine which types of models were relevant for our study is explained in the methodology section. We put together Table 2, also from Desmarais and Baker (2011), to show the techniques that are used to model skills. We use this table to infer the type of model from the technique used in an ITS. For example, an ITS that uses knowledge tracing (KT) will use a sequential model, whereas an ITS that uses "noise input deterministic and" (NIDA) will use a non-sequential model. NIDA is used in situations where several skills are required to perform a task. It uses a Q-matrix where every skill in an item must be mastered to succeed. From these we can assume that one model will detect the development of a skill using KT, and the other will detect the presence or absence of a skill using NIDA.

Kobsa (2001) and Kobsa et al. (2007) analysed Generic User Modeling Systems (GUMS). GUMSs were amongst the first approaches for sharing user model information between several systems. The initial approach in GUMS consisted of the creation of systems that built generic models which could store any kind of information from different domains. The goal was to have different systems consult a shared model. The second approach consisted of having a third party system regulate a standard model format such that different applications could exchange their models. GUMS are viable for models that work as repositories, but not for models that emulate behaviours. In reality there are no GUMS, or user models for that matter, which can store every fact, represent every pattern, and emulate every behaviour already modeled in computer science. In other words, it is very difficult to have a single model that solves every problem and has the benefit of every model. This idea is often referred to as the "no free lunch problem" in computer science (Dembski 2001). Projects that attempt this can be found in the more fundamental field of knowledge representation, where frames (Minsky 1975) and ontologies (Lenat and Guha 1989) have been amongst the largest initiatives. Kobsa's analysis of GUMS, however, provides insight as to what factors impact the capacity of sharing models, in particular those related to architecture of the models.

Carmagnola et al. (2011) studied user model interoperability. Although interoperability and portability are not the same thing, there are several attributes that both share. Carmagnola reviewed systems that exchange UMs, systems that provide adaptation services, and systems that share UMs across multiple applications. For each of these systems she analysed their architectures, like Kobsa, and their implementations to grade their interoperability. A similar approach can be used to grade portability.

In the following sections we will use the ideas and concepts presented in these surveys to create our own factors which will be used to measure student model portability. We believe that having a clear understanding of the traits that make a SM portable is necessary to increase the use of UM sharing systems (Kobsa 2001; Vassileva et al. 2003; Niu et al. 2004; Lorenz 2005; Schwartz et al. 2006; Zhang et al. 2006; Carmagnola and Dimitrova 2008; Cena and Furnari 2009) and less formal approaches for sharing student models. We now explain how metrics to measure

Table 1 Types of models

Type of Models	Skill	Affect	Motivation	Engagement	Other	Language
Elements or factors the models detect	Presence of skill (non-sequential models) Development of skill through time (sequential models)	Anger Happiness Boredom Frustration Uncertainty Confidence Excitement Interest	Effort Challenge Independence Confidence	Gaming Behaviours Off-task Behaviours	Rule Based CBM Sequencing Rule Based Unclassified	Narrative Text Cohesion Complexity Correctness

portability were derived from previous work, and our approach for grading model portability.

Methodology

As Washizaki et al. (2003) explain "...in application development with reuse, it is difficult to use conventional metrics because the source codes of components cannot be obtained". For this reason metrics are based in questions that allow us to analyze student models without having access to the source code.

Our goal is to create a classification that can be extended and improved in the longer term. In accordance with this idea, we chose simpler metrics. More precise metrics would be harder to calculate and simpler metrics can be refined upon. We used the metrics to calculate factors and we grouped the factors into 4 general dimensions. Table 3 shows the four dimensions: accessibility, complexity, architecture, and

Table 2 Non-sequential and sequential models

Models that detect the presence of a skill (non-sequential models)	Models that detect the development of a skill through time (sequential models)
Q-Matrix: defines the links between items and skills, it is a form of transfer model which can link items to concepts, or even concepts together.	Matrix Factorization Model: skill or concept learning over time.
Noise Input Deterministic And (NIDA): uses q-matrix in a conjunctive approach. Noise Input Deterministic Or (NIDO): uses q-matrix in a disjunctive approach.	Bayesian Knowledge Tracing (BKT): Based in Bayesian Theory it model's the learning of skills over time. They are also considered simple Bayesian Networks
Deterministic Input Noise And (DINA): associates the guess and slip parameters to items instead of skills. Also uses q-matrix	Principal Factor Analysis (PFA): used to represent changes in knowledge over time, uses PFA technique
Deterministic Input Noise Or (DINO): associates the guess and slip parameters to items instead of skills. Also uses q-matrix it is the compensatory part of DINA.	
IRT is "he ultimate skill transfer model" (single skill)	Learning Factor Analysis (LFA): used to represent changes in knowledge over time, uses LFA technique
Multi-Dimensional IRT	Logistic Regression (LR): simple description of a pattern over time, uses LR technique.

Table 3 Dimensions

Dimensions	Accessibility	Complexity	Architecture	Popularity	Description
Factors	Open Source	Inputs	Model Origin	Re-implemented	Type of Model
	Availability	Type of Data	Implementation		Pedagogical Theory
		Input Formats	Modularity		Approach
		Computational Models	Server/Client		Purpose
		Used	Has been Ported		
		Data Mining			

popularity, each with its corresponding factors. The description column was not used for calculating a grade for the models, instead it was used for the analysis presented in the section analysis.

We have 3 levels of grades. The first level is for each factor, the second level is for each dimension, and the third is the general grade for each model. A detailed description of how the grade of each factor is calculated is presented in the next section. Dimension grades are calculated by summing the grades of each factor and then normalizing the result. This normalized value is then mapped to values between 1 and 4. These values represent the portability restrictions of a student model in that dimension: 1 means low portability restrictions, 2 and 3 mean mid and high respectively. Grade 4 means that there was not enough information to calculate the grade of the model in that domain. Our posture in this regard is that if there is no available information of a model it is even harder to port that if it had high restrictions for portability. The general grade of the model is calculated by summing the grades of the dimensions. It is important to note that the fact that we normalize the factor for each dimension means that we assume that each dimension is equally important. This is a strong assumption to make and may bias our analysis, nevertheless, we believe that it is a valid assumption to be used as our initial approach.

Notes on Factors

There are several notes about the factors that should be pointed out. The first is that several of them show interdependence. We explain the possible dependencies in each factor where we detected it. The second is that we penalized the lack of information about a factor. Our rationale behind this was that if you cannot find information about the model, it will be harder to port. This is represented in our mappings for the grades of each particular factor in the following way:

1. low restrictions, in regard to this factor the model would be the easy to port.
2. mid restrictions, in regard to this factor the model would require significant work to be re-used in another software.
3. high restrictions, in regard to this factor the model requires a lot of work to be re-used in another system.
4. no information, there is not enough information of this factor to re-use the model.

The third note to consider is that these labels are comparative by nature, we chose the low mid high scheme because it seemed the natural way to compare very heterogeneous categories. But it does not mean that they are equivalent in nature, though we do make this assumption for our comparison. The reason for this is that there is no previous work or stepping stone in the field regarding this aspect. The most similar approaches are the ones cited from which we inspired our factors e.g. Kobsa (2001), Carmagnola and Dimitrova (2008), and Washizaki et al. (2003). Hence in our first approach we make this assumption. Finally there were some factor that we could not incorporate, such as the number of systems that use a particular model and the domain in dependence of each model. This is one of the reasons why the popularity domain only has 1 factor. The reasons this factor was to measurement difficulties.

Choosing the Models

The models were selected from the full papers presented in the AIED 2013, EDM 2014 and ITS 2014 conferences. These conferences represent the community involved in developing SMs. We chose this approach instead of a more historic one (e.g. review models published in journals in the last 10 years), because we aim to classify models that have been used recently. Working with recent models will allow the classification to be used in a practical way with student models that are still being used. The tradeoff of this approach is that some types of models will be overrepresented while others might not show at all, this reflects the trends of the conferences in those years.

When searching for the models we proceeded as follows: If a model was only mentioned but not described in the proceedings, we included previous work where the model was properly described and contacted the authors of the model for further information. We were interested in SMs for single users, so collaborative user models were omitted to reduce the scope of the search and will be addressed in future work. Open learner user models that do not have data mining or knowledge inference components were also not included, since these type of models are already analysed in Kobsa (2001) and Carmagnola and Dimitrova (2008). Classic approaches for student modeling such as Knowledge Tracing KT and Q-matrix approaches were included through their instantiations in the systems reported in the consulted papers.

Table 4 shows the SMs identified in the literature. The table shows the ID we assigned to each model and the references to the papers in which it appeared. A model can be presented in several papers, and a single paper can contain several models. Papers such as San-Pedro et al. (2013) model ids 10 to 15, and Paquette et al. (2014), model ids 37 to 48, contain more than 4 models each. An example of this can be seen in Paquette et al. (2014) where model id 38 is used to detect confusion at the hypothesis stage of an assignment, models id 37 and id 36 are different versions of the same model and are used at different stages of the assignment. Another common case for several models to be present in the same paper is the comparison of several models. An example of this can be found in Käser et al. (2014b) where a user-disaggregated model id 26, an additive factor model id 24, and a knowledge tracing model id 24 are

