OATutor: An Open-source Adaptive Tutoring System and Curated Content Library for Learning Sciences Research

Zachary A. Pardos
UC Berkeley School of Education
Berkeley, CA, USA
pardos@berkeley.edu

Matthew Tang*
UC Berkeley EECS
Berkeley, CA, USA
matthewtang@berkeley.edu

Ioannis Anastasopoulos*
UC Berkeley School of Education
Berkeley, CA, USA
ioannisa@berkeley.edu

Shreya K. Sheel*
UC Berkeley School of Education
Berkeley, CA, USA
shreya_sheel@berkeley.edu

Ethan Zhang
UC Berkeley EECS
Berkeley, CA, USA
ethanzhang@berkeley.edu

ABSTRACT

Despite decades long establishment of effective tutoring principles, no adaptive tutoring system has been developed and open-sourced to the research community. The absence of such a system inhibits researchers from replicating adaptive learning studies and extending and experimenting with various tutoring system design directions. For this reason, adaptive learning research is primarily conducted on a small number of proprietary platforms. In this work, we aim to democratize adaptive learning research with the introduction of the first open-source adaptive tutoring system based on Intelligent Tutoring System principles. The system, we call Open Adaptive Tutor (OATutor), has been iteratively developed over three years with field trials in classrooms drawing feedback from students, teachers, and researchers. The MIT-licensed source code includes three creative commons (CC BY) textbooks worth of algebra problems, with tutoring supports authored by the OATutor project. Knowledge Tracing, an A/B testing framework, and LTI support are included.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

Adaptive learning, intelligent tutoring systems, open source, OER, content authoring, research through design, replicable research

ACM Reference Format:
Zachary A. Pardos, Matthew Tang, Ioannis Anastasopoulos, Shreya K. Sheel, and Ethan Zhang. 2023. OATutor: An Open-source Adaptive Tutoring System and Curated Content Library for Learning Sciences Research. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3544548.3581574

*Authors contributed equally to this research.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581574

1 INTRODUCTION

Adaptive learning systems based on Intelligent Tutoring System (ITS) principles have long been established as effective tools for improving learning gains, particularly in algebra and statistics [16, 37]. Despite their proven tutoring capabilities, no ITS, nor any other type of adaptive tutor, has yet been open-sourced. The absence of such a system has meant a high barrier to entry for those interested in replicating adaptive learning experiments or extending and fielding components of their design.

We introduce Open Adaptive Tutor (OATutor), an open-source[1] adaptive tutoring system and curated content library based on ITS principles [4], designed for the learning sciences research community. OATutor makes use of existing creative common content pools, not present at the inception of ITS, with OATutor authoring tools designed for learners to build its adaptive content library in a tractable way. With a completely engineered adaptive tutoring system made available, researchers can experiment and innovate with individual components of the system (e.g., adaptive algorithm, user interface, content modifications) without the burden of needing to create it all from scratch, including content management, individual mastery estimation, adaptive item selection, logging, and learning management systems (LMS) integration. Furthermore, OATutor is designed so that any arbitrary piece of code can be A/B tested, enabling the instrumentation of a wide variety of research questions the community can ask and reasonably endeavor to answer.

The OATutor project created a publisher-like authoring framework, focused around learners, to produce its Creative Commons (CC BY) adaptive content library. The OATutor content pipeline makes use of Google Spreadsheets, a familiar interface for students and educators. Through crowdsourcing of a community of content creators and editors, our publishing framework allowed for the production and quality iteration of a scalable, replicable, and time-efficient content environment.

Throughout the development of OATutor, we have iterated upon features, improved accessibility, and created a robust content community to demonstrate the effectiveness of the ecosystem. OATutor was successfully piloted in several community college classrooms, where feedback given from instructors contributed to feature design and improvement. Upon release, OATutor will include three textbooks worth of Algebra content, adapted from OpenStax [40].

[1] https://github.com/CAHLR/OATutor

The release of our MIT-licensed open-source adaptive tutoring codebase, combined with complete algebra adaptive content will allow researchers to conduct experiments using OATutor, then make their entire end-to-end experimental framework, content, and platform available as a Github link in a publication for others to replicate, compare to, and build off of.

In this paper, we discuss the design iteration process of OATutor following a research through design approach [53], where our system's features were critiqued and iterated upon by a diverse group of stakeholders, all of whom are committed to creating an ecosystem that transforms the current, propriety state of adaptive tutoring systems to a more open state. We will begin with related work, followed by presenting the OATutor system and the initial design choices that were made based on ITS research, some of which was established at CHI [11, 13]. We then describe the participants and data collection involved in our methodology, followed by results of the data collection and the iterative design decisions and implications that were informed by our stakeholder co-designers. Finally, we reflect on the multi-year field deployment of the system and discuss remaining design considerations presenting future research opportunities for the larger research community.

2 RELATED WORK

Opening up resources, courses, datasets, and algorithms has led to the adoption of digital resources at scale and accelerated the advancement of learning sciences research. We briefly highlight each of these open movements in education technology and argue that it is long past due for an ITS-based platform to be among the open movements.

2.1 Open Educational Resources

Open Educational Resources (OERs), resources that are openly provisioned for use and adaptation by a community of users for non-commercial purposes [23], allow for broader access and dissemination of learning materials such as problem sets, videos, and other web-media resources. With OERs, educators can incorporate material into their syllabus at no charge while reducing their time spent authoring content. Most OERs, like free and open textbooks, focus on accessibility [50]. Others, like Khan Academy, have created alternative recognition systems for skills and educational achievements. Meanwhile, the Scratch software has focused on collective construction of knowledge, where the program is developed and repurposed by others in order to learn from one another on the platform [32]. Kolibri, an online and offline OER learning platform for educators across the globe that contains curated and openly licensed educational content libraries, focuses on including a wide range of curricula containing lessons, assessments, books, and games [24]. However, there is a key gap to be filled in the translation between OERs and computer tutoring systems. The lack of a common structure to OERs largely inhibits them from interoperating with one another and slotting into digital assessment systems [41].

2.2 Massive Open Online Courses

Massive Open Online Courses (MOOCs), open access instructional content released through an online platform, gained traction as digital versions of courses from well-known universities made available to the world [6]. As a result, platforms like Coursera, edX, and Udemy emerged as content providers, with some offering credits counting towards a future degree. Participants interact with assessments, videos, and in some cases teaching staff and their peers, in order to foster discussion and further thinking. Through MOOCs, learners can enroll in courses for free, opening participation to anyone with internet access. However, a certificate of completion in a MOOC costs money, and while MOOCs are open access, their content is propriety and not creative commons like an OER.

2.3 Open Datasets

A variety of educational technology platforms have released anonymized data logs, promoting offline experiment replication and generalization of methods across datasets. Carnegie Learning has released dozens of datasets, which have contributed considerably to educational data mining research [17]. Most notably, they released a dataset in 2010 for the Association for Computing Machinery (ACM) Special Interest Group on Knowledge Discovery and Data-mining's (KDD) annual competition. The dataset contained over 9 GB of student data, becoming the most often cited from an educational technology platform [48] and the largest dataset the competition had seen at the time. ASSISTments, a free web-based tutoring platform, has also been generous with the release of its datasets that have been used in over 100 papers [2]. More recently, large datasets from tutoring systems developed and deployed in South Korea have been released, such as EdNet [10].

2.4 Open Algorithms

Open algorithms remove barriers for the educational data mining community to expand and replicate research. For example, pyBKT provides the first python implementation of a popular algorithm for cognitive mastery estimation [5], which is the same algorithm used in the Cognitive Tutor/MATHia. Open-sourcing of these tools can garner considerable attention from the community, with pyBKT receiving 1,000 installs per month. Many other algorithms are open-sourced as a companion to a research paper. As an example, an adaptive assessment engine used within MOOCs was open-sourced[3] and evaluated for learning gains within several courses [46].

2.5 Open Platforms

A combination of open resources, datasets, and algorithms in the form of an open platform has been rare. Cognitive Tutor's Geometry 1997 dataset is public, but the system itself is not open and not free to use. ASSISTments, an open platform, encourages outside researcher collaboration to run A/B experiments on different content-based approaches [20], but does not release the content of its platform, nor its codebase. We hypothesize that the same acceleration of research outlined in the sections above could be seen in the domain of platform design with the open-sourcing of a full-featured tutoring system and content library. The open-source nature of a system also fosters trust, encouraging wider spread student, teacher, and institution adoptions [28].

[2] https://www.etrialstestbed.org/resources/featured-studies/dataset-papers
[3] https://github.com/harvard-vpal/adaptive-engine

Table 1: Open-source projects supporting ITS-like features

System	License	Language	Supports Authoring	Adaptive Algorithm	A/B Testing
OATutor	MIT License	Python	Yes	Yes	Yes
CalcTutor	Apache 2.0	Python	No	No	No
GnuTutor	GNU GPLv2	C#	Yes	No	No
GuruTutor	Apache 2.0	F#	Yes	No	No
Joots	GNU GPLv2	Java	No	No	No
LogicITS	GNU GPLv3	Java	No	No	No
ThesisITS	Apache 2.0	Java	Yes	No	No

This is not to say that partial implementations have not been made available. Several open-source tutor repositories adopting the moniker of either adaptive or intelligent can be found online (Table 1). All, however, were short-lived projects at the unfielded prototype stages, implementing a small subset of ITS features and with no development activity registered in the past three years. Most had implemented immediate feedback and hints (LogicITS[4], Joots[5], and ThesisITS[6]) but with only one, ThesisITS - a more mature learning management system and java coding interface - having a documented content abstraction or authoring tool allowing for content extension beyond the hard-coded example problems. CalcTutor offered procedural generation of integration and differentiation questions with answer checking. GnuTutor [34] and GuruTutor [35], the only projects with accompanying papers, were inspired by past, proprietary discourse-based tutor interfaces. Another system, GRAPPLE [8], provides an open-source and adaptive hypermedia framework, but is explicitly not a tutor, as it only supports passive resource use, not problem-solving activities. None of the systems, at this time, adhered to the majority of ITS principles, and namely lacked support to assess skill mastery and adapt instruction accordingly, a core capability of Intelligent Tutoring Systems.

2.6 Content Authoring Tools

Various authoring tools exist to populate tutoring environments with learning material. Cognitive Tutor Authoring Tools (CTAT) has been the most developed toolset for producing ITS content with only minimal programming background [52]. For this reason, many tutor developers have built upon the CTAT framework due to flexible content authoring, creating example tools for a variety of learning tasks and domains. A well-known drawback to content authoring for ITS has been the time taken for content creation. CTAT has made progress on this front, reducing the estimate of 200-300 hours of development to produce content for one hour of instruction [2] to only 50-100 hours [52].

