To appear in Journal of Learning Engineering
Volume 1, Issue 1, 2026

A Taylorizable Process for Textual Detector Development for Research-Practice Partnerships

Ryan S. Baker, Adelaide University, Adelaide, SA Australia

Caitlin Mills, University of Minnesota, AugmentEd, Minneapolis, MN, USA

Andrew Lan, University of Massachusetts Amherst, Amherst, MA, USA

Blair Lehman, Brighter Research, Thousand Oaks, CA, USA

Amanda Barany, University of Pennsylvania, Philadelphia, PA, USA

Corresponding Author: Ryan S. Baker, ryan.baker@adelaide.edu

Abstract

Textual detectors of cognitive constructs play a central role in digital learning platforms, enabling scalable intervention and analysis of student thinking, discourse, and learning processes. Increasingly, these detectors are not the endpoint of research, but a key component of learning platforms built through learning engineering efforts and research–practice partnerships. This applied context increases the importance of high-quality measurement: because a detector embedded in a deployed tool shapes how real learners are treated, inaccurate measurement becomes a source of potential harm. And because co-design moves in rapid cycles, detector development must either keep pace or be excluded from final designs. Recent LLM-as-a-judge approaches, which replace traditional supervised learning with rubric-based prompting of large language models, can substantially reduce development effort. But they are often deployed without the forms of rigor historically emphasized in the learning engineering and educational

data mining communities. At the same time, fully rigorous detector development remains costly and slow, creating pressure to adopt lower-quality shortcuts. In this paper, we propose a Taylorizable process for developing textual detectors of cognitive constructs using LLMs—framed as a learning engineering contribution that balances methodological rigor with the efficiency that partnership-based work demands. Drawing on prior detector development practices and principles of scientific management, we decompose detector construction into a standardized, end-to-end workflow spanning construct definition, human coding and reliability checking, LLM-based detector construction, evaluation, and application at scale. We argue that this process supports four essential forms of rigor—assessment of goodness, generalizability, construct validity, and auditability—while enabling streamlined, legible execution by mixed-expertise teams involving researchers and practitioners. The ultimate goal of this process is to increase efficiency on the research side of research-practice partnerships, while maintaining the rigor needed to guarantee students receive interventions accurately.

Keywords: learning design, data analytics, learning engineering, educational data mining, Taylorization

Introduction

Digital learning platforms increasingly rely on automated detectors of the text that learners produce—such as essays, short answers, discussion posts, and dialogue—to decide how the system should respond, to surface information to instructors, and to make sense of learning at scale. The systems that put such detectors to work are increasingly being constructed through research–practice partnerships (RPPs), in which researchers, teachers, and designers co-design tools together and refine them in real classrooms, so that what results is usable and desirable at scale (e.g., Holstein et al., 2019; Heffernan & Heffernan, 2014). In these partnerships—and in contemporary learning engineering efforts in general—accurate and reliable detectors are often key components of interventions that impact real learners.

Many current tools and interventions attempt to detect and adaptively respond to learners' cognitive states. Detection of this kind is possible because many cognitive constructs are reflected in text that humans produce, which makes it possible for them to be automatically detected and/or measured using natural language processing tools. Extended textual data has been used for applications such as predicting the personality type of the author (Yang et al., 2023), cognitive impairment (Zhu et al., 2019), writing style (Kumar et al., 2025), and critical thinking (Peczuh et al., 2026). A range of other behaviors have also been detected in this fashion from shorter text, including detecting confusion in MOOC discussion-forum posts (Zeng et al., 2017), predicting the urgency of forum posts to support instructor triage (Švábenský et al., 2023), and classifying peer-tutoring chat messages into dialog-act categories linked to learning processes (Borchers et al., 2024).

Such detectors can serve as a strong basis for learning platforms. They can guide responses by instructors or teaching assistants. They can also be applied to scalable assessment

and feedback, such as automatically characterizing the quality of student explanations, argumentation, or critical thinking in written work (Attali, 2008; Attali & Sinharay, 2015). In collaborative and dialog-based learning environments, detectors of discourse features and dialog acts have been used to analyze learning processes, revealing which interaction patterns are associated with stronger learning gains (Borchers et al., 2024)—supporting learning engineering efforts to improve educational effectiveness.

The idea for many detectors is to replicate what an experienced practitioner could do, but at a scale and a speed that would otherwise be infeasible. However, the development of detectors up until recently was itself slow and costly enough to preclude uptake in many cases (e.g., Hollands & Bakir, 2015). Before the public use of generative AI models, the mainstream method to detect these types of constructs in text was a supervised learning classification paradigm, where labeled data is obtained, and then used for both training and testing. However, this approach has typically required large amounts of labeled data to train on, which is often scarce and expensive.

Recent work has therefore attempted to circumvent some of these issues using LLM-as-a-judge methods that do not require the same type of training (Zheng et al., 2023), building on the new capacities of pre-trained LLMs. In this approach, humans first need to define a set of rubrics, in textual format, to assess a particular construct; then, the developer prompts LLMs to perform this evaluation according to the rubrics, leveraging their built-in text understanding capabilities. This cost- and time-effective approach has obvious appeal for a faster pace of development cycles, but also comes with a different set of issues. LLM-as-a-judge is a much simpler and more straightforward process than previous work in the educational data mining (EDM) and related communities to detect constructs from text. However, this approach may also

be too simple and straightforward. Specifically, unlike machine learning approaches, researchers using LLM-as-a-judge approaches often skip the step of careful validation on human-created labels which have themselves undergone an intensive validation process (even though canonical papers using the term LLM-as-a-judge do take this validation step—Gu et al., 2024). In general, LLM-as-a-judge approaches have shown promise due to their scalability, but substantial concerns have been raised about construct validity and reliability, such as limited generalizability across tasks and domains (Huang et al., 2025), whether the use of LLMs is prone to intrinsic biases (Chen et al., 2024) and whether this approach relies too much on superficial textual features (Marioriyad et al., 2025). Without the application of careful validation, it remains uncertain whether the resultant labels are valid and trustworthy. This concern becomes especially urgent once detectors enter deployed tools, and particularly problematic if there is false confidence placed in the detector by its users. When a detector is embedded in a tool that shapes how real learners are taught, inaccurate or otherwise poor measurement can negatively impact the students we hope to benefit. It should be pointed out that this is not simply a hypothetical concern. Although we are not aware of an example of an LLM-as-a-judge detector being audited in the fashion necessary to demonstrate harm, past work has shown how this type of error can concretely impact learners and there is no reason to think LLMs are immune to this type of error; if anything, harmful LLM-based errors may simply be more opaque. For example, Wisconsin's Dropout Early Warning System has generated dropout risk predictions for public school students statewide since 2012; an evaluation drawing on roughly 215,000 students and a decade of outcomes found that the system sorts students by dropout risk with reasonable accuracy overall, but is much better calibrated for some groups of students than others (Perdomo et al., 2025), leading to de-prioritizing students from historically disadvantaged groups in the provision of

support. The countermeasure is simple—subgroup-level calibration and error analysis at development time, as has been done in many past projects using traditional machine learning in education (e.g. Ocumpaugh et al., 2014; Hutt et al., 2024), rather than using aggregate performance metrics alone. What is now needed is to bring that rigor into play when using LLMs in the measurement of cognitive constructs, so that the promise of faster development of partnership-driven, adaptive educational tools is not undermined by measurement we cannot trust.

In what follows, we propose a Taylorizable (systematic, designed to be streamlined and reproducible) process for building reliable detectors of cognitive constructs using LLMs. We frame this as a problem of learning engineering: the design of a repeatable, engineered process that lets detector development serve the applied, collaborative work of building educational tools (Kessler et al., 2023; Goodell & Kolodner, 2023; Dede et al., 2019). Two needs each motivate this step. The first is rigor. Because detectors developed in partnerships are destined for deployment, they must meet the standards of validity and reliability that the educational data mining, learning analytics, and related communities have long upheld; learners bear the consequences when we do not meet this standard. The second is efficiency. Rigor of this kind has historically been slow and costly, and partnerships built around co-design cannot wait: co-design proceeds in rapid cycles, funding and attention are finite, and a detector that arrives after a design is completed and is being piloted has arrived too late. Efficiency is not a precondition for partnership in the abstract; but in the fast, iterative partnerships our field prioritizes (Penuel et al., 2011), its absence can be a key factor causing a project to stall or never reach adoption. A Taylorizable process can bring rigor and efficiency together so that the research side of a

partnership can contribute at both the standard and the speed that real-world deployment requires.

Why We Need Attention to Rigor Now

The implicit assumption in much of the use of the LLM-as-a-judge paradigm (again, if not in the initial work establishing this paradigm—Gu et al., 2024) is that we do not need the forms of rigor historically emphasized in the EDM community, the AES (automated essay scoring) community, and related scholarly communities. To be fair, the majority of researchers adopting this approach are probably not so much actively rejecting the standards of rigor used by scholars working on these types of problems over the last decades; instead, they are new to this type of problem and enthusiastic about the possibilities of LLMs (also referred to as the "peak of inflated expectations"; Dedehayir & Steinert, 2016). However, this affirms the need to argue for the importance of rigor, an argument that perhaps eternally needs to be made and re-made (cf. Moorcock, 1970).

We argue that four types of rigor are particularly essential for effective partnership-driven development, without ignoring the importance of a wide range of other types of validity: assessment of goodness, generalizability, construct validity, and auditability.

Assessment of goodness

A first and foundational form of rigor is the assessment of goodness: establishing whether a detector meaningfully distinguishes between the construct that the developer is attempting to identify, compared to cases where that construct is not present. Without such evaluation, it is uncertain whether a detector is genuinely identifying cases that meaningfully correspond to the target construct rather than to superficial properties of the text. This issue is particularly critical in the LLM-as-a-judge paradigm, where models can generate fluent, confident-seeming, and

internally consistent evaluations, but those evaluations may actually be weakly related—or unrelated—to the construct of interest (Krumdick et al., 2025). Because LLM-as-a-judge approaches do not involve training on labeled data, it is easy to implicitly assume that traditional evaluation practices are unnecessary, and that having results with face validity in a small number of cases is sufficient. However, without explicit assessment of goodness, there is no empirical basis for determining whether an LLM is systematically distinguishing categories such as higher-quality explanations, stronger argumentation, or deeper critical thinking, as opposed to making judgments based on length, vocabulary sophistication, or other proxy features (as seen in Dubois et al., 2024; Zambrano et al., 2023).