Table 4 Sources from where each model was consulted

Id	Ref.	Id	Ref.	Id	Ref.	Id	Ref.
1	(Girard et al. 2013) (Gonzalez-Sanchez 2014)	21	(Westerfield et al. 2013)	41	(Paquette et al. 2014) Boredom J48	61	(Chen et al. 2014)
2	(Wang and Beck 2013)	22	(Papoušek et al. 2014) (Pelánek 2014)	42	(Paquette et al. 2014) Conf. JRIP	62	(Hawkins et al. 2014)
3	(Yudelson et al. 2013)	23	(Pelánek 2014)	43	(Paquette et al. 2014) Frust. J48	63	(Käser et al. 2014a)
4	(Wang and Heffernan 2013)	24	(Käser et al. 2014b) AFM	44	(Paquette et al. 2014) Conc. J48	64	(Adamson et al. 2014)
5	(Bondareva et al. 2013)	25	(Käser et al. 2014b) LG	45	(Paquette et al. 2014) Boredom J48	65	(Rivers and Koedinger 2014)
6	(Baker et al. 2012) Concent.	26	(Käser et al. 2014b) DIS	46	(Paquette et al. 2014) Conf. J48	66	(Lipschultz and Litman 2014)
7	(Baker et al. 2012) Boredom	27	(González-Brenes et al. 2014)	47	(Paquette et al. 2014) Frust. J48	67	(Dascalu et al. 2014)
8	(Baker et al. 2012) Conf.	28	(PeddycordIII et al. 2014)	48	(Paquette et al. 2014) Conc. Reg.	68	(Gluz et al. 2014)
9	(Baker et al. 2012) Frust.	29	(Khajah et al. 2014)	49	(Lee et al. 2014)	69	(Wang and Heffernan 2014)
10	(San-Pedro et al. 2013) (Pardos et al. 2013) Concen.	30	(Grafsgaard et al. 2014) Eng.	50	(Lee et al. 2014)	70	(Sahebi et al. 2014)
11	(San-Pedro et al. 2013) (Pardos et al. 2013) Boredom	31	(Grafsgaard et al. 2014) Frust.	51	(Lee et al. 2014)		
12	(San-Pedro et al. 2013) (Pardos et al. 2013) Frust.	32	(Grafsgaard et al. 2014) Affect	52	(Mills et al. 2014)		
13	(San-Pedro et al. 2013) (Pardos et al. 2013) Conf.	33	(Grafsgaard et al. 2014) Confidence	53	(Mills et al. 2014)		
14	(San-Pedro et al. 2013) (Pardos et al. 2013) Off-task	34	(Grafsgaard et al. 2014) Frust.	54	(Mills et al. 2014)		
15	(San-Pedro et al. 2013) (Pardos et al. 2013) Gaming	35	(Grafsgaard et al. 2014) Excitement	55	(Jaques et al. 2014)		
16	(Dzikovska et al. 2013) (Dzikovska et al. 2014)	36	(Grafsgaard et al. 2014) Interest	56	(Bosch et al. 2014)		
17	(Dascalu et al. 2013)	37	(Paquette et al. 2014) Boredom J48	57	(Bosch et al. 2014)		
18	(Käser et al. 2013)	38	(Paquette et al. 2014) Conf. J48	58	(VanLehn et al. 2014)		
19	(Desmarais and Naceur 2013)	39	(Paquette et al. 2014) Frust. JRIP	59	(Kopp et al. 2014)		
20	(Goldin and Carlson 2013)	40	(Paquette et al. 2014) Conc. Reg.	60	(Dowell et al. 2014)		

compared. We will now explain in detail each of the dimensions and their respective factors.

Dimension: Accessibility

Kobsa (2001) analysed the support for open standards that different GUMS have and spoke in favour of this aspect. We extend this idea to models based in open source principles in general. We assume these SMs will be predisposed to be shared because they are free to use and accessible. Besides Kobsa's insights, we believe that access to the model is an important factor; for if the model cannot be accessed, it matters little if it is portable or not. To determine accessibility we ask two questions: "Is the model open source?" and "Is the model available, if so where?".

Open Source

We group, the possible answers of the first question into 4 categories:

1. Completely open source, any one can access and modify the model to better suit their needs.
2. Depends on commercial software or hardware. This includes the models that are in themselves free but that require a commercial hardware or software to be used.
3. Not open source. This includes models that require a license to be used, or that are part of a commercial application.
4. No information. There was no information regarding this aspect.

Availability

We group, the possible answers of the second question into 4 categories:

1. Online, the model can be accessed and downloaded by anyone.
2. Authorization is required. This includes the models that are available only with access to a database, as well as those that are available upon request.
3. No, the model is private and it is not available online.
4. No information, there was no information found in the reported literature

Dimension: Complexity

We assumed that a more complex model would have greater restrictions on portability. We counted the number of inputs, heterogeneous sources, and formats using an approach similar to Washizaki et al. (2003) metrics for complexity of software components. "Simply, the smaller the number of business methods without parameters, the smaller the possibility of having dependency outside the component." Washizaki et al. (2003). By looking at dependencies as requirements we rephrase this as the smaller number of inputs, different types of data used, computational techniques, and formats, the fewer restrictions a SM should have. To address these factors we asked the following questions: "How many inputs or sources of information does

the model require?", "How many different types of data are used?", "How many different formats is this information in?","How many different computational techniques or models does the student model involve?", and "Does the model use data mining?".

Inputs

We decided to use input as a factor because it represents a data requirement. The grade was calculated by summing the number of inputs the model uses and then it was normalized using:

$$\frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

where the highest number of inputs was 11 from models ID 32, 33, 53, and 66 (see Table 4 for a list of all the student models analysed). The lowest number of inputs was 1 for models ID 16, 17, 19, 21, 28, 60, 61, 62, 64, and 68. The normalized value was then mapped to one of the: low ($x < 0.3$), mid ($0.3 \leq x < 0.7$) or high ($0.7 \leq x$) portability restrictions. In Table 5 we present an example of how the grades were assigned.

Since the granularity of inputs reported in the literature varied, it was not enough to look just at the number of inputs. To complement this factor we also checked the type of data used and the format the inputs were in. There would be dependence

Table 5 Inputs

Inputs	Count	Normalized	Grade	Restrictions
The percent of past actions on the skills involved were incorrect Presence of actions in the clip where the student made a wrong answer	2	0.1	1	Low
Student Identity Class Identity Skill Identity Correct answers Log attempts	5	0.4	2	Mid
GOOD_METHOD VERIFY_INFO SINGLE_ANSWER SEVERAL_ANSWERS UNDO_GOOD_WORK GIVE_UP TIME_ON_TASK MISUSED_RUN_MODEL HYBIRD_FUNCTION	9	0.8	3	High

between these factors (input, type of data, and input format), but our method would better represent the cases where a model has a very large number of simple inputs with just one format or a model that has a small number of inputs but from very diverse sources.

Type of Data

When speaking of type of data, we refer to those described in Kobsa (2001) and Carmagnola and Dimitrova (2008), where the type of data was used to describe the data exchanged by user modeling systems. We extracted the following from a larger list of types of data in Carmagnola and Dimitrova (2008):

1. Usage, what the user did during his interaction with the system, i.e. logged behaviors and actions.
2. User, information about attributes of the user: knowledge, skill, and demographic aspects.
3. Social, information about the user's membership to communities and interaction with other users.
4. Domain, information about the application, e.g. contents, products, concepts.

We believe that heterogeneous types of data increase model complexity, as each new type of data must be logged from a different source thus increasing logging and representation requirements. As with the inputs we counted the total number of different types of data used in a model and then we normalized and partitioned the result to obtain 3 categories: low, medium, and high portability restrictions.

Input Format

The input format grade was calculated by summing the number of different formats of inputs, then we normalized and mapped as was done with inputs and types of data.

Computational Techniques used

When talking about re-usability (Washizaki et al. 2003) explain that being able to understand what a component does is an important factor. We try to represent this aspect by assuming that the more computational techniques a model has, the harder it will be to understand and to apply. The grade was calculated by summing the number of different computational techniques used by the SM. This reflected that a model that mixes several techniques will have more restrictions. The total number of computational techniques was then normalized to obtain 3 categories: low, medium, and high portability restrictions. Table 6 shows the way grades were assigned.

Data Mining

It should be noted that data mining approaches have been used to create more general models. The fact that these models are more general should encourage portability, however in order to create the model or update it, data (usually in large amounts) is

Table 6 Computational Technique

Computational Technique	Grade	Restrictions
REPTree	1	Low
Hand made Detectors and hand made rules, Heuristic and Expert rules	2	Mid
ELO system Performance Factor Analysis Dynamic Bayes Network/HMM	3	High

required. For this reason we chose to label the use of data mining as adding restrictions to the portability of the model. This was because in order to use a data mining in the model one needs to already have the data with which to train the model. To determine which models used data mining we compared the computational techniques and student models reported in each paper with Table 1. If the models fitted parameters for knowledge tracing, used natural language processing, or used classifiers, we presume that the authors used data mining. We assume that techniques that use data mining and update their models through time will present higher restrictions depending on the way data mining is used. For example, for sake of the argument lets say we had two student models: a Naive Bayes Network which was constantly updated and a function to predict factor $f(x)$. The Bayesian Network uses data mining to create and to update its conditional probability, but a function such as: $f(x) = FactorA * 0.3 + FactorB * 0.8$ doesn't use data mining at all will not require data, or if the function was fitted using some regression technique, then data mining was used only for its creation. In other words the Bayesian has a continual requirement for data and the function doesn't need it all, or only needs it once. With these scenarios in mind we grouped the data mining answers into 3 categories:

1. No, the model does not use data mining
2. Yes, only for model creation
3. Yes, updates itself at run-time

Dimension: Architecture

The way a student model is implemented provides information regarding practical issues. "External dependency indicates the component's degree of independence from the rest of the software which originally used the component." Washizaki et al. (2003). We counted dependencies and encapsulation to determine model portability. Since we did not have access to the source code, we asked questions that would allow as to infer dependencies and physical restrictions such as: how was the student model built? What frameworks, languages, APIs were used in its implementation? How is the model encapsulated? Does it require a server?

Model Origin

The way the model was conceived carries strong implications as to where the model can be used. From our own observations of the field, we created three groups of approaches which represent the way in which student models are created:

1. Standalone, the model is built by itself hence it is not integrated or restricted to any application, it is equally portable to any system.
2. Created or authored with a tool, the model is therefore restricted to the system it was created within or to the system it was created for.
3. Extension, the model is an extension of a pre-existing system meaning it is customized to a particular software and has strong dependencies to it.

Implementation

In their analysis of interoperability, Carmagnola and Dimitrova (2008) compared the languages user modeling systems use to share information. From this idea and from Washizaki et al. (2003) notion of dependence we decided to count the number of different elements involved in the model's implementation. This factor complements the computational techniques factor. Whereas in computational techniques we checked for model complexity in abstract terms, here we checked for its complexity in terms of its implemented dependencies. The assumption was that a higher number of technologies integrated will make the model harder to port to another environment. The grade was calculated by adding the number of different computer languages, frameworks, and libraries, and then normalizing the count. A higher number of components and tools implies more configuration and interfacing tasks. An example is shown in Table 7.

Modularity

Good modularization leads to loosely coupled systems. The more loosely coupled a systems is, the easier it is to change its components and integrate them into other environments. SMs can be encapsulated into modules to be reused as libraries, software components, web services, or agents among others. We assume that if a SM was generated through data mining using data from a 3rd party system, then the SM is by default encased in a separate module. This is a safe assumption because, other than the data, there is no dependency between the SM and the 3rd party system. We grouped modularity into 4 categories:

1. Single module
2. Not implemented in a system

Table 7 Implementation

Implementation	Grade	Restrictions
Coh-Metrix	1	Low
Java, Swing	2	Mid
Prolog, Java, Android	3	High

3. Several modules
4. Integrated in a system, hence no modularization

Server Client

Kobsa (2001) and Carmagnola and Dimitrova (2008) both analyze the architectures of user modeling servers. This gave us the idea of looking into server architectures. A student model running from a server will have different restrictions than a model running locally in an ITS. It is more likely for a server SM to port to another environment because servers tend to assume they will be consumed as services. Therefore they are designed to communicate with other programs, but not to be constantly ported or relocated. A more complex architecture will need additional interfaces between the server and the client. Besides the design restrictions, server client architectures often require a physical server which also adds to the complexity. In this aspect a standalone client represents fewer restrictions.

