Towards Understanding Expert Coding of Student Disengagement in Online Learning

Luc Paquette (paquette@tc.columbia.edu)
Department of Human Development,
Teachers College, Columbia University, New York, NY, 10027

Adriana M.J.A. de Carvalho (adrianac@cs.cmu.edu)
Human-Computer Interaction Institute,
Carnegie Mellon University, Pittsburgh, PA, 15213

Ryan S. Baker (baker2@exchange.tc.columbia.edu)
Department of Human Development,
Teachers College, Columbia University, New York, NY, 10027

Abstract

Gaming the system, a behavior where students disengage from a learning environment and attempt to succeed by exploiting properties of the system, has been shown to be associated with lower learning. Machine learned and knowledge engineered models have been created to identify gaming behaviors, but few efforts have been made to precisely identify how experts code gaming behaviors. In this paper, we used cognitive task analysis to elicit knowledge about how experts code students as gaming or not in Cognitive Tutor Algebra. We show how building a cognitive model of this process gave us insights about the behaviors gaming is composed of.

Keywords: Gaming the system, Cognitive Tutor, cognitive modeling, expert coding.

Introduction

In recent years, there has been increasing awareness that students often disengage from interactive environments by "gaming the system", defined as "attempting to succeed in an educational environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly" (Baker et al. 2008). Gaming the system has been studied in multiple learning systems (Baker et al. 2008; Baker et al. 2010; Baker, Mitrovic, & Mathews 2010; Beal, Qu, & Lee 2006; Johns & Woolf 2006; Muldner et al. 2011; Walonoski & Heffernan 2006a) and has been shown to be related to poorer learning outcomes (Cocea, Hershkovitz, & Baker 2009; Fancsali 2013; Pardos et al. 2013), boredom (Baker et al. 2010), and poorer long-term academic success (San Pedro et al. 2013).

Models of gaming the system have been created using both knowledge engineering (Aleven et al. 2006; Beal, Qu, & Lee 2006; Gong et al. 2010; Johns, & Woolf 2006; Muldner et al. 2011; Walonoski and Heffernan 2006a) and machine learning (Baker et al. 2008; Baker, Mitrovic, & Mathews 2010; Walonoski, & Heffernan 2006a) approaches. Both approaches have been successful at allowing educational systems to intervene when students are gaming the system (Arroyo et al. 2007; Walonoski & Heffernan 2006b) and understanding the relationship between gaming and other constructs, but they do not implement a deep understanding of the construct of gaming the system and of the context of the student's actions as experts do when coding gaming behaviors.

In particular, instead of studying how experts code gaming behaviors, knowledge engineered models have typically implemented simple versions of two types of behaviors related to gaming (Baker et al. 2008), help abuse (Aleven et al. 2006) and systematic guessing. Help abuse has mainly been modeled using behaviors that include copying the answer from a hint (Gong et al. 2010; Johns, & Woolf 2006; Muldner et al. 2011) and repeated help requests (Beal, Qu, & Lee 2006; Walonoski, & Heffernan 2006a), but few efforts have been made to understand how expert coders determine when those behaviors are instances of gaming. Struggling students will often look at multiple help messages and might sometime copy answers from hints as part of their learning process. As we will discuss, expert coders rely on the context in which those behaviors take place to determine if the student is gaming or not.

Systematic guessing is usually defined as quickly answering after errors (Beal, Qu, & Lee 2006; Gong et al. 2010; Johns, & Woolf 2006; Muldner et al. 2011) and making successive errors (Walonoski, & Heffernan 2006a). Similarly to help abuse, many instances of those behaviors can be associated with normal use of the learning environment. For example some minor errors can be very quickly corrected, a behavior that looks similar to guessing. In those situations, expert coders will once again look at the context in which the actions were executed in order to determine if they are systematic guesses or not.

The machine learning approach, on the other hand, captures part of the context of the student's gaming behaviors, but it is not easy to fully understand what context the models capture and to evaluate whether it corresponds to expert judgments. In other words, machine learning often makes the same predictions as experts, but may be doing so in different ways.

