

Applying a Framework for Student Modeling in Exploratory Learning Environments: Comparing Data Representation Granularity to Handle Environment Complexity

Lauren Fratamico¹ · Cristina Conati¹ ·
Samad Kardan¹ · Ido Roll¹

Published online: 6 January 2017

© International Artificial Intelligence in Education Society 2017

Abstract Interactive simulations can facilitate inquiry learning. However, similarly to other Exploratory Learning Environments, students may not always learn effectively in these unstructured environments. Thus, providing adaptive support has great potential to help improve student learning with these rich activities. Providing adaptive support requires a student model that can both evaluate learning as well inform relevant feedback. Building such a model for interactive simulations is especially challenging because the exploratory nature of the interaction makes it hard to know a priori which behaviors are conducive to learning. To address this problem, in this paper we leverage the student modeling framework proposed in (Kardan and Conati, 2011) to specifically address the challenge of modeling students in interactive simulations. The framework has already been successfully applied to build a student model and to give adaptive interventions for an interactive simulation for constraint satisfaction. We seek to investigate the generality of the framework by building student models for a more complex simulation on electric circuits called Circuit Construction Kit (CCK). We evaluate alternative representations of logged interaction data with CCK, capturing different amounts of granularity and feature engineering. We then apply the student modeling framework proposed in (Kardan and Conati, 2011) to group students based on their interaction behaviors, map these behaviors into learning outcomes and leverage the resulting clusters to classify new learners. Data collected from 100 college students working with the CCK simulation indicates that the proposed framework is able to successfully classify students in groups of high and low learners and identify patterns of productive behaviors that are common across representations that can inform real-time

✉ Lauren Fratamico
fratamic@cs.ubc.ca

¹ University of British Columbia, Vancouver, Canada

feedback. In addition to presenting these results, we discuss trade-offs between levels of granularity and feature engineering in the tested interaction representations in terms of their ability to evaluate learning, classify students, and inform feedback.

Keywords Educational data mining · Clustering · User modeling · Interactive simulations · Exploratory learning environments

Introduction

Over the last several decades, science instruction has focused on facilitating authentic inquiry activities in which learners act as the scientists (Gobert and Buckley 2000; Hmelo-Silver et al. 2007; Kuhn et al. 2000). One method of doing so is with interactive simulations. These simulations are educational tools that support students in engaging in inquiry learning by allowing them to experiment with concepts and processes that would otherwise be limited to abstract principles or would require expensive and time-consuming real-world setups to explore (De Jong and Van Joolingen 1998; De Jong et al. 2013). However, not all students can learn well by working with interactive simulations (Alfieri et al. 2011; Kirschner et al. 2006). One main reason is that these environments are designed to facilitate exploratory processes, which tend to be more unstructured and open ended than those involved in other educational activities (e.g. problem solving). Students can explore the environment as they like. Thus, there is often no clear definition of correct behavior. Furthermore, these environments often offer less feedback than more constrained environments (Roll et al. 2010). These factors contribute to making it harder for some students to engage in productive explorations, monitor their own learning and make sense of their progress. For the students that are not effectively using the simulations, it may be beneficial to receive help that guides their interactions and promotes more effective behaviors (Gobert et al. 2013). As a matter of fact, there is increasing research in Intelligent Tutoring Systems (ITS) to endow these interactive simulations, and other types of Exploratory Learning Environments (ELEs), with the ability to provide student-adaptive support (Borek et al. 2009; Gong et al. 2012; Kardan and Conati 2013, 2015).

Delivering effective student-adaptive support requires having a student model that can identify *which learners* are in need of support, *when* support should be given, and the right level and content of *assistance* (Roll et al. 2010). However, building models for unstructured environments like ELEs is challenging. The unconstrained nature of the interaction and the lack of easily definable correct behaviors make it difficult to know a priori what behaviors are conducive for learning. (Segedy et al. 2016). While theories of inquiry learning offer definitions of the different inquiry phases, they lack operational definitions and thus cannot be used to drive interventions (Gobert and Buckley 2000; de Jong 2006).

One common way to deal with this challenge is to limit the exploratory nature of the interaction (Gobert et al. 2012; Westerfield et al. 2013) by constraining how the students can work with the ELE. In contrast, in our research we seek to support learners working with ELEs while maintaining the open-ended nature of the interaction as much as possible, striking a balance between guidance and exploration. The core of our approach is a student modeling framework, proposed by Kardan and Conati (2011) that

does not require the users interactions to be restricted. This framework uses the logged actions of the students as they are working with an ELE to learn which behaviors should trigger help. To do this, clustering is first applied to identify groups of students who learn similarly with the environment based on their logged data alone. Next, association rule mining is used to extract the distinguishing interaction behaviors of each clustered group of students. These behaviors are then used to classify new users and trigger real-time adaptive interventions built upon these behaviors. This unsupervised student modeling framework was successfully applied to provide adaptive support in the CSP applet, an interactive simulation for the AC3 constraint satisfaction algorithm (Kardan and Conati 2015). That work showed that an adaptive version of the CSP applet derived from the framework generated better learning than its non-adaptive counterpart, and was especially useful for students with lower initial domain knowledge.

In this paper, we investigate whether the student modeling framework in (Kardan and Conati 2015) - FUMELE (Framework for User Modeling for Exploratory Learning Environments) from now on - can be successfully applied to a different interactive simulation, as a step toward showing the generality of that approach. We do not go as far as building and testing adaptive interventions based on FUMELE, however we show that we can derive students models similar in accuracy and performance to the one in (Kardan and Conati 2015). This is in itself an important contribution both because a suitable student model is a main component of successful adaptivity and because the simulation targeted in this paper - the PhET DC Circuit Construction Kit (CCK; <https://phet.colorado.edu/en/simulation/circuit-construction-kit-dc>) - is substantially more complex than the CSP applet.

Our work shows that all but one aspect of the FUMELE could be directly applied to model students interacting with CCK. The part of the approach that needed to be changed pertains to the representation of interface actions that is used for characterizing student interaction behaviors in FUMELE. When FUMELE was applied to build a student model for the CSP applet, we used raw actions to characterize interaction behaviors, which was suitable because the CSP applet involves only seven basic actions. In contrast, CCK supports over a hundred different types of actions that can be applied in different contexts to experiment with building electric circuits. Using raw student actions for representing interaction behaviors with the CCK simulation did not scale up to this increased complexity. Thus, in (Kardan et al. 2014) we proposed a richer representation that captures the relationship between the different actions and offers some level of abstraction. Specifically, we proposed a multi-layer representation of action-events that includes information on individual actions (e.g., join), the components manipulated during those actions (e.g., light bulbs), the relevant family of actions (e.g., revise), and the observed outcome (e.g., changes to light intensity or if a fire started in the circuit). We showed that clustering interaction behaviors based on this representation succeeds in identifying students with different learning outcomes in CCK. In (Conati et al. 2015), we compared this multi-layered representation with additional ones that capture user behaviors at different levels of granularity. Specifically, we provided a comprehensive evaluation of these representations as the basis to apply the student modeling framework proposed in (Kardan and Conati 2011) to CCK. The evaluation was done in terms of: (i) ability to cluster learners with high or low learning gains; (ii) suitability for generating user models that can classify new

students in terms of their learning performance after they finish interacting with CCK; (iii) potential to define quality content of adaptive support to deliver during interaction.

In this paper, we extend the work in (Conati et al. 2015) by adding an evaluation criteria based on the ability to classify students *over time* as they interact with the simulation, an important criterion to judge the suitability of a user model that needs to provide predictions in real time while learning is taking place (Baker et al. 2011; Kardan and Conati 2011). Although classifying students at the end of the interaction (as was done in (Conati et al. 2015)) is valuable to give learners retroactive feedback and suggestions to improve future interactions with the simulation, evaluating classifier performance over time is crucial to understanding how quickly an adaptive version of CCK could begin to provide assistance to the learners in real-time while they are working with the ELE. We also include an additional representation in our comparison that solely relies on information automatically available from action logs. In contrast, all the representations considered in (Conati et al. 2015) relied on some form of knowledge engineering. The addition of the new representation to our investigation enables further insights on the effect of different amounts of feature engineering for representing learner interactions in complex simulations.

Our results show that the student-modeling framework proposed in (Kardan and Conati 2011) can be leveraged to build student models for complex simulations such as CCK, given a suitable representation of student actions. The representations that we discuss in this paper are general in nature, involving elements that characterize actions in term of their common types, the entities they are applied to, and the corresponding outcomes. These models achieve a classification accuracy comparable to what was reported for the CSP applet (Kardan and Conati 2015), both at the end of interaction and over time. They also identify behaviors that intuitively relate to learning and thus can be leveraged to design interactive support. Proof of concept that this form of adaptive guidance can work is provided by our previous results with the CSP applet, showing that adaptive support designed for the CSP applet based on learning-related behaviors identified by FUMELE for that simulation are highly effective for learning (Kardan and Conati 2015). We further discuss tradeoffs between the various representations tested in this paper with regard to model accuracy and behaviors identified, specifically looking at the impact of varied levels of knowledge engineering.

The rest of the paper is structured as follows. We first discuss related work. Then, we describe the CCK simulation and the study used for collecting data. Next, we summarize the workings of FUMELE, discuss the different representations we evaluated, present the evaluation results, and conclude with discussions and future work.

Related Work

There is ample evidence showing the benefits of providing adaptive guidance in learning environments that provide support for structured problem solving activities (e.g. (Najar et al. 2014; Salden et al. 2009; Stamper et al. 2013)). Researchers have also started to show results on the effectiveness of adding adaptive guidance to ELEs (e.g., (Borek et al. 2009; Hussain et al. 2009; Sao Pedro et al. 2012; Westerfield et al. 2013)). However, these results involve relatively structured ELEs where users are limited in the actions that they can perform or in the variety of solutions they can generate. For

instance, the simulations developed by Hussain and colleagues (Hussain et al. 2009) to train Navy sailors on emergency flooding procedures allow users to interact only with parts of the simulated ship relevant to these specific procedures. The augmented reality training system for assembling motherboards created by Westerfield et al. (2013) allows users to interact with any of the components available in the system but only allows for one correct solution, thus the feedback is limited to parts being assembled in the correct/incorrect place as per this solution. Similarly, Chemistry VLab (Borek et al. 2009) and Science Assistments Phase Change Microworld (Sao Pedro et al. 2012), virtual test beds for inquiry-based learning chemical and physical reactions, both provide help only on well-defined steps required to run the predefined scientific experiment.

Other ELEs leverage cognitive tools to reduce the space of behaviors that students can generate. Cognitive Tools are “templates” that help learners organize their learning process, and in the context of inquiry, they often limit (and pre-label) the type of interactions available, making student behaviors easier to interpret and evaluate. For example, SLINQ (Gobert et al. 2012) scaffolds the inquiry process using cognitive tools so it becomes more linearized and tractable. However, constraining the interaction may hinder learning and reduce students’ agency. In addition, linearizing the inquiry process reduces its authenticity, which is one of the goals of interactive simulations such as the PhET Simulations (Wieman et al. 2008).

