Challenges in Adopting Artificial Intelligence Based User Input Verification Framework in Reporting Software Systems

Dong Jae Kim*, Steve Locke*, Tse-Hsun (Peter) Chen*
Andrei Toma†, Steve Sporea†, Laura Weinkam†, Sarah Sajedi†
*Software PEformance, Analysis and Reliability (SPEAR) Lab, Concordia University**
Montreal, Quebec, Canada
ERA Environmental Management Solutions†
Montreal, Quebec, Canada
{k_dongja, s_loc, peterc}@encs.concordia.ca
{andrei.toma, steve.sporea, laura.weinkam, sarah.sajedi}@era-ehs.com

***Abstract*—Artificial intelligence is driving new industrial solutions for challenging problems once considered impossible. Many large-scale companies use AI to identify opportunities to improve business processes and products. Despite the promise and perils of AI, many traditional software systems (e.g., taxation or reporting) are implemented without AI in mind. Adopting AI-based capabilities in such software can be challenging due to a lack of resources and uncertainties in requirements. This paper documents our experience working with our industry partner on adopting AI capabilities in enterprise software. The enterprise software receives and processes thousands of user inputs with different configuration settings daily, which makes manual user input verification infeasible. To assist our industry partner, we design and integrate an AI-based input verification framework into the software. However, during the design and integration of the framework, we encounter many challenges that range from the requirement engineering process to the development, adoption, and verification process. We discuss the challenges we encountered and their corresponding solutions while working with our industrial partner to integrate the AI-based input verification framework into their non-AI software. Our experience report may provide valuable insight to practitioners and researchers on better integrating AI-based capabilities with existing software systems.**

***Index Terms*—User Input, Verification, Testing, Experience Report**

I. Introduction

Many existing traditional software, such as compliance reports, bookkeeping, and management software, relies heavily on user-provided data. They store and process user-provided data to help users achieve business goals (e.g., filing tax returns or generating reports). Hence, from the software engineering perspective, verifying user input quality to detect user mistakes is critical to the software product's success.

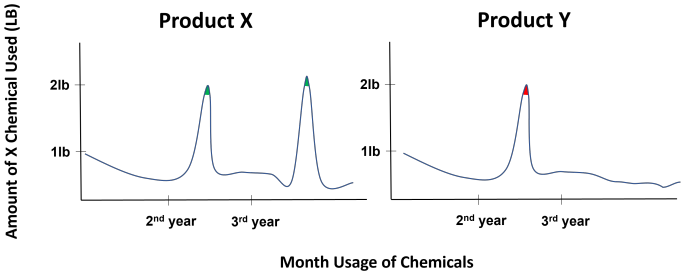
Traditional software testing ensures the functional aspects of user input and data quality, such as testing the input data for illegal characters and incorrect value types. However, context-related issues may hinder data quality. Users can input abnormally high or low inputs instead of expected values. In such a scenario, artificial intelligence (AI) based anomaly detection algorithms are a natural procedure to find irregularities in user-provided data. Using AI has several advantages. Firstly, due to considerable variations in user data, it is improbable for software developers, who may not have a detailed understanding of the client's behavior, to apply the rule-based technique to detect anomalies in the input data. From the users' perspective, it is infeasible to manually investigate tremendous user input data, whereas AI can automatically flag potential anomalies.

AI is rapidly spreading across the global business [1], from autonomous vehicle software (e.g., Tesla) to AI-guided surgeries in medicine [2]. In light of its prevalence, companies new to AI have a misconception that AI is a magic bullet capable of providing answers to any problem. Our experience working with an industrial partner new to AI has let us discover many unexpected challenges that hinder AI implementation. Challenges adopting AI from an engineering perspective are relevant yet rarely discussed in many research-industrial collaborations implementing AI. Hence, this paper documents our experience integrating AI into large-scale reporting software.