1. Client, single standalone client architecture
3. Server-client architecture

Ported to other Systems

We assume, that models that have already been used in other systems are more portable. Hence if a system has been ported before, we say it has low restrictions for portability, if it has not we say it has high restrictions.

1. Yes, the same instance of the model was used in another system.
3. No, similar models were implemented with a similar methodology but they were not the same model.

Dimension: Popularity

Systems that have larger communities of users and developers tend to have better documentation and are easier to use. We asked if the model had been re-implemented in another system. We assumed that SMs that had been re-implemented in more than one system were more popular and therefore should have fewer restrictions.

Re-implemented

There are several standalone models from the field of EDM which have not been integrated into any software, or that haven't been re-implemented, we counted those models as "has not been re-implemented".

1. Has been re-implemented
3. Has not been re-implemented

Context Sensitivity: a Case in Which Ported Models Mights not Work

In Kobsa (2001) the author lists the desirable traits a GUMS should have in order to increase its generality. One of such traits is domain independence. This means that a model can represent the aspect it is meant to represent, regardless of the domain it is in, i.e., regardless of the context. Lets say, for the sake of the argument, that an affect model was put in practice to determine whether a student likes or not working with a math tutor. If the affect model works just as well with a physics tutor; then we could argue that the affect model is independent from the math and physics domain.

Making a model domain independent is hard because the definition of domain can be subjective. For example if we needed to define the domain of physics for and ITS: Should this domain include the concepts of Calculus? Is Calculus a domain in itself? Or is it part of the domain of math and not of physics? Questions like these are common. The answers of these questions lie with the maker of each tutoring system. The domain can be defined as a conceptual label given to a group of concepts or skills by the creator of the tutor.

In practice, the term domain is commonly used to refer to the area of interest of the skills we wish to develop. Therefore to talk about a model that has no domain could be argued to be a model that does not represent anything. We excluded a domain independence factor because it could not be measured in a practical way. But the fact remains that context is an important player in whether a student model might work with another ITS or not. Models that are more related to the student traits than to the contents of a tutor, such as affective, motivation or gaming models are not exempt from context. This is also true for those model involving machine learning, which can overfit to the system that generated the training data (Paquette et al. 2015). This means that a student might behave differently when addressing a math problem compared to a physics problem. The interface also plays an important role in how the student will behave, since the interface dictates what a student can and cannot do.

It seems intuitive to think that the more differences there are in the domain and interfaces, the more changes will be observed in the behavior of a student. The lesson here is that even if a model can easily be ported from one ITS to another, it might not work, so even after porting it, the model should be tested to measure how the new context and new data impacts the ported model. An example of how to measure a model generality can be found in Paquette et al. (2015) where a model for gaming was tested across different ITS for the same domain or Muldner et al. (2011) where a similar comparison was made to determine which was a better predictor for gaming behavior: the problem or the student.

The reason why it is so hard to measure whether a model will work in a different context ahead of time is because, in reality it is not an issue of how general a model is, but rather one of whether it can be used in a particular ITS. We could argue that a model that has been successfully tested with several ITS would be more general than one that has not. But the work involved in porting most models to several ITS just for sake of measuring how sensitive to the context they are is not practical.

Comparison of Student Models

In order to calculate and then rank the models' portability we used the four dimensions specified in Table 3: accessibility, complexity, architecture, popularity. Each of the factors graded in each dimension was mapped in the following way: grade 1 for low restrictions, grade 2 for middle restrictions, grade 3 for high restrictions, and grade 4 for no information. The grades were then added and normalized to calculate the grade of each dimension. To calculate an overall grade for the model we summed the grades of each dimension and mapped in the following way:

- 4 - Represents the models with least types restrictions we found (Very Low).
- 5,6 - Represent the models with low restrictions (Low).
- 7,8 - Represent the models with medium restrictions (Mid).
- 9,10 - Represent the models with high restrictions (High).
- 11,12 - Represent the models with very high (Very High).
- 13 or higher - Represent that there was too little information of the model (No info).

The minimum possible value is 4 because it represents the summing of the 4 categories where each got a grade of 1 (Low restrictions). For a model to get a grade above 12 means that there is very little information about the model which makes it very hard to port or re-implement.

Tables 8, 9, 10, and 11 show the SMs surveyed. Each table groups the models by their overall restrictions for portability. The tables contain the grade of each dimension for that model, what the model is meant to detect and the approach used in the model.

Analysis

The classes defined for this work are coarse because we chose to simplify several aspects to make the classification useful. These means that there might be considerable differences between some models that fall in the same class. Another important aspect to be considered is that the chosen models were surveyed from a 2 year span and so may well be biased to the interests and tracks that were chosen for those years in each conference. Rather than looking at the whole history of SM we have looked only at a brief 2 years snapshot of models. We do not assume that the classified models represent every SM effort in the field. Indeed it would be useful to extend this work to include both past and future SMs. To get a general picture of the distribution of models, we put together in Fig. 2a histogram of portability restrictions showing how many SMs there are in each class i.e. very low, low, mid, high, very high, and no info.

For a model to be classified as having very low restrictions it would need to be: open source, available online, have been re-implemented or ported to other systems, using a single computational technique with a small number of inputs and with little heterogeneity, be implemented in a single language or framework, be encased in a single module or library, and be able to operate as a stand-alone client. This

Table 8 Models with low portability restrictions

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
6	Low	Low	Mid	Low	Low	Concentration	Affect detectors calculated through logs, don't use external sensors	Affect Model
7	Low	Low	Mid	Low	Low	Boredom	Affect detectors calculated through logs, don't use external sensors	Affect Model
8	Low	Low	Mid	Low	Low	Confusion	Affect detectors calculated through logs, don't use external sensors	Affect Model
9	Low	Low	Mid	Low	Low	Frustration	Affect detectors calculated through logs, don't use external sensors	Affect Model
10	Low	Low	Mid	Low	Low	Concentration	Affect detectors calculated through logs, don't use external sensors	Affect Model
11	Low	Low	Mid	Low	Low	Boredom	Affect detectors calculated through logs, don't use external sensors	Affect Model
12	Low	Low	Mid	Low	Low	Frustration	Affect detectors calculated through logs, don't use external sensors	Affect Model
13	Low	Low	Mid	Low	Low	Confusion	Affect detectors calculated through logs, don't use external sensors	Affect Model
14	Low	Low	Mid	Low	Low	Off-Task	Affect detectors calculated through logs, don't use external sensors	Engagement
15	Low	Low	Mid	Low	Low	Gaming	Affect detectors calculated through logs, don't use external sensors	Engagement
21	Low	Low	High	Low	Low	Constraint based model	The model is used inside a tutor in an Augmented Reality Application, It is generated from the ASPIRE authoring tool to provide feedback in and assembly tutor	Learner Model
37	Low	Low	High	Low	Low	Boredom	Use J48 to generate a classification to detect Boredom in Hypothesizing stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
38	Low	Low	High	Low	Low	Confusion	Use J48 to generate a classification to detect Confusion in Hypothesizing stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
39	Low	Low	High	Low	Low	Frustration	Use JRip to generate a classification to detect Frustration in Hypothesizing stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
40	Low	Low	High	Low	Low	Concentration	Use Step Regression to generate a classification to detect Concentration in Hypothesizing stage. Affect detectors calculated through logs, don't use external sensors	Affect Model

Table 8 (continued)

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
41	Low	Low	High	Low	Low	Boredom	Use jRIP to generate a classification to detect Boredom in Experimenting stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
42	Low	Low	High	Low	Low	Confusion	Use jRIP to generate a classification to detect Confusion in Experimenting stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
43	Low	Low	High	Low	Low	Frustration	Use j48 to generate a classifier to detect Frustration in Experimenting stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
44	Low	Low	High	Low	Low	Concentration	Use j48 to generate a classifier to detect Concentration in Experimenting stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
45	Low	Low	High	Low	Low	Boredom	Use j48 to generate a classifier to detect Boredeom in Anaylising stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
46	Low	Low	High	Low	Low	Confusion	Use j48 to generate a classifier to detect Confusion in Anaylising stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
47	Low	Low	High	Low	Low	Frustration	Use j48 to generate a classifier to detect Frustration in Anaylising stage. Affect detectors calculated through logs, don't use external sensors	Affect Model
48	Low	Low	High	Low	Low	Concentration	Use step regression to generate Affect detectors calculated through logs, don't use external sensors. Affect detectors calculated through logs, don't use external sensors	Affect Model
5	Low	Low	Low	High	Low	Development of skill	use eye behaviour to predict if the students is a High or Low Learner	Learner Model

Table 9 Models with mid portability restrictions

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
1	Low	Low	Mid	High	Mid	Engagement	create a Behaviour detection model which is used for an Affective Learning Companion	Engagement
4	Low	Low	Mid	High	Mid	Development of skill	BKT with partial credit which makes it more accurate	Skill Model
16	Low	Low	Mid	High	Mid	Correctness	Creates a model for evaluating student answers that mixes traditional student mapping (Semantic Interpretation) and Statistical Classification to obtain a more robust model.	Language Models
17	Low	Low	Mid	High	Mid	Complexity	Predicts reading comprehension, detects the level of text complexity	Language Models
19	Low	Low	Mid	High	Mid	Presence of skill	use of Matrix factorization with ALS to obtain Q-Matrices with better prediction rates.	Skill Model
23	Low	Low	Mid	High	Mid	Development of skill	Use time Decay functions as a substitue for BKT, time decay fucntions are easier to implement and work almost as well as BKT	Skill Model
60	High	Low	Mid	Low	Mid	Language Models	The authors use the NLP framework Coh Metrix to analyze chat logs from learners. Coh Metrix detects 5 features. Then they realte through mixmodeling this attributes with user performance (best features are Deep Cohesion and Syntax Simplicity)	Language Models
20	Low	Low	Mid	High	Mid	Presence of skill	The model predict performance of the student using student logs including his requests for assistance in the system	Skill Model
27	Low	Mid	Mid	High	Mid	Development of skill	They create a general representation for student model using KT which is extensible and escalable to several skill and items	Skill Model

Table 9 (continued)