In contrast, the EDM, AES, and related communities have long treated quantitative evaluation against held-out data (data selected at the beginning of the process that is only to be used for evaluation, or not yet available at the time of model construction) as a minimum standard, precisely because models that appear plausible in isolation often fail when applied to new students, tasks, or contexts. The same risk applies to LLM-as-a-judge systems, given their known sensitivity to prompt wording, rubric phrasing, and latent biases in pretraining data (Zambrano et al., 2023). For this reason, assessment of goodness remains essential even when no training is involved. The typical standard for doing so is to take cases labeled in advance by human judges, through a process demonstrated to be valid (i.e., high inter-rater agreement on cases rated separately), and evaluate the model on those cases using standard metrics such as AUC ROC and F1 (for LLM-as-a-judge, Gu et al., 2024; for detectors based on machine learning, Baker, 2025). The choice of metric matters here as much as the choice to evaluate at all. In the largest systematic audit of LLM-as-a-judge to date, Norman et al. (2026) report that validation based on exact-match agreement systematically overstates discriminative ability

relative to chance-corrected measures (a finding that matches long-known findings in machine learning; Baker, 2025), and that judges with high test-retest reliability can nonetheless exhibit severe position bias.

Generalizability

A second, closely related form of rigor is assessment of generalizability: whether a detector applies to the full span of cases where we want to use that detector for inference. If a detector is intended only for use in analyzing a specific dataset, this can be simple; generalizability only need be established for that dataset. However, if we want to apply that detector in the future to new data or in real-world educational applications, the task becomes more difficult.

The typical minimum standard in the learning engineering community for generalizability is student-level cross-validation, or a held-out test set of students, where a model is trained on some students and tested on others (Baker, 2025). An analogue for use with LLMs that are not fine-tuned would be to develop a prompt and, where appropriate, associated resources for Retrieval-Augmented Generation (RAG) on one set of students, and then test goodness on a different set of students. This minimal step helps us to reason about whether a model will work on different students than our original sample. This minimal step is unfortunately, but importantly, all too often absent from LLM-as-a-judge research and development.

However, this minimal step is by itself insufficient to establish that a model will function appropriately in new content or for students different from those in the original sample. To validate this, content must be held out, and performance must be tested on unseen groups of students—such as students in a different context such as rural versus urban (Ocumpaugh et al., 2014) or students with different demographic attributes (Hutt et al., 2024). A related type of

validation is checking for algorithmic bias, where models function substantially better for students with some demographic attributes than others (Baker & Hawn, 2022). Given the known biases in LLMs, checking for algorithmic bias is just as important in LLM-as-a-judge approaches as in models developed with machine learning.

Construct Validity

A third essential form of rigor is the assessment of construct validity: whether a detector is measuring what it is intended to measure, rather than some related but distinct construct. Even when a detector demonstrates strong goodness metrics and generalizes well across students and contexts, it may still be identifying patterns that do not genuinely reflect the theoretical construct of interest. Thus, with insufficient attention to construct validity, researchers risk drawing incorrect inferences about student cognition or learning processes based on precise measurements of the wrong construct. Violations of this principle are at odds with the most basic psychometric principles of social science, and most would agree renders an investigation invalid.

If there are sufficiently numerous, diverse, high-quality, and trusted human labels of the same construct, then assessing goodness and generalizability becomes an assessment of construct validity as well. However, there are many cases where humans and computers can become aligned around a definition with questionable construct validity. This issue is particularly common in over-simplified rational definitions of highly complex and multi-faceted constructs such as engagement and metacognition (see discussion in Roll & Winne, 2015). If an LLM is given an over-simplified definition of a construct and asked to code according to it, it will typically not push back—based on its helpful assistant persona, it will do what it is told (as will many human research assistants).

Despite the limitations of many existing systems, communities ranging from educational data mining to psychometrics have long recognized construct validity as a foundational requirement for any detector or form of measurement, and there are several approaches to establishing construct validity, including (1) asking domain experts to review construct definitions (Haynes et al., 1995), (2) checking for convergence with established measures (Ventura & Shute, 2013), and (3) careful review of cases where a detector succeeds and fails, to understand what features the model is actually responding to. This type of analysis can reveal whether an LLM or other detector is making judgments based on the intended construct, has captured a related but different construct, or is focusing on confounding factors that happen to be present in the training examples or prompt. Without such rigor, LLM-as-a-judge approaches risk producing measurements that appear valid on the surface but fundamentally misrepresent the constructs they claim to assess. This is especially important to transparency and communication in RPPs, given that educator partners need to have a clear understanding of what the detector can and cannot do well in order to make informed decisions on the pedagogical implications of using the detector.

Auditability

As part of establishing that a detector or LLM-as-a-judge approach is genuinely capturing what is intended, it is helpful to inspect how the model is functioning. However, inspectability is infeasible both for contemporary machine learning algorithms (the complexity of a random forest or neural network is far too high for casual study) and for LLMs. Thus, in the absence of true interpretability, some combination of auditability and explainability is needed.

A detector is auditable (Biran & Cotton, 2017) if its decisions can be inspected by human reviewers. Without auditability, it becomes difficult to identify systematic errors, diagnose

failures, build confidence in the system's reasoning, or ensure accountability when detectors are used in real-world educational contexts. Hence, auditability is not just about the developers' ability to inspect a detector, but also the detector's availability to external reviewers and stakeholders. Auditability encompasses several complementary practices that enable a person to inspect, understand, and verify the decisions made by a detector. At a minimum, it requires comprehensive documentation of the detection process, including the rubrics or criteria used, the prompts provided to the LLM (where applicable), the version of the model or algorithm employed, and any examples or training data supplied (to the maximum extent possible, also taking privacy concerns into consideration). Ideally, it also involves maintaining records of the detector's decisions alongside the original texts being evaluated, enabling users to spot-check results and identify patterns of error. These practices support the broader research community in scrutinizing methods, replicating findings, and building cumulative knowledge about what works and what doesn't. While auditability does not provide the same depth of understanding as true interpretability, it establishes a foundation for trust and verification. Auditability of this kind does add time and cost, and – if not designed right —becomes an ongoing delay to deployment. However, if designed through instrumentation (setting up ongoing recording of LLM code, prompts, and outputs) and good record-keeping (storing the codebook and other artifacts in an archivable fashion throughout the project), it can be a one-time cost, amortizable across projects. Auditing itself can then be conducted both by external teams, as part of a review by stakeholders prior to or during usage, or as part of the overall quality assurance process discussed below.

Interpretation is supported when models and approaches are not just auditable but also explainable: where there is some capacity to explain why a detector made a particular decision about a particular case. In classical machine learning, explainable AI (xAI) techniques such as

LIME, SHAP, and DiCE have been developed to provide post-hoc explanations of model judgments (Swamy et al., 2022). However, these techniques have significant limitations. Different xAI methods applied to the exact same model judgment decision often provide contradictory explanations (Swamy et al., 2022), raising questions about which (if any) accurately represents the model's reasoning process. For LLM-as-a-judge approaches, a natural solution is to prompt the LLM to explain its coding decisions alongside the codes themselves (Liu et al., 2023). Yet this approach also involves a fundamental limitation: an LLM's explanation represents the most likely textual completion given the prompt and context, rather than a faithful representation of its actual internal reasoning processes regarding the judgment. Thus, the explanation may be fluent and plausible while bearing little relationship to how the model actually arrived at its judgment. As such, while LLM-generated explanations can provide useful hypotheses about decision-making patterns and may help identify obvious errors, they should not be treated as a complete and accurate representation for why the model made a particular judgment. Nonetheless, a model which is auditable and provides explanations—even imperfect ones—enables a level of scrutiny and verification that is impossible with purely black-box approaches. By combining comprehensive documentation, accessible decision records, and model-generated explanations, researchers can support human reviewers and stakeholders in identifying when a detector is making decisions for the wrong reasons, catching systematic biases or failures, and in making informed judgments about when automated codes should be accepted, questioned, or overridden by expert human judgment.

Why We Need Taylorization Now

More rigor is thus needed. But at the same time, increasing rigor can exacerbate a second challenge: detector-building is time-intensive, effortful, and costly (Hollands & Bakir, 2015).

Building detectors more rigorously becomes even more time-intensive, effortful, and costly. Indeed, each of the four types of rigor listed above can increase cost. When applying all four of them, the additional cost and time adds up. If the approaches used are too slow to be practical, they will not be used in practice. Less rigorous, lower-quality approaches may be chosen, with real consequences for the quality of scholarship, transparency in partnerships, and potential negative impacts on learners.

This can be seen concretely in what we believe to be the most detailed cost analysis available for detector development in education using machine learning. Hollands and Bakir (2015) applied the ingredients method of cost analysis to two detector-building efforts and found that developing a suite of affect detectors cost $13,490 USD per detector in one learning platform (ASSISTments), for a total cost over $80,000 USD, and $12,460 USD per detector in another platform (Inq-ITS), including both labeling hundreds of data points and several days of programmer time. When applied to data at scale, this approach was much less expensive per label than classroom observation or video coding, but remains quite expensive overall, especially considering inflation since 2015.

Fortunately, the conditions are present to apply an approach that can substantially cut the time and cost of detector development. First, a substantial body of knowledge has been developed on how to build high-quality detectors, both for classical machine learning and increasingly for LLM-based approaches as well. Second, there is a long-standing body of literature on increasing efficiency in engineering and development. In this section, we draw from one of the most classic and long-standing such methods: Taylorization (Taylor, 1911).