Studying the context used by experts when coding the student's gaming behaviors would be a potentially valuable step towards getting a deeper understanding of this construct and modeling it better, whether using machine learning, knowledge engineering or both. To achieve this, we used cognitive task analysis (Cooke 1994; Clark et al. 2008) to elicit the expert's process of coding gaming behaviors using text replays from Cognitive Tutor Algebra (Koedinger, & Corbett 2006). Using the knowledge we acquired, we built a cognitive model of how an expert codes gaming behaviors and validated it against the expert's judgment.

Method

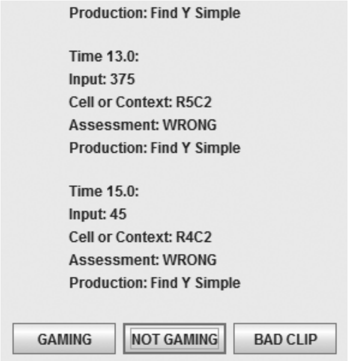
Data

In order to study how experts code gaming behaviors, we used data obtained from 59 students using the Cognitive Tutor Algebra system during an entire school year as part of their regular mathematics curriculum. The Cognitive Tutor environment offers mathematical problems broken down into the steps of the process used to solve the problem. As the student works through the problem, a cognitive model of the task assesses whether the student's answers map to a correct step. Cognitive Tutors also offer multi-step hints. A student who is struggling can ask for help at anytime and the system will provide increasingly specific hints about what should be done next.

Data from 12 tutor lessons was obtained and segmented in sequences of 5 actions, called clips, illustrating the student's behavior. A total of 10,397 clips from this dataset were randomly selected; the chance of a clip being selected was weighted for each lesson according to the total number of clips in that lesson. Those clips were previously coded by an expert in order to develop machine-learned gaming models (e.g. Baker, & de Carvalho 2008), and contains 708 examples of gaming the system and 9,689 examples of behaviors that were not coded as gaming.

Studying expert coding

In order to code clips of student behaviors as gaming the system or not, experts can use text replays (Baker, Corbett, & Wagner 2006; Baker, & de Carvalho 2008), clips of student behaviors in a textual form. A clip of a pre-selected duration (in terms of time or length) is shown in a textual format giving information about the actions and their context. In the example shown in figure 1, the expert sees each action's time (relative to the first action in the clip), the problem context, the input entered, the relevant skill (production) and how the system assessed the action (right, wrong, a help request or a "bug"). The expert can then choose whether the behavior is gaming or not, or indicate that something has gone wrong in the text replay software or logs, making the clip uncodeable. When coding gaming behaviors using text replays, experts typically achieve an inter-rater reliability, measured using Cohen's Kappa, around 0.60 (Baker, Corbett, & Wagner 2006).



Production: Find Y Simple

Time 13.0:
Input: 375
Cell or Context: R5C2
Assessment: WRONG
Production: Find Y Simple

Time 15.0:
Input: 45
Cell or Context: R4C2
Assessment: WRONG
Production: Find Y Simple

Figure 1: The last 2 actions of a 5-action text replay clip.

To elicit knowledge about the expert's process of coding gaming behaviors using text replays, we used a cognitive task analysis (Cooke 1994; Clark et al. 2008) approach in which knowledge was elicited through observations and interviews. The first author acted as elicitor and the second author was the domain expert. The second author, to the best of our knowledge, has conducted more text replay coding of student behavior in intelligent tutoring systems than anyone else in the world, has conducted text replays for a range of problems and learning environments, and has trained others in the method.

We studied how the expert coded gaming behaviors when using text replays from Cognitive Tutor Algebra. To elicit the expert's knowledge, an "active participation" (Cooke 1994; Meyer 1992) approach was used to collect preliminary knowledge of the task. During those sessions, the expert explained to the elicitor how to identify gaming behavior using text replays. The elicitor coded a few clips while thinking aloud as the expert commented and corrected his thought process. This allowed the elicitor to acquire a good understanding of the process, enabling him to better communicate with the expert.