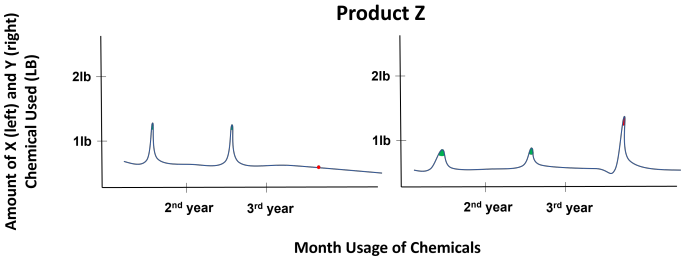
Our subject system is a large-scale business-to-business enterprise software in environmental and manufacturing. It allows manufacturers to maintain and regulate chemical usage to adhere to government health and safety regulations. The software takes thousands of user inputs with many configurations, values, and units daily. Since the quality of data input directly connects to the stakes of the manufacturing companies, our industrial partners have dedicated environmental analysts to verify the quality issues. Maintaining high-quality data is of utmost interest to stakeholders (i.e., manufacturers) since even slight deviations may cause negative consequences for stakeholders against government regulations. To reduce the manual intensive stage of anomaly detection, we implemented and integrated AI to automatically detect anomalies and recommend them to domain experts.

We provide insights on the three stages of challenges when integrating the AI component into a non-AI system: 1) Challenges in the requirement engineering process with

Anomaly Pattern	Description
Pattern 1 - Amount Entry	Irregularities in the product usage reported by the users. Namely, a reported input is too little or too much in value compared to the general amount.
Pattern 2 - Period Entry	Irregularities in the usage period reported by users. These are durations for which a manufacturer may use the product. Hence, a reported input may have a longer or shorter usage period than the general usage period. For example, product X, generally used for 30 days, was reported to be used for 90 days.
Pattern 3 - Report Frequency	Irregularities in the report frequency. These are frequencies in which user may report product usage. For example, product X, generally reported at the beginning of every month, was reported three months later. Hence, there is a missing entry.



SubFig. (1) Two products made from the same chemical X



SubFig. (2) Same products made from the different chemical

Fig. 1: The first table shows different types of anomalies detected in our input-verification framework. The plots below show the complexity of amount distribution difference between different chemical usages. Note, the plot only shows the amount of used chemicals entered by users, it does not show period or report frequency of the data entered.

a development team from a diverse background. 2) Challenges in adopting AI in user-intensive software from an end-user navigation perspective. 3) Challenges in AI verification, specifically anomaly verification, where stakeholders have diverse patterns of operational behavior and context-specific anomalies. We then provide insights into avoidance strategies for how we overcame each of these challenges. For 1) we share our process to facilitate better identification of the scope of problems with domain experts; for 2) we share our process to facilitate better adoption of the AI component; and for 3) we share our process of verifying the robustness of anomaly detection model when ground truth is absent. In particular, the main contributions of this paper are:

- We provide an experience report that discusses the challenges of integrating an AI-based user input verification framework into an existing non-AI software system, which may help other practitioners who face similar issues in the industry.
- We discuss our challenges encountered during requirement engineering, AI adoption, and validation.
- We provide a detailed design decision of our AI-based user input verification framework.

Paper Organization. Section II discusses our background & motivations. Section III presents the design and implementation of our AI-based user-input verification application. Section IV discusses the challenges encountered in the integration and our solutions to these challenges. Section VI surveys related work. Finally, Section VII concludes the paper.

II. Background and Motivation

This section discusses our industry partner's problem statements and motivation for adopting AI (artificial intelligence) to the software. Their software is a large-scale business-to-business software that allows stakeholders to book-keep chemicals used in their manufacturing process. The software then automatically generates reports, enabling stakeholders to report to the government and help them align with environmental regulations. However, the software is not a direct plugin in their manufacturing process, i.e., it does not collect live usage data. It requires humans to manually enter the collected data into our systems after manufacturing, which opens potential data quality issues. Even the slightest mistake in the data input may cause severe consequences for our stakeholders. Hence, our industrial partners' mission is to work closely with stakeholders to improve the quality of data coming into their systems.