Our work in this paper and in (Kardan and Conati 2011; Kardan and Conati 2015) is the first that aims to provide adaptive support in ELEs where students can freely use the functionalities available in the system, with no predefined sets of correct courses of actions and without limiting the exploratory nature of the interaction. Kardan and Conati (2015) provided encouraging evidence in favor of effectiveness of non-constraining adaptive support in ELEs, and in this paper we seek to build student models that eventually can drive similar unconstrained support for the CCK PhET simulation.

As mentioned in the **Introduction** section, our work builds on a student modeling approach (proposed in (Kardan and Conati 2011) and leveraged in (Kardan and Conati 2015) and (Conati et al. 2015)) that uses clustering to form groups of learners, mines the rules that define each of those groups, and uses the rules to classify new users of the simulation. In the field of Educational Data Mining, clustering and other data mining techniques have been applied for off-line analysis of groups of similar users (García et al. 2008; Kardan and Conati 2013; Perera et al. 2009; Shih et al. 2010). For instance, Perera et al. (2009) applied clustering to find groups of students in a collaborative software development tool that revealed teams with good or bad collaboration skills. Shih et al. (2010) used clustering to discover common student learning tactics in a geometry tutor. García et al. (2008) applied clustering and sequential pattern mining to recognize how students navigate through a web-based learning environment. Eagle and Barnes (2014) applied a different data mining technique based on network analysis to discover the different behaviors of two groups of students as they explore possible solution spaces in an ELE aimed at teaching propositional logic concepts. (Kinnebrew et al. 2013; Kinnebrew and Biswas 2012) used differential sequence mining methods to compare productive and unproductive learning behaviors of students when using Betty’s Brain, an environment designed to foster self-regulated learning by allowing students to teach the relevant aspects of this meta-cognitive ability to an artificial

student. As with some of the work discussed above, our work applies clustering to identify different types of learners, but takes the additional step of leveraging clusters to build predictive models of student learning and form the basis for providing adaptive support to learners.

One of the main challenges of providing adaptive support in ELEs is the creation of effective student models without having a strong theoretical and practical knowledge of what constitutes productive online behaviors. Both EXpresser, an ELE for learning algebra (Noss et al. 2012), and ACE, an interactive simulation for mathematical functions, provide feedback based on a set of strategies defined by experts. Experts were also used in (Biswas et al. 2005) to define complex concept maps that could be leveraged to provide scaffolding to students using Betty's Brain. However, relying on expert knowledge is time consuming or even unfeasible for complex ELEs that support a large variety of exploratory behaviors. An approach that is in between relying on experts to identify which behaviors to capture in a student model and a purely data-driven approach is engineering the data features used for the student model. For example, in the context of predicting students' inquiry behaviors with the aforementioned Science Assessments Phase Change Microworld, Sao Pedro et al. (2012) compared two feature sets for prediction. The first set consisted of features based solely on interaction data (e.g., number of hypothesis made). The second set included both features engineered based on theory, as well as a subset of the features from the first set selected by experts based on perceived utility. The comparison revealed that the model with engineered features yielded better overall predictive performance than the non-engineered models. Similarly, Taylor et al. (2014) used crowd-sourcing to identify student actions predictive of MOOC dropout. Some of the features identified were directly present in the data (e.g., total number of forum posts) while others were engineered to capture more complex information such as temporal trends or comparison against other students. It was found that the most predictive features were the engineered ones. In this paper, we included some feature sets with feature engineering and compare them to models with features based solely on raw data to determine if feature engineering provided the same benefit others found, in relation to our comprehensive user modeling framework.

The CCK Simulation

The DC Circuit Construction Kit (CCK) simulation is part of PhET (Wieman et al. 2008), a freely-available and widely-used suite of simulations in different science and math topics. Simulations in the PhET family are being used over 45 M times a year, and CCK is the most popular simulation. CCK includes 124 different types of actions that allow learners to build DC circuits by connecting components (such as wires, light bulbs, resistors, batteries) and test them using measurement instruments such as voltmeters and ammeters (see tool bar in the screenshots in Fig. 1). The available actions include adding, moving, joining, splitting, and removing components, as well as changing the attributes of components (such as voltage and resistance). Additional actions relate to the interface (such as enabling different components) or the simulation itself (such as resetting the simulation). CCK provides animated responses with regard to the state of the circuits on the test bed. For example, when students add light bulbs to

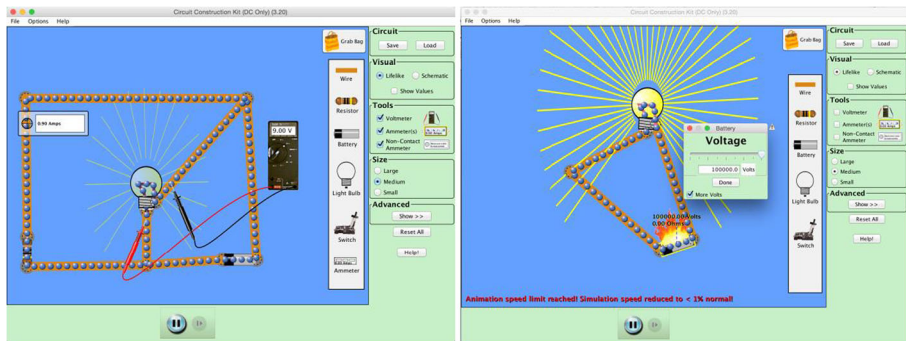


Fig. 1 Sample screenshots of the DC Circuit Construction Kit (CCK) showing the voltmeter (left), fires caused by high battery voltage (right), and changes in illumination intensity of a light bulb (comparing the two)

their circuit, changes in current through the light bulb can be visualized by the varying amount of illumination (as seen by comparing the screenshots in Fig. 1). In addition, the magnitude of the current is visualized by the speed of the “bubbles” of electrons (see blue dots inside the wires in Fig. 1). A faster speed implies that more current is flowing through that wire. If the amount of current through a part of the circuit ever gets too high, the speed of the electron bubbles will become very fast and a fire will start. Notably, CCK is a tool, not an activity. That is, instructors can use CCK with a variety of activities; most of them are given on paper outside the simulated environment.

Our long-term goal is to provide adaptive support for learners in CCK so that they can make optimal use of this simulation in their learning. To achieve that, the system should be able to assess the effectiveness of students’ behaviors and provide explicit support to foster learning. To do that, one should first identify what behaviors are typical of successful learners in this context. A second phase is to create a student model that is capable of differentiating between productive and unproductive behaviors. Last, the model should give suggestions that would help learners change unproductive behaviors as soon as the model identifies them. While eventually support may be productive across activities and simulations, we first address one typical activity in the CCK simulation.

User Study

Data used in this paper is derived from a laboratory user study on the CCK simulation described in (Kardan et al. 2014; Roll et al. 2014b). The overall goal of the study, which involved 100 first-year physics students, was to understand how students learn with the simulation across activities and across levels of scaffolding. Students were given two activities, each of which lasted about 25 min, and all of their actions during interaction were logged. In addition, students were assessed on their conceptual knowledge of circuits before and after both activities. The first activity, on the topic of light bulbs, had two different conditions of external scaffolding whereby half of the students were assigned to a high scaffolding condition and the other half to a low scaffolding condition.

During the first activity, students in both conditions worked with the simulation for roughly 25 min and had the same learning goal, to understand the effect of number and

arrangement of light bulbs on voltage and current. Students in the low scaffolding condition had no additional guidance. Students in the high scaffolding condition received a worksheet consisting of several pages of prompts, including diagrams of circuits to be tested, tables of measurements to be taken, and self-explanation and compare-and-contrast prompts. Notably, while the manipulation had effect on learning in the first activity, it had no impact on post-test scores for the second activity (Roll et al. 2014b). For our purpose, we treat the first activity as an unknown background. In fact, it allows us to see whether our modeling approach is applicable for students with unknown, diverse prior experiences.

The second activity, on the topic of resistors, was identical for all learners, and everyone received low scaffolding. These students received a learning goal, to “investigate how resistors affect the behaviors of circuits”, and three guiding recommendations: (1) to investigate what happens to the current and voltage when resistors with different resistances are used, (2) to investigate circuits that include multiple resistors with different resistances with a variety of arrangements, and (3) to explore the properties of different combinations of resistors with the same resistance. Students were asked to use CCK to help them explore the learning goal and these guiding recommendations. In this paper, we focus only on data from this second activity because it allows us to observe how all students in our study use the CCK tool when they are not guided by strict instructions or scaffolding, that is, in a more exploratory and self-guided manner. In addition, as this was their second activity with the simulation, all students were familiar with the CCK interface and affordances. Notably, students in the study had different backgrounds, based on their condition in the first activity. For example, students who were in the high scaffolding condition in the first activity may have been primed by the diagrams and tables that the other half of the students did not see. This allows us to test our model in an environment with a variety of prior experiences and exposures, as often is the case in educational settings. Indeed, our analysis was still able to pick out patterns of student behaviors related to learning gains and not dependent on prior experiences.

Three students with perfect pre-test scores were not included in our dataset. All remaining students scored below 90% on the pre-test. We also removed one additional student whose system malfunctioned during the study, causing logging errors. This left us with 96 students.

User Modeling Framework

As discussed in the [Introduction](#) section, in this paper we aim to investigate whether the user modeling framework for ELEs, first proposed in (Kardan and Conati 2011), can be used to create reliable student models and inform suitable adaptive feedback for an interactive simulation substantially more complex than those previously tested [e.g. (Eagle and Barnes 2014)]. FUMELE is summarized in Fig. 2, and consists of two main phases: *Behavior Discovery* and *User Classification*. In this section, we provide a high-level description of these phases sufficient for the purposes of this paper. A more detailed description can be found in (Kardan and Conati 2011).

In *Behavior Discovery*, each user’s interaction data is first pre-processed into feature vectors that summarize the relevant elements of the user’s behavior in the simulation,

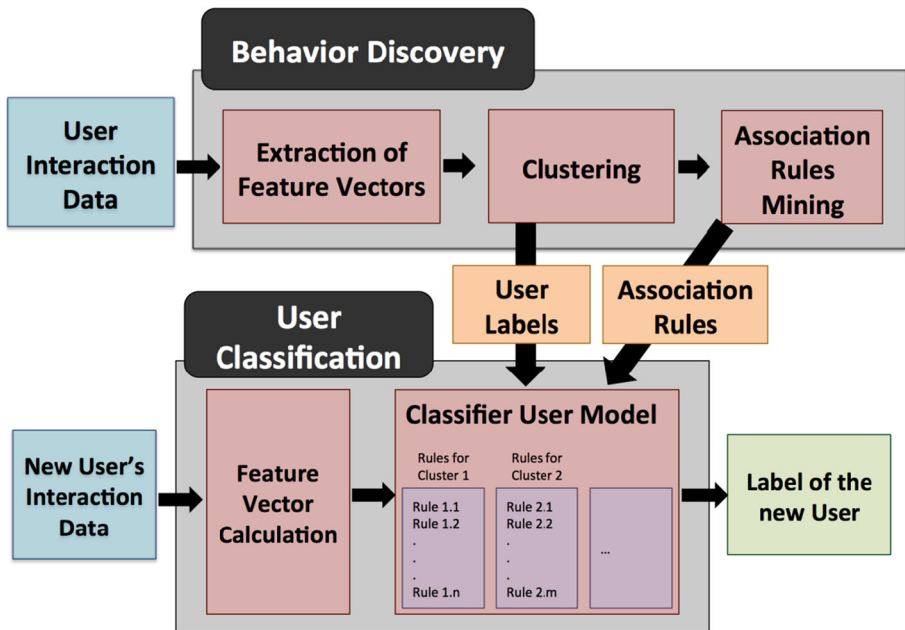


Fig. 2 The user modeling framework used in this paper, highlighting the two main phases of Behavior Discovery and User Classification. The input is user interaction data and the output is the label of the new users interacting with the system

one vector per student. Students are then clustered using these vectors so as to identify groups of learners with similar interaction behaviors. Clustering is done via the K-means algorithm. K-means was initially selected because it is a common and computationally simple clustering algorithm. Other clustering algorithms were experimented with in (Kardan and Conati 2011), including Hierarchical and Expectation Maximization, and none substantially outperformed K-means. Since K-means clustering is subject to converging to local maxima, we controlled the stability of the clustering algorithm in two ways. First, we used three measures that test for validity of the cluster to determine the optimal k : C-index, Calinski-Harabasz (Milligan and Cooper 1985), and Silhouettes (Rousseeuw 1987). We then choose the optimal number of clusters as the lowest k that one of these measures suggests. Second, we used Genetic Algorithm (GA) K-means (Krishna and Narasimha Murty 1999) when performing our clustering. This does an evolutionary search with many different starting centroids to minimize the chance that it converges to a local maximum. These were also the strategies applied in Kardan and Conati (2011), where this framework was first proposed.