Once the elicitor had understood the process of coding clips, additional observations sessions were conducted in which the expert coded clips while thinking aloud (Van Someren et al. 1994) with minimal inputs from the elicitor. Using this method, we were able to observe and collect information about how the expert codes behaviors without biasing her thought process by asking questions about it. Those sessions were recorded and were used by the elicitor to elaborate an initial version of the cognitive model.

This initial version of the model was implemented and executed on a subset of our data. The complete set of 10,397 clips was separated in a training and a test set. The training set was composed of 75% of the clips coded as gaming and 75% of the clips coded as non-gaming, and the test set was composed of the remaining 25%. As such, the training set contained 531 gaming and 7,267 non-gaming clips and the test set contained 177 gaming and 2,422 non-gaming clips. Clips for each subset were chosen at random.

Separating the dataset into two subsets is essential to developing and validating a model. In order to elaborate the model, the knowledge elicitor and the expert need to look at and analyze clips of student behaviors. Having two subsets allows them to look at a restricted part of the data (the training set) while the rest of the data (the test set) remains unseen. The unseen data can then be used to validate the performance of the model elaborated on the training set. This ensures that the model is not overly specific to the data that was used to train it.

An initial model was developed and applied to the training set. Then, the clips that were misclassified by the model were collected. Additional sessions were conducted with the expert to examine those specific clips. The expert coded them while thinking aloud and the knowledge elicitor asked questions to precisely identify what allowed the expert to determine whether a clip was considered as gaming or not. This allowed us to formalize information that might have been omitted by the expert while thinking aloud. These sessions were also recorded.

The knowledge acquired from those sessions was included in a new version of the model that was once again executed on the training set. This process was repeated multiple times to iteratively improve our model by identifying situations where the model was unable to correctly classify gaming behaviors (false negatives) and to make the model more stringent to prevent false positives (non-gaming behaviors classified as gaming by the model).

Model

The cognitive process of an expert judging whether a clip of actions is an example of gaming the system can be separated in two main parts: interpreting the student's actions and identifying patterns of gaming. Although those two parts are usually done simultaneously by the expert, the cognitive model can do them separately without hindering the process.

Interpreting the student's actions

The first thing the expert does when looking at a text replay is to look at the pauses between each action to build a likely interpretation of the student's mental process. Each pauses before or after an action is an opportunity for the student to think about the problem he/she is currently solving. Table 1 provides a list and a description of constituents of the students' behaviors that were identified during the knowledge elicitation process with the expert.

In Cognitive Tutor Algebra, students can execute two types of actions: help requests and step attempts. Help can be requested more than once consecutively to obtain increasingly specific help messages until the tutor provides the answer (within a “bottom-out hint”). The length of the pause before an help request indicates whether the student took the time to think about the next step before asking for help. The length of the pause after receiving help indicates whether the student took the time to read the help message, was scanning the help message for specific information or was searching for a bottom-out hint.

Step attempts can be assessed by the system as being correct (right) or incorrect (wrong or bug). Bugs are incorrect answers expected by the system. When a bug is entered, the system provides the student with a message explaining the error.

Pauses before step attempts indicate whether the student took the time to think about the step before attempting it or if it is more likely that he/she was guessing the answer. In cases where this pause is short, it is also possible that the student planned one step ahead. In those cases, the previous step attempt should have been right and should have a long pause before it. When a step is assessed as incorrect, a pause after that step indicates that the student is thinking about his/her mistake. A bug message for an incorrect step gives additional information that can be used to interpret the student's action. Since bugs are expected mistakes, a long pause before the bug is usually an indicator of an unsuccessful but sincere attempt to solve the problem, whereas a short pause is usually an indicator of students trying to guess the answer by entering values taken from the problem statement. In addition, bugs are followed by an error message and thus, as for help requests, a long pause after a bug is an indicator that the student read the message provided by the system.

In addition to interpreting the pauses before and after actions, the expert also identifies relevant constituents in the answers inputted by the student. We identified four such constituents. First, the student might reuse the same answer multiple times in different contexts of the user interface. For example, the student might input the answer 5 in every text field. Second, the student might stop working on a part of the problem that he/she doesn't know how to solve and start working on a different part of the problem ([switched context before correct] in table 1). For example, attempting to answer part of the problem, getting an incorrect answer and then asking for help on a different part of the problem. Third, the student might input an answer that is similar to his/her previous input. For example, entering the sequence of answers 10, 20, 30, etc. Finally, the student might repeat the exact same step. For example, entering the same answer in the same text field.