Taylorization, named after Frederick Winslow Taylor's principles of scientific management (Taylor, 1911), refers to the process of breaking down complex work into smaller,

well-defined, standardized tasks that can be executed more efficiently and, crucially, by individuals with less specialized expertise than would otherwise be required. Taylor's original framework emphasized systematic study of workflows to identify inefficiencies, decomposition of skilled labor into discrete steps, standardization of procedures, and clear specification of how each step should be performed. While Taylorization has been critiqued in manufacturing and labor contexts for its potential to deskill workers and reduce autonomy (Braverman, 1974), its core principle—that complex processes can be made more efficient through careful decomposition and standardization—has proven valuable in many domains. In software engineering, for instance, design patterns, coding standards, and automated testing frameworks represent forms of Taylorization that allow developers to work more efficiently without reinventing solutions to common problems (Gamma et al., 1994; Beck, 2023). We do, however, acknowledge that the usefulness of the principle in other domains does not by itself assuage the critique as it relates to our proposed context. We return to this tension, and to what it implies for process in detector development, in our concluding discussion of limitations.

In the context of automated detectors, this means shifting away from building detectors using an ad-hoc, custom process, treated each time as a unique problem to be solved step by step with the process revised as it goes. Instead, we can adopt a clear and standard process, based on what we have learned as a field. The idea of Taylorization also tells us that if we can sufficiently simplify, streamline, and standardize each step in that process, we do not need a superstar-level researcher/developer to supervise (or even participate) at each step. Instead, more junior developers may be able to take on specific steps where appropriate, safe in the knowledge that they are following an established process which, if followed, will yield valid and rigorous detectors. It is important to note that adopting a Taylorized process does not mean skipping key

steps around theoretical reasoning or construct definition. It means using a standardized process to conduct these steps in an efficient, legible, rigorous way—one where the right expertise is brought to bear at the right stage, where no one's time is wasted by reinvention or reimplementation, and where decisions do not need to be endlessly revisited because key tasks weren't done right the first time.

Thus, for detector development, Taylorization means identifying the key steps in creating rigorous detectors, standardizing those steps into a clear workflow, and providing tools and guidance that allow researchers to execute that workflow efficiently and with less specialized expertise than was previously needed. In doing so, we can reduce many of the inefficiencies that characterize detector development today: the frequent reinvention of very subtly different methods, ad-hoc adoption of slightly different decisions, and reimplementation of the exact same code. Avoiding these inefficiencies can make it easier and less costly to build rigorous, high-quality detectors in most cases; a well-designed process will also recognize those rare cases where expert attention or reconsideration is needed. By adopting Taylorization, we expect to increase accessibility, improve speed, reduce cost, and maintain rigor and quality -- claims we are currently testing in an ongoing partnership (see Documentation for Process Improvement, below).

Later in this paper, we will demonstrate how this can be done, presenting a Taylorized process for developing LLM-based detectors of cognitive constructs, realized as a concrete, step-by-step workflow. Before diving into the process, we review related work from adjacent fields that have attempted to standardize methods.

Related Work on Process

In thinking about processes for detector development, three bodies of literature are relevant. Two of them involve detector development process itself: qualitative coding, and past work in the educational data mining community and in broader machine learning on process. The third, research–practice partnerships and co-design practices, establish the collaborative context that motivates our process and the demands it must meet; we begin there.

Research–Practice Partnerships and Co-design

Our process is motivated by the needs of research–practice partnerships—long-term collaborations in which researchers and practitioners jointly investigate and address problems of practice (Coburn & Penuel, 2016). Design-based implementation research organizes research and development around problems and goals in the world through cycles of collaborative design (Penuel et al., 2011), and scholarship around co-design studies document how researchers, teachers, and designers develop educational tools together through rapid rounds of prototyping and feedback (Penuel et al., 2007; Holstein et al., 2019). This literature has largely focused on the dynamics of the collaboration itself—how partners negotiate goals, build trust, and sustain joint work—and on the design and implementation of interventions. Yet it has paid far less attention to the methodological machinery on the research side: how a team produces the rigorous measurement that data-driven intervention depends on, as well as how this can happen quickly enough to keep pace with product development cycles. That is the gap our process addresses. Where the RPP literature describes the partnership and the tool, we focus on how one essential research contribution—the detectors that make a tool adaptive and informative—can be built rigorously, while doing so efficiently enough to be a genuine partner in the co-design cycle rather than a bottleneck in it.

Process in Educational Data Mining

Within the educational data mining field, there is a fairly lengthy literature of individual case studies of detector development that describe their methods in a few paragraphs, but relatively few treatments of process as the object of study itself. These descriptions of methods have converged on some broad elements. Perhaps the best review of this is seen in de Morais et al. (2023), who describe a common five-step process for sensor-free affect detectors, consisting of data collection, feature engineering, model development, performance analysis, and application. However, their treatment of this process consists of describing its manifestations and variations across papers, rather than recommending specific best practices for this process. A portion of a formalized process is also seen in discussion of the EDM Workbench, a tool developed to support developing detectors based on text replays (human coding of pretty-printed log files) (Rodrigo et al., 2012). The EDM Workbench supported a process—enacted by the tool but not specifically argued for or justified—that consisted of importing log files, distilling features, generating clips (segments of log data which are the grain-size of a construct label), sampling data, and labeling data (by human coders), including inter-rater checking. The Workbench then allowed researchers to export data to machine learning packages for development of actual detectors. Similarly, the BROMP protocol (Baker et al., 2020) offered a more formal and repeated process for the limited step of collecting and validating classroom observations for use as training data for detectors.

More recently, process work in this area has begun to address LLM-based detection directly. Xu et al. (2026) present a curriculum-grounded, configurable LLM-as-judge pipeline for question-level marking, in which curriculum intent is operationalized through a known set of syllabus artefacts—prescribed verbs and outcomes, performance band descriptors, glossary

definitions, and marking-guideline principles—and a staged workflow generates question-specific rubrics before deriving the criteria used to allocate marks. That pipeline governs how a deployed judge marks against an existing authorized rubric; the process we propose later in this paper addresses the prior problem of how a team defines, operationalizes, and validates a construct for which a detailed rubric of this nature does not exist and is likely infeasible to create.

Process in Data Mining/Machine Learning

Outside of educational uses of machine learning, the Cross-Industry Standard Process for Data Mining (CRISP-DM) has been proposed and discussed as a process for machine learning more generally (Chapman et al., 1999). CRISP-DM articulates a six-step process for how to conduct machine learning, which fits many projects, but does not attempt to use principles of scientific management to optimize processes through standardization or division of labor (nor was it intended to). Later work has built on CRISP-DM to try to incorporate principles of scientific management. For example, QM-CRISP-DM (Schäfer et al., 2018) considers ways that Six Sigma quality management tools could be incorporated into CRISP-DM's process, albeit at a fairly high level where the exact selection and application of tools must be conducted on a project-by-project basis. Zwetsloot et al. (2018) propose a more formal linkage between Six Sigma and CRISP-DM at each of the stages of CRISP-DM, and suggests some principles, such as having data scientists embedded into teams doing data science and making sure that data scientists are familiar with Lean Six Sigma principles. Studer et al. (2021) build on CRISP-DM with a formalized approach to identifying and mitigating risks, as well as a list of risks that can occur at each step of CRISP-DM. Microsoft's Team Data Science process (TDSP; Guhathakurta, 2017) provides an alternate process for the steps of building data science models, with a focus on tools and infrastructure for implementation and deployment, and consideration of the technical

roles needed to implement such a project (but not the other types of expertise needed for many data science projects, such as domain expertise). A more specific process for creating data science driven apps proposed by Weigand & Kindsmüller (2021) clearly articulates process steps, but neither operationalizes how rigor and efficiency are achieved nor formalizes the types of expertise needed.

A broader standard for the data management dimension of this kind of work is provided by DAMA International's Data Management Body of Knowledge (DAMA-DMBOK; DAMA International, 2017), which organizes enterprise data management into knowledge areas spanning governance, architecture, quality, metadata, and stewardship, and defines the specialist roles attached to each. DMBOK is written for organizations treating data as a long-lived enterprise asset, where provenance and stewardship must hold across many systems and many years, and a full DMBOK implementation is well beyond what most research–practice partnerships can staff. It may be relevant to situations where detector development sits inside a larger data infrastructure—for instance where detector outputs feed institutional reporting, or where a partner institution already maintains formal data governance that the detector pipeline must conform to.

Overall, while several frameworks have incorporated scientific management principles into data science processes, there remains much to be done to reach the full vision of Taylorization. Taken together, these frameworks articulate what steps occur in data science projects, and occasionally which tools may support them, but they stop short of specifying how those steps can be decomposed, standardized, and distributed across roles in a way that systematically increases efficiency while preserving scientific rigor. They continue to treat model development as a one-off process, relying on continual bespoke decisions by experts, rather than

as something that can be rigorously repeated across projects with minimal redundant effort. And indeed, achieving this level of Taylorization may be challenging to accomplish in a generic way, across all of data science—but may be more achievable within a narrower focus such as developing detectors of cognitive constructs. We would note that the gap is narrowing at the level of individual stages: recent work has addressed the grounding of judgments in pre-selected curriculum artefacts (Xu et al., 2026), the verification of LLM annotations with human adjudication (Ahtisham et al., 2026), and the validation of judges themselves (Norman et al., 2026). What remains absent, and what we attempt to provide here, is an end-to-end workflow that connects these stages, assigns them to roles, and specifies what must be produced at each stage.

Process in Qualitative Coding

Qualitative coding, as a research tradition, has taken a different path than machine learning/EDM on the characteristics of good process of identifying specific constructs, in line with the different nature of who identifies constructs (typically a fully human process) and the different research values often seen in qualitative research as compared to machine learning. The process we propose here is targeted to detection applications rather than qualitative research applications, but the qualitative research community's scholarship on process may still be relevant to process for detection. Qualitative methods scholars often describe a series of recurring analytic activities, such as framing guiding questions, conducting initial data preparation and sensemaking, developing and refining codes, and interpreting patterns. These activities are explicitly characterized as iterative, reflexive, and deeply shaped by theoretical and contextual commitments (Maxwell, 2013; Braun & Clarke, 2006; Saldaña, 2011).

Importantly, descriptions of qualitative coding processes often reject the idea of a fixed or linear process "pipeline." Braun and Clarke's (2006) model of thematic analysis articulates phases such as familiarization, coding, theme development, and refinement, while noting that researchers routinely move back and forth between phases rather than progressing in a straight line. Saldaña (2011) conceptualizes coding as occurring across multiple cycles, with early, provisional codes giving way to more focused and theoretically informed categories through sustained comparison and memoing. Maxwell's (2013) interactive model of qualitative research design further emphasizes that goals, conceptual frameworks, research questions, data, and analytic strategies continually shape one another over the life of a study. Across process models there is consensus that making analytic steps explicit can help support goals such as transparency, auditability, reflexivity, and coherence (Lincoln & Guba, 1985; Tracy, 2010; Ye et al., 2025).