ASSISTments is another example of a tutoring system that has focused on authoring tools. These tools also match the 50 hour estimate, having shown to be around four times faster than original ITS authoring tools, while also not requiring any programming background whatsoever [44]. However, a new user must spend time learning the ASSISTments builder interface, as it is an original design allowing for nested hints and scaffolds. A simpler "quick builder" also exists within ASSISTments that simplifies the interface but does not provide the ability for scaffolding and hinting to be added [21].

[4]https://github.com/IgorFonck/logicits
[5]https://sourceforge.net/projects/joots
[6]https://github.com/robertoguazon/Thesis-ITS

Additional authoring tools such as ASPIRE have also emerged, with a focus on easing the creation of ITS [33]. Such tools aim to simplify the process in a form that allows domain experts who are not experienced in the realm of ITS or software engineering to create their own curricular content [49]. The downside of such tools, however, lies in them not being able to overcome old time burdens for content creation, resulting in development times similar to those CTAT previously helped overcome [49].

2.7 Crowdsourcing and Learnersourcing

The content found in the aforementioned OERs is an example of crowdsourcing, a "participative online activity" involving a proposer who reaches out to a group of individuals in order to request a "voluntary undertaking of a task" [15]. In most forms of crowdsourcing, a financial or other benefit is provided to possible participants to encourage the undertaking of the task. Due to the common need for an incentive and the necessary effort to find individuals, crowdsourcing on a large scale over a significant time period can be challenging [31]. In order to create a sustainable crowdsourcing environment, it is critical to manage work-allocation in a flexible manner, and ensure the incentive is encouraging for participants to remain on the project [31].

In addition to OERs, certain tutoring systems feature their own pool of crowdsourced and learnersourced content. As mentioned above, ASSISTments allows teachers to view, use, and sometimes build upon content from other users. This type of content has been shown to be beneficial for students during the learning process and validates the reliability of sourcing educational content from users knowledgeable of the subject [42]. Educational crowdsourcing has also seen success in a real-time environment with TeacherASSIST, a feature created by ASSISTments to allow teachers to provide more on-demand crowdsourced help [39].

Micro-task markets such as Amazon's Mechanical Turk have emerged as efficient crowdsourcing environments [27]. However, when it comes to educational content, crowdsourcing from such a platform could raise various concerns. Low quality material with errors could result in misconceptions, disengagement, and could even have negative socio-economic consequences by adversely impacting students' grades in formal education environments. Thus, there is a significant requirement in order to qualify as a contributor to educational crowdsourced content (such as being a teacher, mentor, tutor, and having experience with the subject matter). With these criteria being very particular, it falls outside the scope of typical crowdsourcing platform communities. Where there are appropriate community members, the cost for having such specific criteria increases considerably. Thus, educational platforms tend to resort to known communities with the appropriate background.

To avoid possible financial barriers while seeking qualified participants, one can shift to a more focused category of crowdsourcing called learnersourcing. Learnersourcing is suitable for content authoring tasks as it is focused on featuring a recent or current learner of the related academic subject as the individual participating in the voluntary crowdsourced task [25]. Additionally, learnersourcing revolves around the creation of systems and tasks related to the subject being learned and incorporates the experience and creativity of the learner for the successful completion of the task.

In the process of learnersourcing, individuals can be identified into three separate groups: contributors, beneficiaries, and members of the instructional team [47]. Contributors are the individual learners who create the necessary learning artifacts, while beneficiaries are the learners on the receiving end of such artifacts. The incentive structure for contributors lies in the "Generation Effect", students better recall information they have generated themselves [47]. Thus, the contributors also participate in learning, and strengthen their own understanding of the subject.

3 OPEN ADAPTIVE TUTOR

OATutor began development at the end 2019. Throughout the past three years, A/B testing capabilities, data listeners, LTI support for systems such as Canvas, new problem capabilities such as variabilization, and various other features have been added. OpenStax's three Algebra textbooks (Elementary, Intermediate, and College) have had questions from every section curated into the system and help added by the OATutor content team, with work underway to support Introductory Statistics (all under CC BY 4.0). The system has been piloted at a community college in seven classes over six terms (on the quarter system). In this section, we will introduce how OATutor's preliminary model was grounded on ITS design principles, before outlining the system and its various features.

3.1 Research-based ITS design choices

Intelligent Tutoring Systems have been researched for many decades with a strong foundation of design principles [4] that have led to robust learning gain evaluations [37]. In Table 2, we detail the initial feature set adopted for OATutor and the justification for adoption based on past research.

3.2 Content Structure

Inspired by Intelligent Tutoring Systems, OATutor uses a hierarchical structure for its content pool to generalize the expressiveness of existing popular tutoring systems in the literature (e.g., ASSISTments, Cog Tutor). The base unit of content is referred to as a problem which contains a title, a body, and a collection of subparts of the problem referred to as steps. An example is shown in Fig. 1 instructing the student to find the surface area of a cylinder.

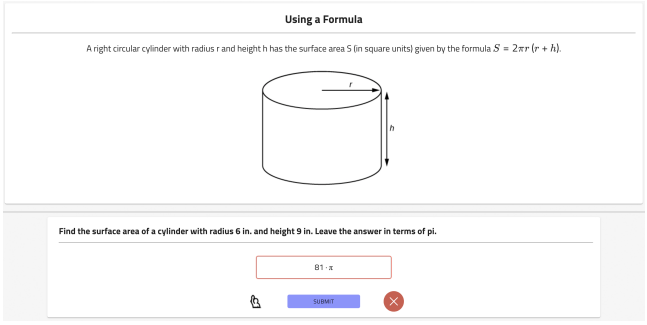


Figure 1: OATutor interface showing a surface-of-a-cylinder problem and problem figure, with a step that the student has submitted an incorrect answer to. The student can re-attempt or click the raise hand icon to open the tutoring pathway.

The step is the finest granular level, and the level at which students enter their answer. This structure inherently encourages content authors to provide varying levels of grain size of instruction. Each step contains a title, body, and an answer field which can be either a textbox (set to exact answer or algebra answer) or a multiple-choice selection. Steps are typically structured as independent subparts of the problem that can be solved in any order. To provide additional help, each step contains a collection of help items called a tutoring pathway. The step shown in Fig. 1 instructs the student to calculate the surface area of the cylinder. However, the student has a conceptual error with the order of operations and ignores the parentheses, which the system marks as an incorrect answer. The student can retry the problem again on their own or click on the raised hand icon to begin the step's tutoring pathway.

In Fig. 2, a step is decomposed into a mixture of hints and scaffolding. This serves to communicate the underlying goal structure within a problem step and walk the student through the problem solving process. Hints are unlocked and shown one at a time to reduce cognitive load. The hint unlocking order structure is configurable, with hint dependencies specified in the tutoring pathway content files. If a student's answer is incorrect, they can reattempt the problem or choose to open any number of help items. Students can leave the help items to (re)attempt the original step at any point. All questions have a bottom-out hint which displays the correct answer to help the student move on to the next question if the previous hints were insufficient. Bottom-out hints can be turned on or off in a global configuration file. In Fig. 2, the student is shown to have finished two hints and a scaffold. Upon solving the scaffold, the student realizes their error in the order of operations, skips the remaining hints, and correctly solves the step from Fig. 1.

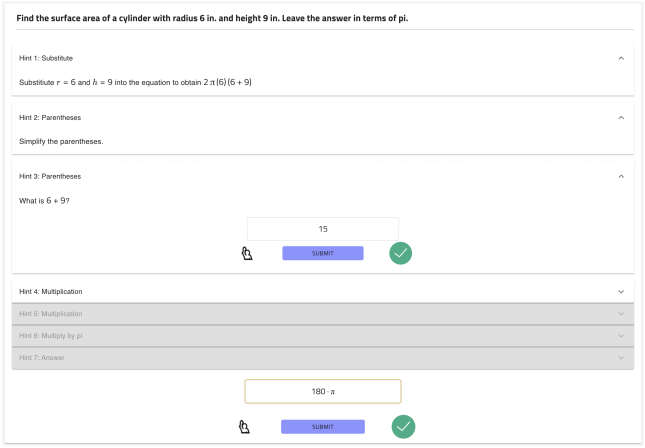
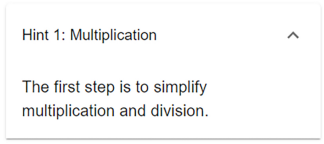
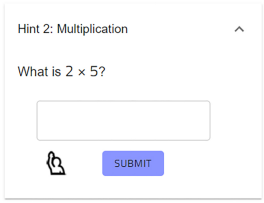
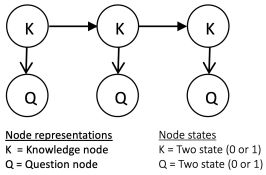
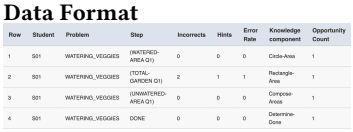
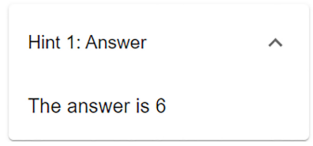
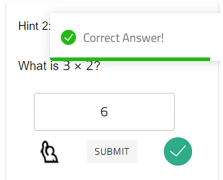


Figure 2: Tutoring pathway for the problem in Fig. 1

The content pool exists as a modular folder within the OATutor repository, with problems and their tutoring represented in JSON format (Figures 3 and 4). Content authors do not need familiarity with JSON and instead can author using a Google Spreadsheet. An automated conversion script is included and responsible for converting content in these spreadsheets to the JSON format required by OATutor's content pool.