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
28	High	Low	Low	High	Mid	Not contemplated	create networks of student interaction with the tutor to make recommendations to students who have similar networks	Skill Model
61	Low	Low	Mid	High	Mid	Presence of skill	Used a modified version of NIDA to detect which are the faulty concepts in a Q matrix. They establish a hierarchy between concepts	Skill Model
62	Mid	Low	Mid	High	Mid	Presence of skill	BKT that uses heuristics to estimate when a student learns a skill, and use this estimate to calculte the 4 probability paramters, prior knowledge, slip, guess, and will learn skill after question	Skill Model
55	Low	Mid	Mid	High	Mid	Boredom	Use gaze information to predict Boredom and Curiosity. Uses several classifiers, but doesn't choose one. Though the work resembles more a study it has the ideas of a model	Affect Model
66	Low	Mid	Mid	High	Mid	type of visual representation	The authors use stepwise linear rergession to generate models the help select the type of visual representation that best suits a student. The visual inputa can be illustratiosn or graphs	Not contemplated

Table 10 Models with high portability restrictions

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
2	Mid	Mid	Mid	High	High	Presence of skill	BKT for single user and class, contextualizing the user within the class	Skill Model
24	No Info	Low	Mid	High	High	Development of skill	Use Additive Factor Model AFM to predict student performance as a substitute to BKT	Skill Model
25	No Info	Low	Mid	High	High	Development of skill	Use Learning Gain Model LG to predict student performance as a substitute to BKT	Skill Model
26	No Info	Low	Mid	High	High	Development of skill	Use Disaggregated Model DIS to predict student performance as a substitute to BKT	Skill Model
49	High	Low	Mid	High	High	Boredom	Use step regression to generate a model that detects if student is in Boredom. Affect detectors calculated through logs, don't use external sensors	Affect Model
50	High	Low	Mid	High	High	Concentration	Use step regression to generate a model that detects if student is in flow state. Affect detectors calculated through logs, don't use external sensors	Affect Model
51	High	Low	Mid	High	High	Frustration	Use step regression to generate a model that detects if student is in Frustration. Affect detectors calculated through logs, don't use external sensors	Affect Model
56	High	Low	Mid	High	High	Confusion	Use facial expression, through vision recognition, to recognize emotions Affect. Confusion	Affect Model
57	High	Low	Mid	High	High	Frustration	Use facial expression, through vision recognition, to recognize emotions Affect. Frustration	Affect Model

Table 10 (continued)

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
68	No Info	Low	Mid	High	High	Rules Based Model	Uses a multiagent system to analyse each of the students step in the prooving of a mathematical theorem. Using the analysis it estimates the probability of the student completing the proof succesfully. The student model is therefore not explictly instantiated, but calculated based on the knowledge of the specialist agent, and the estimation of success is estimated from such calculation.	Rules Based Model
3	High	Low	High	High	High	Development of skill	BKT with specific calculated values for starting	Skill Model
22	Low	Mid	High	High	High	Development of skill	It is a probability model that uses ELO system and PFA in two stages, first to estimate previous knowledge second to use that indicator and predict current level of knowledge in a student	Skill Model
33	No Info	Low	Mid	High	High	Confidence	Use logs from Wayang Tutor to create Affect detectors for confidence	Affect Model
34	No Info	Low	Mid	High	High	Frustration	Use logs from Wayang Tutor to create Affect detectors for confidence	Affect Model
35	No Info	Low	Mid	High	High	Excitement	Use logs from Wayang Tutor to create Affect detectors for confidence	Affect Model
36	No Info	Low	Mid	High	High	Interest	Use logs from Wayang Tutor to create Affect detectors for confidence	Affect Model

Table 10 (continued)

ID	Accessibility	Complexity	Architecture	Popularity	Overall grade	Detects	Approach	Type of model
52	No Info	Low	Mid	High	High	Interest	a predicted model is created to detect and predict disengagement behaviors, using only previous interaction data	Motivation
59	No Info	Mid	Low	High	High	Off-Task	Use sueprvised classification to detect the best conditions to avoid mind wandering	Engagement
70	No Info	Low	Mid	High	High	Development of skill	Uses a Bayesian Probabilistic Tensor factorization to predict student perfomance	Skill Model

Table 11 Models with very high portability restrictions

ID	Accessibility	Complexity	Architecture	Popularity	Overall Grade	Detects	Approach	Type of model
30	No Info	Mid	Mid	High	VeryHigh	Engagement	Generates models that integrate visual recognition, user interaction, and postures to predict Affect and Learning. This is submodel predicting engament	Engagement
53	No Info	Mid	Mid	High	VeryHigh	Interest	a predicted model is created to detect and predict disengagement behaviors, using only previous interaction data	Motivation
54	No Info	Mid	Mid	High	VeryHigh	Interest	a predicted model is created to detect and predict disengagement behaviors, using only previous interaction data	Motivation
63	No Info	Low	High	High	VeryHigh	Presence of skill	Use a Dynamic Bayesian Network, to integrate a Topology of skills in a domain for Knowledge Tracing	Skill Model
64	No Info	Low	High	High	VeryHigh	Language Models	Use of language tagging and HMM to predict learning from observing student communication	Language Models
65	No Info	Low	High	High	VeryHigh	Learning Path Models	generates paths of students posible solutions to suggest hints, the paths themselves are a Student Model	Sequencing model
67	No Info	Low	High	High	VeryHigh	Language Models	Use of learning strategies with SVM to predict Reading Comprehension	Language Models
29	No Info	Low	High	High	VeryHigh	Development of skill	Student model is a hybrid approach between BKT and LF to yield better perfomarnace predictions	Skill Model
31	No Info	Mid	Mid	High	VeryHigh	Frustration	Generates models that integrate visual recognition, user interaction, and postures to predict Affect and Learning. This is submodel for predicting frustration	Affect Model
32	No Info	Mid	Mid	High	VeryHigh	Affect Model	Generates models that integrate visual recognition, user interaction, and postures to predict Affect and Learning. This is submodel for predicting learning gain	Affect Model
69	No Info	Mid	High	High	VeryHigh	Development of skill	Linear Regression is used to try to get a long term prediction and reassessment of student knolwedge. What a student learns in long term (Retention of knowledge after 1 week until 6 months)	Skill Model

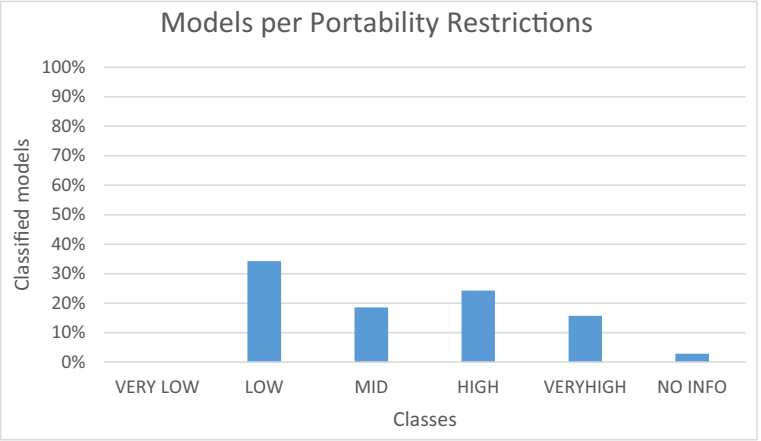

Fig. 2 Comparative chart of the population of our classes

is an utopian view since matching all these traits would mean a trade-off against the model's capacity to perform adequately. We therefore looked not for a particular ideal model, but for the ideal traits that the models in the area share, and so hoped to provide insight as to where improvements can be made. We observe that 34 % of the classified models have low restrictions, almost 19 % of them have medium restrictions, and 40 % the SM (24 % and 16 % respectively) are hard or very hard to port. The fact that 53 % of the models have low or mid portability restrictions can be interpreted as half the models show promise to be ported. In Fig. 3 we show the histograms for the restrictions of each dimension. At a first glance we can see that the accessibility and complexity dimensions present the overall lowest restrictions. Popularity and architecture present the higher restrictions. This could reflect the aspect in which SMs are more mature, i.e., the community is more concerned with making their models available and understandable, than with their architectures and replication.

Analysis of Accessibility

Accessibility is a dimension in which several models have low restrictions, from which we can assume that many researchers in the ITS, AIED, and EDM

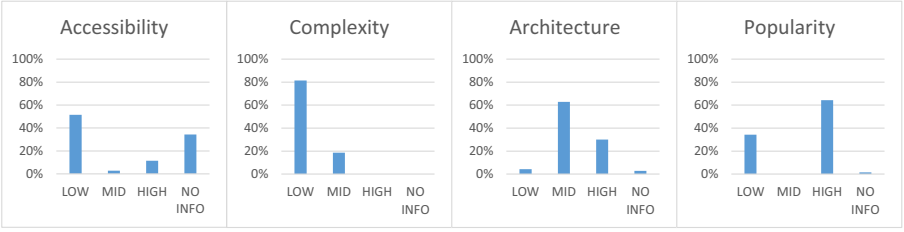

Fig. 3 Comparative chart of the population of our classes per dimension

Table 12 Analysis accessibility

Open Source restrictions		Availability restrictions	
Low	49 %	Low	10 %
Mid	4 %	Mid	47 %
High	11 %	High	13 %
No Info	36 %	No Info	30 %

communities are providing ways for other researchers to access their models. In Table 12 we looked at the individual factors used to calculate the accessibility grade: open source restrictions and availability restrictions. As we can see in Table 12 half of the models are open source, which means that we can freely reuse and modify the SM. However, access to the models' implementations and source code requires authorization. In many cases this minor restriction can be explained by the requirement of web repositories like github https://github.com/ that users register with before accessing private project branches. Private branches are a form of access regulated repositories for ongoing software projects. The problem with private access repositories is that sometimes the developers or creators of the systems are extremely busy or are no longer around to grant access to other researchers to their projects. An interesting fact is that 30 % of the SM do not report how to access or if their SMs can be used.

Analysis of Complexity

When looking at the factors used to calculate complexity presented in Table 13, we can observe that most restrictions spring from the use of data mining. The data to be mined is likely to incorporate restrictions of its own. To reduce the restrictions in the complexity dimensions implies reducing the model's complexity and this is not trivial a problem. Reduction of complexity by itself is considered a contribution; in fact every year contributions to the area are made by presenting models that perform similarly but with a reduced complexity (Käser et al. 2014b; Clement et al. 2014), which impacts on ease of development or processing time.