From this perspective, qualitative research still relies on process thinking, even when it rejects standardized pipelines. What is often absent, however, is systematic attention to efficiency, division of labor, and reuse across projects. Because of the emphasis on flexibility in qualitative research practice, decisions are often idiosyncratic to a specific project or context, with construct definitions that are refined ad hoc, or when reliability or validation assessments are conducted inconsistently. Some research communities have worked to formalize decisions more explicitly (Arastoopour Irgens & Eagan, 2022), with consideration to rigor, but without a corresponding focus on efficiency, division of labor, or systematic reuse of analytic work across studies. As such, qualitative coding can be rigorous and trustworthy, but it is also usually labor-intensive and difficult to scale or reproduce. Furthermore, best practices and lessons learned

from the idiosyncratic choices of individual projects can often be lost and are relatively rarely transmitted across research groups.

This gap is especially consequential when considering the role that LLMs can play in analyzing data. Qualitative methods process research has offered rich guidance on how constructs should be defined, interpreted, and validated, but share fewer insights into how these practices can be operationalized repeatedly and efficiently across many constructs, datasets, and teams. This is acceptable for research intended to be small-scale and highly in-depth, but remains a limitation for larger-scale mixed methods research and practice intended to impact many students. Our proposed Taylorized process builds directly on qualitative coding traditions by preserving their core analytic commitments, such as construct clarity, iterative refinement, human judgment, and validation, while introducing systematic decomposition and standardization where possible. In this way, we aim to treat the detection of cognitive constructs not as an artisanal, idiosyncratic activity, but as a rigorously structured analytic process that can be made more efficient without sacrificing epistemic integrity or other core qualitative research values.

The Process

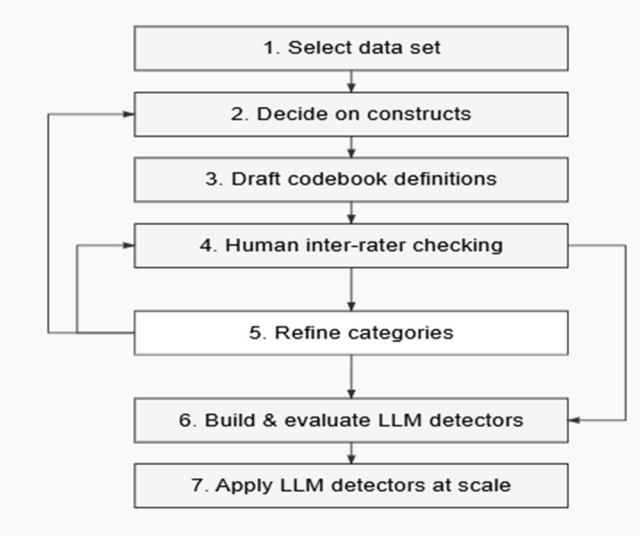
Key Steps

Our proposed Taylorized process for developing textual detectors is given in Figure 1. The first step is to select an appropriate dataset for analysis; note that in many cases, this first step is already implicitly solved. For example, when this process is a step in a larger process of building detectors for a specific learning platform or learning activity, with both the platform/activity and detector goal designed by the research–practice partnership team, the dataset will naturally be drawn from the pilot use of that platform or activity. Then, the second

step is to decide on the constructs of interest for detecting within the dataset, guided by theory, prior literature, bottom-up identification of coding categories (such as in grounded theory—Corbin & Strauss, 1990), and the platform/activity design goal(s). Next, the third step is to use the data to draft clear operational codebook definitions that specify inclusion and exclusion criteria and illustrative examples. The fourth step is to conduct human inter-rater checking, in which multiple coders apply the codebook to a shared subset of data and inter-rater agreement is computed. The agreement for each construct is then checked to see if it falls above or below a cut-off selected in advance. Codebook definitions are refined for any constructs with unacceptable agreement (step 5) and then the process of human inter-rater checking is repeated (step 4). The process alternates between step 4 and step 5 until either acceptable reliability is achieved or it is decided that human beings cannot reach acceptable interrater reliability for a construct, and the construct is either abandoned or heavily redesigned (step 2). The sixth step is to build detectors based on LLMs, using the final round of human coding as ground truth for evaluation. This step will typically involve some iteration. Finally, the seventh step is to use the validated detectors at scale, embedding them into the learning platform to drive intervention or inform educators, or in some cases to support analysis.

Figure 1

The High-level Taylorizable Process for Detector Development.



Key Team Personnel and Steps Where They Are Needed

Although this process can be done by smaller teams, and larger teams can bring additional perspectives (particularly for step 2) and help to scale key steps of the process faster (particularly in steps 3 and 4), we envision that the minimum optimal team has the following roles: A) a project lead who oversees the entire process; B) a domain expert who understands the domain of application and the theory and practice relevant to the types of constructs being studied; C) a researcher/developer experienced in using LLMs to develop and refine codebooks; D) at least two qualitative coders, one of whom is experienced; the other can be less experienced or even a novice; E) a developer experienced in using LLMs to develop detectors of specific textual constructs (or, when unavailable, a developer experienced in educational data mining, machine learning, or data science more generally); F) where the work is embedded in a design or co-design partnership, one or more practitioners—typically teachers who will use the tool, together with members of its design team—who represent the goals of the learning activity and the realities of its eventual use. In some cases, it may also be possible to substitute an

unavailable role with an AI agent whose output is inspected and edited by a human reviewer (Ryng & Suhr, 2026), but this emerging possibility needs further study before full inclusion and recommendation in a process of this nature.

Some members of this team are particularly essential in specific steps. The domain expert, LLM codebook developer, and—where applicable—the practitioners are particularly needed in step 2: beyond supplying design goals, the practitioners help decide which constructs are worth detecting for their setting and what some of the constraints will be, based on how the detectors will be used. The domain expert, experienced qualitative coder, and detector developer are particularly needed in step 3, defining the categories. The qualitative coders are needed in steps 4 and 5, human coding and checking; the domain expert and LLM codebook developer can also assist with refinement in step 5. The detector development team is key to steps 6 and 7, building, validating, and scaling detectors. The practitioners are also essential to step 7, when detectors are integrated into the learning platform/activity.

Details of Each Step

In this subsection, we unpack the sub-steps and key considerations for each of the process's seven core steps.

Step 1: Select the dataset. The project lead begins by identifying the goal for dataset selection, taking into account the type of domain, learning context, learning activity, or general type of categories to be examined. The entire team then engages in a brainstorming session to think of possible datasets that could be obtained or collected. Each of the candidate datasets are then reviewed along multiple dimensions: the domain expert assesses the relevance of the learning context and activity to the target domain and goals; the experienced qualitative coder evaluates amenability to qualitative coding; the detector development lead examines amenability

to analysis; the project lead or their designee investigates any data access or publication issues; and practitioners weigh in on feasibility of collection and relevance to the eventual context of use. Based on this comprehensive review, either the project lead selects the dataset that will be used for analysis (increasing efficiency, in line with the goals of Taylorization) or a consensus-based process is used (increasing the likelihood of success, when all voices are heard).

Step 2: Decide on constructs. The analysis team starts this decision process by reviewing the overall guiding definitions of the project. As this process often takes place in the context of developing detectors for an existing learning platform or activity, the analysis team also reviews design goals and design notes from the design team where available. The domain expert identifies relevant theory or papers (if not already known) and reviews the data in terms of that theory to suggest initial categories, with a particular focus on ensuring the operationalization of the construct is (1) consistent with existing research and (2) indicative of the construct in the specific context in which it is being analyzed. In parallel, a researcher experienced in using LLMs to develop codebooks runs both a bottom-up codebook discovery process (e.g., Barany 2et al., 2024) and a theoretical codebook discovery process (Zambrano et al., 2026). This developer also reviews constructs from past project work to identify categories that might be included in the current effort. The superset of the codebooks generated through each of these processes becomes the draft of the full category set. Next, a pair of design team members (ideally a combination of a practitioner/teacher and a researcher) assess the categories against the design goals and intended uses of the tool, flagging which constructs matter most for those goals and which would not be actionable in practice (e.g., Barany et al., 2024). Finally, the project lead reviews the category set with the domain expert and the practitioners, and (in the most

efficiency-oriented version of the process) selects a final set that reflects both construct validity and the tool's intended use.

Step 3: Draft codebook definitions. The domain expert and the experienced qualitative coder work together to draft an initial set of codebook definitions, building on initial definitions suggested by the review of past coding categories and the LLM's coding category discovery where available. The LLM is then prompted by the development team to suggest refinements to those coding categories that may make the coding process more feasible (e.g., Barany et al., 2024). The domain expert and experienced qualitative coder review these suggested refinements and accept or reject them as appropriate. They then work together to create the first set of draft codebook definitions that will be subject to human coding. During this process, the grain-size at which categories are identified and the data segmentation approach will necessarily be chosen. Sometimes this process is very quick or even implicit (e.g., code every utterance separately), and sometimes more judgment is needed (segmenting log files almost always involves decisions about trade-offs). In some cases, if a category cannot feasibly be identified at any grain-size of data available, that category may be dropped or redefined (returning to step 2), or the dataset may need to be reconsidered (returning to step 1).

Step 4: Human inter-rater checking. The experienced qualitative coder and a second qualitative coder (perhaps less expert; perhaps a novice) begin by coding a small number of cases together, to make sure they have the same understanding of each category. They then select the number of cases for each round of coding (using, perhaps, a power calculator for inter-rater agreement; e.g., He et al., 2022) and code independently. They then check their agreement using metrics such as Cohen's/Fleiss's Kappa, Krippendorff's alpha, or Shaffer's Rho (Shaffer, 2017). If the metrics indicate that agreement is not good enough, the coders discuss the cases where

they disagreed, code a small number of cases together again, return to coding separately, and check their inter-rater agreement once more. Alternatively, inter-rater reliability can be used as a diagnostic signal for localizing disagreement and refining constructs rather than for a pre-chosen threshold (Thomas et al., 2026). We suggest using a pre-selected threshold because a team executing a process needs a defined point at which to stop iterating and move on, and in the absence of one, refinement can continue indefinitely at considerable cost, or be closed off in an inconsistent and ad hoc fashion.