Table 2: OATutor features incorporated based on ITS principles and past research

Hints 	Hints "provide instruction in the problem solving context" (ITS Principle 3) [4] with figures or text shown to students upon request [3]. They can be used to convey short declarative information. Alternatively, they can contain a full solution to a similar problem, called a worked example, which has been found to be effective for learning, particularly with tasks focused more on concepts than procedures [26].
Scaffolding 	Scaffolding are similar to hints but include one or more questions posed to the student. Most often, scaffolding are part of a tutor strategy called tutored problem solving, which breaks a problem down into multiple steps and in doing so can "communicate the goal structure underlying the problem solving" (ITS Principle 2) [4]. Several studies have found this approach to produced higher learning gains than a worked example hint strategy [13, 19], particularly with learning tasks focused on procedural knowledge [26], while other studies have found no significant difference [43].
Mastery Estimation 	ITS are adaptive in the amount of practice they prescribe to a student. The ability of the system to avoid under or over practice is dependent on how well it is able to estimate student mastery of a skill [9]. Knowledge Tracing [12], an algorithm based on a Hidden Markov Model, is used to estimate cognitive mastery for each skill in an ITS [45]. This assessment is important for the tutor to be able to withhold problems a student is not ready for yet (ITS Principle 8) and to advance them on to more abstract material when appropriate (ITS Principle 7) [4].
Data Format 	The data logged from student use of ITS have been the subject of large data mining competitions [48] and has provided fertile ground for education data science [17]. We adopt a tutor logging format familiar to researchers, that creates a row for each student interaction with the tutor [30].
Knowledge Component Model Skills Problems $\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$	A Knowledge Component (KC) model [1, 29] is a definition of the skills expected to be assessed and tutored in the system and the problems associated with those skills. It is prescribed that the skills be coarse-grained enough to "promote an abstract understanding of the problem-solving knowledge" (ITS Principle 4) but fine-grained enough to "represent student competence as a production set" (ITS Principle 1) and inform the adaptive selection of the next problem [4].
Variabilization person: Xia|Samantha r: 1|2|3|4|5 h: 6|7|8|9|10 ans: (r*h)pi	Variabilization allows many instances of a problem template to be generated based on variables taking on a predefined set of values or ranges of values. Problem answer checking and tutoring can also be based on instantiating of variables, and can help save content authors time [44].
Bottom-out Hints 	A bottom-out hint serves as the final hint for a problem and the final opportunity to "provide instruction in the problem solving context" (ITS Principle 3) [4] usually containing the answer to the problem. Bottom-out hints are usually provided to prevent the student from dwelling too long on a problem after completing the hints and scaffolds. They have been shown to improve student performance on problems following the problem where they were shown the bottom-out hint [18].
Immediate Feedback (Correctness) 	Upon submitting an answer the student is immediately informed of its correctness. This feature helps "provide immediate feedback on errors" (ITS Principle 6) [4] and has been shown to have a statistically significant positive effect on learning gain [13, 14].

```
// real11a.json

[ {
    "id": "real11a",
    "stepAnswer": ["2@{r}@{r}pi + 2@{r}@{h}pi"],
    "problemType": "TextBox",
    "stepTitle": "Find the surface area of a cylinder
        with radius @{r} in. and height @{h} in. Leave
        the answer in terms of pi.",
    "stepBody": "",
    "answerType": "arithmetic",
    "variabilization": {
        "r": [1, 2, 3, 4, 5],
        "h": [6, 7, 8, 9, 10]
    }
} ]
```

Figure 3: Problem step definition in JSON for the step shown in Fig. 1, implemented using variabilization

```
// real11aDefaultPathway.json

[  {
        "id": "real11a-h1",
        "type": "hint",
        "dependencies": [],
        "title": "Substitute",
        "text": "Substitute @{r} and @{h} into the
            equation to obtain $$2\\pi(@{r})((@{r})+(@{h
            }))$$"
    }, {
        "id": "real11a-h2",
        "type": "hint",
        "dependencies": ["real11a-h1"],
        "title": "Parentheses",
        "text": "Simplify the parentheses."
    }, {
        "id": "real11a-h3",
        "type": "scaffold",
        "problemType": "TextBox",
        "answerType": "arithmetic",
        "hintAnswer": ["@{r}"],
        "dependencies": ["real11a-h2"],
        "title": "Parentheses",
        "text": "What is $$\\left(6\\right)+\\left(9\\
            right)$$?"
    },
    ...
```

Figure 4: Tutoring pathway definition in JSON for the hints and scaffold shown in Fig. 2, implemented referencing the variables from Fig. 3

Content is tagged with skills at the step level. For OpenStax content, the learning objective of the problem within the lesson was used. In order to support easy definition and redefinition of skill mappings for researchers to experiment with, OATutor places these tags in a centralized configuration file called the skill model, shown in Fig. 5. This avoids the need to edit the fields of every individual step in the content pool. The skill model represents a matrix of all the steps and the skills they are associated with. The skill model matrix was stored in a javascript file in an earlier iteration of the system, but was later changed to a JSON file to better support a researcher participant who had explored re-tagging all OATutor content with skills from the US Common Core and desired a more structured skill association file format to output to.

```
// skillModel.json

[ {
    ...
    "real10a": ["classifying_a_real_number"],
    "real10b": ["classifying_a_real_number"],
    "real10c": ["classifying_a_real_number"],
    "real11a": ["evaluating_algebraic_expressions"],
    "real12a": ["evaluating_algebraic_expressions"],
    "real12b": ["evaluating_algebraic_expressions"],
    ...
```

Figure 5: Skill mappings configuration file (JSON). The step in Fig. 1, referred to as real11a, is tagged with the `evaluating_algebraic_expressions` skill.

Courses are the top level of content organization in the OATutor interface (Fig. 6), which contain lessons (Fig. 7). Instead of lessons explicitly containing problems, each lesson defines a list of skills, also called learning objectives, in the tutor. Problems are then adaptively chosen based on their association with the skills of the lesson according to the skill mapping, or KC model configuration (Fig. 5). The course, lessons, and learning objectives structure is designed to follow the structure of a typical textbook, intended to be integrated into the syllabus of a formal course. Fig. 8 shows how courses and lessons are defined using JSON in the system. Lessons can be configured to recycle problems in the event that a student has exhausted all problems associated with the skills of a lesson, but has not yet reached the mastery threshold. Variabilization increases the value of recycled problems (see section 3.4), changing the numbers within a problem each time it is shown. Additional lesson toggles of `giveStuFeedback` (correctness feedback), `giveStuHints` (allowing hints to be requested), `doMasteryUpdate` (using mastery learning as a stopping criterion), and `showStuMastery` (showing students their mastery estimate) are available and turned on by default. Researchers and teachers may like to set these to false when creating lessons that serve as tests.

Figure 6: OpenStax-based Algebra courses in OATutor

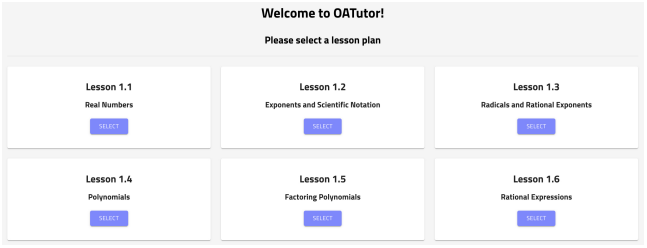


Figure 7: The first six lessons from OpenStax: College Algebra

```
// coursePlans.json

[
   {
      "courseName":"OpenStax: College Algebra",
      "lessons":[
         {
            "id":"lesson1.1",
            "name":"Lesson 1.1",
            "topics":"Real Numbers",
            "allowRecyle":true,
            "learningObjectives":{
               "classifying_a_real_number":0.85,
               "performing_calculations_using_the_
order_of_operations":0.85,
               "evaluating_algebraic_expressions":0.85
            }
         },
         ...
      ]
   },
   {
      "courseName":"OpenStax: Elementary Algebra",
      "lessons":[
         {
            "id":"lesson1.1",
            "name":"Lesson 1.1",
            "topics":"Introduction to Whole Numbers",
            "allowRecyle":true,
            "learningObjectives":{
               "identify_multiples_and_apply_
divisibility_tests":0.85,
               "find_prime_factorizations_and_
least_common_multiples":0.85,
               "use_place_value_with_whole_numbers":0.85
            }
         },
        ...
```

Figure 8: Courses and lessons configuration file (JSON). Lesson titles are defined here, as well as a list of the skills (or learning objectives) they should cover.

3.3 Mastery-based Adaptive Problem Selection

Bayesian Knowledge Tracing (BKT) [12] is used to model student mastery of skills. BKT models student knowledge using a Hidden Markov Model in which observations are the correctness of a student's answer for a problem. Each skill has four model parameters which are stored in a file called bktParams: $p^{(t)}_{\text{mastery}}$, p_{guess}, p_{slip}, p_{transit}. $p^{(t)}_{\text{mastery}}$ represents the probability of the student having mastered the skill at time t, p_{guess} represents the probability of the student solving a problem without having mastered the skill, and p_{slip} represents the probability of the student making a mistake applying a skill they have mastered. p_{transit} is used to compute the probability a student will master a skill at each answer opportunity. OATutor uses the Bayesian update shown in Equation 1 depending on the correctness of the student answer, where mastery at t is the model's prior belief before observing a response at time t and $t|ans$ is the updated posterior after ovserving a student's response:

$$p^{(t|ans)}_{\text{mastery}} = \begin{cases} \frac{p^{(t)}_{\text{mastery}} * (1 - p_{\text{slip}})}{p^{(t)}_{\text{mastery}} * (1 - p_{\text{slip}}) + (1 - p^{(t)}_{\text{mastery}}) * p_{\text{guess}}} & \text{Correct} \\ \frac{p^{(t)}_{\text{mastery}} * p_{\text{slip}}}{p^{(t)}_{\text{mastery}} * p_{\text{slip}} + (1 - p^{(t)}_{\text{mastery}}) * (1 - p_{\text{guess}})} & \text{Incorrect} \end{cases} \quad (1)$$

Each problem is manually tagged by content authors with a skill (ex. `evaluating_algebraic_expressions`). Each lesson contains skills and a specified target mastery threshold for each skill. After a student finishes a problem, the system chooses a new problem that the student has not mastered yet using a configurable heuristic function. The default heuristic function iterates through all problems and chooses the one with the lowest average $p^{(t)}_{\text{mastery}}$, averaged among all the skills the problem is tagged with.

Learning continues until all lesson skills have been mastered, at which point the system informs the student that they have mastered all skills. Progress is stored locally within the student's browser as a cookie, avoiding the need for a backend. Although this means that OATutor lacks cross-device progress tracking, the lessons are designed to be completed in a 30 minute session, mitigating this need. If the student exhausts the entire problem pool without mastering the skills, completed problems can be recycled.

3.4 Variabilization and Answer Checking

Problem variabilization is a feature that reduces the burden on content creators by allowing content to be instantiated with different numbers, creating more versions of content. Content authors specify all variables in the problem and the possible sets of values the variable can take on, as seen in the example in Fig. 3. A random integer is chosen as the index for all of the variable possible value arrays (the modulo operator ensures there are no index out of bounds errors for varying length value arrays). OATutor uses predefined groupings of variables to improve the student experience, since different combinations of numbers can significantly change the difficulty of a problem. An example of this would be that certain randomly chosen numbers will not divide evenly into whole numbers for fraction problems. Answers will often need to be stated in terms of variables given earlier in the problem. Variabilization can be defined at the problem, step, or hint/scaffold scope, similar to the variable scope of a computer program. Variable values at lower granular levels take precedence in the case of a duplicated variable name.