Table 13 Analysis complexity

Computational Techniques restrictions		Data Mining restrictions		Inputs restrictions		Type of Data restrictions		Input Format restrictions	
Low	96 %	Low	10 %	Low	47 %	Low	90 %	Low	90 %
Mid	3 %	Mid	44 %	Mid	27 %	Mid	9 %	Mid	4 %
High	1 %	High	46 %	High	26 %	High	1 %	High	6 %
No Info	0 %	No Info	0 %	No Info	0 %	No Info	0 %	No Info	0 %

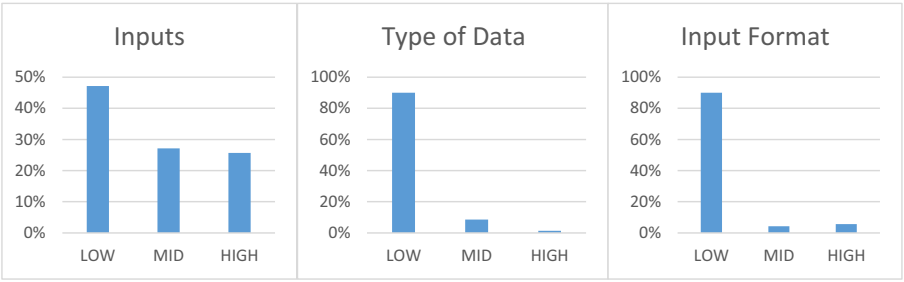

Fig. 4 Comparative chart of the population of classes per factor in the complexity dimension

Figure 4 shows the inputs, input format, and types of data factors that are used for calculating model complexity. We make this graphical comparison because we found an interesting relation between the factors. The input factor by itself is not enough to determine complexity, since a system could have hundreds of inputs which come from the same source. To balance this we incorporated the input format and the types of data the SM uses. This reflects that a SM can have few, but very hard to get or very hard to process inputs. The input data also restricts what kind of model is to be used and different types of data will be predisposed to different models. The grouping criteria for types of data are taken from Carmagnola et al. (2011) and they are: user, usage, domain, social, environment, inferred, and reasoning. When looking at Fig. 4 we see that there is little heterogeneity in the portability restrictions arising from the types of data and the input formats present in each SM. In general "usage data" is the most used type of data, which is common in educational data mining. This is because most usage data is extracted from "user logs" which is the most common type of input format. It is interesting to observe that there is no correlation between the inputs factor and the other two, which we would expect to show some degree of dependency.

Analysis of Architecture

Table 14 shows the factors used to calculate the architecture dimension. Only 3 % of the models reviewed have been ported which means that the little re-utilization there is, is mainly done by re-implementation. Re-utilization of the models is achieved

Table 14 Analysis architecture

Origin restrictions		Implementation restrictions		Modularity restrictions		Server Client restrictions		Ported restrictions	
Low	27 %	Low	56 %	Low	44 %	Low	46 %	Low	3 %
Mid	27 %	Mid	13 %	Mid	41 %	Mid	0 %	Mid	0 %
High	46 %	High	1 %	High	6 %	High	30 %	High	93 %
No Info	0 %	No Info	30 %	No Info	9 %	No Info	24 %	No Info	4 %

to a degree by re-implementing models or by extending the software that contains the model, as seen in the origin column of Table 14. The origin represents the approach or the historic relation of the SM and the system, i.e., if the SM was developed as a standalone model, as the heart of the system, or as an extension of an ongoing system. It should also be noted that most of the models that have low restrictions in the origin column come from the EDM conference, where authors commonly present their models and predictors without attaching them to any systems. As stated by Beck (2014), models presented in EDM have great potential to solve problems or complement the work done in other conferences such as AIED and ITS. A natural way to perform this task would be to make the SM as portable as possible.

Analysis of Popularity

As can be seen in Table 15, 34 % of the SMs analysed have been Re-implemented. Most of these models are affect models, which is logical since affect models complement skill models. The reason behind this is historical. Originally SMs focused on knowledge and later on skill modeling became the main concern. Since one of the main goals of the field is the development of skills, new SMs for affect, motivation, and engagement (Baker et al. 2012; San-Pedro et al. 2013; Paquette et al. 2014) are used, not to substitute, but to improve previously existing ITS which only used SM for skills. The bigger picture here being the joint effort to improve the learning process through the cooperative work of several student models.

Analysis of Types of Models

In Fig. 5 we present the relative proportions of different types of SM. We include the different types found in the surveyed papers and in the not contemplated category we put the models which did not fit with any of the well known models or approaches. We can see that there is a strong trend for affective models. This is not surprising given the time period for which the literature was sampled. In Fig. 6 we present the overall grades for portability restriction in each type of model. Here we can observe which types of model tends to have fewer restrictions. Affect models overall show the highest number of models with low restrictions. This might be explained by the fact that affect models complement other student models and therefore have been developed in way that reduces their portability restrictions. A bimodal distribution

Table 15 Analysis popularity

Re-implemented Restrictions	
Low	34 %
Mid	0 %
High	64 %
No Info	1 %

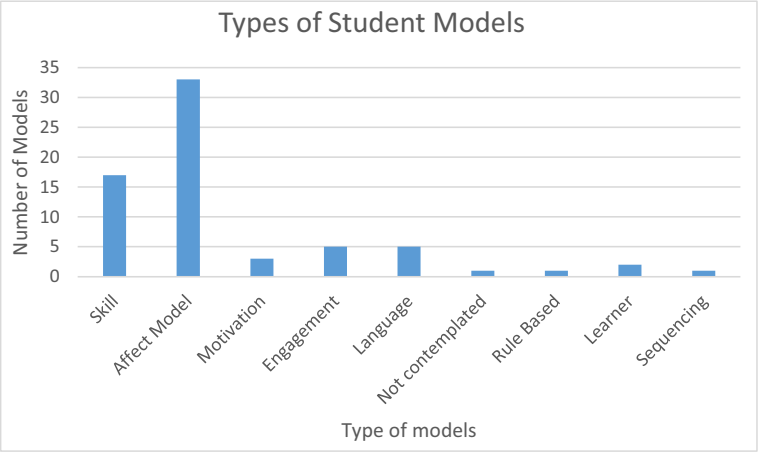

Fig. 5 Comparative chart of the population per type of model

can be seen in the affect models in Fig. 6. This distribution can be explained by two strong trends within affect models. The original trend was the use of external sensors to find physiological traits, such as eye movement, heart rate, pressure, posture, and humidity among others. The use of external sensors significantly increases model restrictions, since the SM needs to deal with additional hardware, additional software interfaces, additional types of data, and more computational techniques. The second approach is to use data mining in users' logs to infer a student's emotional status (Jaques et al. 2014). This approach has significantly fewer requirements since it does not depend on external sensors as it only uses system logs (Baker et al. 2012; San-Pedro et al. 2013). It must be noted though that most of the models in this category come only from a small group of authors (Paquette et al. 2014; Grafsgaard et al. 2014; San-Pedro et al. 2013; Pardos et al. 2013), however the bimodal distribution makes sense given the two main approaches taken in the field.

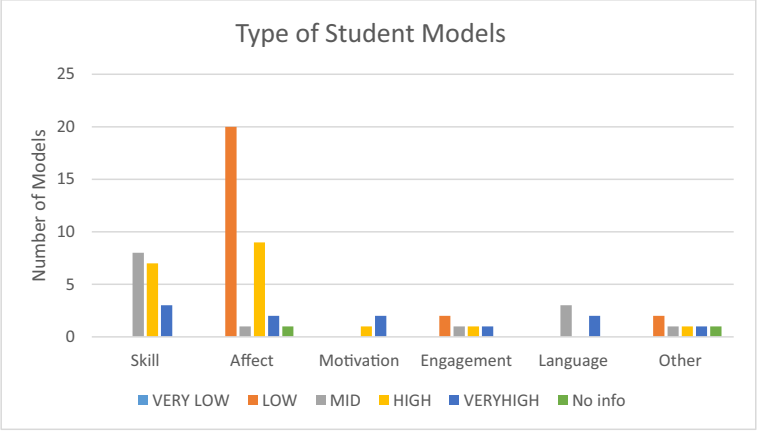

Fig. 6 Comparative chart of the population of classification per type of model

Reimplementation Difficulty

Besides measuring portability we also measured reimplementation restrictions. If a model is not portable then at least we could estimate the amount of work it would take to manually replicate it. So models where the source code can be extracted and adapted are portable; and those where it is easier to re-implement the entire model itself are easily replicable. By this definition every model listed in this work should be replicable since they are published in scientific sources, but the amount of work required to do so would vary greatly from model to model. We established 2 categories: low difficulty and high difficulty. Low difficulty means that, assuming that the inputs for the model are available, the SM could be re-implemented in a few lines of code and would take little time, (as in hours). High difficulty means that the model implementation is more complex and the amount of code or time required to re-implement it is unknown.

A Neural Network, or a Q-matrix model will be much harder to re-implement than a math formula, like a linear equation. For reimplementation difficulty, we consider 2 factors:

1. If the model is a single equation, e.g., a model for affect detection is "AffectValence(x) = Engagement*.3 + StressLevel*.8";
2. If the model is explicitly published including its optimal weights or configuration values.

If the model did not meet both of this requirements, it was considered to have high reimplementation difficulty.

In Table 16 we show a comparison between the difficulty in manually re-implementing a SM and the overall portability grade of a SM. The comparison highlights for which cases it would be better to port the SM and for which it would be easier to re-implement it.

We can see in Fig. 7 that most models are difficult to re-implement. In Fig. 8 we show a graphical representation of Table 16 where portability and reimplementation difficulty are compared for each model. Here we can see that, though some models have high restrictions for porting, they can be re-implemented more easily, but in general portability tends to be a more viable option than reimplementation.