Step 5: Refine categories. If acceptable agreement has not been achieved after two rounds of Step 4, the novice qualitative coder feeds the definitions and disagreement cases into an LLM, which proposes definition refinements (e.g., Zambrano et al., 2023). The experienced qualitative coder and novice qualitative coder review these LLM-generated refinements, make their own additional refinements, and propose a new version of the definitions. The domain expert then reviews this new version and refines it further. The refined definitions are then used for a new round of qualitative coding; Step 4 is repeated for a maximum of two cycles. If Step 4 fails a second time, the affected categories are dropped from the project.

Step 6: Build and evaluate detectors. The development team sets up the collection of prompts and approaches for the LLM and selects the set of LLMs that will be used. For each category, the development team runs this full set of approaches. The dev team then selects the best approach for each coding category. If any category cannot be successfully modeled, the development team tries custom approaches while immediately advancing to Step 7 for categories where an approach has already worked.

Step 7: Apply detectors at scale. The development team applies the successfully developed coding categories to all remaining data. The development team also packages the

successful approaches as Dockerized containers for inclusion in the learning platform or activity, with practitioners and designers leading the process of determining how each detector's output drives adaptivity or feeds reporting to teachers.

Key Tools for Process

A data repository of papers and datasets reviewed. Such a repository can help prevent duplication of effort when selecting datasets, and can also make it easier to keep track of what theories were used and where some categories originally came from.

A category repository. A repository is needed to store each coding definition and each variant of each definition. This repository should mark the current variant in use for each definition, allowing the team to track the evolution of definitions over time. This repository can be useful from step 2 onwards.

Inter-rater reliability process tools. Several tools can support the inter-rater reliability process. A tool is needed to pull a certain number of cases for coding, with the capability to ensure those cases are never pulled again in subsequent rounds. This can be achieved informally, but risks repeating cases or other inefficiencies. Another tool is needed to record each coder's decisions for later comparison (both to the other human coder and to LLMs). A third tool can rapidly calculate inter-rater reliability metrics within a round, enabling the team to quickly assess whether agreement is sufficient or whether additional rounds are needed. A fourth tool can support refinement by collating disagreements between coders and feeding them to an LLM or a human conducting codebook refinement, along with the relevant definition.

LLM prompting and evaluation tools. The development team can benefit from the creation of a set of standardized prompts and approaches that will be consistently used across categories. Developing a tool that runs each of these prompts and approaches and outputs the

results to developers in easy-to-review form will streamline them in comparing many variants of many models. Such a tool can also select and output the best prompt and approach for each category, along with model quality measures, for review. Finally, a tool can be developed to quickly turn the best prompt and approach for each category into a Docker container, facilitating deployment and integration into learning platforms and activities.

Past example artifacts. In general, having past examples of key artifacts – such as a sample codebook entry, a sample prompt, a sample reliability output table – gives practitioners a concrete template to imitate rather than only a description to interpret. This helps to increase consistency and quality across projects.

Major Pitfalls at Each Step

Each of these steps can have pitfalls where the process can either fail in ways that jeopardize the successful development of detectors, or where the process becomes slow and inefficient in a way that jeopardizes the overall success of a project depending on the detectors, even if the detectors themselves are eventually developed. Careful attention to these pitfalls is important for project success, and it may be worth reviewing this set of pitfalls both before and after conducting the step. Setting explicit targets in advance is important for each sub-step of the process (including deadlines for completion). Getting commitment from key project members to be available at the right times can also make the difference between a process that runs smoothly and a process that drags on.

Step 1: Select dataset. In this step, it could become apparent that the existing datasets available are insufficient for detecting the types of constructs desired, necessitating expensive and time-consuming data collection. Alternatively, the chosen dataset could be too small, too

coarse-grained, or too low-quality (including needing extensive pre-processing) to afford the development and validation of a model. Thirdly, a dataset could be chosen that is itself high-quality, but is not representative of the eventual contexts that the team wants to deploy detection in. A fourth risk is that an appropriate (or better) dataset exists but is not found by the team due to an insufficiently comprehensive search; it is also possible to spend too long searching for ideal data and delay settling on datasets that are perfectly appropriate for the team's research goals.

Step 2: Decide on constructs. In this step, the core risks emerge from the breakdown of each of the processes used to identify candidate categories. For instance, the domain expert could fail to identify the most relevant theory or papers or could spend too long searching for prior literature and mining it for categories. Selection of an inappropriate or overly abstract theory can hinder both human and LLM processes for using theory to derive codes (although reference to past applications of a theory can mitigate this issue—cf. Zambrano et al., 2026). Insufficient attention to prompt design can reduce the quality of any LLM-based discovery process. Insufficient time spent on filtering categories, or choosing categories that are rare (and not sufficiently important to justify including despite their rarity), can lead to delays further down the pipeline. The number of individuals involved in this step of the process may also cause coordination delays, requiring high-level attention to how the process is progressing.

Step 3: Draft codebook definitions. This step is dependent, as with earlier steps, on the thoroughness and thoughtfulness of human revision, the appropriateness of LLM prompts, and the efficient collaboration of the domain expert and experienced qualitative coder to select the first set of draft definitions that will be subject to human coding.

Steps 4 and 5: Human inter-rater checking and refining categories. An unrealistically high criterion value for inter-rater agreement can also leave coders cycling between steps 4 and 5

for an inefficient amount of time. Careful attention to whether agreement looks infeasible at the very beginning and to the amount of time spent in this part of the process can prevent delays in a stage where delays are particularly common.

These phases can also be delayed if coding is too difficult due to ambiguities or other issues with the categories. Early attention to this issue can prevent delays, but steps taken to improve agreement during refinement can risk filtering out key aspects of the construct, particularly if a category is narrowed to an oversimplified set of rules that can be applied without judgment. Finally, some refinements may solve current disagreements while breaking the parts of the definition that were working effectively, creating new disagreements within the next round.

Additionally, if the coding sample is poorly chosen, the coders may end up coding data subsets where codes of interest are insufficiently common, or data subsets that are not representative of the full data, leading to poor generalizability later on in step 7. Careful attention to sampling, such as stratification on key variables, can reduce these risks. Including too little or too much context for coding can also make coding difficult.

Finally, the availability of experts needed for step 5 should be secured in advance, so that there can be rapid turnover of refined categories back to the coders, if a repetition of step 4 is needed.

Step 6: Build and evaluate detectors. The core risk at this step is that the LLMs are unable to effectively detect the constructs of interest. This can be mitigated through trying a sufficiently large number of LLMs and prompting approaches; building a library of approaches in advance (from published literature and past experience) can reduce the time spent tinkering at this stage. Using the type of tools mentioned in the previous section rather than running each

prompt one by one (assembly-lining the process of trying different approaches) can also improve the efficiency of this stage. Too much tinkering can however lead to over-estimation of model goodness; this can be prevented by holding out some human labels until the very end of the process.

Another risk is found with models that may be deprecated or replaced by their developers. Running open models locally can prevent the best model from suddenly becoming ineffective or unavailable. This practice also reduces privacy risks and concerns but can introduce scaling issues that may not be present with large-scale corporate LLMs.

Step 7: Apply detectors at scale. This stage should be relatively free of risk, if the code to run the right prompt on the right LLM has been effectively distilled during stage 6. The process of creating a Dockerized container can be time-consuming the first time, but if set up correctly, should be rapid to scale across detectors afterwards. However, it is important to re-check model performance once generalization has commenced (and perhaps even later in the process—e.g., Levin et al., 2022), to make sure that the model is functioning as intended in the context of deployment. Note that a successfully functioning detector does not necessarily mean that interventions based on it will work—in that step, iterative co-design involving the right group of practitioners and stakeholders is essential.

Criteria for Evaluating a Process

Up to this point, we have outlined a Taylorizable process for developing LLM-based textual detectors of inherently cognitive constructs. The question is how we can assess whether this process successfully resolves the challenges that made such a process necessary. We therefore propose six criteria that processes should be weighed against in order to evaluate their effectiveness for detector development in educational data mining and related fields.

Does it get things done efficiently? Taylorization aims to increase efficiency by standardizing repetitive processes and tasks and enabling parallel work streams. An efficient process should reduce the time from project initiation to completion (whether it be deployment or publication, etc.), particularly when building multiple detectors or scaling across domains. If a process is rigorous but very slow, or if it requires highly specialized expertise at every step, it may not be efficient enough to be scalable (the concern that motivates Taylorization in the first place). Increased efficiency should not come at the cost of quality (as we discuss below). It should instead remove unnecessary variation and decision-making, and avoid time-consuming mistakes.

Does it reduce early planning? One key advantage of our proposed process is that it distributes the workload and decision making across a standardized workflow, making the affordances and constraints of building a new detector a bit more straightforward. This should therefore reduce the need for extensive upfront planning and reduce the amount of subsequent re-planning. Planning is not eliminated within our process; instead, it is scaffolded by the process. This should reduce the number of person-hours dedicated to custom planning for each new detector, as well as the overall burden of setting up and managing the overarching process from scratch.

Does it avoid unexpected derailments and unnecessary iterations? As most of us who have engaged in the detector building process know, issues can (and often do) arise at various stages. Such unanticipated obstacles can force researchers to backtrack or even restart parts of the detector building process, thus losing valuable time and resources. We propose that risk of discovering flaws at late stages (e.g., poorly operationalized constructs or failures of basic cross validation techniques) can be reduced with a thoughtful sequential process, which helps mitigate

such inevitable derailments by anticipating common failures as part of the process and having validation checkpoints that catch issues before they cascade into later stages. Back-tracking is sometimes necessary (and our process explicitly incorporates it); the key is to limit how far back the research team needs to go, and how much work needs to be redone. As an example, our process recommends establishing human-human inter-rater reliability before any LLM prompt development, which ensures that construct definitions are clear, reproducible, and codeable before substantial effort is invested in automation. Skipping any part of this step may produce results that are not rubric aligned or have questionable construct validity, necessitating repeating LLM coding work after fixing issues that could have been resolved beforehand.