Because we aim to adapt feedback based on anticipated learning gains, the clusters are valuable when they group students with similar learning gains within-clusters and different gains between-groups. This is evaluated using one-way ANCOVAs with post-test scores as dependent measures, clusters as factors, and pre-test as covariate to control for the variance in post-test scores explained by pre-test scores.

When significant learning differences between clusters are found, the next step is to identify the distinctive interaction behaviors in each cluster via association rule mining using Weka's Hotspot algorithm. This process generates association rules for each

cluster in the form of a tree and extracts the common behavior patterns in terms of class association rules in the form of $X \rightarrow c$, where X is a set of feature-value pairs and c is the predicted class label for the data points where X applies. For example, a pattern for low learners could be that they test with the voltmeter with frequency less than 0.075, meaning that, of all the actions they have done, less than 7.5% of them have been testing with the voltmeter. During the association rule mining process, the values of features are discretized into bins (Kardan and Conati 2011). The optimal number of bins for each feature is determined automatically based on estimated entropy of each feature.¹ In the case of the last example, the feature is divided into two bins, labeled as lowest and highest, where the lowest bin (0 to .075) is assigned to low learners and highest bin (.075 to 1) is assigned to high learners.

In *User Classification*, the labeled clusters and the corresponding association rules extracted in Behavior Discovery are used to train a classifier student model (see Classifier User Model box in Fig. 2). As new users interact with the system, they are classified, by this rule-based classifier, in real-time into one of the identified clusters. To do this, we calculate a membership score that summarizes how well the user's behaviors match the association rules for each cluster. The cluster with the largest percent of rules satisfied is the class that the new user is labeled as. Thus, in addition to classifying students in terms of learning, this phase returns the specific association rules describing the learner's behaviors that caused the classification. These behaviors can then be used to trigger real-time interventions designed to encourage productive behaviors and discourage detrimental ones, as described in (Kardan and Conati 2015).

It should be noted that, when learning gains are available, the traditional approach to building a classifier user model entails creating the training classes by dividing the range of learning gains into equal intervals (e.g. two intervals by using median split) and putting students within each interval into one class. How many classes to include would be an arbitrary choice, and data points at the boundaries are also forced into one class somewhat arbitrarily. Our approach selects the training classes based on the outcome of clustering on interaction behaviors, with the additional constraint that the clusters differ in terms of learning gains. Kardan and Conati (2011; Kardan et al. 2014) showed that this process results in training classes that generate classifiers with better classification accuracy than building training classes based on learning gains. This was the case for a battery of standard classification algorithms, and even more so for the rule-based classifier described above. This classifier, trained on clustering-based classes, reached the highest accuracy among all the other tested combinations of classifiers and training sets.

Representing the User Actions in CCK

Structured Representation of Action Events

Clustering students based on their actions is the first step of our user-modeling framework, and it requires a representation that captures important aspects of these actions. CCK logs three pieces of interaction information: the type of action (e.g.,

¹ The number of bins per feature is limited to a maximum that is set by cross validation.

connect), the component used (e.g., a wire), and the response of the physical model (e.g., current change). However, within CCK, outcomes of actions depend on their context. For example, connecting a wire may lead to different outcomes based on the state of the circuit. If the student attaches the wire to a dead circuit, there will be no outcome. If the student attaches the wire to a live circuit, and both ends of the wire are attached to different components, current will flow through the wire, assuming that there is no open switch in the circuit, etc. In addition, actions with one component often affect other components. For instance, changes to batteries can affect existing light bulbs and the readings on attached measuring instruments. Last, interpretation of actions depends on their context. For example, a user who connects one probe of a voltmeter, a testing instrument that measures voltage, is actively testing only if the other probe is connected as well.

As described in (Kardan et al. 2014), we created a structured representation that can capture these *action-events*, i.e., user actions and their relevant contextual information, at different levels of granularity. This representation contains four layers shown in Fig. 3 along with the elements they contain and their frequency of occurrence in our dataset: The “Actions” layer describes the types of actions available in CCK (e.g., *add*) and includes 25 different action types. “Components” describes components available to be manipulated by actions (e.g., *wire*) and includes 22 items. “Outcomes” capture what happens in the circuit after an action is performed. There are 6 types of outcomes, including: *None*, *Deliberate-measure* (the value displayed on a measurement device is updated as a result of using it), *Current-change* (a change in the current occurred, reflected in the speed of movement of electrons), and *Light-intensity-change* (the brightness of a light bulb changes). It should be noted that an action may cause more

Outcomes (6)	Frequency
None	0.6986
deliberate_measure	0.1303
reading_updated	0.0832
current_change	0.0537
light_intensity	0.0267
fire_started	0.0074

Family (8)	Frequency
Test	0.3369
Build	0.2422
Organize	0.1791
Revise	0.1583
Interface	0.0371
Pause	0.0334
Extra	0.0102
Reset	0.0029

Action (25)	Frequency
organizeWorkspace	0.1803
endMeasure	0.1591
startMeasure	0.1550
join	0.1313
add	0.1052
changeResistance	0.0733
pause	0.0334
remove	0.0328
joinX	0.0265
split	0.0207
view	0.0195
traceMeasure	0.0180
sliderEndDrag	0.0123
enableComponent	0.0116
disableComponent	0.0054
playPause	0.0048
changeVoltage	0.0046
reset	0.0029
switch	0.0014
reverse	0.0009
moreVoltsOption	0.0004
exitSim	0.0003
help	0.0002
deiconified	0.0001
iconified	0.0001

Component (22)	Frequency
nonContactAmmeter	0.2109
wire	0.2090
resistor	0.1705
junction	0.1472
voltmeter	0.1332
battery	0.0342
pause	0.0334
lightBulb	0.0149
showValues	0.0135
seriesAmmeter	0.0081
grabBagResistor	0.0075
zoom	0.0034
circuitSwitch	0.0033
stepButton	0.0030
resetAll	0.0029
schematic	0.0022
playPause	0.0018
hideElectrons	0.0004
exitSim	0.0003
phetFrame	0.0002
help	0.0001
hide	0.0001

Fig. 3 The four layers of action-events: outcome, family, action, and component along with the elements they contain and the frequency of each

than one outcome. For example, changing the battery voltage may affect light intensity, current, and measured voltage. Our last layer, “Family”, denotes one of 8 general types of action, and its purpose is to abstract actions into more general activities that students can perform in CCK. For example, the *Test* family includes different ways to perform active measurements of the circuit using the available measurement instruments. It includes both testing a single component (“test”), as well as tracing a section of the circuit along one loop using a testing instrument (“trace measure”). The *Build* family captures all actions performed to create a new circuit, before the circuit is live (e.g., adding wires, joining light bulbs, removing resistors), whereas the *Revise* family includes actions performed to modify an existing circuit, after the circuit has gone live. Thus, the same action can belong to a different family depending upon the context in which it was performed. The *Pause* family was added to have a coarse-grained representation of the amount of time a student spends in between actions, as an indication that they might be pausing to plan or reflect on their actions (we will see in a later section that we also represent finer-grained time information as the average time that students spend between specific types of actions). In its current incarnation, *Pause* includes only one event, defined as inactivity for longer than 15 s, but more events representing different extents of pausing could be included once we have a better sense of the value of this Family. We chose the 15 s threshold because, when plotting frequency of pause-lengths, 15 s marked the beginning of the long tail. That is, around this value, the rate of pauses is fairly insensitive to the specific value of this parameter.

It should be noted that, while the Actions, Components, and Outcomes layers are logged directly by the simulation, the Family layer was defined by the authors via extensive discussion about which actions are important to abstract to this level and how. The goal of adding the family layer is to ascertain whether modeling learning with CCK requires capturing interaction behaviors in terms of the specific actions and components, or whether learning depends mainly on the type of activity being performed. For example, does it matter how students test, or only that they test? Similarly, when building circuits, does information regarding specific actions such as add and attach matter, or is only the general “build” characteristic that informs learning? Table 1 shows the complete mapping of Actions to Families.

This structured representation adds contextual information to the data. For example the action-event `current_change.revise.join.wire` describes joining (action) a wire (component) that led to a current-change (outcome) when revising a circuit (family). If a light bulb were also connected properly to the circuit, the action-event `light_intensity.revise.join.wire` would also occur at the same time, describing a second outcome of joining the wire, namely a change in the luminosity of the light bulb. Figure 3 shows the frequency of each element in the action-event representation.

While in (Kardan et al. 2014) all 4 layers of the structure were used to represent actions-events, subsets of the layers can represent events at different levels of granularity. For example, just using the action layer would provide less information about what behavior the student exhibited than if both the action and outcome layers were used. Each additional layer gives a more complete picture of the student’s behavior. The different combinations of layers also give rise to feature sets with different amounts of feature engineering. Specifically, as the *family* layer was defined by the authors, it is a completely engineered level. The *action* and *component* layers, on the other hand,

Table 1 Summary of all families and the actions that comprise them

Family	Abstracted Actions
Build	Add changeResistance changeVoltage join joinX reverse sliderEndDrag switch
Revise	add changeResistance changeVoltage join joinX remove reverse sliderEndDrag split switch
Extra	add changeResistance join joinX moreVoltsOption organizeWorkspace reverse sliderEndDrag
Organize	organizeWorkspace
Test	endMeasure playPause startMeasure traceMeasure
Interface	deiconified disableComponent enableComponent exitSim help iconified view
Reset	reset
Pause	pause

require only a minimal level of engineering, as they are present in the logs. Manipulations that were engineered for these layers included the addition of two actions, *joinX* and *traceMeasure*. *JoinX* was introduced as a finer resolution of joining component. In the logs joining any number of components is logged as “*join*”. Thus, complex circuits will have a similar sequence of actions to that of simple single-loop circuits. However, we wanted to be able to distinguish the joining of just 2 components to create a single line and the connection of junctions that create more than one loop in a single circuit, hence, more complex circuits. We introduced *joinX* to serve this purpose, and called all *join* actions that involved more than 2 components *joinX*. *TraceMeasure* was

introduced to allow us to distinguish different types of testing, as described above. Students either used their testing instruments to examine one location on the circuit at a time or moved the testing instrument over many parts of the circuit (“tracing” over many components). The latter action was defined as *traceMeasure*.