Two main elements of the process of interpreting the student's actions remain ambiguous and difficult to accurately represent in the current version of our model: the time thresholds used to determine whether a pause is short or long and deciding whether two answers are similar. The time thresholds for pauses were defined using numbers that were often used by the expert as rules of thumb (table 1). An alternate approach would be to try to determine time thresholds empirically (cf. Baker et al. 2011).

To decide whether two answers are similar, we computed the Levenshtein distance (Levenshtein 1966) between the answer strings. This distance is defined by the number of edits (deletion, insertion or substitution of individual symbols) required to transform an answer and make it identical to the other one. Answers with a distance of 1 or 2 were considered as being similar. This approach worked well for numerical answers, but had issues with other types of answers. For example, it does not allow the detection of semantic similarities between text inputs such as seconds, minutes and hours. It also has issues dealing with equations. For this type of answer, "425x+150" will be considered similar to "425x-150", but "150x-425" will not.

Once we had identified the different constituents of the student's behavior that the expert pays attention to, we implemented them in our model. Our model simply iterates through all of the student's actions and label each of them using the rules defined in table 1.

Table 1 : List of the interpretations considered by the coder. Time thresholds are selected by the expert coder.

Identifier	Description
[did not think before help request]	Pause smaller or equal to 5 seconds before a help request
[thought before help request]	Pause greater or equal to 6 seconds before a help request
[read help messages]	Pause greater or equal to 9 seconds per help message after a help request
[scanning help messages]	Pause between 4 and 8 seconds per help message after a help request
[searching for bottom-out hint]	Pause smaller or equal to 3 seconds per help message after a help request
[thought before attempt]	Pause greater or equal to 6 seconds before step attempt
[planned ahead]	Last action was a correct step attempt with a pause greater or equal to 11 seconds
[guess]	Pause smaller or equal to 5 seconds before step attempt
[unsuccessful but sincere attempt]	Pause greater than or equal to 6 seconds before a bug
[guessing with values from problem]	Pause smaller than or equal to 5 seconds before a bug
[read error message]	Pause greater than or equal to 9 seconds after a bug
[did not read error message]	Pause smaller than or equal to 8 seconds after a bug
[thought about error]	Pause greater than or equal to 6 seconds after an incorrect step attempt
[same answer/diff. context]	Answer was the same as the previous action, but in a different context
[similar answer]	Answer was similar to the previous action (Levenshtein distance of 1 or 2)
[switched context before right]	Context of the current action is not the same as the context for the previous (incorrect) action (referred to as “soft underbelly” in Baker, Mitrovic, & Mathews 2010)
[same context]	Context of the current action is the same as the previous action
[repeated step]	Answer and context are the same as the previous action
[diff. answer AND/OR diff. context]	Answer or context is not the same as the previous action

Table 2 : Action patterns considered gaming the system. Constituents associated with each action are between brackets.