OATutor also supports localization in the form of global variable sets, which allow templated words to be replaced in all problems to accommodate different contexts. For example, if a problem contains the word "elevator" as the default, it may be desirable to use the word "lift" for a UK audience. Localization offers to reduce language confusing without undue content duplication, improving the quality of students' interaction with the platform.

The platform was first designed for math courses, so the system needed to allow content creators to typeset math expressions and also allow students to input math expressions. OATutor's frontend was created using React, enabling us to reuse common UI components from open-source React libraries such as MaterialUI, saving programmer development hours. LaTeX rendering libraries were integrated to support rendering mathematical expressions. Custom markdown syntax was formulated to denote LaTeX and inline images. For problems with textbox input fields, student answers are automatically converted to LaTeX in real time. This allows students to input their answers in a more natural way that requires no knowledge of LaTeX.

Algebra type answer-field checking is performed using Khan Academy's open-source computer algebra system (KAS) which can verify if two expressions are algebraically equivalent. This gives students greater flexibility in entering their answers into the system, avoiding the need for the system to specify input ordering or answer formats.

3.5 Extensibility and A/B Testing

OATutor was designed to be as modular and simple as possible in supporting randomized A/B testing. Researchers can conduct A/B testing by navigating to the desired logic within the source of the platform and inserting a conditional statement. Common A/B testing entry points exist in the platform as examples for researchers, such as A/B testing different BKT learning parameters and different tutoring pathways (Fig. 9). When a student visits OATutor, condition assignment is generated randomly and saved into the student's browser. All subsequent student actions will then be logged with the current condition for future analysis.

```
// App.js:

import { bktParams as bktParams1 } from './config/
    bktParams/bktParams1.js';
import { bktParams as bktParams2 } from './config/
    bktParams/bktParams2.js';

<ThemeContext.Provider value={{
    ...
    bktParams: this.getCondition() === 0 ?
        bktParams1 : bktParams2,
    tutoringPathway: this.getCondition() === 0 ?
        "HintPathway" : "ScaffoldPathway"
    ...
}}>
```

Figure 9: A/B testing code excerpt, showing support for testing different BKT parameters and tutoring pathways

3.6 Accessibility Standards

In an effort to make our system accessible to all users who wish to learn or conduct research with our platform, we have ensured that every page complies with the Web Content Accessibility Guidelines 2.0[7] (level AA) and the federal Revised Section 508 Standards[8]. To do this, the website is composed of modular components that were built to be accessible. As demonstrated in Fig. 10, our code has Accessible Rich Internet Applications (ARIA) attributes embedded in the application. Additionally, we tag our graphical components with alternative texts so that screen readers can easily identify figures in problems or buttons in our interface (Fig. 10). OATutor's accessibility compliance with the Revised Section 508 standards is documented using the Voluntary Product Accessibility Template (VPAT). Using the Section 508 version of the template, our publicly available document[9] lays out the required components OATutor has for educational institutes that receive US federal funding to adopt the system.

[7]https://www.w3.org/TR/2008/REC-WCAG20-20081211
[8]https://www.access-board.gov/ict
[9]https://cahlr.github.io/OATutor/static/documents/OATutor_Sec508_WCAG.pdf

```
// ProblemLayout/LessonSelection.js

<IconButton
    aria-label={`View Course ${i}`}
    aria-roledescription={`Navigate to course ${i}'s page
        to view available lessons`}
    role={"link"}
    onClick={() => {
        this.props.history.push(`/courses/${i}`)
        this.props.selectCourse(course)
    }}>
    <img src={`${process.env.PUBLIC_URL}/static/images/
        icons/folder.png`}
        alt="folder icon"/>
</IconButton>
```

Figure 10: Accessibility example, showing our IconButton component making use of descriptions, labels, and alt text

Going beyond the codified accessibility standards, our system incorporates many of the modern best practices for complex input transformations, dynamic content notifications, and error handling. Since users are able to enter mathematical symbols, operands, and matrices, the system makes use of the Math-quill library and a custom equation interpreter to generate English text for screen readers to consume. In the event that a user using a screen reader submits an answer to be checked, they will also receive real-time feedback via a "role=alert" notification popup. To ensure that component malfunctions do not affect the overall user experience, every component is wrapped with at least one error boundary. In addition to logging an exception event, the error boundary allows us to customize a fallback view for the system.

3.7 Ease of Adoption

OATutor was designed with simplicity in mind. It takes two clicks to set up an entire learning system, run a pre-made AB test, and deploy the system as a github-pages website on GitHub.io[10] . By offloading data storage needs to the users' browsers and making our models browser-friendly, OATutor does not need any servers beyond basic static file hosting. Nevertheless, OATutor is built with several features that could be enabled if the orchestrating user points the configuration file to their own or cloud hosted database.

3.7.1 Data logging. By default, a student's progress through lessons is stored in their browser using IndexedDB or WebSQL (whichever is available). However, a database is still useful for the purpose of storing data logs and other feedback entries for later analysis. As such, an administrator can opt into using Firebase's Firestore which is a NoSQL cloud database to reduce the time and complexity of researchers setting up an instance of OATutor.

Data logs are created as students submit answers to problem steps and scaffolds and unlock hints. Every student interaction with the platform is logged as its own entry to enable researchers and teachers to reconstruct the user's interaction history. This allows for hint usage analytics and efficacy of A/B testing to be evaluated. The data logs are stored in NoSQL JSON, accessible through the Firebase web interface and can be easily exported to a CSV. All fields are present in all event type logs, as researchers had

[10]https://cahlr.github.io/OATutor

Table 3: Log data of student answering step from Fig. 1

answerStep event Data Log

Field	Value	Field	Value
problemID:	"real11"	helpID:	null
stepID:	"real11a"	helpIsCorrect:	null
input:	"81*pi"	helpAnswer:	null
correctAnswer:	"180pi"	helpInput:	null
isCorrect:	false	helpFinished:	[0,0...]
condition:	0	siteVersion:	1.108
oat_user_ID:	999999999		
timeStamp:	"09-05-2021 11:18:00"		
skill:	evaluating_algebraic_expressions		

Table 4: Log data of student unlocking hint from Fig. 2

unlockHelp event Data Log

Field	Value	Field	Value
problemID:	"real11"	helpID:	real11h3
stepID:	"real11a"	helpIsCorrect:	null
input:	null	helpAnswer:	null
correctAnswer:	"180pi"	helpInput:	null
isCorrect:	null	helpFinished:	[1,1...]
condition:	0	siteVersion:	1.108
oat_user_ID:	999999999		
timeStamp:	"09-05-2021 11:21:23"		
skill:	evaluating_algebraic_expressions		

challenges exporting to CSV when NoSQL JSON formatted MOOC data contained different fields for different events [38].

Tables 3 and 4 show the fields of two example data log entries. The `helpFinished` field stores the number of hints that the user has unlocked at the current time as an array. An array value of 1 represents that the hint was unlocked, a value of 0.5 represents a scaffold that was unlocked but not yet solved, and a value of 0 represents a locked help item. The condition field refers to the random condition a user would be assigned from A/B testing (e.g., Fig. 9). Also included in each log entry, but not shown in the Tables for brevity, are the lesson and book title names the problem was selected from. The `oat_user_ID` is a non-identifiable random ID that is produced by the front end and is persistent across sessions with the same browser.

Users can also submit feedback reports on problems through the system UI for any tutor or content errors that may impede their progress. Although most errors are caught when vetted by content editors or through automated checks, students do find and report errors they encounter.

3.7.2 Data for BKT. The format of OATutor's data makes it convenient to train knowledge tracing models, such as with pyBKT, an open-source Python library of BKT models [5]. The log data format can be easily used to fit parameters in just a few lines. In Fig. 11, data from OATutor is loaded as a CSV. Using pyBKT, a model is initialized. A BKT model is trained on all the unique skills in the dataset, and the learning parameters (prior, learns, guesses, slips, and forgets) are fit for each skill. The learned values of each parameter per skill are shown in Table 5.

Table 5: Skill parameters after fitting to OATutor pilot data

pyBKT sample outputs

Skill	Prior	Learns	Guesses	Slips	Forgets
evaluating_algebraic_expressions	0.70324	0.02174	0.48241	0.17582	0.00000
finding_x_intercepts_and_y_intercepts	0.45737	0.31908	0.07976	0.32756	0.00000
classifying_a_real_number	0.29347	0.42512	0.78437	0.34095	0.00000
the_discriminant	0.11926	0.17466	0.67566	0.51459	0.00000
solving_a_rational_equation	0.677738	0.07468	0.46928	0.25739	0.00000

```
1
2  oat = pd.read_csv('oat-full-data.csv')
3
4  model = Model(seed=42, num_fits=1, parallel=True)
5
6  paramslst = []
7  for skill in oat['skill_name'].unique():
8      model.fit(data=oat, skills=skill, num_fits=20)
9      paramslst.append(model.params().values)
10
11 pd.DataFrame(columns=['prior', 'learns', 'guesses', '
       slips', 'forgets'], data=np.hstack(paramslst).T).
       set_index(oat['skill_name'].unique()).rename_axis('
       skill')
```

Figure 11: An excerpt of the model fitting code using pyBKT

3.7.3 LMS Interoperability. OATutor supports the Learning Tools Interoperability (LTI) standard to integrate the tutor with classroom Learning Management Systems (LMS) such as Canvas. This allows teachers to view student mastery of skills within the tutor from their LMS gradebook. To support LTI, we created a middleware server implemented with Node.js to handle LTI authentication. Teachers simply specify the middleware as an external app when creating assignments, then link to a lesson on OATutor through a graphical selection screen. Students can then click on the assignment link when logged in to their LMS account to be redirected to the respective lesson on the platform. This removes the necessity for students to create a separate account on the platform and also simplifies the platform, avoiding storage and management of user credentials.

When OATutor is accessed by students via LTI, the same logging format applies; however, the additional fields of unique `course_id` and unique `lms_user_id` are populated, both of which are anonymized but persistent across devices and sessions as per the LTI protocol. The field `course_name` is also populated, which reflects the name of the course set by the teacher in the LMS.

Scores are passed back to the LMS as students finish problems. These scores are available for teachers to view in the gradebook of the LMS, which shows a detailed breakdown of the student's mastery probability for each skill within the lesson. Each skill score is calculated by dividing the student's mastery by the mastery threshold, capped at 100%. The overall score is calculated as the average of all skill component scores. An example grade report is shown in Fig. 12. This also shows a detailed log of actions in the lesson associated with the student to support the teacher in diagnosing a student's engagement level and quality of understanding, by viewing the history of answers they have given and their pace of interactions with the system.