The *outcome* layer required more modifications, such as adding the additional outcomes *reading updated*, *deliberate measure*, and *light intensity*. The only outcomes logged in the log files were *current change*, *fire started*, and whether there was a change in the reading on a testing instrument. We defined *light intensity* to represent the change in light bulb luminosity that occurs when there was a current change and a live light bulb in the circuit. We felt it was important to capture this outcome as many students alluded to observing the light bulbs to gauge changes to their circuits. Another distinction was added to testing-related outcomes. We wanted to create a distinction between changes in measurements that occurred while the instrument was purposely used (e.g., making an active measurement) vs. changes that occurred as a side effect of manipulating another component while (e.g., adding a battery while a testing instrument was connected to the circuit). Thus, we introduced *deliberate measure* as the outcome of actively using testing instrument, and *reading updated* as the side effect outcome of testing instruments due to other components). An additional level of engineering at the *outcome* layer relates to the fact that outcomes had to be associated with the relevant action-event during post-processing, as they are logged as independent system-event.

It should be noted that all forms of feature engineering described above are relatively lightweight compared to more extensive engineering efforts described in other work (eg, (Roll et al. 2014a; Sao Pedro et al. 2011; Taylor et al. 2014)). Still, they go beyond using log files as is, as was done in the previous application of our user-modeling framework (Kardan and Conati 2013).

Generating Feature Sets for the Student-Modeling Framework

Each representation at the different levels of granularity can be used to generate different feature sets, based on the information and calculation used to summarize the action-events performed by a user. These summative measures include:

- *Frequency* of the action-event, i.e., the proportion of each type of action-event over the total number of action-events
- Timing information in terms of *mean* and *standard deviation* of the *time* spent between each type of action-event and its preceding event

The timing information gives an indication of the time it took for the student to plan and carry out each action event. When students took pauses (i.e. let more than 15 s pass in between two consecutive action events), we kept their average time for that action, and treated the rest of the inactive period as a pause. For example, if a student took 21 s to attach a certain wire (longer than 15 s, hence including a pause), and their mean time before attaching wires is

2 s, then we relabeled 19 s as “pause”, and kept 2 s (their mean) for attaching the wire.

In (Kardan et al. 2014), we described the performance of a feature set built on all 4 layers in Fig. 3 and on all three summative measures above (frequency, mean time, and standard deviation of time), when used to cluster students who learn similarly with CCK. Here, we generated feature sets that use different subsets of layers in the action-event structure, in order to investigate the effect of representation granularity on both generating meaningful clusters, as well as on building effective user models and informing feedback that can improve student learning from CCK, as in (Kardan and Conati 2015). We investigated 11 of the possible 15 combinations of layers. For each representation, we also experimented with using only frequencies vs. adding time-related summative measures, for a total of 22 different feature sets.

We also generated an additional representation that does not have any feature engineering and strictly uses only information present in the log files. This representation is similar to the one that only included the *outcome*, *action*, and *component* layers in that it did not include the *family* layer, but the feature engineered actions and outcomes were also removed so that the action-events present are only the ones that exist in the log files. In addition, this representation no longer includes the “pause” actions as the authors defined the length of those. For this representation, as was done with the others, we experimented with using only frequencies vs. adding time-related summative measures. This gave us an additional 2 feature sets. All feature sets investigated can be seen in Table 2 which, for each feature set, lists whether it was engineered or not, the representation layers included, the summative measures leveraged, and the overall number of features present.

Selecting Relevant Representations

As previously mentioned, the first step in the user modeling framework is to see if we are able to identify groups of students that differ in terms of the amount they learned while working with the simulation. For each of the 25 feature sets described above (and listed in Table 2), we first clustered them using K-means clustering. We then looked at if there was a statistically significant difference in learning gain between the clusters, by running one-way ANCOVAs on the post-test scores in each cluster, controlling for pre-test scores. Only 4 feature sets generated clusters of students with statistically different learning gains. Feature sets with only frequency as a summative measure are denoted *_f* while feature sets that include frequency, mean time, and standard deviation of time information are denoted *_fms*. The four feature sets are:

- 1) FOAC_*f*: Set including all action-events elements (Family, Outcome, Action, Component) with frequency information (211 features)
- 2) FAC_*f*: Same as the first feature set, but without the Outcome layer (99 features)
- 3) OAC_*f*: Same as the first feature set, but without the Family layer (207 features)

Table 2 Summary of all feature sets explored. Bolded ones are those giving useful clusters, and thus compared further in this paper

Event type	Statistical measures	Engineered	Number of features
Action	Frequency	Minimally	25
Action	Frequency, Mean Time, Std. Dev. Time	Minimally	75
Family	Frequency	Yes	8
Family	Frequency, Mean Time, Std. Dev. Time	Yes	24
Outcome	Frequency	Minimally	6
Outcome	Frequency, Mean Time, Std. Dev. Time	Minimally	18
Action, Component	Frequency	Minimally	76
Action, Component	Frequency, Mean Time, Std. Dev. Time	Minimally	228
Family, Component	Frequency	Yes	44
Family, Component	Frequency, Mean Time, Std. Dev. Time	Yes	132
Outcome, Family	Frequency	Yes	21
Outcome, Family	Frequency, Mean Time, Std. Dev. Time	Yes	63
Family, Outcome, Component	Frequency	Yes	99
Family, Outcome, Component	Frequency, Mean Time, Std. Dev. Time	Yes	297
Family, Action, Component	Frequency	Yes	99
Family, Action, Component	Frequency, Mean Time, Std. Dev. Time	Yes	297
Outcome, Action, Component	Frequency	Minimally	207
Outcome, Action, Component	Frequency, Mean Time, Std. Dev. Time	Minimally	621
Outcome, Family, Action	Frequency	Yes	102
Outcome, Family, Action	Frequency, Mean Time, Std. Dev. Time	Yes	306
Outcome, Family, Action, Component	Frequency	Yes	211
Outcome, Family, Action, Component	Frequency, Mean Time, Std. Dev. Time	Yes	633
Outcome, Action, Component	Frequency	No	203
Outcome, Action, Component	Frequency, Mean Time, Std. Dev. Time	No	609

- 4) *Raw_OAC_fms*. Same as the third feature set, but with no engineered features, as described above, and with time-related summative measures in addition to frequency (609 features: 203 action-events \times 3 types of summative measures)

Interestingly, the three feature sets that have some amount of feature engineering include only information on action frequency, indicating that summative statistics capturing how much time students spend before actions are not contributing to identify different learning outcomes. This can be explained by the fact that we capture significant inactivity before actions via *pauses*. The *Raw_OAC_fms* feature set, on the other hand, does not have pause information, and thus requires the timing information to form meaningful clusters.

While we had defined the family layer to abstract the action and component layers, we were not able to find a significant feature set that included the family layer without the action and component layers. This suggests that, in this case, the specific actions taken by the users provide us with additional distinguishing information on top of the

abstracted family layer. The fact that the abstracted layer is also included suggests that both the detailed account of students' actions, as well as the abstract level, convey different information regarding students' learning.

Evaluating Representations for Assessment and Support

In the previous section we identified four feature sets that allow our student-modeling framework to identify students that behave and learn differently with CCK. In the rest of this paper, we focus only on these four sets. Next, we use FUMELE to evaluate the suitability of each set for modeling students and providing adaptive support with CCK along these three measures: (i) Quality of the generated clusters, measured by effect size of difference in learning performance between students in the different clusters. That is, how informative are the resulting clusters with regard to learning gains? (ii) Classification accuracy of user models trained on the obtained clusters evaluated both at the end of interaction and over time throughout interaction. That is, how accurately can the model assign students to their respective cluster, and how quickly can it do it during interaction? (iii) Usefulness of the generated association rules in identifying behavior patterns that can be used to design and trigger support to students. That is, do the clusters provide information to inform the provision of relevant feedback for users who need it?

Quality of the Clusters

Table 3 shows the outcome of clustering on the four feature sets remaining in our analysis. Each cluster has one row per cluster, as defined by the optimal number of clusters identified. Each cluster is labeled based on the learning performance of the users in that cluster. Pre- and post-test scores are reported in the fourth and fifth columns in Table 3, respectively. In general, there was significant learning during the study, measured as the difference between pre and post test scores: $t(96) = 6.1$, $p < 0.0005$, Cohen's d is 0.74 (Mean (SD) for pretest and posttest were 0.47 (0.17) and 0.62 (0.23), respectively) (Roll et al. 2014b). The third column reports the cluster

Table 3 Summary statistics for the clustering results

Feature set	Cluster label	#Members	Average post-test score (SD)	Average pre-test score (SD)	p -value	Effect size (partial eta squared)
FOAC_f	High	65	.609 (.190)	.465 (.169)	.013	.065
	Low	31	.511 (.170)	.470 (.181)		
FAC_f	High	67	.596 (.197)	.455 (.166)	.048	.041
	Low	29	.534 (.163)	.494 (.184)		
OAC_f	High	66	.609 (.189)	.463 (.168)	.007	.076
	Low	30	.509 (.171)	.475 (.182)		
Raw_OAC_fms	Very high	3	.840 (.117)	.513 (.197)	.003	.122
	High	67	.595 (.177)	.475 (.170)		
	Low	22	.489 (.162)	.445 (.182)		

size. Clusters that contained only one member (singletons) resulted in that member being removed as an outlier and the clustering algorithm rerun. This impacted only the *Raw_OAC_fms* feature set, reducing the number of students from 96 to 92. The last two columns report the *p*-value and effect size of the difference in learning performance among the clusters found for each feature set, obtained via an ANCOVA on the post-test scores, controlling for pre-test. Thus, a larger effect-size suggests a representation that better separates students with different learning levels.

All feature sets except for *Raw_OAC_fms* generated two clusters, identifying groups of students with high vs. low learning. *Raw_OAC_fms* generated 3 clusters. Post-hoc pairwise comparison with Holm-Bonferroni corrections showed that all three clusters had statistically different learning performances from each of the other two.

Effect sizes of the difference in learning performance varied for different feature sets. *Raw_OAC_fms* achieves the best score with a medium effect size of 0.122. Its better performance is probably due to having three clusters and larger learning differences between the spectrum of clusters. Of the feature sets with 2 clusters, effect size ranges from small effect size (.041 for *FAC_f*) to medium-small effect (0.076 and 0.065 for *OAC_f* and *FOAC_f* respectively). The fact that the feature set that does not include outcomes (*FAC_f*) performs worse than the other three suggests that outcome may be an important separator between high and low learners. Interestingly, the addition of Outcome to a feature set is most effective (i.e. achieves the highest effect size) when information on Family is not present (see *OAC_f* in the table). This suggests that the addition of slightly more feature-engineered information (the Family) reduced the ability of information on action outcomes to detect differences in learning.

It should be noted that one of the clusters in *Raw_OAC_fms*, the one corresponding to a very high level of learning, included only 3 students. Thus, we will only focus on the two larger clusters in *Raw_OAC_fms* for the rest of our analysis. To ensure that the clustered groups remained significantly different, we reran the ANCOVA with only the two groups. The differences in learning between the two clusters remained significant: $F(1) = 4.730, p = .032, \text{partial } \eta^2 = .052$.