Pattern	Training			Test		
	TP	FP	Kappa	TP	FP	Kappa
incorrect → [guess] & [same answer/diff. context] & **incorrect**	92 17.33%	66 0.91%	0.243	47 26.55%	20 0.83%	0.219
incorrect → [similar answer] [same context] & **incorrect** → [similar answer] & [same context] & **attempt**	112 21.09%	173 2.38%	0.238	25 14.12%	62 2.56%	0.151
incorrect → [similar answer] & **incorrect** → [same answer/diff. context] & **attempt**	33 6.21%	19 0.26%	0.102	10 5.65%	6 0.25%	0.093
[guess] & **incorrect** → [guess] & [diff. answer AND/OR diff. context] & **incorrect** → [guess] & [diff. answer AND/OR diff. context & **attempt**	55 10.36%	92 1.27%	0.137	22 12.43%	33 1.36%	0.163
incorrect → [similar answer] & **incorrect** → [guess] & **attempt**	118 22.22%	204 2.81%	0.237	33 18.64%	70 2.89%	0.195
help & [searching for bottom-out hint] → **incorrect** → [similar answer] & **incorrect**	23 4.33%	52 0.72%	0.060	11 6.21%	24 0.99%	0.083
incorrect → [same answer/diff. context] & **incorrect** → [switched context before correct] & **attempt/help**	49 9.23%	72 0.99%	0.201	24 13.56%	19 0.78%	0.197
bug → [same answer/diff. context] & **correct** → **bug**	20 3.77%	16 0.22%	0.062	9 5.08%	5 0.21%	0.085
incorrect → [similar answer] & **incorrect** → [switched context before correct] & **incorrect**	50 9.42%	52 0.72%	0.146	16 9.04%	16 0.66%	0.135
incorrect → [switched context before correct] & **incorrect** → [similar answer] & **incorrect**	58 10.92%	55 0.76%	0.151	20 11.30%	17 0.70%	0.167
incorrect → [similar answer] & **incorrect** → [did not think before help] & **help** → **incorrect** (with first or second answer similar to the last one)	47 8.85%	57 0.78%	0.129	10 5.65%	24 0.99%	0.074
help → **incorrect** → **incorrect** → **incorrect** (with at least one similar answer between steps)	45 8.47%	83 1.14%	0.113	15 8.47%	32 1.32%	0.108
incorrect → **incorrect** → **incorrect** → [did not think before help request] & **help** (at least one similar answer between steps)	83 15.63%	154 2.12%	0.182	23 12.99%	60 2.48%	0.140

Identifying patterns of gaming the system

Once the expert has interpreted the student's actions, she uses the interpretation's constituents to judge whether the student is gaming the system. Although some constituents, such as searching for bottom-out hints and guessing the answer, are usually associated with gaming and others, such as getting the right answer, are usually associated with non-gaming behaviors, looking at each constituent individually is not sufficient to code the student's behavior. The expert instead tries to find a pattern of actions and constituents that acts as strong evidence that the student is gaming.

Some patterns might involve sequences of incorrect actions -- for example, reusing the same number in multiple different contexts of the interface to try to guess where the number goes. This can also be combined with bottom-out hints -- for example, searching for a bottom-out hint, getting the wrong answer and then trying a similar answer.

It is also possible for gaming patterns to include constituents that are usually associated to non-gaming behaviors. For example, getting the right answers is not evidence of gaming, but a student who enters a bug, fixes it by trying the same answer in a different context, and enters a second bug, may be considered as gaming. In this pattern, the first action might have been a small mistake that was easily corrected by entering the same answer in a different context, but the second bug makes it a strong indicator that the correct step was a guess rather than an intentional correction and that the student is trying to game the system.

During the knowledge elicitation process, we identified 13 patterns (table 2) that the expert considers sufficient to code a clip as an example of gaming the system. Additional patterns were developed, but were not included in the final model if: 1) they were only found in a small number of gaming clips, 2) they had too much overlap with other patterns or 3) they produced too many false positives.

Our final model uses a sliding window approach iterating over the student's actions to identify gaming clips. Every sequence of 2, 3 or 4 actions in a clip are examined to determine if they match any of the 13 patterns from table 2. If at least one such match is found, the clip is classified as gaming, otherwise it is classified as non-gaming.

Validation

The performance of our model was assessed using Cohen's Kappa (Cohen 1960), a metric that assesses the degree to which the model is better than chance at identifying gaming clips. A Kappa of 0 indicates that the model performs at chance and a Kappa of 1 indicates that it performs perfectly.

We validated our model by computing its performance for both the complete model composed of the 13 patterns and sub-models that matched only one pattern. Table 2 lists the performance for each patterns by indicating their Kappa, how many clips containing this pattern had been coded by the expert as gaming (true positives or TP) and how many had been coded as non-gaming (false positives or FP).