Discriminating the Importance of Factor in our Categorization

Bobadilla et al. (2013) used data mining to organize the most relevant papers in the area of recommender systems. To determine which factors have more impact in portability we used a similar idea. Instead of using publication attributes and keywords as Bobadilla et al. (2013); we used the factors we proposed and their corresponding grades. We generated a minimal expansion trees using J48 (from the Weka data mining suite available at http://www.cs.waikato.ac.nz/ml/weka/) to find the determining factors in our categorization. The J48 algorithm creates a tree where the top node is the attribute that reduces entropy the most, which means the attribute at the root of the tree helps classify data in the smallest number of steps. The algorithm then repeats

Table 16 Reimplementation difficulty vs overall grade

ID	Reimplementation difficulty	Grade	Purpose of the model:
1	High	Mid	Affective intervention
2	High	High	Detect knowledge:predict performance
3	High	High	Detect knowledge:predict performance
4	High	Mid	Detect knowledge:predict performance
5	High	Low	Predict learning with eyetracking
6	High	Low	Detect concentration
7	High	Low	Detect boredom
8	High	Low	Detect confusion
9	High	Low	Detect frustration
10	High	Low	Detect concentration
11	High	Low	Detect boredom
12	High	Low	Detect frustration
13	High	Low	Detect confusion
14	High	Low	Detect off-task
15	High	Low	Detect gaming
16	High	Mid	Classify students answer
17	High	Mid	Detect level of compexity
18	High	Veryhigh	Predict performance
19	High	Mid	Predict performance
20	High	Mid	Predict performance
21	High	Low	Detection of errors
22	High	High	To estimate current level of knowledge
23	High	Mid	Predict performance
24	High	High	Predict performance
25	High	High	Predict performance
26	High	High	Predict performance
27	High	Mid	Emulate KT models and combine it with IRT to get evaluation of subsets of skills
28	High	Mid	the model is used to choose Hints strategies from different students and suggest it to othe students
29	High	Veryhigh	Predict performance
30	Low	Veryhigh	Predict Engagement
31	Low	Veryhigh	Predict Frustration
32	Low	Veryhigh	Predict Student Learning
33	Low	High	detect leve of confidence
34	Low	High	Detect Frustration
35	Low	High	Detect Excitement
36	Low	High	Detect interest
37	High	Low	Detect Boredom in a student's Hypothesizing stage

Table 16 (continued)

ID	Reimplementation difficulty	Grade	Purpose of the model:
38	High	Low	Detect Confusion in a student's Hypothesizing stage
39	High	Low	Detect Frustration in a student's Hypothesizing stage
40	High	Low	Detect Concentration in a student's Hypothesizing stage
41	High	Low	Detect Boredom in a student's Experimenting stage
42	High	Low	Detect Confusion in a student's Experimenting stage
43	High	Low	Detect Frustration in a student's Experimenting stage
44	High	Low	Detect Concentration in a student's Experimenting stage
45	High	Low	Detect Boredom in a student's Analyzing stage
46	High	Low	Detect Confusion in a student's Analyzing stage
47	High	Low	Detect Frustration in a student's Analyzing stage
48	High	Low	Detect Concentration in a student's Analyzing stage
49	Low	High	Detect Boredom
50	Low	High	Detect Flow
51	Low	High	Detect Frustation
52	High	High	Predict Disengagement at any page
53	High	VeryHigh	Predict Disengagement at first page
54	High	VeryHigh	Predict Disengagement after first page
55	High	MID	Predict Boredom and Curiosity with eyetracking alone
56	High	High	Detect confusion using face recognition
57	High	High	Detect frustration using face recognition
58	High	VeryHigh	Detect an emotional state and use that to generate a message
59	High	High	Detect conditions to avoid mind wondering
60	High	MID	Detect learning performance
61	Low	MID	Diagnoses concept knowledge
62	High	MID	To estimate current level of knowledge of a skill.
63	High	VeryHigh	To estimate current level of knowledge of a skill.
64	High	VeryHigh	To estimate learning of an individual
65	High	VeryHigh	Suggest Hint to get to the desired state of knowledge
66	High	MID	Detect visual representation more akin to user
67	High	VeryHigh	determine reading comprehension level
68	High	High	Determine if a student is able to use natural deduction in propositional logic correctly
69	High	VeryHigh	To reasses student knowledge and estimate it in long term.
70	High	High	Predict student success or failure

the process for each branch in a recursive way until all the members left to classify belong to the same class. We trained the algorithm using our portability factors and their calculated grades, and as an output extracted a tree whose top levels contained

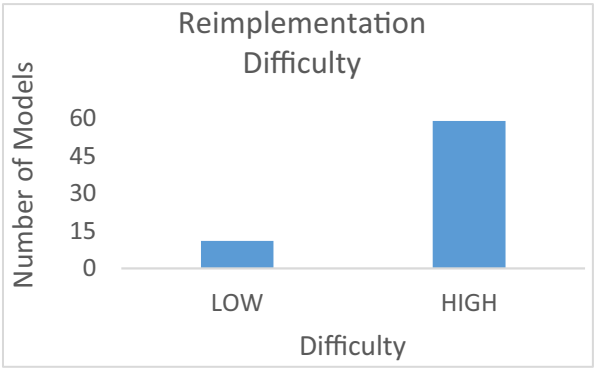

Fig. 7 Comparative chart of number of models with their reimplementation difficulty

the factors that best discriminated the SMs in terms of portability. We excluded the unique attributes such as approach and references, and we used a 10 fold cross validation. The class we used was the models general grade. The resulting tree can be seen in Table 17. The tree is built on the assumptions that we have explained in this paper. This is because our assumptions impact our data and tree reflects the relations between such data.

J48 points towards *has been reimplemented* (reimplementation in Table 17), *open source*, and *model origin* as the highest impact factors. The *has been reimplemented* factor should not be confused with the *reimplementation difficulty*. Interpreting the tree we can say that to quickly determine if a model is portable or not, one should first check if the model has been re-implemented, if the model is open source, and how was the model implemented. We think *has been reimplemented* is the main factor because if a student model has been reimplemented or replicated, the odds are that it was designed to be shared since its conception. Another possibility is that the community shaped model in such way as to be able to reuse it. It makes sense that open source is an important factor because it is likely that authors that make their model available online would take design and implementation decisions that would favor re-usability. If there is no information regarding whether the model is open source then one should look at the origin of the SM. As we mentioned before, the way a

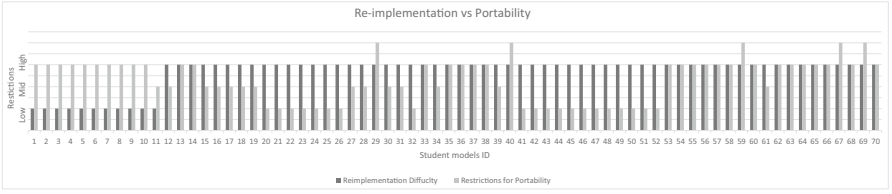

Fig. 8 Comparison between each model's reimplementation difficulty and portability restrictions. In the vertical axis 0 indicates very low portability restrictions or low difficulty of reimplementation. 5 means that the model has high difficulty or very high restrictions. This figure is a graphical visualisation of Table 16

Table 17 Discrimination of factors using J48

```
Tree J48:
        Reimplementation = HIGH
        |  Open source = MID: LOW (3.0/2.0)
        |  Open source = LOW: MID (11.0/1.0)
        |  Open source = HIGH: HIGH (7.0/2.0)
        |  Open source = NO INFO
        |  |  Model Origin = MID
        |  |  |  Input Format = LOW: HIGH (9.0/1.0)
        |  |  |  Input Format = MID: HIGH (0.0)
        |  |  |  Input Format = HIGH: VERYHIGH (3.0)
        |  |  Model Origin = HIGH: NO INFO (1.0)
        |  |  Model Origin = LOW
        |  |  |  Implementation = LOW
        |  |  |  |  Modularity = LOW: HIGH (0.0)
        |  |  |  |  Modularity = HIGH: HIGH (0.0)
        |  |  |  |  Modularity = MID: HIGH (2.0)
        |  |  |  |  Modularity = NO INFO: VERYHIGH (3.0/1.0)
        |  |  |  Implementation = MID: VERYHIGH (0.0)
        |  |  |  Implementation = NO INFO: VERYHIGH (5.0)
        |  |  |  Implementation = HIGH: HIGH (1.0)
        Reimplementation = LOW: LOW (24.0/1.0)
        Reimplementation = NO INFO: NO INFO (1.0)
```

=== Summary ===

Correctly Classified Instances	54	77.1429 %
Incorrectly Classified Instances	16	22.8571 %
Kappa statistic		0.6902
Mean absolute error		0.1194
Root mean squared error		0.2788
Relative absolute error		39.8527 %
Root relative squared error		72.1175 %
Total Number of Instances		70

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.958	0.087	0.852	0.958	0.902	0.958	LOW
	0.714	0.036	0.833	0.714	0.769	0.857	MID
	0.632	0.078	0.75	0.632	0.686	0.78	HIGH
	0.818	0.102	0.6	0.818	0.692	0.901	VERYHIGH
	0	0	0	0	0	0.684	NO INFO
Weighted Avg.	0.771	0.074	0.757	0.771	0.758	0.872	

=== Confusion Matrix ===

a	b	c	d	e	<-- classified as
23	1	0	0	0	a = LOW
2	10	2	0	0	b = MID
1	1	12	5	0	c = HIGH
0	0	2	9	0	d = VERYHIGH
1	0	0	1	0	e = NO INFO

model is built (as an extension, a standalone model, or created with some authoring tool) implies several design decisions which will impact on whether a SM can be shared or not. The other 3 factors that show up in the tree are input format, the

number of components used in the implementation, and modularity. All the factors present in the tree make sense for they are all desirable traits in shareable software, but we are trying to distinguish why some might be better indicators than others for portability. J48 points towards *has been reimplemented* (reimplementation), *open source*, and *model origin* as the highest impact factors. The *has been reimplemented* factor should not be confused with the reimplemented difficulty. In other words, to quickly determine if a model is portable or not, one should first check if the model has been re-implemented, if the model is open source, and how was the model implemented: as an extension, a standalone model, or created with some authoring tool. We think reimplementation is the main factor because if a student model has been reimplemented or replicated, the odds are that it was designed to be shared since its conception or the community shaped in such way as to be able to reuse it. It makes sense that open source is an important factor because it is more likely that authors that intend to share their model by making it available online would take design and implementation decisions that would favor re-usability and as a consequence portability. If there is no information regarding whether the model is open source then one should look at the origin of the SM. As we mentioned before, the way a model is built implies several design decisions which will impact on whether a SM can be shared or not. The other 3 factors that show up in the tree are input format, the number of components used in the implementation, and modularity. All the factors present in the tree make sense for they are all desirable traits in shareable software, but we are trying to distinguish why some might be better indicators than others for portability.

The input format as explained in the factors section carries strong implications regarding interfaces. The more different formats the data is in, the more interfaces will be required to process the data and hence the more work will be required. The implementation factor presents and similar situation where the more technologies involved the harder it becomes to understand and build interfaces for all of them.

Modularity is largely related to the other architecture parameters and in general it is used as a criteria for sharing and interconnecting systems, so it is not surprising that it should be present amongst the determining factors for measuring the portability of a SM as well.

Wrapping Student Models

An important aspect which has not be covered so far is how to describe student models. Standards such as IEEE Public and Private Information for Learners (PAPI for Learners) (IEEE-LTSC 2001), IMS Learner Information Package (LIP) (IMS GlobalLearning Consortium 2005b), and IMS eportfolio (IMS GlobalLearning Consortium 2005a) are used to "wrap" student information in meta-data descriptors. This was originally done to share learner information between different learning management systems like Blackboard and Moodle. These XML wrappers are meant to work with best practices guides and information models to safeguard the use the information and provide transparency between the systems that share the content. The standards used

to describe user information could also be used to make cleaner and more practical descriptions of student models.