Classification Accuracy

Here we evaluate the classification accuracy of user models trained on each set of clusters generated by different feature sets. Namely, we evaluate how accurate each model is in assigning each student to his respective cluster (e.g. High vs. Low learning). Classification accuracy was evaluated in two ways. First, we looked at just the accuracy at the end of interaction, after all interaction data had been seen for a new user. This accuracy can be relevant to ascertain the feasibility of a form of “retroactive feedback” that gives students advice on how to improve their interaction with CCK the next time they use it based on their performance at the end of the last session they had with the system. We also use this accuracy to weed out any classifiers that did not even perform well after seeing all the data. For the ones that did, we then evaluated classification accuracy over time, focusing on ascertaining at which point *during* a student’s interaction with CCK the classifier can reliably start to classify students. This is crucial to understand how quickly an adaptive version of CCK could begin to provide hints to the user in real-time while they are working with CCK. We’ll first look at classification accuracy at end of interaction, then classification accuracy over time.

Classification Accuracy at End of Interaction

For each of the four feature sets, a classifier user model is trained on the generated clusters, using 8-fold nested cross validation to set the model's parameters and find its cross-validated accuracy at the end of the interaction (i.e., the classifier is trained on the complete interaction logs). 8-fold cross-validation was chosen so that we would have even sized folds for our 96 users. We performed nested cross-validation as it is a standard machine learning technique that involves running sub-cycles (inner levels) of cross validation within main cross-validation cycles (outer levels) to set model parameters while avoiding contaminating the test set. Thus, model fitting is performed in the inner cross-validation cycles, and then model accuracy is assessed in the outer cross validation cycles (Conati et al. 2015). The parameters we fit for our classifiers in the inner cross-validation cycle relate, for instance, to thresholds defining when new association rules were generated in clusters, such as the minimum required rule support (percentage of users in the cluster that the rule must describe).

Table 4 reports the classification performance of each classifier in terms of overall accuracy, class accuracy for high and low learners, and kappa scores. The table also reports, for each classifier, the accuracy of a baseline majority classifier that predicts the accuracy of assigning all students to the largest cluster found with the corresponding feature set in the previous evaluation phase (see Table 3). Since, as Table 3, shows, the size of the larger cluster changes with feature set, the accuracy of the majority classifier for each feature set is also different. We use Kappa scores as one of our performance measures because they account for agreement by chance, thus providing a fair comparison among classifiers with different baselines. At the same time, Kappa scores are hard to interpret in terms of a model's practical effectiveness, whereas overall accuracy is more informative in this respect. Thus, we also discuss classifiers performance in terms of whether their overall accuracy is statistically better than each corresponding baseline. In order to do so, one-sample t-tests were conducted to compare classifier accuracy and baseline accuracy for each feature set (4 total). The overall accuracies for three of the action-event sets, *FOAC_f*, *FAC_f*, and *OAC_f* are significantly above their respective baselines, with kappa scores values ranging from 0.564 to 0.702, indicating that our user-modeling framework can effectively classify students at the end of their interaction with CCK with three of the feature-sets. The *Raw_OAC_fms* feature set, on the other hand, does not significantly beat its baseline, has the lowest overall accuracy, the lowest kappa-score, and extremely imbalanced classes. It is also worse than the *OAC_f* feature set on all measures, signifying that even some minimal feature

Table 4 Classifier accuracy measures for different feature sets. Baseline is the accuracy of the most likely classifier

Feature set	Baseline %	Overall accuracy % (Std. dev.)	High learner class accuracy %	Low learner class accuracy %	Kappa
<i>FOAC_f</i>	65.3	86.5 (8.8)	91.8	77.1	.702
<i>FAC_f</i>	69.8	83.3 (5.9)	85.1	72.4	.564
<i>OAC_f</i>	68.8	84.4 (9.4)	90.9	70.0	.626
<i>Raw_OAC_fms</i>	75.3	79.8 (10.6)	88.1	54.5	.439

engineering is helpful for more accurately classifying students. Because *Raw_OAC_fms* does not beat its baseline after all data has been observed, we exclude it from all remaining evaluation, as it cannot be used to classify students better than chance.

The feature set based on the most detailed representation, *FOAC_f*, is superior to the other 3 sets in terms of accuracy at the end of the interaction. In particular, although its overall accuracy, at 86.5% is higher but comparable to *FAC_f* and *OAC_f*, its kappa-score reaches 0.7 compared to the second best value of 0.626 for *OAC_f*, and it is better than *OAC_f* at classifying low learners (77% vs. 70% accuracy for this class), with on-par performance for high learners.² This indicates that the additional level of representation added by Family is beneficial for classifier accuracy even though all the information is already captured in more granular features (action, outcome, component). Also, of the three feature-engineered feature sets, the two that include Outcome show higher accuracy compared with *FAC_f*, suggesting that the outcome of students' actions, rather than the actions themselves, are most beneficial to identify low vs. high learners.

High classification accuracy is critical to providing effective adaptive support to students. Ideally, we would only give interventions to low learning students, so as to be giving help to students who need it without interrupting students who don't. In general, all our classifiers have better accuracy at classifying high learners than low learners, mostly due to the fact that there are more high learners in the training data. Looking at false-positive and negative rates in the confusion matrices for our classifiers, in the case of the *FOAC_f* classifier for instance (Table 5), only 10.4% of all students were clustered as high learners but classified as low learners. These are the students that will be receiving interventions that they do not need, potentially being distracted or confused by this. On the other hand, only 13.5% of all students were low learners but not classified as such. Thus, the system will miss the chance to help these students when they need it. Still, even for low learners, the percentage of misclassified students is rather low. The confusion matrices of the other two classifiers that significantly beat baseline are similar, with low learners misclassified as high learners ranging from 8.3 to 13.5% of students, whereas high learners misclassified as low learners make up only 6.3 to 10.4% of the students.

Classification Accuracy over Time

The accuracies reported in the previous section relate to the performance of the classifier after seeing all the data for one student at the end of one session with CCK. While this information could still be leveraged for providing the student with adaptive summative feedback, or for personalized instructions the next time a student uses the system, we also want to know whether our classifiers can be used for providing adaptive interventions *during* a specific session of CCK. Thus, for each of the three feature sets that beat the baseline at the end of interaction, we calculated accuracy over time. Namely, for each feature set we trained and evaluated different classifiers on

² We don't perform a formal statistical analysis in this section comparing the accuracies of our classifiers against each other because that will be done as part of the subsequent analysis of accuracies over the whole course of the interaction.

Table 5 Confusion matrix for the *FOAC_f* feature set. Percentages displayed are percent of students in that category out of all students

		<i>FOAC_f</i>	
		Classified class	
		HL	LL
Clustered class	HL	55 (57.3%)	10 (10.4%)
	LL	13 (13.5%)	18 (18.8%)

incremental portions of interaction data (time slices), effectively simulating what happens when a new user gets classified in real-time while using CCK. We calculated the data points at cumulative 10% time slices, each slice including ~2.5 min more interaction data than the time slice before it. For example, the 50% time slice includes all actions taken up to half of the way through the interaction (~12.5 min). We used percent of interaction observed instead of minutes of interaction for ease of presentation, given that each student worked with the PhET simulation for different amounts of time (average length = 24.7 min, standard deviation = 4.3 min). The analysis of computed over time accuracies that we present in the rest of this section aims to answer the following questions:

- 1) Which classifiers significantly outperform their corresponding baselines in terms of accuracy over time?
- 2) When do the classifiers begin to significantly outperform the baseline?
- 3) Which classifier has the best over time performance?

The over time accuracy of each of the *FOAC_f*, *OAC_f* and *FAC_f* feature sets are plotted in Fig. 4.

To assess which of the three classifiers outperforms its baseline (not reported in Fig. 4), we first ran a 2-way repeated measures ANOVA for each classifier, with classification accuracy as the dependent measure. The factors in the ANOVAs were *classifier type* (with 2 levels: baseline or classifier) and *time slice* (with 10 levels: one level for each 10% of interaction data observed). This gives us 20 cells (2 classifier types × 10 time slices). Naturally, for each feature set, the baseline vector of 10 time slices was a flat line, as the baseline assigns all students to majority cluster regardless of task phase. 8-fold cross-validation was again used. For the ANOVAs, we chose to use the accuracies from 10 runs of the cross-validation instead of the accuracies of each fold in one run of the cross-validation so that we could get a more accurate standard deviation between the data points. There was high variation amongst the accuracies of the folds due to uneven splits of user types into different folds. For each of the three ANOVAs, Mauchly's Test of Sphericity indicated that the assumption of sphericity had been violated. Mauchly's Test of Sphericity determines if the variances of the differences across each pair of time slices in the ANOVA are uniform. Since this test was violated, and the variances were not uniform, we applied a standard Greenhouse-Geisser correction. This decreases the degrees of freedom in the ANOVA to counteract the inflated F-statistic that results when sphericity is violated.

We found a significant effect for classifier type for each of the three feature sets: $FOAC_f$ ($F(1,9) = 864.794, p < .001, \eta^2 = .990$), FAC_f ($F(1,9) = 14.192, p < .01, \eta^2 = .612$), and OAC_f ($F(1,9) = 550.159, p < .001, \eta^2 = .984$). This indicates that each classifier outperforms its baseline on average in terms of accuracy over time (the means for each are shown in Table 6).

There was also a significant effect for time slice for each of the three feature sets: $FOAC_f$ ($F(3.833, 34.494) = 82.579, p < .001, \eta^2 = .902$), FAC_f ($F(4.715, 42.438) = 60.117, p < .001, \eta^2 = .870$), and OAC_f ($F(4.206, 37.855) = 56.045, p < .001, \eta^2 = .862$). This indicates, not surprisingly, that classifier accuracy is significantly improved as the classifier sees more data from the user.

However, we also found a significant interaction between classifier type and time slice for each of the three feature sets: $FOAC_f$ ($F(3.833, 34.494) = 82.579, p < .001, \eta^2 = .902$), FAC_f ($F(4.715, 42.438) = 60.117, p < .001, \eta^2 = .870$), and OAC_f ($F(4.206, 37.855) = 56.045, p < .001, \eta^2 = .862$). This indicates that the performance of each classifier compared to its baseline differed as a function of time. To determine at which time points each classifier beats its baseline, a t-test was run for each of the time slices for each of the classifiers (30 total t-tests). Table 7 shows the outcome of the t-tests. Reported effects are significant at the $p < .05$ level after all tests are corrected for multiple comparisons using the Holm-Bonferroni correction. We see that FAC_f does not beat the baseline until 70% of interaction has been observed. However, OAC_f and $FOAC_f$ are able to consistently beat their baselines after observing only 30 and 40% of data, respectively. In addition, they are never statistically worse than the baseline implying that they could be used at any point during interaction and do no worse than baseline at classifying the students. These results suggest that Outcome adds additional information allowing for better early classification of the students.