Each individual pattern has an above chance performance at detecting gaming clips, with a Kappa of 0.060 or better on the training set. Although some patterns detect a higher number of FP than TP, each pattern detects a higher percentage of the TP than of the FP (as there are many more non-gaming clips than gaming clips in this dataset). Some of the patterns with a low number of TP were included in the model as they still contributed to the performance of the model when all patterns were combined. This is usually an indicator that they capture a different subset of gaming behaviors than the other patterns.

When the complete model was applied to the training set, it accurately detected 340 (64.03%) gaming clips, misdiagnosed 551 (7.07%) non-gaming clips and obtained a Kappa of 0.430. For the test set, the model accurately detected 93 (52.54%) gaming clips, misdiagnosed 210 (8.67%) non-gaming clips and obtained a Kappa of 0.330.

The performance of our cognitive model was similar to the best machine learned model of gaming that was developed for the same dataset (Baker, & de Carvalho 2008). The machine learned model obtained a Kappa of 0.40 when the model was trained and tested on the same dataset. This is a little below the Kappa of 0.430 that was achieved by our model on the training set. To further validate our model, we applied it to an unseen test set and obtained a Kappa of 0.330. Although performance decreased, the Kappa we obtained indicates that an important part of our model generalizes to new data. In as yet unpublished work, the modeling approach used by Baker, & de Carvalho (2008) achieved a Kappa of 0.24 when applied to held-out data, suggesting that our cognitive model is mildly better than the one from Baker, & de Carvalho (2008).

Discussion and conclusion

In this paper, we used cognitive task analysis (Cooke 1994; Clark et al. 2008) to understand how an expert codes sequences of actions (clips) in an interactive learning environment as gaming the system or not. This allowed us to acquire a better understanding of what experts include in their definition of this disengaged behavior. We used the knowledge elicited from the expert to implement a cognitive model of the process of coding gaming behaviors using text replays from Cognitive Tutor Algebra.

The patterns that we identified when building our model provided insights about what behaviors experts consider as gaming the system. Gaming has been defined by two main types of behaviors: systematic guessing and help abuse. In previous knowledge engineered models (Beal, Qu, & Lee 2006; Gong et al. 2010; Johns, & Woolf 2006; Muldner et al. 2011), systematic guessing has mainly been defined as short pauses between step attempts. Although this behavior was included in our model, we also found that entering the same answer in multiple contexts and entering similar answers are two key aspects of the students' guessing behaviors that have often been overlooked.

Surprisingly few of our gaming patterns included searching for bottom-out hints. As our model currently correctly classifies 61.15% of the instances of gaming from our dataset, it is possible that additional gaming behaviors that include searching for bottom-out hints could be discovered in the future. Other occurrences of help requests we observed in our patterns included quickly asking for help without thinking about the step and entering multiple incorrect answers despite asking for help.

Our current model was built using data from only one system (Cognitive Tutor Algebra) and the insights of only one expert. As such, it will be important to study whether different experts consider different behaviors to be gaming the system and whether our model is effective when applied to data from other systems. Examining the performance of our complete model and of individual patterns on those new systems would provide us with interesting insights about differences in how students game in different systems. Similarly, our model could be used as a tool for studying the cultural aspects of gaming the system by comparing the frequency of each pattern of gaming in different cultures.

Acknowledgments

This research was supported by a Fonds de recherche du Québec - Nature et technologies post-doctoral fellowship and by National Science Foundation grant #SBE-0836012.

References

Aleven, V., McLaren, B. M., Roll, I., Koedinger, K. R. (2006). Toward Meta-Cognitive Tutoring: A Model of Help Seeking with a Cognitive Tutor. *Int'l Journal of Artificial Intelligence in Eduaction*, 16, 101-130.

Arroyo, I., et al. (2007). Repairing Disengagement with Non-Invasive Interventions. *Proc of AIED 2007*, 195-202.

Baker, R. S. J. d., Corbett, A. T., Roll, I., Koedinger, K. R. (2008). Developing a Generalizable Detector of When Students Game the System. *User Modeling and User Adapted Interaction*, 18, 287-314.

Baker, R. S. J. d., Corbett, A. T., Wagner, A. Z. (2006). Human Classification of Low-Fidelity Replays of Student Actions. *Proc of EDM Workshop at ITS 2006*, 29-36.