PAPI, LIP, and eportfolio can be seen as structures to be filled with the learner's goals, skills, objectives, preferences, and evidence of work among others. These wrappers however are not straight forward to use with more complex student models. To store student models such as the ones classified in this paper, extension tags would need to be used. IMS standards, like LIP and eportfolio, incorporate extension tags in their XML schema to deal with extra content. The advantage of using wrappers is that student models might then be easier to search for and classify with automated tools and inference engines like the one used by Heckmann (2005). This ideal scenario however seems a faraway reality, for wrapping all these models would require a field-wide standardization effort.

Using the Classification

There are several ways in which our classification can be used:

- *Find a suitable SM.* When a developer wishes to create an ITS or ILE without having to create a new SM, he or she can use Tables 8, 9, and 10 to find a SM suitable to his needs. If developers wish more detailed information about the models, they can consult the complete table at http://goo.gl/6jQSgY.
- *Estimate a SM portability.* When developers wish to estimate if a SM would be easy to port they could ask the questions provided by the J48 in Table 4:
  - Has the SM been reimplemented before?
  - Is the SM open source?
  - How many technologies or framework are used in the implementation of the model?
- *Benchmarking.* When developers wish to create a flexible portable SM, they can use our classification and grading scheme to compare his SM to others. If any authors wish to update the information on their models they can send the updated information about their models and we will update the main table at http://goo.gl/6jQSgY.

We should note that after the classification for finding student models or estimating portability, the researchers should still check for aspects related to context sensitivity domain dependence. This aspect could very well determine whether it is useful to port the student model or not.

Conclusions

Based upon previous work in systems for sharing student models and software portability we have proposed a set of factors to measure student model portability. The factors are subject to the several assumptions we made and do not contemplate the domain independence, which means that even if a SM is successfully ported it might

not perform well. A way to reduce this difficulty is by trying to make sure that the ITS for which the model was built and the one the model will be used in are as similar as possible, especially in their interface and domain.

We applied the factors to a group of SMs from the AIED, EDM, and ITS conferences from 2013 and 2014 to generate a classification of models. The models were classified by their portability restrictions. The classes were low, high, and mid. The models with low restrictions are those which should be easier to re-use in other ITSs. From this we found that in the models we analysed affective models were the ones that showed the least restrictions for portability.

The levels of restrictions can be seen as the amount of problems a researcher might face when porting a model. It is up to the tutor designer to choose whether to implement a new model from zero or to reuse an existing one. A key factor in this decision should be the experience of the designer in making tutors. Designers who are creating a tutor for the first time should probably implement the models from zero so they can better understand how tutoring system work. More experienced designers, however, might make better use of this classification. They can use it to quickly find candidate systems and make estimations based on the metrics proposed here to see whether the effort involved in porting a student model out-weights the effort of implementing it from zero.

In general not every model should be ported. There are several authoring tools that can be used to quickly develop tutoring systems which can probably be used to quickly create student models as well. But authoring tools take some time to catch up to state of the art student model because new techniques are constantly being used in the area. It is here where we think that porting presents a plausible solution. Those models that are part of the state of the art but have not been integrated into some authoring tool are the ideal models to be ported.

To better understand the importance of each factor we used a machine learning algorithm, J48, to discriminate among the factors' classification properties. There was not enough data in general to build a solid classifier, so we used the J48 to make educated guesses regarding the importance of the factors. In general the models precision is bound to be low (around 70 %) due to the large amount of attributes, the reduced number of instances, and noise in the data. The results of the classification show that the quickest way to determine if a model is easy or hard to port is by asking if it has been re-implemented before, if it is open source, and how many technologies are involved in its implementation. The analysis of all of these factors might seem obvious since we are comparing desired characteristics in models. But what we think is interesting from the analysis is the priority between the factors, i.e. which ones have more importance when determining portability. To provide further guidelines as to how SMs might be made more portable, we discussed standards for sharing student information such as PAPI and LIP, and how they might be used to wrap student models in meta-data in order for the to be found more easily and easier to understand.

It is clear that our classification can be improved. There are many determining factors that we might have overlooked, or simply could not get our hands on. We tried to counter this by using simple metrics (counting repetitions and normalizing into low, mid, and high restrictions) so that future work can easily extend or modify our classification. A simpler approach is a good starting point for building a classification,

and will be a good baseline for future more refined classifications to come. We think this is a step towards a sharing set of mind. With regard to guidelines and advice, we have compared the difficulty of re-implementing current models with the restrictions of portability, so authors wishing to reuse any of the SMs analysed may choose the simpler path, i.e. importing the SM and creating interfaces vs implementing the complete model using the material published in recent papers.

As for future work, besides the refinement of its factors and extending the coverage of literature, several issues remain open. One is how to represent or measure domain independence of implemented SMs in a practical way. Another is the extension of the classification and its factors to include collaborative SMs and other type of models that were excluded from this work. Finally there is the idea a about a repository similar to Datashop where models could be hosted and shared using the IMS standards and organized perhaps by their portability, purpose and domain. All in all, we hope this initial analysis will start a conversation on student model portability, and how to better measure this attribute in components of intelligent software for education.

Acknowledgments We thank all the researchers that provided information of their models for this analysis and the journal reviewers for their accurate feedback. We also thank CONACYT and Tecnologico de Monterrey Campus Estado de Mexico and Campus Queretaro for their financial support.

References

Adamson, D., Bharadwaj, A., Singh, A., Ashe, C., Yaron, D., & Rosé, C.P. (2014). Predicting student learning from conversational cues. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 220–229). Honolulu: Springer International Publishing.

Baker, R.S., Gowda, S.M., Wixon, M., Kalka, J., Wagner, A.Z., Salvi, A., Aleven, V., Kusbit, G.W., Ocumpaugh, J., & Rossi, L. (2012). Towards sensor-free affect detection in cognitive tutor algebra. In *Conference on educational data mining* (pp. 126–133).

Beck, J. (2000). High-level student modeling with machine learning. In *Intelligent tutoring systems* (pp. 584–593). Montreal, Canada: Springer.

Beck, J. (2014). The field of EDM: where we came from and where we're going. In *Proceedings of the 7th international conference on educational data mining* (p. 2). London, United Kingdom.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132.

Bondareva, D., Conati, C., Feyzi-Behnagh, R., Harley, J.M., Azevedo, R., & Bouchet, F. (2013). Inferring learning from gaze data during interaction with an environment to support self-regulated learning. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 229–238). Berlin, Heidelberg: Springer.

Bosch, N., Chen, Y., & D'mello, S. (2014). It's written on your face: detecting states from facial expressions. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 39–44). Switzerland: Springer International Publishing.

Brusilovsky, P. (1994). Student model centered architecture for intelligent learning environments. In *Fourth international conference on user modeling* (pp. 30–36). Hyannis, Massachusetts, USA: MITRE Corporation.

Carmagnola, F., Cena, F., & Gena, C. (2011). User model interoperability: a survey. User Modeling and User-Adapted Interaction. *The Journal of Personalization Research*, *21*(3), 285–331.

Carmagnola, F., & Dimitrova, V. (2008). An evidence-based approach to handle semantic heterogeneity in interoperable distributed user models, In Nejdl, W., Kay, J., Pu, P., & Herder, E. (Eds.) *Adaptive hypermedia and adaptive web-based systems* (pp. 73–82). Hannover, Germany: Springer.

Cena, F., & Furnari, R. (2009). A model for feature-based user model interoperability on the web, In Kuflik, T., Berkovsky, S., Carmagnola, F., Heckmann, D., & Krüger, A. (Eds.) *Advances in ubiquitous user modelling* (pp. 37–54): Springer.

Chen, S., & Liu, X. (2008). An integrated approach for modeling learning patterns of students in web-based instruction: a cognitive style perspective. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *15*(1), 1.

Chen, Y., Wuillemin, P.-H., & Labat, J.-M. (2014). Bayesian student modeling improved by diagnostic items. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 144–149). Honolulu: Springer International Publishing.

Chou, C.-Y., Chan, T.-W., & Lin, C.-J. (2003). Redefining the learning companion: the past, present, and future of educational agents. *Computers & Education*, *40*(3), 255–269.

Clement, B., Roy, D., & Oudeyer, P.-Y. (2014). Online optimization of teaching sequences with multi-armed bandits. In *Proceedings of the 7th international conference on educational data mining* (pp. 269–272). London, United Kingdom.

Dascalu, M., Dessus, P., Bianco, M., & Trausan-Matu, S. (2014). Are automatically identified reading strategies reliable predictors? In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 456–465). Honolulu: Springer International Publishing Switzerland.

Dascalu, M., Dessus, P., Trausan-Matu, S., Bianco, M., & Nardy, A. (2013). Readerbench, an enviroment for analyzing text complexity and reading strategies. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 379–388). Berlin, Heidelberg: Springer.

Dembski, W.A. (2001). *No free lunch why specified complexity cannot be purchased without intelligence*: Rowman & Littlefield.

Desmarais, M.C., & Baker, R.S.J.D. (2011). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, *22*(1-2), 9–38.

Desmarais, M.C., & Naceur, R. (2013). A matrix factorization method for mapping items to skills for enhancing Expert-Based Q-Matrices. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 441–450). Berlin, Heidelberg: Springer.

Dowell, N.M., Cade, W.L., Tausczik, Y., Pennebaker, J., & Graesser, A.C. (2014). What works creating adaptive and intelligent systems for collaborative learning support. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 124–133). Switzerland: Springer International Publishing.

Dzikovska, M., Steinhauser, N., & Farrow, E. (2014). BEETLE II: deep natural language understanding and automatic feedback generation for intelligent tutoring in basic electricity and electronics. *International Journal of Artificial Intelligence in Education*, *24*(3), 284–332.

Dzikovska, M.O., Farrow, E., & Moore, J.D. (2013). Combining semantic interpretation and statistical classification for improved explanation processing in a tutorial dialogue system. In *Artificial intelligence in education* (pp. 279–288). Berlin, Heidelberg: Springer.

Girard, S., Chavez-Echeagaray, M.E., Gonzalez-Sanchez, J., Hidalgo-Pontet, Y., Zhang, L., Burleson, W., & VanLehn, K. (2013). Defining the behavior of an affective learning companion in the affective Meta-Tutor project, In Lane, H.C., Yacef, K., Mostow, J., & Pavlik, P. (Eds.) *Proceedings of 16th international conference on artificial intelligence in education* (pp. 21–30). Memphis: Springer.

Gluz, J.C., Penteado, F., Mossmann, M., Gomes, L., & Vicari, R. (2014). A student model for teaching natural deduction based on a prover that mimics student reasoning. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 482–489). Honolulu: Springer International Publishing Switzerland.

Goldin, I.M., & Carlson, R. (2013). Learner differences and hint content. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 522–531). Berlin, Heidelberg: Springer.