While our industrial partner has a great data quality assurance team, it takes time to validate every piece of data entered into the system. Dealing with 100+ clients with various manufacturing products and chemicals used in the process exacerbates manual labor, which could cause many unchecked anomalies to manifest in the data. Hence, our industrial partner expressed interest in using AI to detect data anomalies

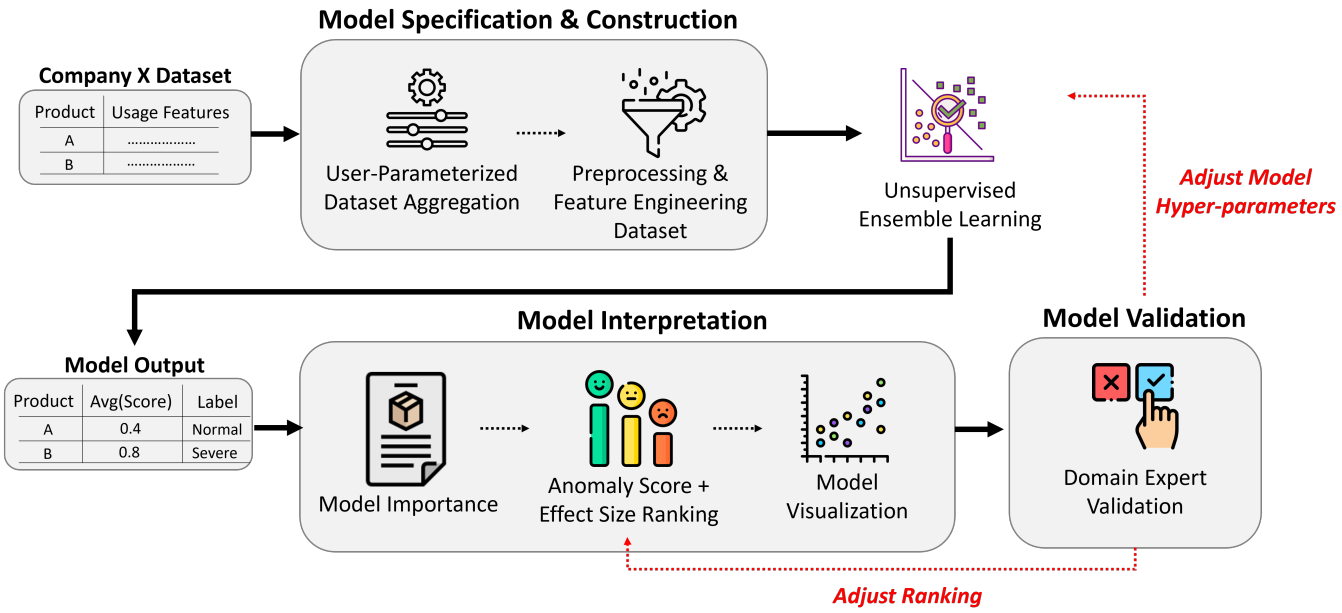


Fig. 2: The overview of the end-to-end process in our outlier detection strategy.

automatically. We currently detect three types of anomalies suggested as critical by domain experts. The table in Figure 1 summarizes the initial types of anomalies detected in our AI-based input verification framework: Anomalous amount, period, and frequency, and their correct and incorrect values.

While seemingly straightforward, identifying quality issues (e.g., anomalies) in these chemical usage inputs is challenging as it differs on the products manufactured and stakeholders. For instance, let us look at the amount of the same chemical used (e.g., lb) during manufacturing products X and Y in SubFigure 1 in Figure 1. In such case, some AI models would algorithmically flag the high usage amount to manufacture product X in the second year as normal (e.g., recurring pattern) and flag product Y's second year as anomalous. However, the high peak in product Y is actually due to increased manufacturing demand and is normal behavior. SubFigure 2 shows another distribution change when using a different chemical for manufacturing Z. The model may flag the first two maxima usage as non-anomalous and the final data point as anomalous as it does not agree with recurring patterns. However, we find that stakeholders may replace materials in manufacturing, hence a decrease in usage for the third year for the first plot and its corresponding increase in the second plot. Hence, many context-specific anomalies can cause false alarm. Since the software is critical for stakeholders, our partner is interested in decreasing false negatives by flagging as many potentially anomalous data points as possible at the expense of allowing false positives.

III. The Design and Implementation of Our AI-Based User-Input Verification Framework

This section discusses our AI-based user input verification framework, which may help other practitioners who want to adopt similar frameworks to their systems. Figure 2 shows an overview of the framework. The pipeline acts as a channel between stored data and downstream anomaly recommendations. In particular, our framework has three main processes given the user input data: (1) Model specification and construction, (2) Model interpretation, and (3) Model validation. Section 4 will further discuss our design decisions and the challenges they helped to overcome in AI integration.