It is important to know whether a classifier can beat its baseline to understand if the classifier has any predictive value at all. However, beating the baseline does not say much about the value of the classifier in practice, especially if, like in our case, one wants to compare classifiers with different baselines that might be easier or more difficult to beat. To assess which of the 3 classifiers performs best in terms of sheer value of accuracy over time, we ran a 2-way repeated measures ANOVA with classification accuracy as the dependent measure, and factors consisting of *classifier* (with 3 levels: $FOAC_f$, FAC_f , and OAC_f) and *time slice* (with 10 levels: one level for each 10% of interaction data observed).

There was a significant effect for classifier type ($F(2,18) = 78.513, p < .001, \eta^2 = .897$) and for time slice ($F(9,81) = 164.094, p < .001, \eta^2 = .948$). There was also a significant interaction between classifier type and time slice ($F(18,162) = 11.632, p < .001, \eta^2 = .564$). The post-hoc analysis for the classifier type main effect shows that

Table 6 Average accuracies for each classifier and its baseline

Feature set	Average classifier accuracy over time %	Baseline accuracy %
$FOAC_f$	73.9	63.5
FAC_f	71.0	69.8
OAC_f	76.5	68.8

Table 7 Classifier accuracy over time for each of our three feature sets. Bolded values highlight the time Slices where the classifier is significantly beating the baseline

Time slice	FOAC_f						OAC_f							
	FAC_f			FAC_f			FAC_f			FAC_f				
	Classifier accuracy % (SD)	Baseline	t-value	p-value	Time slice	Classifier accuracy % (SD)	Baseline	t-value	p-value	Time slice	Classifier accuracy % (SD)	Baseline	t-value	p-value
10%	63.9 (1.5)	63.5	.758	.468	10%	64.3 (3.5)	69.8	-4.984	.001	10%	68.4 (1.0)	68.8	-1.160	.276
20%	68.0 (4.0)	63.5	3.582	.006	20%	59.8 (4.1)	69.8	-7.805	.000	20%	70.4 (1.8)	68.8	2.866	.019
30%	67.0 (4.3)	63.5	2.536	.032	30%	69.6 (2.9)	69.8	-0.240	.816	30%	78.3 (2.4)	68.8	12.328	.000
40%	68.3 (2.4)	63.5	6.462	.000	40%	64.4 (2.1)	69.8	-8.057	.000	40%	72.3 (3.1)	68.8	3.591	.006
50%	69.1 (2.1)	63.5	8.432	.000	50%	69.2 (2.2)	69.8	-0.931	.376	50%	77.2 (2.8)	68.8	9.632	.000
60%	73.5 (2.6)	63.5	12.399	.000	60%	69.1 (3.3)	69.8	-0.708	.497	60%	73.6 (1.7)	68.8	8.990	.000
70%	79.8 (1.9)	63.5	27.842	.000	70%	76.4 (1.9)	69.8	10.880	.000	70%	83.9 (2.9)	68.8	16.329	.000
80%	81.9 (2.2)	63.5	27.006	.000	80%	75.8 (2.5)	69.8	7.649	.000	80%	77.1 (2.6)	68.8	10.266	.000
90%	81.9 (2.4)	63.5	24.567	.000	90%	79.8 (2.9)	69.8	10.845	.000	90%	79.9 (2.4)	68.8	14.571	.000
100%	85.2 (3.0)	63.5	22.738	.000	100%	81.9 (3.7)	69.8	10.373	.000	100%	84.1 (2.4)	68.8	20.472	.000

all three classifiers are statistically different from each other at the $p < .01$ level (after all tests are corrected for multiple comparisons using the Holm-Bonferroni correction). The ordering, from best to worst feature sets, in terms of over time classifier accuracy is: *OAC_f*, *FOAC_f*, *FAC_f*. This may seem to contradict what we found in the last section, where we had seen that *FOAC_f* was the classifier that was able to classify students with the highest accuracy. However, *OAC_f* is able to perform better over time because it outperforms *FOAC_f* in many of the earlier time slices. Figure 4 and Table 7 show that *OAC_f* is the classifier that be used the earliest. Starting after only 20% of the interaction has been observed (roughly 5 min in), *OAC_f* is able to correctly classify at least 70% of the students. The other two feature sets are not able to achieve this high of accuracy until about 60% (*FOAC_f*) and 70% (*FAC_f*) of the interaction has been observed.

Usefulness for Informing Adaptive Support

In this section we focus on the third criterion we leverage to evaluate our feature sets, namely, what is the usefulness of the association rules uncovered in the corresponding clusters for informing the design of adaptive support to students. Association rules identify behavioral patterns that are representative of what students in a given cluster do with CCK (see (Kardan and Conati 2013) for a discussion of how patterns are derived from rules). These patterns are useful if they are associated with low (or high) learning performance that can inform adaptive interventions. Specifically, if a student is classified as a “Low Learner” (LL) at any given point of working with CCK, adaptive interventions can be provided to discourage the LL patterns she is showing and to encourage the HL patterns she is not showing. We discuss usefulness of association rules in terms of number/meaningfulness of the patterns identified, as well as their potential for informing the design of adaptive hints.

Patterns Identified

The number of identified patterns varies to some degree among feature sets, ranging from 15 in *OAC_f* to 17 in *FAC_f* to 23 in *FOAC_f*, showing that the most complex representation captures finer grained variations in learner behaviors. With more distinct

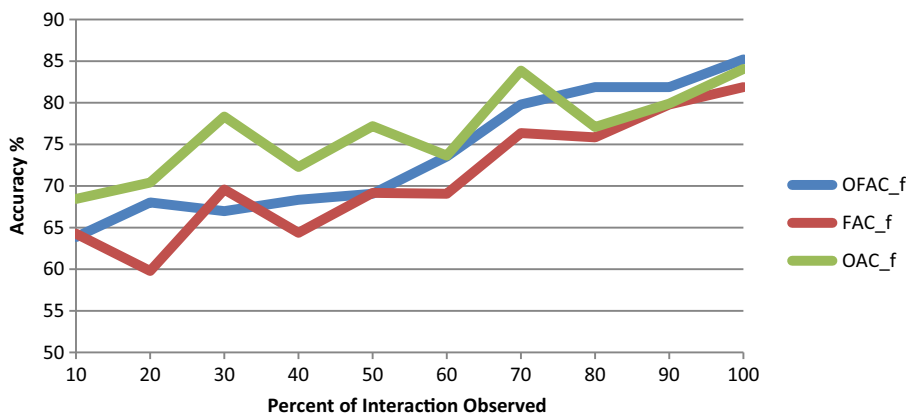


Fig. 4 Over time performance of each of the three feature set's rule based classifiers

patterns, we are able to deliver a wider variety of interventions to the students. More distinct patterns also mean more diverse behaviors that can trigger an intervention, possibly allowing us to intervene in different situations during a student's interaction. The breakdown of patterns by high and low learners is shown in Table 8 and example patterns are shown in Table 9. We observe that in all feature sets, LL clusters produce more distinct patterns than the HL ones, suggesting that low learners behave less uniformly in terms of the detrimental behaviors they show, than high learners do with respect to successful behaviors.

While the patterns produced by the three feature sets varied, we identified 4 trends that occurred in at least two feature sets each. This shows that our general approach for behavior discovery is able to uncover core behaviors that are stable across representations. The following patterns are presented in their raw form for brevity, but the English description can be found in Table 9.

One of the trends found is related to the use of the voltmeter and ammeter (to measure the voltmeter and current of different parts of the circuit, respectively). Intuitively, using testing devices is an effective behavior for understanding the changes in a circuit. We observed that LLs perform testing with low frequency. This pattern occurs for LLs in all feature sets: *FOAC_f* (*deliberate_measure.Test.startMeasure.voltmeter_f = Low*), *FAC_f* (*Test.endMeasure.voltmeter_f = Low*), and *OAC_f* (*deliberate_measure.traceMeasure.nonContactAmmeter_f = Low*).

The next trend we identified is related to changing the resistance of resistors. Engaging in this behavior is consistent with experimenting with a range of resistances, as suggested by the study activity, and is an effective behavior for understanding how resistors work. However, we see that LLs engage in this pattern infrequently, and the trend is observed in 2 feature sets: *FAC_f* (*Build.changeResistance.resistor_f = Low* and *Revise.changeResistance.resistor_f = Low*) and *OAC_f* (*fire_started.changeResistance.resistor_f = Low*). While this last pattern may seem counterintuitive because it suggests that it is productive to start fires, fires only occur in a circuit when the current through a component reaches a very high value. This happens when that segment of the circuit has relatively little resistance and a very high voltage. By causing a fire to happen after changing resistances, students are likely experimenting with extremes of resistance and gaining an understanding about that. Yet, the pattern in *OAC_f* indicates that low learners engage in this behavior infrequently.

The next trend we observed is related to the frequency of pausing (possibly to plan, reflect, and take notes), and, as with the two before, it is only present in LL patterns. Specifically, LLs show patterns of pausing infrequently, suggesting that they are not taking adequate time to reflect on the circuit behavior. This trend is identified in both the *FOAC_f* (*pause_f = Low*) and *FAC_f* feature sets.

Table 8 Number of distinct patterns for each of the feature sets. This table shows the breakdown of patterns coming from high learners and low learners

Feature set	HL distinct patterns	LL distinct patterns	Total distinct patterns
FOAC_f	11	12	23
FAC_f	6	11	17
OAC_f	6	9	15

Table 9 Sample patterns for each feature set (raw form and English description)

Feature set	Cluster	Pattern [Description]
FOAC_f	HL	None.Build.add.lightBulb_f = Low [When building, they added light bulbs resulting in no outcome with low frequency] light_intensity.Revise.split.junction_f = Low [When revising, they split junctions resulting in changes to light intensity with low frequency]
	LL	deliberate_measure.Test.startMeasure.voltmeter_f = Low [When testing, they used the voltmeter with low frequency] pause_f = Low [They paused with low frequency]
FAC_f	HL	Build.add.lightBulb_f = Low [When building, they added light bulbs with low frequency]
	LL	Build.changeResistance.resistor_f = Low [When building, they changed the resistance of resistors with low frequency] Revise.changeResistance.resistor_f = Low [When revising, they changed the resistance of resistors with low frequency] Test.endMeasure.voltmeter_f = Low [When testing, they used the voltmeter with low frequency] pause_f = Low [They paused with low frequency]
	LL	deliberate_measure.traceMeasure.nonContactAmmeter_f = Low [They used the non contact ammeter by tracing with low frequency] fire_started.changeResistance.resistor_f = Low [They changed the resistance of resistors resulting in a fire with low frequency]
OAC_f	HL	light_intensity.join.wire_f = Low [They joined wires resulting in light intensity change with lower frequency]
	LL	deliberate_measure.traceMeasure.nonContactAmmeter_f = Low [They used the non contact ammeter by tracing with low frequency] fire_started.changeResistance.resistor_f = Low [They changed the resistance of resistors resulting in a fire with low frequency]

The last trend is related to the addition of light bulbs and changes in light intensity. HLs both add light bulbs infrequently and make infrequent changes in light intensity. Since this activity was focused on understanding how resistors work in circuits, light bulbs were likely distractors at best, and possibly interfered with observing the behavior of other resistors (light bulbs are a type of resistor too). Thus, it makes sense that those who learn well with CCK use and manipulate light bulbs infrequently. We see behaviors associated with HLs adding light bulbs infrequently in both the *FAC_f* feature set (*Build.add.lightBulb_f = Low*) as well as in the *FOAC_f*. Notably, the *FOAC_f* feature set further qualified this action-event with the outcome (*None.Build.add.lightBulb_f = Low*), as shown in Table 9. The pattern related to HLs making changes to light intensity with low frequency appears in both *FOAC_f* (*light_intensity.Revise.split.junction_f = Low*) and *OAC_f* (*light_intensity.join.wire_f = Low*), although related to two different ways to create changes in light intensity in a circuit (splitting a junction and joining wires).