Does it lead to high rigor? Rigorous detector development should include a few key dimensions, listed above: reproducibility (i.e., can others follow the same steps and achieve similar results?), auditability (i.e., is there a clear record of decisions and validations?), construct validity (i.e., does the detector measure what it claims to measure?), and generalizability (i.e., does performance hold across datasets or contexts?). Auditability provides the foundation for other forms of rigor: a rigorous process produces artifacts at each stage, such as codebooks, reliability statistics, evaluation results, documentation, that provide evidence for forms of rigor mentioned above. This should be embedded in the workflow itself with high standards at each step.

Does it avoid introducing ethical concerns? Any attempts at detecting cognitive constructs must include careful attention to ethics like privacy, algorithmic bias, and the potential of misuse. A Taylorizable process should therefore include these considerations as part of the process. This includes taking steps such as validating across diverse populations to check for bias (Baker & Hawn, 2022) and clear documentation that supports informed decision-making about

appropriate deployment contexts and model uses based on the rates of incorrectness (i.e., false negatives and false positives).

Is it useable across domains? Perhaps the ultimate test of any Taylorizable process is whether it functions as a reusable methodology across different cognitive constructs or across different contexts for the same cognitive construct. A process that meets this criterion is one whose core workflow remains applicable with some necessary adaptation, whether the target is critical thinking in student essays or confusion in peer tutoring dialogues. If the core processes remain stable, then the approach is successful at saving time while maintaining rigor. However, this is difficult to assess in the initial context; ultimately, it can only be assessed in the eventual domain where re-application is attempted. But it can be facilitated by the work in the original domain by documenting the assumptions and constraints that factored into the development of the original model set, as well as contextual, population, and design factors that influenced those models. Eventually, through greater study of this type of process, we may be able to infer even during work within the original domain whether the process is likely to succeed in new domains – but more work is needed to get to that point.

Using the Process: An Ongoing Research–Practice Partnership

A Multi-detector Project with Co-design

The issues we discuss here emerge for us within an ongoing program of research–practice partnerships in which we are developing and stress-testing this process; the idea for this paper was partly inspired by the challenges of this project. In this project, we are co-designing educational tools to support learning experiences with teachers, researchers, and engineers at the center of the co-design teams. These tools will eventually depend on detectors to support adaptivity and reporting to teachers. The teams are designing learning experiences from scratch,

meaning they have the opportunity to build tools around what's now possible given theoretically driven hypotheses and knowledge about what is now possible with detectors.

This is different from the more commonly-seen case where detectors are developed for existing technologies. In such a case, researchers typically wait until data is available from an already deployed system, then build and validate detectors for inclusion and sometimes retrofitting into the system. We are hoping to instead enable an approach where teams can build tools that are informed by what is possible, given the state of detector development at the time.

To do this, teams need to understand what constructs can be detected reliably, what constructs will need more time for development, and what constructs may not be feasible given current methods. This orientation of detector development to the needs of a design project in progress also emphasizes designing learning experiences around "good enough" detectors, rather than spending a ton of time developing a hypothetical "best" detector that ultimately misses the window for inclusion in the design itself. Each decision in such an approach needs to be shared between practitioners, designers, and researchers. Because teachers and designers sit at the center of the co-design teams, they help decide which constructs are worth detecting for their learning goals and what counts as "good enough" for the decisions their tool will make—while the research team contributes what is technically feasible and how reliable a given detector is likely to be.

The challenge, of course, is speed. Co-design processes move quickly, with teams iterating ideas, testing prototypes with teachers and students, and refining based on feedback in rapid cycles (Penuel et al., 2007). Traditional detector development, in contrast, moves more slowly, and has—without Taylorization—historically required considerable time and effort (Hollands & Bakir, 2015). The mismatch in timelines creates a fundamental tension: either slow

down co-design to wait for detector development (sacrificing the iterative responsiveness that is one of the major pluses of co-design) or proceed with co-design without detector input (risking design decisions that cannot be adequately supported, or not fully leveraging the opportunities detectors create).

Our Taylorizable process is designed to address this challenge in two key ways. The first is the standardization of workflows and clear guidance at each step. This process saves time that would otherwise be spent on process decisions, debates about methodology, or trial-and-error approaches to common challenges. Teams can focus their effort on the substantive work of defining constructs and building detectors, rather than continually reinventing processes. The efficiency gains may be realized even more when developing multiple detectors in parallel, as lessons learned and infrastructure developed for one detector can be directly applied to other detectors. The second way is that the process is designed to be adaptable across different forms of data. Imagine collecting different types of student data that are all text-based, like essays, short-answer responses, discussion posts, transcribed dialogues, or other forms of writing. The core workflow remains stable across these production types, and the same technology can be used for all these types of data. This consistency means that teams can develop expertise in the process itself, rather than needing to develop entirely new approaches for each detector, and can build shared infrastructure (such as coding interfaces, prompt evaluation tools, and deployment pipelines) that accelerates work across multiple detectors. To be clear, we do not think this process only applies when co-designing tools that are amenable to multi-detector solutions. Instead, we think this is an example of how such a process would enable others to adopt similar and efficient processes when rigor, speed, and design are all priorities.

Documentation for Process Improvement and Refinement

Our research agenda treats the proposed Taylorizable process's benefit as a hypothesis to be tested and refined through actual use. We would argue that the best way to validate these claims is through systematic documentation of how the process performs in practice. This requires tracking not only the outcomes of various stages of detector development (such as inter-rater reliability statistics, detector performance metrics, and deployment results) but also tracking the process itself, such as timelines, decision points, iterations, and failures. Concrete examples might include: calendar time and person-hours at each step in the process, the number of coding rounds needed to reach acceptable inter-rater reliability (and the level reached for each construct at each round), how often and in what ways the team backtracks, and the constructs abandoned or substantially redesigned late in the process rather than early. By clearly documenting each of these, a team develops the basis to study the success of this (or any such) process, and to reflect and report on lessons learned.

Meticulous documentation thus serves a forward-looking function by enabling systematic refinement of the process over time. The current formulation represents our best thinking based on prior literature and experience, but it will inevitably have limitations and gaps that only become apparent through actual implementation. Some steps may take longer than anticipated; some transitions between steps may be unclear or awkward. It's also possible that some risks or failures may emerge that we did not anticipate. By documenting our experience systematically, we create the empirical foundation for revising and improving the process for future detector development projects, both within our own team and for others who may adopt or adapt this approach. To this end, it will be important for anyone using these types of processes to carefully document failures and challenges—even small-scale ones that were resolved but decreased

efficiency—rather than only presenting success stories alone. And then – equally importantly – they will need to take the next step of improving their process and reporting those improvements – treating a process such as the one discussed here as living rather than static, with a review at the close of each project asking specifically what should be done differently going forward.

An Alternate Path: Decreasing Total Calendar Time Using Existing Datasets for Early Development

The process outlined in this paper attempts to reduce both total time and effort. In some cases cost may be less of a concern than calendar time spent, or vice versa. In this section, we discuss one of the ways that the Taylorizable process outlined in this paper could be optimized for faster completion at the cost of spending more resources. This earlier-completion approach involves using existing datasets for early development. We are currently testing this approach in parallel with the previously discussed approach, in the context of our current research agenda.

As mentioned previously, in that research agenda, the goal is for multiple co-design teams to develop new tools from scratch. This means that data from those tools will not exist until the tools are sufficiently developed to pilot with actual users, which typically occurs well into the design process. Waiting for that data to become available would delay detector development, and major design decisions may be made and difficult to revise by then. Using other datasets can help kickstart the process while also testing the feasibility of various design features. As an example, an existing corpus might contain middle school science argumentative writing, while the eventual tool might target high school science argumentation. The dataset is not a perfect match but may be similar enough along key dimensions to support initial construct definition, human coding, and even detector development.

This process adjustment has several benefits. Overall, it puts us early in stage 6 of the process before the first data has even been collected from the final tool: a considerable head start in the process, even if work needs to be redone on earlier data stages when data from the final tool is available. Adjacent dataset exploration allows for the early testing of construct definitions, particularly in terms of whether the definitions are sufficiently clear and operational for reliable human coding. If multiple trained coders cannot achieve acceptable inter-rater reliability when coding an existing dataset, this may signal that the construct definition needs refinement before proceeding further. This is a problem that is better to discover early, as opposed to, for example, at the conclusion of months of tool design.

Adjusting the process in this fashion also enables early assessment of LLM capabilities for detecting the constructs of interest, allowing teams to determine which models perform best, which prompting strategies are most effective, and which constructs may be particularly challenging to detect reliably. An added benefit is the exploration of generalizability by testing whether detectors developed on one dataset (e.g., middle school writing) can transfer to somewhat different contexts (e.g., high school writing). Of course, using a "similar enough" dataset requires judgment about how much mismatch between existing datasets and target contexts is acceptable in the first place. This depends on both the nature of the construct and the intended application. Some constructs are highly domain-specific, which may result in cross-domain datasets that may produce conclusions that are not applicable to the current project. Constructs may not always manifest in the same way, and some constructs may be differentially present or absent. Similarly, some applications may be more sensitive to context mismatch than others. Adjusting the process in the fashion discussed here inherently involves repeated work, increasing cost but possibly producing models months earlier.

Framing this adjustment purely as spending resources to buy calendar time, however, may understate the costs that it has the potential of avoiding. First, it avoids cost due to failing late in the process. For one thing, our primary Taylorized approach risk is discovering that a construct cannot reliably be coded by the human or LLM only after the tool has been piloted and its data collected; under the existing-data variant, this failure may surface earlier, if that construct is relevant within the existing data. Given that human labeling and programmer time dominate the cost of detector development (Hollands & Bakir, 2015), avoiding even one such failure can offset a substantial fraction of the duplicated effort this variant introduces. A second factor is the cost of late design revision. Because detector development on the tool's data can only begin once major decisions in that tool's design have already been made, problems that detector development would have surfaced—constructs that are not reliably present in the text learners actually produce, for instance—may be found too late to practically address. At the same time, there may be less duplication of effort than it would initially seem: construct definitions, codebooks, prompt designs, and evaluation processes developed in working with the adjacent dataset have the potential to transfer to the final dataset even if the labels themselves do not. As such, there may be ways for this alternate approach to come closer to the anticipated cost of the primary Taylorized approach than it might initially seem. In part, this depends on how closely the adjacent dataset matches the eventual learning system and context – something that the value of an adjacent dataset approach depends on in general.