Baker, R. S. J. d., D'Mello, S. K., Rodrigo, M. M. T., Graesser, A. C. (2010). Better to Be Frustrated than Bored: The Incidence, Persistence, and Impact of Learners' Cognitive-Affective States During Interactions with Three Different Computer-Based Learning Environments. *Int'l Journal of Human-Computer Studies*, 68, 223-241.

Baker, R. S. J. d., de Carvalho, A. M. J. A. (2008). Labeling Student Behavior Faster and More Precisely with Text Replays. *Proc of EDM 2008*, 38-47.

Baker, R. S. J. d., Gowda, S., Corbett, A. T. (2011). Towards Predicting Future Transfer of Learning. *Proc of AIED 2011*, 23-30.

Baker, R. S. J. d., Mitrovic, A., Mathews, M. (2010). Detecting Gaming the System in Constraint-Based Tutors. *Proc of UMAP 2010*, 267-278.

Beal, C. R., Qu, L., Lee, H. (2006). Classifying Learner Engagement Through Intergration of Multiple Data Sources. *Proc of AAAI-06*, 2-8.

Clark, R. E., Feldon, D., van Merriënboer, J., Yates, K., Early, S. (2008). Cognitive Task Analysis. In J. M. Spector, M. D. Merrill, J. J. G. van Merriënboer, & M. P. Driscoll (Eds.), *Handbook of Research on Educational Communications and Technology (3rd ed.)*, 575-593.

Cocea, M., Hershkovitz, A., Baker, R. S. J. d. (2009). The Impact of Off-Task and Gaming Behaviors on Learning: Immediate or Aggregate? *Proc of AIED 2009*, 507-514.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20 (1), 37-46.

Cooke, N. J. (1994). Varieties of Knowledge Elicitation Techniques. *Int'l Journal of Human-Computer Studies*, 41, 801-849.

Fancsali, S. E. (2013). Data-Driven Causal Modeling of "Gaming the System" and Off-Task Behavior in Cognitive Tutor Algebra. *NIPS Workshop on Data Driven Education*.

Gong, Y., Beck, J., Heffernan, N. T., Forbes-Summers, E. (2010). The Fine-Grained Impact of Gaming (?) on Learning. *Proc of ITS 2010*, 194-203.

Johns, J., Woolf, B. (2006). A Dynamic Mixture Model to Detect Student Motivation and Proficiency. *Proc of AAAI-06*, 163-168.

Koedinger, K. R., Corbett, A. T. (2006). Cognitive Tutors: Technology Bringing Learning Sciences to the Classroom. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences*, 61-77.

Levenshtein, A. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10 (8), 707-710.

Meyer, M. A. (1992). How to Apply the Anthropological Technique of Participant Observation to Knowledge Acquisition for Expert Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 22, 983-991.

Muldner, K., Burleson, W., Van de Sande, B., VanLehn, K. (2011). An Analysis of Students' Gaming Behaviors in an Intelligent Tutoring System: Predictors and Impact. *User Modeling and User Adapted Interaction*, 21, 99-135.

Pardos, Z. A., Baker, R. S. J. d., San Pedro, M. O. C. Z., Gowda, S. M., Gowda, S. M. (2013). Affective States and State Tests: Investigating how Affect Throughout the School Year Predicts End of Year Learning Outcomes. *Proc of LAK 2013*, 117-124.

San Pedro, M. O. Z., Baker, R. S. J. d., Bowers, A., J., Heffernan, N. T. (2013). Predicting College Enrolment from Student Interaction with an Intelligent Tutoring System in Middle School. *Proc of EDM 2013*, 177-184.

Van Someren, M. W., Barnard, Y. F., Sandberg, J. A. C. (1994). *The Think Aloud Method: A Practical Guide to Modeling Cognitive Processes*.

Walonoski, J. A., Heffernan, N. T. (2006a). Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. *Proc of ITS 2006*, 382-391.

Walonoski, J. A., Heffernan, N. T. (2006b). Prevention of Off-Task Gaming Behavior in Intelligent Tutoring Systems. *Proc of ITS 2006*, 722-724.