In summary, the trends uncovered by association rules in the different feature sets indicate that it is productive to do the following activities: (i) frequently use testing

devices, (ii) frequently change the resistance of resistors, (iii) frequently pause, and (iv) infrequently use light bulbs and change the light intensity. It is interesting to note that as part of interpretation of these trends, we approached the instructor of the course, as well as other expert physicists. They confirmed that these are meaningful observations and mapped these trends on elements such as focus on the task at hand and reflection. While these were not a-priori hypotheses (as our rule-mining was unsupervised), the fact that the rules are easily interpretable by experts adds some weight to their validity.

One limitation of our approach is that the number of features (behaviors) in our data often exceeds the number of items being clustered (students). However, several aspects suggest that our results are still meaningful. First, notably, we observed similar patterns across data representations. That is, the detected rules were very similar, regardless of the structure and content of the original data (e.g., inclusion of timing information). Thus, these rules are likely to be inherent to the data, and the fact that our patterns transcend beyond specific representations suggests that are valuable patterns.

Second, these rules match our and experts' intuition about what behaviors are conducive to learning, showing ecological validity. The fact that these rules are highly interpretable again supports their value. Third, the clustering process was unsupervised. Thus, if the rules had not been meaningful, we would not have received differences in learning between clusters. The fact that clusters in these specific representations translated to differences in learning supports the interpretation of the identified rules. Finally, classifiers trained on some of the detected sets of clusters could predict learning better than a baseline. Thus, the classifiers do not overfit the training sets (if they did, they would have poor accuracy on the test sets during cross validation), an additional indication that our datasets are meaningful despite the fact that they are not large.

Patterns Potential for Supporting Adaptive Hints

Next we discuss the usefulness of these patterns to inform potential adaptive support. One criterion for doing so is level of detail at which the support can be provided. Naturally, this depends on the granularity of the corresponding features in the different representations. Thus, behaviors in *FOAC_f* give the most contextual information and can be used to give students feedback with regard to the *outcome* of desired actions, *family of actions* that achieve that outcome in terms of a high level behavior, and *how to* achieve it using specific actions and components. For example, a hint relating to the “frequently change the resistance of resistors” trend could deliver a variety of levels of support depending on the representation. In *FAC_f*, the LL rule is “*Revise.changeResistance.Resistor_f = Low*”. In this representation, a hint could tell students to revise more (*what to do generally*), and then give the specific suggestion of doing this by changing the resistance of resistors (*how to do it*). It is missing the layer of feedback related to outcomes and is therefore not able to emphasize the desired outcome the student should attain. This could be a drawback if certain outcomes are more productive than others. For example, it may be more useful if students have a testing instrument attached to the circuit when they are changing the resistance so that they can be observing changes on the reading of the testing device. Alternatively it may be better if they are creating fires as it likely means they are experimenting with

extreme resistances. In the absence of the outcome level, a hint based on this representation may get students to do a desired action without knowing what to look for. In contrast, the *OAC_f* representation is able to provide the desired outcome but is not able to provide the level about what to do in terms of families of behaviors. The related LL rule for *OAC_f* is “*fire_started.changeResistance.resistor_f = Low*”. With this representation, we can tell the students to start more fires (*outcome*), and then to specifically do this by changing the resistance of resistors (*how to do it*). We cannot provide information about what high level behaviors to perform (e.g., should they be revising or building their circuit while doing this?). Based on the identified relevance of the Outcome level, we believe that the information conveyed by *OAC_f* is more suited for informing feedback, compared with *FAC_f*. Naturally, for hints coming from the *FOAC_f* representation, we would be able to provide all levels of detail.

The richer level of detail available due to the nature of the *FOAC_f* representation lends itself well to provide sequences of hints with narrowing specificity (a well-established approach to hint provision in ITS (Lepper et al. 1993)). For instance, if the following LL pattern existed for *FOAC_f* (*fire_started.Revise.changeResistance.Resistor_f = Low*), a first-level hint could tell the student the outcome that they should try to achieve (“Try to generate more fires.”), then, if needed a second level, a hint could suggest the family (what to do at the high level – “Do this by revising more.”), followed by a hint on how to do it in terms of a specific action and component involved (“Do this by changing the resistance of the resistors.”). The *OAC_f* and *FAC_f* feature sets do not support this hint progression.

Discussion, Conclusion and Future Work

In this paper, we aimed to provide a comprehensive evaluation of FUMELE, the student modeling framework for exploratory learning proposed in (Kardan and Conati 2011), when applied to several multi-layer representations of student interactions with CCK, a complex simulation for learning about electrical circuits. These representations differ in their level of granularity and amount of feature engineering to deal with the high level of complexity in the interaction behaviors supported by CCK.

We evaluated the representations in terms of ability to identify learners with high- or low- learning gains, suitability for student modeling (i.e. ability to classify new students in terms of their learning performance as they work with the simulation), and for informing the content of adaptive support during interaction.

The results in this paper represent an important step towards providing evidence for the *generality* of FUMELE. This framework had already been successfully applied for modeling students and providing support with the CSP Applet, a rather simple simulation for an AI algorithm (Kardan and Conati 2015). Here, we showed that it can transfer to a substantially more complex ELE, in terms of both creating suitable student models as well as identifying interaction behaviors that can be leveraged to inform the design of student adaptive support.

Evaluation Results In relation to student modeling, most classifiers built using FUMELE were significantly better than baseline at classifying whether a student showed behaviors conducive to high vs low learning with CCK. The best classifier

was able to classify over 70% of students correctly after observing just 20% of interaction data, an accuracy comparable to the one reported in (Kardan and Conati 2015) for the student model built for the AIspace applet using the same framework. Using that model, an adaptive version of the CSP applet generated higher student learning than the non-adaptive version, providing evidence that a student model with the aforementioned levels of accuracy can support successful real-time adaptation.

In the case of the CSP applet, FUMELE was applied to features that represented basic summative statistics of raw interaction data. In this paper, we showed that some level of feature engineering is necessary for reliable student classification when the interactive simulation gets more complex. Notably, when we tested a feature set that had no feature engineering (*Raw_OAC_fms*), it was the best at clustering learners in our dataset, showing that the information in the logs is suitable for distinguishing behaviors that generate different learning outcomes. However, this feature set could not beat the baseline in classifying learners, proving unsuitable for student modeling and provision of adaptive support.

For the remaining representations that we tested, the *FAC_f* feature set (including information on the interface actions applied, their general family as well as the components involved, but no action outcome) is also unsuitable as it does not provide classification that beats the baseline until late in the interaction. The two better representations included information on both the action performed and its outcome. The one with less feature engineering, *OAC_f* (which misses information about which family an action belongs to) generates rules that come from higher quality clusters (i.e. with the largest difference in learning gains), and can target the correct students sooner, reaching 70% accuracy after only 20% of interaction. The classifier built over the most complex representation (*FOAC_f*) reaches the same accuracy only after seeing 60% of the interaction. However, this representation might still be worth considering because it provides richer information than *OAC_f* for defining adaptive hints for CCK, both in terms of number of relevant behaviors identified as representative of higher and lower learning, as well as in terms of the level of hint detail that it can support.

Implications for Adaptive Interventions The next step of this research is to build adaptive versions of the CCK simulation based on the student models and interaction behaviors identified in this paper, to ascertain whether they can foster student learning. In particular, the trade-off we have identified with *FOAC_f* and *OAC_f* between accuracy of student modeling and richness of hints supported suggests that an adaptive CCK version should be built for each representation, and evaluated, to ascertain if and how the tradeoff impacts the effectiveness of adaptivity in CCK.

It should be noted that the interventions that can be generated solely based on the behaviors discovered during student modeling consist of suggesting to low learners behaviors that were identified as being conducive to learning or discouraging detrimental ones. Our models have no information to diagnose the potential causes underlying the suboptimal interactions exhibited by low learners (e.g., lack of relevant cognitive and meta-cognitive skills, cognitive overload, etc.). In this respect, our models and resulting hints are rather shallow. They attempt to correct the behaviors, but cannot do anything to address their causes. Although several researchers have argued that analyzing student learning behaviors in exploratory learning environments may require analysis of students cognitive and metacognitive skills and processes

(Land 2000; Segedy et al. 2016), We already have evidence that our “shallow” student modeling and adaptive interventions can foster learning. The adaptive version that was built for the CSP applet, the previous simulation targeted in our research, included 9 hints derived from the behavior discovery process described in the “[User Modeling Framework](#)” section. Some of these hints discouraged behaviors identified as typical to low learners with that simulation (e.g. overusing a functionality that allows one to step through the simulation of the AC3 algorithm to the end), others suggested behaviors that worked well for high learners (e.g. spending more time in between actions). A formal evaluation of the adaptive CSP applet showed that it generated over one-standard deviation higher learning gains than its non-adaptive counterpart, with students following an average of 73% of the hints they received (Kardan and Conati 2015).

Building and evaluating adaptive versions of CCK based on the *FOAC_f* and *OAC_f* representations will allow us to verify whether the positive outcomes obtained with the CSP applet generalize to this more complex simulation. It will also help shed light on whether it is more important to have an accurate student model or richer (albeit still shallow) hints to foster student learning. It should be noted that the evaluations discussed here relate to using CCK in exploratory mode to learn about resistors, because the behaviors that FUMELE deemed useful or detrimental to learning when students were working with CCK in this way. As one of the reviewers of this paper pointed out, the relation of some of these behaviors with learning might change if CCK were used to support a different learning activity, e.g. an exercise designed by a teacher to illustrate how light bulbs impact a circuit or how to create fires. In this case, a different set of hints would have to be created.

Another avenue of future research relates to comparing the pedagogical outcomes that can be obtained by our data-driven, lightweight approach to generating adaptive hints, against an approach that relies on a knowledge-based student model for CCK. Such a model would encode the relationships between relevant interaction behaviors and the underlying domain knowledge and cognitive/meta-cognitive skills, as was done for instance in Conati and Merten (2007) for a simulation for mathematical functions. This model could perform knowledge and skills assessment from the observed student interaction behaviors (Conati and Merten 2007; Gobert et al. 2012), which in turn, can drive adaptive interventions richer than the strictly behavioral suggestions supported by our current approach (e.g. re-teaching skills that the student model indicates to be poorly understood). Understanding and representing the relationships between knowledge, student skills and behaviors, however, is usually a laborious and time consuming process, even in relatively well understood problem-solving domains (Kodaganallur and Weitz 2005). It is an open question worthy of investigation: whether having the set of behaviors identified by our framework as important for learning (or lack thereof) in open-ended interactions can facilitate the knowledge engineering process.