Overall, adjustments of this nature make it possible to optimize differently. The Taylorizable process offered earlier in this paper can be applied to many situations where detectors need to be efficiently developed, but will not fit all situations. When adapting or deviating from such a process, it is important to consider what goals are being optimized for,

what the downsides are for other goals, and how the process can be adjusted in advance to plan for and mitigate possible risks. Ad hoc, unplanned shifts risk unintended consequences.

Conclusions

The emergence of LLMs has fundamentally changed the landscape of how automated detectors are built. Recent work has shifted to considerable use of LLM-as-judge approaches to infer a range of constructs in textual data. This has been a positive development in many ways: less intensive background is needed for researchers and developers to build detectors, and LLMs can make many parts of the process more efficient. At the same time, without careful attention to important methodological considerations, we run the risk of degrading the quality of the inferences made about students and, in turn, reducing the appropriateness of the interventions those students receive. In this paper, we propose a solution to this by decomposing the stages of detector development of cognitive constructs using textual data into a standardized, end-to-end Taylorizable process, with the goal of making rigorous practices more efficient (and also making efficient practices more rigorous).

Learning engineering teams and RPPs interested in using this approach can introduce it into their practice in several ways, in a fairly immediate fashion. At minimum, the seven-step sequence we introduce can serve as an initial project plan for the next detector a team builds, with the personnel commitments described and explicit target dates attached to each step. Alternatively, the six criteria we propose can be applied as a rubric for auditing a process a team already has underway, in order to locate ways a process is breaking down/could break down, or is simply not achieving its goals. Similarly, the pitfalls we catalogue can be used as a pre-mortem checklist, reviewed on entering each step and again on exiting it. A team whose binding constraint is calendar time rather than cost can adopt in advance the existing-data variant we

describe. None of this requires the tooling we recommend to be in place beforehand; that infrastructure can be developed along the way, and accumulate across successive projects. Finally, a team that documents its timelines, decision points, and failures as we recommend produces exactly the evidence a team needs to refine its process. These are some of the ways a team could immediately use the ideas in this paper to improve their processes.

That said, our proposal is not intended to simply provide a step-by-step roadmap for what steps should be taken; it also represents a perspective and commentary on how the engineering of automated detectors and the interventions they enable can evolve in the era of LLMs, while continuing to center core principles of validity and reliability. While fields like educational data mining have implicitly standardized approaches in terms of expectations for what must be reported in each paper, each research project is often built from scratch with limited reuse of methods across teams and contexts. This approach has enabled creativity and flexibility, but as the deployment of detector-based learning technologies continues to grow, we are at a critical juncture where we need to ensure consistent quality standards across projects. More systematic engineering practices will pay off by enabling more scalability and infrastructure that enables others to build upon prior work.

Taylorization, as presented here, represents a step toward a more engineered field—a contribution to learning engineering, and specifically to the measurement and detector-development work on which data-driven learning tools depend. The hope is that this relieves some of the burden in terms of resource planning and rigor, without stifling innovation or creativity. We hope that this method also leaves room to place more emphasis on cognitive constructs that have not traditionally been a focus in past learning systems, particularly constructs that require nuanced and complex operationalizations such as complex forms of

comprehension and critical thinking. Similarly, we hope this method will encourage sharable standardized artifacts like codebooks, datasets, and documented prompts, which could help create more robust shared infrastructure that enables generalizable knowledge and (hopefully) impact.

None of this is to suggest that Taylorization will be without downsides and limitations. Anytime something is automated and standardized (even if customizable), there are corresponding trade-offs. Some product goals may require bespoke approaches. To this end, we do not suggest that every project attempting to detect cognitive constructs using textual data—or even every process for every single construct, within-project—must follow this approach. That said, well-designed and repeated processes with clear and rigorous checkpoints still present a meaningful starting point to consider, so that a decision to deviate from standard procedure is carefully thought-out and justified. We also want to emphasize, as scholars conducting work in learning engineering, in EDM, and in mixed-method projects involving qualitative research, that this paper's process is intended for detector development rather than for qualitative research approaches involving qualitative coding (such as thematic analysis or grounded theory).

One trade-off deserves specific attention, as it is a classical objection to Taylorization (e.g. Braverman, 1974). Our process reduces the expertise required to execute several steps, and in doing so concentrates the use of expert judgment into a smaller number of points, principally construct definition and the review of refined definitions. This concentration of the use of expertise does create some risk: a team member who executes only standardized steps may trust that the process will yield a valid result and therefore not develop the judgment needed to recognize when the process is not working for a particular construct. We would note, though, that the more ad-hoc process that has typically been used may not lead to the development of that

judgment either. When each project is bespoke, expert judgment is exercised continuously but may be done so invisibly; if an expert makes the key decisions but does not explain them, a junior project member may not learn them in this situation either. A standardized process at least makes the points at which judgment is required explicit, and that legibility can support teaching it: junior coders working through codebook development alongside a domain expert can be engaged in an apprenticeship, a catalogue of possible pitfalls surfaces tacit craft knowledge, and documenting why decisions were made preserves reasoning that a later reader (external or within the team) would otherwise have to reconstruct. We do not claim these practices fully resolve this tension, but it suggests the relevant comparison is not between a Taylorized process and one in which experts are involved in all stages of the process in a less planned-out fashion, but between a process in which the exercise of judgment is made visible and one in which it is not.

Ultimately, the value of our Taylorization process should be based on the practical changes it can support within the research–practice partnerships it is designed to serve. We share this framework now, at a time when we believe it can help standardize rigorous processes with LLMs, and we will document our own efforts to apply it in the future. If nothing else, we hope to invite others into a dialogue about how we can collectively build a more rigorous, efficient, and impactful field.

Acknowledgments

The research reported here was funded by AugmentED, which is supported by the Advanced Education Research and Development Fund (AERDF). The opinions expressed are those of the authors and do not necessarily represent the views of the AERDF. We also thank Chelsea Porter for assistance with document preparation.

Conflicts of Interest Statement

There are no known conflicts of interest to disclose.

GenAI Disclosure Statement

LLMs were used to support the creation of this manuscript – multiple versions of Claude and ChatGPT – to suggest draft sentences, critique the article, proofread, and create a diagram. Most of the em-dashes are our own. All content has been verified by human co-authors.

References

Ahtisham, B., Vanacore, K., Lee, J., Zhou, Z., Pietrzak, D., & Kizilcec, R. F. (2026). AI annotation orchestration: Evaluating LLM verifiers to improve the quality of LLM annotations in learning analytics. In Proceedings of the 16th International Learning Analytics and Knowledge Conference (LAK26). https://doi.org/10.1145/3785022.3785095

Arastoopour Irgens, G., & Eagan, B. (2022, October). The foundations and fundamentals of quantitative ethnography. In International Conference on Quantitative Ethnography, 3-16. Cham: Springer Nature Switzerland.

Attali, Y. (2008). Automated scoring of short-answer open-ended GRE Subject Test items. Research Report. Educational Testing Service.

Attali, Y., & Sinharay, S. (2015). Automated trait scores for" GRE"® writing tasks. Research Report. ETS RR-15-15. ETS Research Report Series.

Baker, R.S. (2025). Big Data and Education. 9th Edition. Philadelphia, PA: University of Pennsylvania.

Baker, R. S., & Hawn, M. A. (2022). Algorithmic bias in education. International Journal of Artificial Intelligence and Education, 32, 1052-1092.

Baker, R. S., Ocumpaugh, J. L., Andres, J. M. A. L. (2020). BROMP quantitative field observations: A review. In R. Feldman (Ed.) Learning Science: Theory, Research, and Practice, 127-156. New York, NY: McGraw-Hill.

Barany, A., Nasiar, N., Porter, C., Zambrano, A. F., Andres, J. M. A. L., Bright, D., Choi, J., Gao, S., Giordano, C., Liu, X., Mehta, S., Shah, M., Zhang, J., Baker, R. S. (2024). ChatGPT for education research: Exploring the potential of large language models for qualitative codebook development. In Proceedings of the 25th International Conference on Artificial Intelligence in Education.

Beck, K. (2003). Test-Driven Development: By Example. Addison-Wesley.

Biran, O., & Cotton, C. (2017, August). Explanation and justification in machine learning: A survey. In IJCAI-17 workshop on explainable AI (XAI), 8(1), 8-13.

Borchers, C., Yang, K., Lin, J., Rummel, N., Koedinger, K. R., & Aleven, V. (2024). Combining dialog acts and skill modeling: What chat interactions enhance learning rates during AI-supported peer tutoring? In Proceedings of the 17th International Conference on Educational Data Mining, 117–130. International Educational Data Mining Society. https://doi.org/10.5281/zenodo.12729784

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3, 77-101.

Braverman, H. (1974). Labor and Monopoly Capital: The Degradation of Work in the Twentieth Century. Monthly Review Press.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (1999, March). The CRISP-DM user guide. In 4th CRISP-DM SIG Workshop in Brussels in March (Vol. 1999).

Chen, G., Chen, S., Liu, Z., Jiang, F., & Wang, B. (2024, November). Humans or LLMs as the judge? A Study on judgement bias. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 8301-8327.

Coburn, C. E., & Penuel, W. R. (2016). Research-practice partnerships in education: Outcomes, dynamics, and open questions. Educational Researcher, 45(1), 48–54.

Corbin, J. M., & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. Qualitative Sociology, 13(1), 3-21.

DAMA International. (2017). DAMA-DMBOK: Data Management Body of Knowledge (2nd ed.). Technics Publications.

Dede, C., Richards, J., & Saxberg, B. (Eds.). (2019). Learning Engineering for Online Education: Theoretical Contexts and Design-Based Examples. New York: Routledge.

Dedehayir, O., & Steinert, M. (2016). The hype cycle model: A review and future directions. Technological Forecasting and Social Change, 108, 28-41.

de Morais, F., Goldoni, D., Kautzmann, T., da Silva, R., & Jaques, P. A. (2023). Automatic sensor-free affect detection: A systematic literature review. arXiv preprint arXiv:2310.13711.