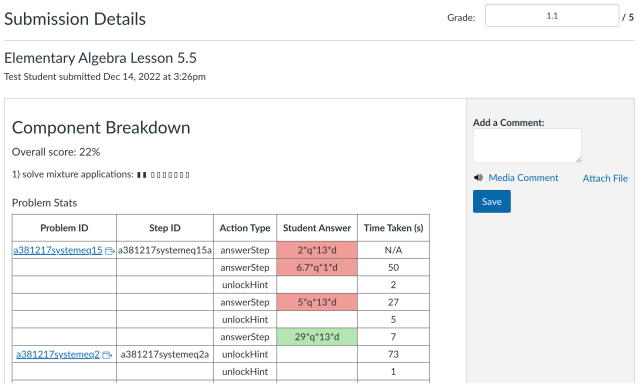


Figure 12: An example score breakdown and student action log in Canvas, accessible in the teacher's gradebook

3.7.4 Usage Statistics. Since the first deployment, the system has received over 70,000 problem submissions, of which 5,000 were from pilot courses - identified by the presence of a `lms_user_id` in the logs. Through these pilot courses at a local community college, we found 150 unique students using the system and 10 who had completed an entire course (predicted via their mastery). Thanks to our early adopters and educators, we have also been able to review nearly 2,000 pieces of feedback and subsequently made significant changes to the tutor and its content.

4 CREATING THE CONTENT LIBRARY

Creating three textbooks of adaptive content for OATutor required building a mini publishing framework (Fig. 13). Two main roles are in charge of content creation in OATutor: creators and editors. Creators curate problem content from acceptable sources and then author hints and scaffolds for the problems. They are selected from a pool of experienced learners who have interacted with the subject previously. Editors serve as reviewers and quality checkers for content made by creators, while also sometimes contributing content on their own. Editor selection was not only based on someone having learned the material, but also having prior experience with tutoring it, usually resulting in editors being college and university students. In the context of OpenStax's College Algebra, the first book we undertook, a total of 16 creators participated throughout the project, with about five active at a time. A total of five editors participated, with an average of two active at a particular time. With creators signing up for an average of four hours of work per week, the entire book was completed in about 6 months.

4.1 Selection of Creative Commons Source

In order to ensure the quality of the curated content, we established a methodology for selecting qualified resources for content curation. Our examination of various Open Educational Resources is shown in Table 6. Multiple OERs were selected through research on the web, including searching databases such as Amazing Educational Resources[11]. First, we examined the license provided by each OER. The goal was to identify which resources had usable creative commons licenses that would allow for curation of their content. Then, it was critical to identify the types of resources commonly found in the OER. Our goal was to find an active pool of problem resources that could be transcribed. The next step was analyzing if the provided problem resources offered help. Already having some sort of answer or explanation could significantly reduce the time of curation, since less time would need to be spent on creating hints and scaffolds from scratch. Finally, we identified the types of community input available for the OER, which would allow us to better determine the quality of content available. After examining all these variables, we compiled a score for each resource we examined. We marked the top five scores in green, and the scores that came close to the top five with yellow, and in our case, we chose OpenStax as most suitable.

[11] https://www.amazingeducationalresources.com

4.2 Building Community

Our system utilizes an aspect of crowdsourcing bordering on the definition of learnersourcing to create a sustainable content-authoring environment. Our process involves the recruitment of qualified learners to help with content curation of the desired respective subject (such as college-level math students to help with algebra content). We request a "voluntary undertaking of a task" while having our group of individuals who are contributors be former learners of the subject. However, we do not fully meet the definition of learnersourcing, as our contributors may not necessarily have recently learned the material they are authoring but instead may have mastered it previously in addition to having tutoring experience on the subject. Furthermore, the main benefit provided to said contributors is not rooted in learning. The qualified learners become involved through an Undergraduate Research Apprenticeship program providing course credit, and join with the motivation of collaborating with the sponsoring research lab.

After a learner has been recruited as a content creator, they undergo an introductory training course, consisting of an introduction to the system, a video guide on how to create problems, and an introductory "assignment" asking the creator to make five sample problems. The goal of the course is to prepare creators for their task in a time-efficient manner, while providing them with a hands-on introduction to the system through the sample problems.

In order to focus on fostering a healthy content curation environment, the content team held weekly half-hour meetings. This allowed authoring issues to be voiced and gave the opportunity for editors to directly respond to questions from the content creators.

Additionally, the community building that took place fostered a collaborative crowdsourcing environment to help meet the expectations of participant content authors while also allowing them to individually convey critiques and applicable feedback, a necessary factor in learnersourcing-like environments [7]. Furthermore, the meetings served as an integral method to keep the team engaged. Communication was a prime factor in showing project activity in addition to consistent feedback through emails, allowing further room for discussion. This in turn allowed emulation of a learnersourced environment as the crowdsourced community of content authors were given the foundations to interact with the content and apply their knowledge, though their incentives may not primarily be focused on learning.

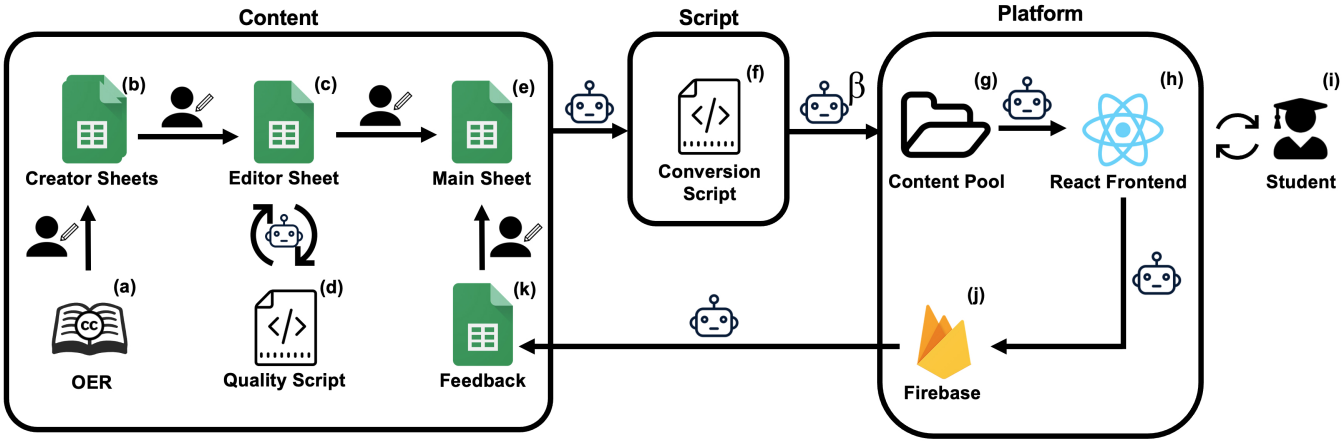


Figure 13: The stages of the content authoring and publishing pipeline

Table 6: OER websites surveyed and the criteria we considered when choosing the source of our question content

OER Name	Commercial License?	Non-Commercial License?	Problem Resources?	Problem Resources With Help?	Math Content?	Community Input?
OpenStax	Yes	Yes	Yes	Yes	Yes	No
Illustrative Mathematics	N/A	Yes	Yes	No	Yes	No
OER Commons	N/A	Yes	Yes	Yes	Yes	Yes
Curriki	N/A	Yes	Yes	Yes	Yes	Yes
National Science DL	N/A	Yes	Yes	Yes	Yes	Yes
MIT Open Courseware	N/A	Yes	Yes	Yes	Yes	No
S.O.S. Mathematics	N/A	N/A	Yes	No	Yes	No
Khan Academy	N/A	Yes	Yes	Yes	Yes	Yes
BiteScis	N/A	N/A	Yes	Yes	N/A	Yes
teachit Maths	N/A	N/A	Yes	Yes	Yes	Yes
teachit Science	N/A	N/A	Yes	Yes	No	Yes
OpenLearn	N/A	Yes	Yes	No	Yes	Yes
BC Capus OpenEd	N/A	Yes	Yes	No	Yes	Yes
ACT Academy	N/A	N/A	Yes	Yes	Yes	No
CUNY Academy Works	Both	Both	Yes	Yes	Yes	No
Academic Earth	N/A	N/A	No	No	Yes	Yes
Stem Resource Finder	N/A	N/A	Yes	Yes	Yes	No
Math Res. for Students	No	No	Yes	Yes	Yes	No
LUMEN Open NYS	Both	Both	Yes	No	Yes	No
OASIS	N/A	N/A	Yes	Yes	Yes	No
iBiology	N/A	Yes	No	No	No	Yes
MERLOT	N/A	N/A	Yes	Yes	Yes	Yes
Math Video Library	No	No	Yes	No	No	No
QUBES	N/A	Yes	Yes	No	No	Yes
Xpert	No	No	Yes	Yes	Yes	No
ELA Free Resources	No	No	Yes	No	No	Yes
Wisc-Online	N/A	Yes	Yes	No	Yes	Yes
Transum Mathematics	N/A	N/A	Yes	Yes	No	No

4.3 Author Spreadsheets and Conversion

Content creators used Google Spreadsheets to author content, first transcribing OER problems to the sheet and then adding appropriate hints and scaffolding to them. A sheet consisted of the columns: `Problem Name`, `Row Name`, `Title`, `Body Text`, `Answer`, `answerType`, `KC (skill)`, `Image url (optional)`. Generally, a single tab of a sheet corresponded to a lesson in a book. Creator lesson tabs were copied to Editor sheets, which were copied to the Main Spreadsheet, which was populated with an entire book's lessons. Using Google apps had the added benefit of creators being able to see each other's work. OATutor's authoring tools make use of a script to automate conversion from problem data in the tabular sheets to JSON format in the problem pool for the frontend to render, allowing creators to curate material without requiring CS knowledge. The sheet must be shared with a Google service account to allow the script to make google sheet API calls. The script also automatically converts plaintext math expressions to LaTeX using a library called pytexit.

Another feature of the script is automated feedback. The script runs a quality check daily to ensure that there are no errors in the sheet that would cause a runtime error in the tutor. Another series of heuristic-based checks are also performed, such as ensuring that multiple-choice questions have multiple answer options. The outputs from these quality checks are automatically placed inline with the problems in the google sheets, allowing content editors to recognize and correct them.