Generality of the Approach Results from (Kardan and Conati 2015) and from this paper provide evidence that some important outcomes obtained by applying FUMELE to the CSP applet transferred to the more complex CCK. However, this transfer required extending the approach with a process to add levels of knowledge engineering (albeit simple) to the creation of the feature sets necessary for user modeling. Whereas some aspects of the knowledge engineering that we performed are specific to the CCK

simulation (e.g. adding actions to better qualify the operations available on a DC circuit, such as *joinX* and *traceMeasure*, as described in “Representing user actions in CCK”), the underlying approach of providing a structured representation of action-events that contextualize actions in terms of their family, components and outcomes is general. Thus, another step of future work is to test the generality of FUMELE with the extended representation for actions by applying it to other simulations (e.g. other simulations in the PhET family or in AISpace, the suite of interactive simulations of AI algorithms that include the CSP applet). Open research questions that could be pursued include if and how well FUMELE with the structured representation proposed here transfers to other complex simulations, as well as whether it is possible to identify general principles that describe when raw data vs engineered features should be used based on properties of the target ELE.

References

- Alfieri, L., Brooks, P. J., Aldrich, N. J., & Tenenbaum, H. R. (2011). Does discovery-based instruction enhance learning? *Journal of Educational Psychology*, *103*(1), 1–18. doi:10.1037/a0021017.
- Baker, R. S. J., Gowda, S. M., & Corbett, A. T. (2011). Towards predicting future transfer of learning. *15th International Conference, Artificial Intelligence in Education, 2011*, 23–30. doi:10.1007/978-3-642-21869-9_6.
- Biswas, G., Leelawong, K., Schwartz, D., Vye, N., & The Teachable Agents Group at Vande (2005). Learning by teaching: a new agent paradigm for educational software. *Applied Artificial Intelligence*, *19*(3–4), 363–392. doi:10.1080/08839510590910200.
- Borek, A., McLaren, B., Karabinos, M., Yaron, D. (2009). How much assistance is helpful to students in discovery learning? *Learning in the synergy of multiple disciplines 4th European Conference on Technology Enhanced Learning*, Springer, Berlin / Heidelberg, 391–404.
- Conati, C., & Merten, C. (2007). Eye-tracking for user modeling in exploratory learning environments: an empirical evaluation. *Knowledge-Based Systems*, *20*(6), 557–574. doi:10.1016/j.knosys.2007.04.010.
- Conati, C., Fratamico, L., Kardan, S., Roll, I. (2015). Comparing representations for learner models in interactive simulations. *Artificial Intelligence in Education* (pp. 74–83). Springer International Publishing. doi:10.1007/978-3-319-19773-9_8.
- De Jong, T. (2006). Technological advances in inquiry learning. *Science*, *312*(5773), 532–533. doi:10.1126/science.1127750.
- De Jong, T., & Van Joolingen, W. R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research*, *68*(2), 179–201.
- De Jong, T., Linn, M. C., & Zacharia, Z. C. (2013). Physical and virtual laboratories in science and engineering education. *Science*, *340*(6130), 305–308. doi:10.1126/science.1230579.
- Eagle, M., & Barnes, T. (2014). Exploring differences in problem solving with data-driven approach maps. *Educational Data Mining*, 76–83.
- García, E., Romero, C., Ventura, S., & de Castro, C. (2008). An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Modeling and User-Adapted Interaction*, *19*(1–2), 99–132. doi:10.1007/s11257-008-9047-z.
- Gobert, J. D., & Buckley, B. C. (2000). Introduction to model-based teaching and learning in science education. *International Journal of Science Education*, *22*(9), 891–894. doi:10.1080/095006900416839.
- Gobert, J. D., Sao Pedro, M. A., Baker, R. S. J. d., Toto, E., & Montalvo, O. (2012). Leveraging educational data mining for real-time performance assessment of scientific inquiry skills within microworlds. *JEDM - Journal of Educational Data Mining*, *4*(1), 111–143.
- Gobert, J. D., Sao Pedro, M., Raziuddin, J., & Baker, R. S. (2013). From log files to assessment metrics: measuring students’ science inquiry skills using educational data mining. *The Journal of the Learning Sciences*, *22*(4), 521–563. doi:10.1080/10508406.2013.837391.
- Gong, Y., Beck, J. E., Ruiz, C. (2012). Modeling multiple distributions of student performances to improve predictive accuracy. *User Modeling, Adaptation, and Personalization*, Springer Berlin Heidelberg, 102–113.

- Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: a response to Kirschner, Sweller, and. *Educational Psychologist*, 42(2), 99–107. doi:[10.1080/00461520701263368](https://doi.org/10.1080/00461520701263368).
- Hussain, T. S., Roberts, B., Menaker, E. S., et al. 2009. Designing and developing effective training games for the US Navy. *The Interservice/Industry Training, Simulation & Education Conference (IITSEC)*, NTSA.
- Kardan, S., & Conati, C. (2011). A Framework for capturing distinguishing user interaction behaviours in novel interfaces. *Proceedings of the 4th International Conference on Educational Data Mining*, 159–168.
- Kardan, S., & Conati, C. (2013). Evaluation of a data mining approach to providing adaptive support in an open-ended learning environment: a pilot study. *AIED 2013 Workshops Proceedings Volume 2 Scaffolding in Open-Ended Learning Environments (OELEs)*, 41–48.
- Kardan, S., & Conati, C. (2015). Providing adaptive support in an interactive simulation for learning. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, ACM Press, 3671–3680. doi:[10.1145/2702123.2702424](https://doi.org/10.1145/2702123.2702424).
- Kardan, S., Roll, I., & Conati, C. (2014). The usefulness of log based clustering in a complex simulation environment. *Intelligent tutoring systems*, 168–177.
- Kinnebrew, J. S., & Biswas, G. (2012). Identifying learning behaviors by contextualizing differential sequence mining with action features and performance evolution. *Educational Data Mining*, 57–64.
- Kinnebrew, J. S., Loretz, K. M., & Biswas, G. (2013). A contextualized, differential sequence mining method to derive students' learning behavior patterns. *JEDM - Journal of Educational Data Mining*, 5(1), 190–219 Retrieved from <http://www.educationaldatamining.org/JEDM/index.php/JEDM/article/view/34>.
- Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41, 75–86. doi:[10.1207/s15326985Sep4102_1](https://doi.org/10.1207/s15326985Sep4102_1).
- Kodaganallur, V., & Weitz, R. R. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education*, 15(2), 117–144 Retrieved from <http://dl.acm.org/citation.cfm?id=1434928>.
- Krishna, K., & Narasimha Murty, M. (1999). Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(3), 433–439. doi:[10.1109/3477.764879](https://doi.org/10.1109/3477.764879).
- Kuhn, D., Black, J., Keselman, A., & Kaplan, D. (2000). The development of cognitive skills to support inquiry learning. *Cognition and Instruction*, 18(4), 495–523. doi:[10.1207/S1532690XCII1804_3](https://doi.org/10.1207/S1532690XCII1804_3).
- Land, S. M. (2000). Cognitive requirements for learning with open-ended learning environments. *Educational Technology Research and Development*, 48(3), 61–78. doi:[10.1007/BF02319858](https://doi.org/10.1007/BF02319858).
- Lepper, M. R., Woolverton, M., Mumme, D. L., Gurtner, J.-L. (1993). Motivational techniques of expert human tutors: lessons for the design of computer-based tutors. In *Computers as Cognitive Tools*. 75–105.
- Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2), 159–179. doi:[10.1007/BF02294245](https://doi.org/10.1007/BF02294245).
- Najar, A. S., Mitrovic, A., McLaren, B. M. (2014). Adaptive support versus alternating worked examples and tutored problems: which leads to better learning? In *User Modeling, Adaptation, and Personalization*. Springer, 171–182.
- Noss, R., Poulouvassilis, A., Geraniou, E., et al. (2012). The design of a system to support exploratory learning of algebraic generalisation. *Computers & Education*, 63–81. doi:[10.1016/j.compedu.2011.09.021](https://doi.org/10.1016/j.compedu.2011.09.021).
- Perera, D., Kay, J., Koprinska, I., Yacef, K., & Zaiane, O. R. (2009). Clustering and sequential pattern Mining of Online Collaborative Learning Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(6), 759–772. doi:[10.1109/TKDE.2008.138](https://doi.org/10.1109/TKDE.2008.138).
- Roll, I., Aleven, V., & Koedinger, K. R. (2010). The invention lab: using a hybrid of model tracing and constraint-based modeling to offer intelligent support in inquiry environments. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Intelligent Tutoring Systems* (pp. 115–124). Berlin Heidelberg: Springer.
- Roll, I., Baker, R. S. J. d., Aleven, V., & Koedinger, K. R. (2014a). On the benefits of seeking (and avoiding) help in online problem-solving environments. *The Journal of the Learning Sciences*, 23(4), 537–560. doi:[10.1080/10508406.2014.883977](https://doi.org/10.1080/10508406.2014.883977).
- Roll, I., Yee, N., Cervantes, A. (2014b). Not a magic bullet: the effect of scaffolding on knowledge and attitudes in online simulations. *International Conference of the Learning Sciences*, 879–886.
- Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1), 53–65. doi:[10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7).
- Salden, R. J. C. M., Aleven, V. A. W. M. M., Renkl, A., & Schwonke, R. (2009). Worked examples and tutored problem solving: redundant or synergistic forms of support? *Topics in Cognitive Science*, 1(1), 203–213. doi:[10.1111/j.1756-8765.2008.01011.x](https://doi.org/10.1111/j.1756-8765.2008.01011.x).

- Sao Pedro, M. A., de Baker, R. S. J., Gobert, J. D., Montalvo, O., & Nakama, A. (2011). Leveraging machine-learned detectors of systematic inquiry behavior to estimate and predict transfer of inquiry skill. *User Modeling and User-Adapted Interaction*, 23(1), 1–39. doi:10.1007/s11257-011-9101-0.
- Sao Pedro, M. A., Baker, R. S. J. D., Gobert, J. D. (2012). Improving construct validity yields better models of systematic inquiry, even with less information. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, 249–260. doi: 10.1007/978-3-642-31454-4_21.
- Segedy, J. R., Kinnebrew, J. S., Biswas, G. (2016). Using coherence analysis to characterize self-regulated learning behaviours in open-ended learning environments. 2, 1. Retrieved July 19, 2015 from http://www.researchgate.net/publication/279753076_Using_Coherence_Analysis_to_Characterize_Self-Regulated_Learning_Behaviours_in_Open-Ended_Learning_Environments.
- Shih, B., Koedinger, K. R., Scheines, R. (2010). Unsupervised discovery of student strategies. *Proceedings of the 3rd International Conference on Educational Data Mining*, 201–210.
- Stamper, J., Eagle, M., Barnes, T., & Croy, M. (2013). Experimental evaluation of automatic hint generation for a logic tutor. *International Journal of Artificial Intelligence in Education*, 22, 3–17. doi:10.3233/JAI-130029.
- Taylor, C., Veeramachaneni, K., O'Reilly, U-M. (2014). Likely to stop? Predicting Stopout in Massive Open Online Courses. Retrieved March 29, 2016 from <http://arxiv.org/abs/1408.3382>.
- Westerfield, G., Mitrovic, A., & Billingham, M. (2013). Intelligent augmented reality training for assembly tasks. In H. Chad Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial intelligence in education* (pp. 542–551). Berlin Heidelberg: Springer.
- Wieman, C. E., Adams, W. K., & Perkins, K. K. (2008). PhET: simulations that enhance learning. *Science*, 322(5902), 682–683. doi:10.1126/science.1161948.