Dubois, Y., Galambosi, B., Liang, P., & Hashimoto, T. B. (2024). Length-controlled alpacaeval: A simple way to debias automatic evaluators. arXiv preprint arXiv:2404.04475.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Goodell, J., & Kolodner, J. (Eds.). (2023). Learning Engineering Toolkit: Evidence-Based Practices from the Learning Sciences, Instructional Design, and Beyond. New York: Routledge.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., Wang, S., Zhang, K., Lin, Z., Zhang, B., Ni, L., Gao, W., Wang, Y., & Guo, J. (2024). A survey on LLM-as-a-judge. The Innovation, 0(0), 101253.

Guhathakurta, D. (2017). How to implement a standardized process for execution and delivery of data science solutions: Microsoft's Team Data Science Process (TDSP). Presentation at the Global Big Data Conference, Santa Clara, CA USA.

Haynes, S. N., Richard, D. C. S., & Kubany, E. S. (1995). Content validity in psychological assessment: A functional approach to concepts and methods. Psychological Assessment, 7(3), 238–247.

He, M., Baker, R.S., Hutt, S., Zhang, J. (2022). A less conservative method for reliability estimation for Cohen's Kappa. Proceedings of the International Conference on Quantitative Ethnography.

Heffernan, N. T., & Heffernan, C. L. (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), 470-497.

Hollands, F., & Bakir, I. (2015). Efficiency of automated detectors of learner engagement and affect compared with traditional observation methods. New York, NY: Center for Benefit-Cost Studies of Education, Teachers College, Columbia University.

Holstein, K., McLaren, B. M., & Aleven, V. (2019). Co-designing a real-time classroom orchestration tool to support teacher–AI complementarity. Journal of Learning Analytics, 6(2), 27-52.

Huang, H., Bu, X., Zhou, H., Qu, Y., Liu, J., Yang, M., Xu, B., & Zhao, T. (2025, July). An empirical study of LLM-as-a-judge for LLM evaluation: Fine-tuned judge model is not a general substitute for GPT-4. In Findings of the Association for Computational Linguistics: ACL 2025, 5880-5895.

Hutt, S., Wong, A., Papoutsaki, A., Baker, R. S., Gold, J. I., & Mills, C. (2024). Webcam-based eye tracking to detect mind wandering and comprehension errors. Behavior Research Methods, 56(1), 1-17.

Kessler, A., Craig, S. D., Goodell, J., Kurzweil, D., & Greenwald, S. W. (2023). Learning engineering is a process. In J. Goodell & J. Kolodner (Eds.), Learning Engineering Toolkit: Evidence-Based Practices from the Learning Sciences, Instructional Design, and Beyond. New York: Routledge.

Krumdick, M., Lovering, C., Reddy, V., Ebner, S., & Tanner, C. (2025). No free labels: Limitations of LLM-as-a-judge without human grounding. arXiv preprint arXiv:2503.05061.

Kumar, N. A., Pham, C. M., Iyyer, M., & Lan, A. (2025). Whose story is it? Personalizing story generation by inferring author styles. arXiv preprint arXiv:2502.13028.

Levin, N., Baker, R.S., Nasiar, N., Fancsali, S., Hutt, S. (2022). Evaluating gaming detector model robustness over time. Proceedings of the 15th International Conference on Educational Data Mining.

Lincoln, Y. & Guba, E. (1985). Naturalistic Inquiry. Beverly Hills, Calif. : Sage Publications.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., & Zhu, C. (2023). G-Eval: NLG evaluation using Gpt-4 with better human alignment. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Marioriyad, A., Rohban, M. H., & Baghshah, M. S. (2025). The silent judge: Unacknowledged shortcut bias in LLM-as-a-judge. In NeurIPS 2025 Workshop: Reliable ML from Unreliable Data.

Maxwell, J. A. (2013). Qualitative Research Design: An Interactive Approach. Sage.

Moorcock, M. (1970). The Eternal Champion. London, UK: Grafton.

Norman, J. D., Rivera, M. U., & Hughes, D. A. (2026). Reliability without validity: A systematic, large-scale evaluation of LLM-as-a-judge models across agreement, consistency, and bias. arXiv preprint arXiv:2606.19544.

Ocumpaugh, J., Baker, R., Gowda, S., Heffernan, N., Heffernan, C. (2014). Population validity for educational data mining models: A case study in affect detection. British Journal of Educational Technology, 45(3), 487-501.

Peczuh, M. C., Kumar, N. A., Baker, R., Lehman, B., Eisenberg, D., Mills, C., Chebrolu, K., Nashi, S., Young, C., Liu, B., Lachman, S.. & Lan, A. (2026). Toward LLM-supported automated assessment of critical thinking subskills. Proceedings of the International Conference on Educational Data Mining.

Penuel, W. R., Fishman, B. J., Cheng, B. H., & Sabelli, N. (2011). Organizing research and development at the intersection of learning, implementation, and design. Educational Researcher, 40(7), 331–337.

Penuel, W. R., Roschelle, J., & Shechtman, N. (2007). The WHIRL co-design process: Participant experiences. Research and Practice in Technology Enhanced Learning, 2(1), 51–74.

Perdomo, J. C., Britton, T., Hardt, M., & Abebe, R. (2025). Difficult lessons on social prediction from Wisconsin public schools. In Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency (FAccT '25). https://doi.org/10.1145/3715275.3732175

Rodrigo, M. M. T., Baker, R. S. J. d., McLaren, B., Jayme, A., Dy, T. (2012). Development of a workbench to address the educational data mining bottleneck. Proceedings of the 5th International Conference on Educational Data Mining, 152-155.

Roll, I., & Winne, P. H. (2015). Understanding, evaluating, and supporting self-regulated learning using learning analytics. Journal of Learning Analytics, 2(1), 7–12.

Ryng, H., & Suhr, D. (2026). Learning engineering by design: An agentic AI application for rapid, personalized health & safety training in disaster response and hazardous environments. In Proceedings of the Learning Engineering Research Network Convening (LERN 2026).

Saldaña, J. (2011). Fundamentals of Qualitative Research. Oxford university press.

Schäfer, F., Zeiselmair, C., Becker, J., & Otten, H. (2018, November). Synthesizing CRISP-DM and quality management: A data mining approach for production processes. In 2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD), 190-195. IEEE.

Shaffer, D. W. (2017). Quantitative Ethnography. Cathcart Press.

Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K. R. (2021). Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology. Machine Learning and Knowledge Extraction, 3(2), 392-413.

Švábenský, V., Baker, R. S., Zambrano, A., Zou, Y., & Slater, S. (2023). Towards generalizable detection of urgency of discussion forum posts. In Proceedings of the 16th International Conference on Educational Data Mining, 302–309. International Educational Data Mining Society. https://doi.org/10.5281/zenodo.8115790

Swamy, V., Radmehr, B., Krco, N., Marras, M., & Käser, T. (2022). Evaluating the explainers: Black-box explainable machine learning for student success prediction in MOOCs. Proceedings of the 15th International Conference on Educational Data Mining, 98.

Taylor, F. W. (1911). The Principles of Scientific Management. NuVision Publications, LLC.

Thomas, D. R., Borchers, C., Vanacore, K. P., Koedinger, K. R., & Kizilcec, R. F. (2026). Modernizing ground truth: Four shifts toward improving reliability and validity in AI in education. In Proceedings of the 27th International Conference on Artificial Intelligence in Education (AIED 2026).

Tracy, S. J. (2010). Qualitative quality: Eight “big-tent” criteria for excellent qualitative research. Qualitative Inquiry, 16(10), 837-851.

Ventura, M., & Shute, V. (2013). The validity of a game-based assessment of persistence. Computers in Human Behavior, 29(6), 2568-2572.

Weigand, A. C., & Kindsmüller, M. C. (2021, July). HCD3A: An HCD model to design data-driven apps. In International Conference on Human-Computer Interaction, 285-297. Cham: Springer International Publishing.

Xu, X., Wang, C., Jiang, J., Yang, P., Fu, Q., Dhall, M., Zhang, W., & Zhu, L. (2026). LLM-as-judge in education: A curriculum-grounded marking pipeline. arXiv preprint arXiv:2606.17507.

Yang, T., Shi, T., Wan, F., Quan, X., Wang, Q., Wu, B., & Wu, J. (2023, December). PsyCoT: Psychological questionnaire as powerful chain-of-thought for personality detection. In Findings of the Association for Computational Linguistics: EMNLP 2023, 3305-3320.

Yardley, L. (2000). Dilemmas in qualitative health research. Psychology and Health, 15(2), 215-228.

Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P.-Y., Chawla, N., & Zhang, X. (2025, April). Justice or prejudice? Quantifying biases in LLM-as-a-judge. In International Conference on Learning Representations.

Zambrano, A.F., Liu, X., Barany, A., Baker, R.S., Kim, J., Nasiar, N. (2023). From nCoder to ChatGPT: From automated coding to refining human coding. Proceedings of the International Conference on Quantitative Ethnography.

Zambrano, A.F., Wei, Z., Zhang, J., Baker. R.S., Ocumpaugh, J., Barany, A., Liu, X., Zhou, Y., Paquette, L., Ginger, J., Borchers, C. (2026). Data plus theory equals codebook: Leveraging LLMs for human-AI codebook development. Journal of Educational Data Mining, 18(1), 25-65.

Zeng, Z., Chaturvedi, S., & Bhat, S. (2017). Learner affect through the looking glass: Characterization and detection of confusion in online courses. In Proceedings of the 10th International Conference on Educational Data Mining (EDM 2017). International Educational Data Mining Society.

Zheng, L., Chiang, W. L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. Advances in Neural Information Processing Systems, 36, 46595-46623.

Zhu, Z., Novikova, J., & Rudzicz, F. (2019, June). Detecting cognitive impairments by agreeing on interpretations of linguistic features. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 1431-1441.

Zwetsloot, I. M., Kuiper, A., Akkerhuis, T. S., & de Koning, H. (2018). Lean Six Sigma meets data science: Integrating two approaches based on three case studies. Quality Engineering, 30(3), 419-431.