González-Brenes, J., Huang, Y., & Brusilovsky, P. (2014). General features in knowledge tracing applications to multiple subskills, temporal item response theory, and expert knowledge. In *Proceedings of the 7th international conference on educational data mining, pages 84 –91, london, United Kingdom*.

Gonzalez-Sanchez, J. (2014). A system architecture for affective meta intelligent tutoring systems. In *Proceedings of the 12th international conference on intelligent tutoring systems, pages 529–534, Switzerland. Springer international*.

Grafsgaard, J.F., Wiggins, J.B., Boyer, K.E., Wiebe, E.N., & Lester, J.C. (2014). Predicting learning and affect from multimodal data streams in Task-Oriented tutorial dialogue. In *Proceedings of the 7th international conference on educational data mining* (pp. 122-129). London: United Kingdom.

Hawkins, W.J., Heffernan, N.T., & Baker, R.S.J.D. (2014). Learning bayesian knowledge tracing parameters with a knowledge heuristic. In *Proceedings of the 12th international conference on intelligent tutoring systems, pages 150–155, honolulu. Springer international publishing*.

Heckmann, D. (2005). Ubiquitous User Modeling, volume 297. IOS Press.

Hussain, M., AlZoubi, O., & Calvo, R. (2011). Affect detection from multichannel physiology during learning sessions with AutoTutor. In *Artificial intelligence in education, pages 131–137, New Zealand. Springer*.

IEEE-LTSC (2001). IEEE P1484.2.22/D8, PAPI Learner - Learner Relations.

IMS GlobalLearning Consortium, I. (2005a). IMS ePortfolio. Version 1.0 Final Specification. Technical Report June.

IMS GlobalLearning Consortium, I. (2005b). IMS Learner Information Package Summary of Changes. Technical Report January.

Jaques, N., Conati, C., Harley, J.M., & Azevedo, R. (2014). Predicting affect from gaze data during interaction with an intelligent tutoring system. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 29–38). Switzerland: Springer international publishing.

Käser, T., Bussetto, A.G., Solenthaler, B., Kohn, J., Aster, M.V., & Gross, M. (2013). Cluster-Based Prediction of mathematical learning patterns. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 389–399). Berlin: Springer.

Käser, T., Klingler, S., Schwing, A.G., & Gross, M. (2014a). Modeling skill topologies with bayesian networks. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 188–198). Honolulu: Springer international publishing.

Käser, T., Koedinger, K.R., & Gross, M. (2014b). Different parameters - same prediction: an analysis of learning curves. In *Proceedings of the 7th international conference on educational data mining, 51–59, London, United Kingdom*.

Khajah, M.M., Wing, R.M., Lindsey, R.V., & Mozer, M.C. (2014). Integrating Latent-Factor and Knowledge-Tracing models to predict individual differences in learning. In *Proceedings of the 7th international conference on educational data mining, 99-106, London, United Kingdom*.

Kobsa, A. (2001). Generic user modeling systems. *User Modeling and User-Adapted Interaction*, *11*, 49–63.

Kobsa, A., Brusilovsky, P., Kobsa, A., & Nejdl, W. (2007). Generic user modeling systems. In *The adaptive web LNCS, chapter 4* (pp. 136-154). Berlin: Springer.

Kopp, K., Bixler, R., & D'mello, S. (2014). Identifying learning conditions that minimize mind wandering by modeling individual attributes. In *ITSProceedings of the 12th international conference on intelligent tutoring systems* (pp. 94–103). Switzerland: Springer international publishing.

Kort, B., Reilly, R., & Picard, R. (2001). An affective model of interplay between emotions and learning: reengineering educational pedagogy-building a learning companion. In *Proceedings IEEE international conference on advanced learning technologies* (pp. 43–46).

Lee, P.-M., Jheng, S.-Y., & Hsiao, T.-C. (2014). Towards automatic detecting wether student is in flow. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 11–18). Switzerland: Springer international publishing.

Lenat, D., & Guha, R.V. (1989). *Building large knowledge based systems: Representation and inference in the cyc project Addison-Wesley*. Technical report: Addison-Wesley.

Li, N., Cohen, W.W., Koedinger, K.R., & Matsuda, N. (2011). A Machine Learning Approach for Automatic Student Model Discovery. In *Proceedings of the 4th International Conference on Educational Data Mining* (pp. pages 31–40). Netherlands: Eindhoven. www.educationaldatamining.org.

Lipschultz, M., & Litman, D. (2014). Modeling student benefit from illustrations and graphs. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 436–441). Honolulu: Springer International Publishing Switzerland.

Lorenz, A. (2005). Agent-based ubiquitous user modeling, In Ardissono, L., Brna, P., & Mitrovic, A. (Eds.) *User Modeling* (pp. 512–514). Berlin: Springer.

Mills, C., Bosch, N., Graesser, A., & D'mello, S. (2014). To Quit or Not to Quit: Predicting Future behavioral disengagement from reading patterns. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 19–28). Switzerland: Springer International Publishing.

Minsky, M. (1975). A framework for representing knowledge. In *The psychology of computer vision. mcgrawhill, massachusett*.

Muldner, K., Burleson, W., Van De Sande, B., & Vanlehn, K. (2011). An analysis of students' gaming behaviors in an intelligent tutoring system: Predictors and impacts. *User Modelling and User-Adapted Interaction*, *21*(1-2), 99–135.

Niu, X., McCalla, G., & Vassileva, J. (2004). Purpose-Based Expert finding in a portfolio management system. *Computational Intelligence*, *20*(4).

Nwana, H. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, *4*(4), 251–277.

Papoušek, J., Pelánek, R., & Stanislav, V. (2014). Adaptive practice of facts in domains with varied prior knowledge, In Stamper, J., Pardos, Z., Mavrikis, M., & McLaren, B.M. (Eds.) *Proceedings of the 7th international conference on educational data mining, London, United Kingdom*.

Paquette, L., Baker, R.S., Carvalho, M.J.A.D., & Ocumpaugh, J. (2015). Cross-System Transfer of machine learned and knowledge engineered models of gaming the system. In *International conference on user modeling, adaptation and personalization*.

Paquette, L., Baker, R.S.J.D., Sao Pedro, M.A., Gobert, J.D., Rossi, L., Nakama, A., & Kauffman-Rogoff, Z. (2014). Sensor-Free Affect Detection for a Simulation-Based Science Inquiry Learning Environment. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 1–10). Switzerland: Springer International Publishing.

Pardos, Z.A., Baker, R.S.J., & Pedro, M.O.C.Z.S. (2013). Affective states and state tests : Investigating how affect throughout the school year predicts end of year learning outcomes. In *Third international conference on learning analytics and knowledge, 117–124, New York, NY USA. ACM*.

PeddycordIII, B., Hicks, A., & Barnes, T. (2014). Generating hints for programming problems using intermediate output. In *Proceedings of the 7th international conference on educational data mining, 92–98, London, United Kingdom*.

Pelánek, R. (2014). Application of time decay functions and the elo system in student modeling, In Stamper, J., Pardos, Z., Mavrikis, M., & McLaren, B. (Eds.) *Proceedings of the 7th international conference on educational data mining, 21–27, London, United Kingdom*.

Pu, P., Chen, L., & Hu, R. (2012). Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, *22*(4-5), 317–355.

Rivers, K., & Koedinger, K.R. (2014). Automating hint generation with solution space path construction. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 329–339). Honolulu: Springer International Publishing.

Sahebi, S., Huang, Y., & Brusilovsky, P. (2014). Predicting student performance in solving parameterized exercises. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 496–503). Honolulu: Springer International Publishing Switzerland.

San-Pedro, M., & Baker, R. (2011). Detecting carelessness through contextual estimation of slip probabilities among students using an intelligent tutor for mathematics. In *Artificial intelligence in education*. New Zealand: Springer.

San-Pedro, M.O.C.Z., Baker, R.S.J.D., Gowda, S.M., & Heffernan, N.T. (2013). Towards an understanding of affect and knowledge from student interaction. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 41–50). Berlin: Springer.

Schwartz, T., Heckmann, D., & Baus, J. (2006). Sharing sensor data in intelligent environments, In Berendt, B., & Menasalvas, E. (Eds.) *Workshop ubiquitous computing* (pp. 81–88). Berlin.

Sison, R., & Shimura, M. (1998). Student modeling and machine learning. *International Journal of Artificial Intelligence in Education*, *9*, 128–158.

Stern, M., Beck, J., & Woolf, B. (1999). Naive Bayes classifiers for user modeling. In *Proceedings of the conference on user modeling*.

VanLehn, K., Burleson, W., Girard, S., Chavez-Echeagaray, M.E., Gonzalez-Sanchez, J., Hidalgo-Pontet, Y., & Zhang, L. (2014). The affective Meta-Tutoring project lessons learned. In *Proceedings of the 12th international conference on intelligent tutoring systems* (pp. 84–93). Switzerland: Springer International Publishing.

Vassileva, J., Mccalla, G., & Greer, J. (2003). Multi-agent multi-user modeling in I-Help. *User Modeling and User-Adapted Interaction*, *13*(1-2), 179–210.

Wang, Y., & Beck, J. (2013). Class vs. Student in a Bayesian Network Student Model, In Lane, H.C., Yacef, K., Mostow, J., & Pavlik, P. (Eds.) *Proceedings of 16th international conference on artificial intelligence in education, volume 7926 of lecture notes in computer science* (pp. 151–160). Berlin: Springer.

Wang, Y., & Heffernan, N.T. (2013). Extending knowledge tracing to allow partial credit using continuous versus binary nodes. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 181–188). Berlin: Springer.

Wang, Y., & Heffernan, N.T. (2014). The effect of automatic reassessment and relearning on assessing student Long-Term knowledge in mathmatics. In *Proceedings of the 12th international*

conference on intelligent tutoring systems (pp. 490–495). Honolulu: Springer International Publishing Switzerland.

Washizaki, H., Yamamoto, H., & Fukazawa, Y. (2003). A metrics suite for measuring reusability of software components. Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No.03EX717), 211–223.

Westerfield, G., Mitrovic, A., & Billinghurst, M. (2013). Intelligent augmented reality training for assembly tasks. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 542–551). Berlin: Springer.

Woolf, B.P. (2009). Building Intelligent Interactive Tutors for revolutionizing e-learning. Morgan Kaufman Elsevier, US, 1 edition.

Yudelson, M.V., Koedinger, K.R., & Gordon, G.J. (2013). Individualized bayesian knowledge tracing models. In *Proceedings of 16th international conference on artificial intelligence in education* (pp. 171-180). Berlin: Springer.

Zhang, F., Song, Z., & Zhang, H. (2006). Web service based architecture and ontology based user model for cross-system personalization. In *IEEE/WIC/ACM International conference on web intelligence* (pp. 849–852). Washington D.C.