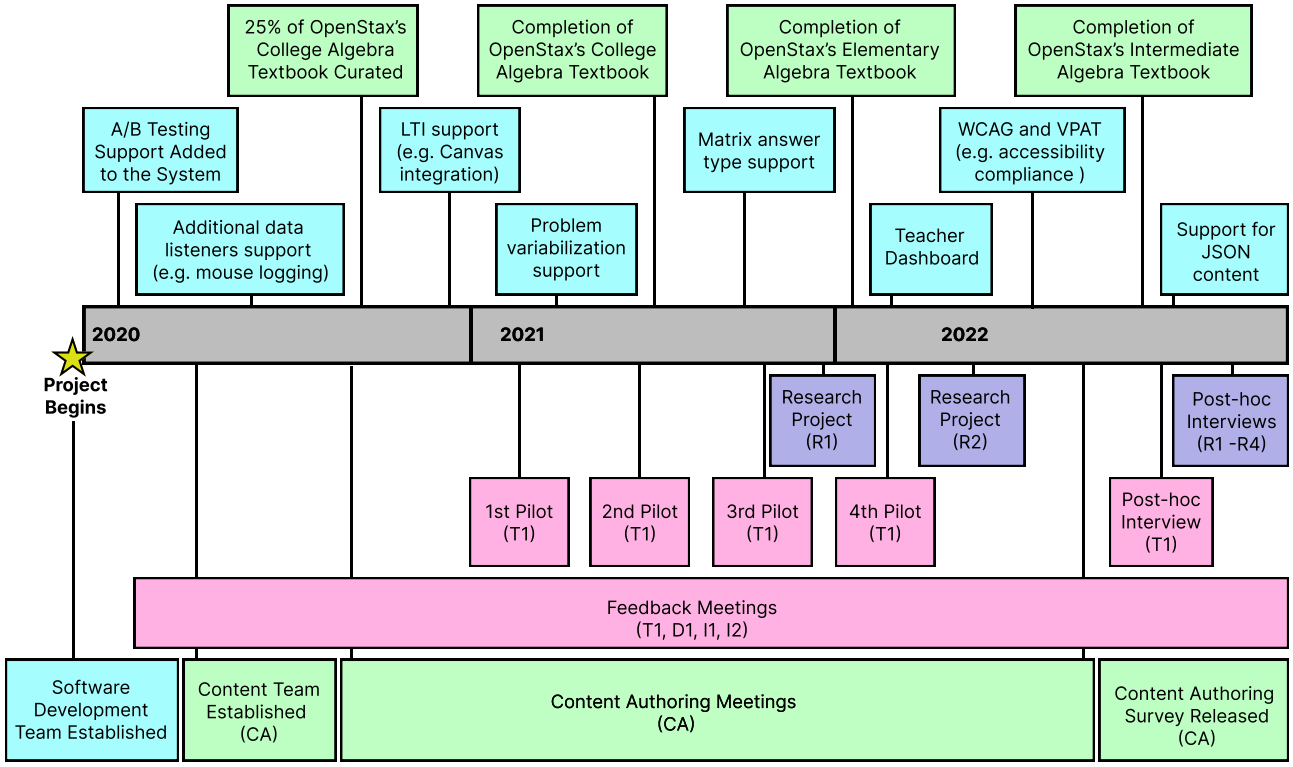


Figure 14: The three-year timeline of OATutor's development cycle. Feature and content milestones are above the timeline, while participant data collection opportunities are below.

5 METHODOLOGY

Our research through design methodology was carried out with three main stakeholder groups: content creators, teachers, and researchers over three years (Fig. 14). By expanding design participation to these three primary groups, we were able to better understand how each of our target stakeholders would interact with the system and to uncover their remaining needs. In this section, we describe the participants and methods used for collecting data from them. We then outline how each stakeholder group contributed to the iterative design decisions of OATutor to date.

5.1 Participants and Study Context

Our content team consisted of 25 total undergraduate students participating for a semester or two over three years. The team was led by the third author, a sophomore majoring in Mathematics with a minor in Education at the time of joining, who continued as a Master's student in Education in Fall 2022. He was an unpaid intern on the team for the first two months, then transitioned to an hourly paid position onwards. The 25 content team members joined through an Undergraduate Research Apprenticeship Program (URAP) at UC Berkeley (on a semester system), with the exception of two students who joined from the local community college under an internship funded by their college. The URAP program provides course credit for a semester to students accepted into research labs and allows for an extension to a subsequent semester.

The content team held weekly half-hour meetings in order to ensure authoring issues were voiced, in addition to allowing editors to directly respond to questions from the content authors. At the beginning and end of the semester, the first author also joined the content meetings to sync on any tasks, blockers, insights, and high-level ideas for teaching content. Content team members' feedback was crucial as they were the first to encounter the material in OATutor and identify anything that they believed was missing or confusing.

The software development team consisted of five members. The second author, a junior in EECS at time of joining, was lead developer from the start of 2020 through mid 2021. The fifth author, a sophomore in EECS at the time of joining, became lead developer from mid 2021 onwards. Two additional EECS students contributed additional code support and the first author set design objectives and supervised development.

We conducted four pilots over 7 course offerings with a local community college algebra teacher (T1). She was introduced to the team by the college's Dean of Math, Science, and Engineering (D1) as a teacher who "always has their eyes and ears on the newest pedagogies using technology". Our pilot teacher taught online math

courses for the past 15 years and advocates for using open OERs in her classroom to alleviate the financial burden on students. Because of her experience with OERs and use of OpenStax Algebra textbooks, she was identified as an ideal teacher participant to give feedback.

Finally, we collaborated with four researchers whose academic and research interests aligned with OATutor. Two researchers (R1 and R2) were also recruited through URAP and distinguished by their research interests at the intersection of data science and education. The other two researchers were Ph.D. candidates, one in Education (R3) and another in Electrical Engineering and Computer Science (R4). R3 was co-advised by the first author and R4 was a past collaborator. Both had deep experience in computer tutoring but had not been involved in OATutor's development. As first consumers of our data exports, the researchers' feedback was critical in identifying valuable additions, challenges, and future design opportunities.

5.2 Data Collection

We collected data from our participants using a combination of interviews, meeting notes, system-submitted user feedback, and surveys.

5.2.1 Interviews. We conducted a one hour interview with the piloting teacher who had used the system for six quarters (T1), two undergraduate students who had conducted research with the system for two semesters (R1 & R2), and two fourth year PhDs who were evaluating the applicability of OATutor to their future dissertation work (R3 & R4). All five participants were interviewed separately in the third year of the study by the fourth author using a semi-structured interview protocol with fixed questions and follow-up.

5.2.2 Meeting notes. Notes were taken on the feedback voiced by members of the content authoring team during weekly content team meetings which took place starting in Fall 2020 and were ongoing. Additionally, notes were taken during monthly co-design meetings that were held with T1 and D1, which took place starting in Summer 2020 and were also ongoing throughout the project. These meetings were also attended by collaborators from a small non-profit, I1 and I2, who worked with D1 on complementary education technology projects.

5.2.3 Student feedback. In the first two terms of piloting OATutor in classrooms, students left feedback about the system in a forum created in the Canvas Learning Management System for their Algebra course. Starting in the third term of piloting, students were asked to submit feedback via a feature in the system that logged the student's comments and contextual information, such as the current problem being displayed.

5.2.4 Surveys. The 25 past and present OATutor authoring team members were surveyed on their motivation for joining and staying on the team. They were also asked to recall how long they spent in our new author training and how long it took them on average to complete the creation of a problem in OATutor.

6 RESULTS

6.1 Case Studies of OATutor Supporting Researchers

As our first case study, we interviewed R1 whose research involved producing the CSV export for BKT models to be trained by R2. It also involved producing analytics on the most accessed lessons by students in the pilot. As a result of her project, several attributes were added to the logging schema to make analysis easier. These changes included added attributes to easily identify when a question was being answered for the first time. This was important because BKT only considers first attempts at each question. Secondly, R1 previously had to access the content files in order to add lesson titles to the usage analytics. To avoid this inconvenience, the logging data schema changed and the database was back-populated with the lesson name and course name associated with each action.

As a second case study, we interviewed, R3, regarding the suitability of the system for conducting research related to their dissertation. They raised that it would be desirable to A/B test the system's default tutorial help (static hints and scaffolds) against an experimental dynamic hint generation algorithm, similar to [36]. To implement and conduct A/B testing for this feature, only one line of core logic had to be edited to include a handler for executing functions (Fig. 15). An example of a generator that we can A/B test (Fig. 16) is demonstrated in Figure 17 where it uses the problem and answer context to give a hint dynamically generated by the black box function 'modelGenerateHints'.

```
1 // ProblemLayout/ProblemCard.js:
2
3 -       this.hints = this.step.hints[context.
      tutoringPathway];
4 +       if ( typeof(context.tutoringPathway) === "
      function") {
5 +           this.hints = context.tutoringPathway(context
      , this.step, this.state.inputVal)
6 +       } else {
7 +           this.hints = this.step.hints[context.
      tutoringPathway]
8 +       }
```

Figure 15: Simple extension of the hint selector module to support R3's requested dynamic hint generation feature

```
1 // App.js:
2
3 + import dynamicHint from "src/config/hintGenerators/
      dynamicHint";
4
5 <ThemeContext.Provider value={{
6     ...
7 -    tutoringPathway: "DefaultPathway"
8 +    tutoringPathway: this.getCondition() === 0 ?
9 +        "DefaultPathway" : "dynamicHint"
```

Figure 16: First step of A/B Testing of R3's "dynamicHint" against the default tutoring pathway

```
// hintGenerators/dynamicHint.js

/**This hint generator is an example that uses the user's
      current question attempt to generate new hints
 * @param appContext contains the context/state of the
      app at the time of function execution
 * @param problemContext contains the context/state of
      the problem at the time of function execution
 * @param studentAnswer is the raw input of what the
      student have answered
 * @returns {Hint[]}
 */
export default function dynamicHint(appContext,
     problemContext, studentAnswer) {
    const { bktParams } = appContext
    const { knowledgeComponents, hints } = problemContext

    console.debug(`bktParams: ${bktParams},
        knowledgeComponents: ${knowledgeComponents}`)

    return modelGenerateHints(problemContext,
        studentAnswer)
}
```

Figure 17: An example dynamic hint generation function

6.2 Findings from Pilot Teacher Feedback

During OATutor's deployment in our pilot teacher's classes, T1 noted in our interview with her that the students who interacted with OATutor were generally the more motivated ones who wanted extra practice. Students would leave feedback on their experience with the system, including bugs and questions, on Canvas's discussion board in the first two pilots. The content team lead would then relay the feedback to editors, who would subsequently make the necessary fixes. However, because students did not receive any credit for using the system, their usage waned as the term progressed. T1 recalled noticing frustration among some students as they experienced content bugs, particularly in the first few pilots. She remarked that some students' frustrations were also due to not knowing how to enter certain operators in their answers, such as square root, sharing that, "Math is already frustrating. I didn't want to require them to use the system and be frustrated with the technology too." However, she recognized that as the system got used more and received more feedback, the interface and experience improved and she continues to use the system in her classes.

Our pilot teacher's general feedback in her meetings with the monthly OATutor co-design team was to make the system's features similar to other web systems, particularly learning management systems, for consistency and ease of use. The ability to see a student's log of actions was a feature she observed in other systems. Additionally, other homework systems she has used with her students have rich text editors for students to input equations and symbols into their answers. She also remarked that all software she adopts in her classroom must be accessible and meet ADA requirements. The ability to see student progress on a teacher-facing dashboard was also important to her to track time spent on mastery per student. Finally, an expansion of question types supported such as graphing questions was expressed as helpful to include to allow for a wider variety of topics to be practiced. In her ideal world, she would like more control over the questions, in terms of adding, removing, and editing problems to cater to her students.

6.2.1 Design implications. We did our best to address the needs expressed by T1 throughout the different pilot stages. As described in section 3.7.3, we implemented a teacher dashboard with detailed logs of students' actions and implemented WCAG standards to be ADA compliant. Additionally, to more rapidly catch content bugs, we implemented an in-tutor feedback system so students could log any bugs observed directly, with our logs noting the ID of the problem and the comment. Additionally, we increased the number of editor passes of our content before release and increased our content authors' ability to see their problems rendered in the developer system soon after authoring to give them an opportunity to catch render issues. Finally, to address student frustration with the entry of mathematical operators, we produced a help sheet on answer entry and linked to it on every problem page. These design choices were direct responses to T1's feedback. Nevertheless, there are still design implications that are unaddressed by us that would be useful for the research community to investigate moving forward. For example, T1's desire for her students to have a rich text editor. Off-the-shelf editors will be explored, but allowing students to express equations precisely and with little friction is an open usability challenge. Additionally, how to engineer a content system with the constraints of an Intelligent Tutoring System that also allows teachers to directly edit or suggest edits to content is an open challenge.

6.3 Findings from the Content Team

At each content team meeting, the content team lead recorded feedback and discussed suggestions with the first author. In addition, the content team lead released a survey at the end of the students' research programs to collect feedback on their overall authoring experience. The form was sent to 25 members who had been part of the OATutor editor team. Six members were still part of the team at the time and 18 members had, historically, continued past their first semester. A total of 16 responses were received, one of which was excluded due to being incomplete. Survey responses were anonymously submitted. When asked what factors led the content team to join OATutor (multi-response allowed), 80% of respondents joined because of the social good mission of the project, 53.3% joined to add a UC Berkeley project to their CV, and 53.3% joined because of the open-source-free nature of what OATutor is producing. With the majority of respondents joining because they believed in the mission of the project, it was possible to count on their feedback and commitment to bettering the content library.

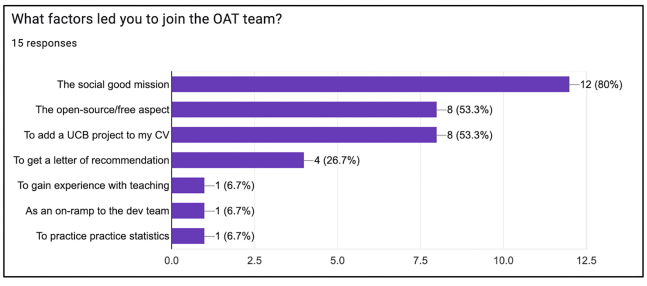


Figure 18: Multi-response survey results on factors that led students to join the content team

The form also asked the team members to estimate how long the introductory training course for the system took (free-response allowed, with time estimated to the nearest hour), as well as the average time to create one problem (free-response allowed, with time estimated to the nearest minute). Among the 15 measured responses, the average introductory course length was 2.27 hours (with a standard deviation of 1.10) and the average for problem creation was 11.03 minutes (with a standard deviation of 9.11). These numbers are in line with ASSISTments training and creation times, which cite four hours of training and problem creation time of 25.4 minutes [51].

During the weekly meetings, two themes arose from feedback that would inform iterations of the authoring framework. Two members of the team reflected on stylistic differences they would observe among problems within lessons. Among these were differences in terminology as well as problem-solving strategies used in similar problems stemming from each problem being authored by a different member of the team. While students being exposed to several approaches to the same problem could help them generalize, it also could be overwhelming when just starting to learn the subject. The second theme was content authors voicing a desire to see what problems looked like rendered in the system shortly after authoring content in order to help catch mistakes. At the time, content was only rendered every 4-7 days.

6.3.1 Design Implications. In order to mitigate differences in authoring styles, the team decided to assign one author per chapter. Upon further review of the chapters, the team saw a decrease in differences among stylistic choices, helping to lessen confusion for students with the problem at hand. Additionally, to address the content authors' feedback that they would like to see how the problem looked in the system ideally shortly after authoring, the team leads created a development server where their content could be published on a nightly basis. The content team lead noticed this feature was used extensively by the rest of the team after observing a decrease in rendering-based mistakes in problem authoring. Content authors would also contact the content team lead whenever the development server was down, indicating they used this feature to check authoring issues that editors would normally be tasked with fixing.

7 DISCUSSION AND FUTURE WORK

In this paper, we presented a full-featured open-source adaptive tutor based on principles. We have used our content pipeline to efficiently curate a content library by enlisting recent learners. Our library covers three OpenStax Algebra textbooks (Elementary, Intermediate, and College), and can be easily extended to include additional material by using our spreadsheet authoring tools or directly writing to our JSON problem format. To test our system and gather feedback, we deployed OATutor to several Algebra classroom settings using LTI for LMS integration with Canvas. Teachers were under no obligation to begin using or continue using our system and one teacher has used OATutor for six terms now with another teacher requesting to try OATutor, demonstrating that our system is gaining traction.

Through a research-through-design methodology, we gathered three main takeaways from our stakeholder groups. First, OATutor demonstrates basic usability as a system that teachers can use for students to supplement their knowledge base. T1 continued using OATutor for six terms without incentives from the team and continues to use the system to date. Second, our authoring system is reasonably efficient for authors to transcribe content. In the OATutor content authors survey, responses (N=15) indicated that authors take 2.27 hours (with a standard deviation of 1.10) for training and 11.03 minutes (with a standard deviation of 9.11) for creating each problem. Both training and creation times are in line with ASSISTments. Third, OATutor is researcher-friendly in providing an easily adoptable platform for researchers to conduct experimentation with. R1's project improved log files and R2's project improved KC model files for future researchers' analyses (see section 3.2). R3's research use case demonstrated OATutor's extensibility with respect to A/B testing of experimental dynamic hint generation strategies.

In order to better understand what limitations OATutor may still have for some researchers, we interviewed R4. R4's research interests were in training users in virtual/augmented reality (VR/AR) in domains where collaborative physical teamwork is necessary. In order to execute their research, they are in search of an interface that could accommodate both cognitive and motor skills. They need an interaction between a VR headset and an ITS, with the ability for the ITS to send a training program scenarios to the VR headset. R4 believes OATutor should be able to hold training content to then subsequently display that to users. While OATutor can be modified to suit R4's needs, it would take a significant effort but one that is not overly daunting due to the design of the existing architecture.

For future work, we plan to run an A/B test using OATutor and ASSISTments with teachers to explore the affordances and limitations of spreadsheet and GUI interfaces in relation to authoring content with images, special characters, hints, and scaffolds. We also plan to conduct a more formal study following a medium-sized cohort of educators to garner more feedback and identify the different modes in which OATutor is used (as homework, enrichment, or for exams). We would also like to conduct a study to assess how researchers make use of our platform in running A/B testing of their own and what other design optimizations we can make to support those tests.

Educators may want to combine content from different OER sources or align existing content to a new taxonomy or curricular standard. Research could be done to evaluate the utility to students and teachers of machine assistance with labeling and reorganization of content in OATutor's library around a standard other than the one given by default. Currently, one of the most time-consuming parts of curating content is the creation of hints and scaffolds. Machine learning could be applied to generate draft help items and reduce the amount of time spent authoring per problem.

We see OATutor as having the potential to contribute to inventive HCI research [22], in enabling investigators to address their domain-specific and cross-disciplinary questions through the development and deployment of our tool. OATutor is both an adaptive tutor and a curated content library, revealing a system that brings together a set of capabilities to lower barriers to experimentation. Further research on OATutor usage is needed to study the magnitude and types of impacts the system can have on the broader research community.

ACKNOWLEDGMENTS

We would like to thank OpenStax for producing high-quality, open textbooks under the Creative Commons Attribution 4.0 International license (CC BY 4.0) which allowed for our adaptation of material. We would also like to thank the Office of Undergraduate Research & Scholarships at UC Berkeley for facilitating the Undergraduate Research Apprenticeship Program that supported our content team members. Additionally, we would like to thank RJ Pimentel and Tinna Liu for their contributions to the tutor codebase and to the CHI reviewers of this paper for their exceptionally thoughtful feedback which led to meaningful additions. Finally, we would like to thank the participants in this project who contributed to the design decisions of the system. This work was partially supported by an educational data science grant from Schmidt Futures. This research was approved by the UC Berkeley Committee for the Protection of Human Subjects under IRB Protocol 2021-10-14690.

REFERENCES

[1] Vincent Aleven and Kenneth R Koedinger. 2013. Knowledge component (KC) approaches to learner modeling. *Design Recommendations for Intelligent Tutoring Systems* 1 (2013), 165–182.

[2] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. 2006. The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. Springer, 61–70.

[3] Vincent Aleven, Ido Roll, Bruce M McLaren, and Kenneth R Koedinger. 2016. Help helps, but only so much: Research on help seeking with intelligent tutoring systems. *International Journal of Artificial Intelligence in Education* 26, 1 (2016), 205–223.

[4] John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. 1995. Cognitive tutors: Lessons learned. *The journal of the learning sciences* 4, 2 (1995), 167–207.

[5] Anirudhan Badrinath, Frederic Wang, and Zachary Pardos. 2021. pyBKT: An Accessible Python Library of Bayesian Knowledge Tracing Models. In *Proceedings of the 14th International Conference on Educational Data Mining*. 468–474.

[6] Meltem Huri Baturay. 2015. An overview of the world of MOOCs. *Procedia-Social and Behavioral Sciences* 174 (2015), 427–433.

[7] Sameer Bhatnagar, Amal Zouaq, Michel C Desmarais, and Elizabeth Charles. 2020. Learnersourcing Quality Assessment of Explanations for Peer Instruction. In *European Conference on Technology Enhanced Learning*. Springer, 144–157.

[8] Paul De Bra, David Smits, Kees van der Sluijs, Alexandra I Cristea, Jonathan Foss, Christian Glahn, and Christina M Steiner. 2013. GRAPPLE: Learning management systems meet adaptive learning environments. In *Intelligent and adaptive educational-learning systems*. Springer, 133–160.

[9] Hao Cen, Kenneth R Koedinger, and Brian Junker. 2007. Is Over Practice Necessary?-Improving Learning Efficiency with the Cognitive Tutor through Educational Data Mining. *Frontiers in artificial intelligence and applications* 158 (2007), 511.

[10] Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, and Jaewe Heo. 2020. Ednet: A large-scale hierarchical dataset in education. In *Proceedings of the 21st International Conference on Artificial Intelligence in Education*. Springer, 69–73.

[11] Albert Corbett and Holly Trask. 2000. Instructional Interventions in Computer-Based Tutoring: Differential Impact on Learning Time and Accuracy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands) *(CHI '00)*. ACM, 97–104. https://doi.org/10.1145/332040.332412

[12] Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction* 4, 4 (1994), 253–278.

[13] Albert T. Corbett and John R. Anderson. 2001. Locus of Feedback Control in Computer-Based Tutoring: Impact on Learning Rate, Achievement and Attitudes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, USA) *(CHI '01)*. ACM, 245–252. https://doi.org/10.1145/365024.365111

[14] Gilan M El Saadawi, Roger Azevedo, Melissa Castine, Velma Payne, Olga Medvedeva, Eugene Tseytlin, Elizabeth Legowski, Drazen Jukic, and Rebecca S Crowley. 2010. Factors affecting feeling-of-knowing in a medical intelligent tutoring system: the role of immediate feedback as a metacognitive scaffold. *Advances in Health Sciences Education* 15, 1 (2010), 9–30.

[15] Enrique Estellés-Arolas and Fernando González-Ladrón de Guevara. 2012. Towards an integrated crowdsourcing definition. *Journal of Information Science* 38, 2 (2012), 189–200. https://doi.org/10.1177/0165551512437638

[16] Mingyu Feng, Neil Heffernan, and Joseph E Beck. 2009. Using learning decomposition to analyze instructional effectiveness in the ASSISTment system. In *Proceedings of the 14th International Conference Artificial Intelligence in Education*. IOS Press, 523–530.

[17] Christian Fischer, Zachary A Pardos, Ryan Shaun Baker, Joseph Jay Williams, Padhraic Smyth, Renzhe Yu, Stefan Slater, Rachel Baker, and Mark Warschauer. 2020. Mining big data in education: Affordances and challenges. *Review of Research in Education* 44, 1 (2020), 130–160. https://doi.org/10.3102/0091732X20903304

[18] Ilya M Goldin, Kenneth R Koedinger, and Vincent Aleven. 2012. Learner Differences in Hint Processing. In *Proceedings of the 5th International Conference on Educational Data Mining*. ERIC, 73–80.

[19] Cristina Heffernan, Neil Heffernan, Ashish Maharjan, Leena Razzaq, Prawal Shrestha, and Xing Wei. 2009. Are Worked Examples an Effective Feedback Mechanism During Problem Solving?. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*. 1294–1299.

[20] Neil T Heffernan and Cristina Lindquist Heffernan. 2014. The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education* 24, 4 (2014), 470–497.

[21] Bryan Hobbs. 2013. *Improving Educational Content–A Web-based Intelligent Tutoring System with Support for Teacher Collaboration*. Ph. D. Dissertation. Worcester Polytechnic Institute.

[22] Scott E. Hudson and Jennifer Mankoff. 2014. *Concepts, Values, and Methods for Technical Human–Computer Interaction Research*. Springer New York, 69–93. https://doi.org/10.1007/978-1-4939-0378-8_4

[23] Sally Johnstone. 2005. Forum on the Impact of Open Courseware for Higher Education in Developing Countries–Final Report. *Education Quarterly* 3 (2005), 15–18.

[24] David Kabugo. 2020. Utilizing Open Education Resources to Enhance Students' Learning Outcomes during the COVID-19 Schools Lockdown: A Case of Using Kolibri by Selected Government Schools in Uganda. *Journal of Learning for Development* 7, 3 (2020), 447–458.

[25] Juho Kim. 2015. *Learnersourcing: improving learning with collective learner activity*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[26] Ryung S Kim, Rob Weitz, Neil T Heffernan, and Nathan Krach. 2009. Tutored problem solving vs."pure" worked examples. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*. 3121–3126.

[27] Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing User Studies with Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. ACM, 453–456. https://doi.org/10.1145/1357054.1357127

[28] René F. Kizilcec. 2016. How Much Information? Effects of Transparency on Trust in an Algorithmic Interface. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. ACM, 2390–2395. https://doi.org/10.1145/2858036.2858402

[29] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science* 36, 5 (2012), 757–798.

[30] Kenneth R Koedinger, Ryan SJ d Baker, Kyle Cunningham, Alida Skogsholm, Brett Leber, and John Stamper. 2010. A Data Repository for the EDM Community: The PSLC DataShop. In *Handbook of Educational Data Mining*. CRC Press, 65–78.

[31] Timothy Shin Heng Mak and Albert Lam. 2022. Two-Stage Auction Mechanism for Long-Term Participation in Crowdsourcing. *arXiv preprint arXiv:2202.10064* (2022).

[32] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *ACM Trans. Comput. Educ.* 10, 4, Article 16 (nov 2010), 15 pages. https://doi.org/10.1145/1868358.1868363

[33] Antonija Mitrovic, Brent Martin, Pramuditha Suraweera, Konstantin Pavlovich Zakharov, Nancy Milik, Jay Holland, and Nicholas McGuigan. 2009. ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors. *International Journal of Artificial Intelligence in Education* 19 (2009), 155–188.

[34] Andrew McGregor Olney. 2009. GnuTutor: An open source intelligent tutoring system based on AutoTutor. In *2009 AAAI Fall Symposium Series*.

[35] Andrew M Olney, Sidney D'Mello, Natalie Person, Whitney Cade, Patrick Hays, Claire Williams, Blair Lehman, and Arthur Graesser. 2012. Guru: A Computer Tutor That Models Expert Human Tutors. In *Proceedings of the 11th International Intelligent Tutoring Systems Conference*. Springer, 256–261.

[36] Eleanor O'Rourke, Erik Andersen, Sumit Gulwani, and Zoran Popović. 2015. A Framework for Automatically Generating Interactive Instructional Scaffolding. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. ACM, 1545–1554. https://doi.org/10.1145/2702123.2702580

[37] John F Pane, Beth Ann Griffin, Daniel F McCaffrey, and Rita Karam. 2014. Effectiveness of cognitive tutor algebra I at scale. *Educational Evaluation and Policy Analysis* 36, 2 (2014), 127–144.

[38] Zachary A Pardos, Anthony Whyte, and Kevin Kao. 2016. moocRP: Enabling open learning analytics with an open source platform for data distribution, analysis, and visualization. *Technology, Knowledge and Learning* 21, 1 (2016), 75–98. https://doi.org/10.1007/s10758-015-9268-2

[39] Thanaporn Patikorn and Neil T. Heffernan. 2020. Effectiveness of Crowd-Sourcing On-Demand Assistance from Teachers in Online Learning Platforms. In *Proceedings of the Seventh ACM Conference on Learning @ Scale* (Virtual Event, USA) *(L@S '20)*. ACM, New York, NY, USA, 115–124. https://doi.org/10.1145/3386527.3405912

[40] Rebecca Pitt. 2015. Mainstreaming open textbooks: Educator perspectives on the impact of OpenStax college open textbooks. *International Review of Research in Open and Distributed Learning* 16, 4 (2015).

[41] Darrell Porcello and Sherry Hsi. 2013. Crowdsourcing and curating online education resources. *Science* 341, 6143 (2013), 240–241.

[42] Ethan Prihar, Thanaporn Patikorn, Anthony Botelho, Adam Sales, and Neil Heffernan. 2021. *Toward Personalizing Students' Education with Crowdsourced Tutoring*. ACM, New York, NY, USA, 37–45. https://doi.org/10.1145/3430895.3460130

[43] Leena Razzaq and Neil T Heffernan. 2006. Scaffolding vs. hints in the Assistment System. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. Springer, 635–644.

[44] Leena Razzaq, Jozsef Patvarczki, Shane F. Almeida, Manasi Vartak, Mingyu Feng, Neil T. Heffernan, and Kenneth R. Koedinger. 2009. The ASSISTment Builder: Supporting the Life Cycle of Tutoring System Content Creation. *IEEE Transactions on Learning Technologies* 2, 2 (2009), 157–166. https://doi.org/10.1109/TLT.2009.23

[45] Steven Ritter, Thomas K Harris, Tristan Nixon, Daniel Dickison, R Charles Murray, and Brendon Towle. 2009. Reducing the Knowledge Tracing Space.. In *Proceedings of the 2nd International Conference on Educational Data Mining*. 151–160.

[46] Yigal Rosen, Ilia Rushkin, Rob Rubin, Liberty Munson, Andrew Ang, Gregory Weber, Glenn Lopez, and Dustin Tingley. 2018. The Effects of Adaptive Learning in a Massive Open Online Course on Learners' Skill Development. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale* (London, United Kingdom) *(L@S '18)*. ACM, Article 6, 8 pages. https://doi.org/10.1145/3231644.3231651

[47] Anjali Singh, Christopher Brooks, and Shayan Doroudi. 2022. Learnersourcing in Theory and Practice: Synthesizing the Literature and Charting the Future. In *Proceedings of the Ninth ACM Conference on Learning @ Scale* (New York City, NY, USA) *(L@S '22)*. Association for Computing Machinery, New York, NY, USA, 234–245. https://doi.org/10.1145/3491140.3528277

[48] John Stamper and Zachary A Pardos. 2016. The 2010 KDD Cup Competition Dataset: Engaging the machine learning community in predictive learning analytics. *Journal of Learning Analytics* 3, 2 (2016), 312–316. https://doi.org/10.18608/jla.2016.32.16

[49] Robert G. Taylor, A. Smith, Samuel P. Leeman-Munk, Bradford W. Mott, and James C. Lester. 2014. Towards ITS Authoring Tools for Domain Experts. In *IntelliMedia Group*.

[50] Ilkka Tuomi. 2013. Open educational resources and the transformation of education. *European Journal of Education* 48, 1 (2013), 58–78.

[51] Terrence E Turner, Michael A Macasek, Goss Nuzzo-Jones, Neil T Heffernan, and Kenneth R Koedinger. 2005. The Assistment Builder: A Rapid Development Tool for ITS.. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*. 929–931.

[52] Daniel Weitekamp, Erik Harpstead, and Ken R. Koedinger. 2020. *An Interaction Design for Machine Teaching to Develop AI Tutors*. ACM, 1–11. https://doi.org/10.1145/3313831.3376226

[53] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. 2007. Research through Design as a Method for Interaction Design Research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. ACM, 493–502. https://doi.org/10.1145/1240624.1240704