A. (Component-1) Model Specification & Construction

(A) *User-Provided Parameterization.* We allow users to decide how to aggregate the dataset before executing the AI model. For example, each client has various facilities for manufacturing their products. Depending on its facility, the exact product X could also have different usage behaviors.

(B) *Preprocessing & Feature Engineering.* We preprocess (e.g., feature engineer) our dataset to allow the data to be learnable by our AI model and detect anomalies indicated in the top table of Figure 1.

1) Amount Entry - To detect anomalies in amount entry, we preprocess amount entry by converting all unit scales to the same scale. For example, product X may utilize volume or mass units, which are inconsistent.

2) Period Entry - To detect anomalies in period entry, we preprocess the period entry by finding the difference between the start and end date of usage for product X.

3) Report Frequency - Finally, we need at least two rows of entries to find anomalies in report frequency. For example, given Xi_{start}, Xi_{end} and Xj_{start}, Xj_{end}, which are the start and end date of usage for product X for time i and j, entry frequency is $Xj_{start} - Xi_{end}$. This value tells us the report duration between each row of entries.

(C) *Ensemble Unsupervised Learning.* After data is aggregated and preprocessed according to the user-configured parameterization, the next step is to run our AI model to detect

anomalies. Since we do not know the exact recall of the outliers, we chose an ensemble of unsupervised anomaly detection algorithms, including isolation forest, local isolation factor, density-based spatial clustering, and the traditional Z-score statistic. Each then outputs a score indicating the likeliness of an outlier. We calculated the weighted average of their score to obtain the unified anomaly score. Namely, we have different techniques to help us flag many suspicious anomalies.

B. (Component-2) User Parameterized Post-Processing

(D) *Rank Anomalies based on Anomaly Score.* We sort the weighted average anomaly score from the ensemble model to prioritize anomalies that may require immediate attention, as domain experts may not have the time to validate all potential anomalies. We consider moderate anomalies to lie between 0.5 and 0.75 and severe ones to be greater than 0.75. Scores less than 0.5 are considered non-anomalous. These anomaly severity cut-off scores are initial heuristics deployed by our pipeline and may be subject to modifications by end users.

(E) *Ranking the Anomalies based on Effect Size.* One limitation of unsupervised outlier detection algorithms is the lack of effect size. For example, multiple outliers have similar outlier scores but different sizes in magnitude. Hence, we calculate the absolute and relative difference for outliers against their median distribution, allowing users' to select anomalies based on effect size.

(F) *Model Importance in Unsupervised Ensemble Learning.* Clients may want to understand the result of the ensemble models. Since the different models in the ensemble capture different characteristics in the data, we describe the model's importance in the ensemble. Namely, we specify which model contributes to a higher weighted average anomaly score.

(G) *Visualization Techniques.* We also implemented a user interface to visualize our anomaly detection. We mainly focus on two graphs: time-series and cluster, which are easier to comprehend for wider audiences. For example, the scatter plot will show a product's historical usage pattern color-coded into the three severity of anomalies (non-anomalous, moderate, and severe).

C. (Component-4) Model Validation

(H) *Domain Expert Validation.* We do not have a ground truth for the anomalies in the user-provided inputs. The ground truth may vary between manufacturers, making the model validation especially challenging. Thus, our goal is to fine-tune result based on the direct feedback given by the domain experts and hide the expert-flagged false positives in future reports. We currently implemented a report generation module scheduled by users to run our AI model, generate a report and send it automatically to end-users.

IV. Challenges in AI Integration in Traditional Companies

In this section, we share our challenges encountered in AI integration. We arrange our findings according to the three main activities *Requirement Elicitation*, *Adoption*, and *Evaluation*. For each, we discuss concrete challenges that are especially important for practitioners and AI engineers. We arrange our discussion into the following flow, 1) a description of the challenge, 2) a misstep that led to the manifestation of the challenge, 3) share lessons learned in the form of avoidance strategy to help practitioners, and 4) additional discussion on how this challenge fit in the scope of our overall system.

A. Requirement Elicitation between AI and non-AI experts

Implementing AI is challenging from requirement specifications. This section demonstrates how to manage requirements.

A.1) Data Source Requirement.

Challenge A.1. Training data is an integral part of the AI system. The model's performance depends heavily on the data example size that the model could learn. However, finding all relevant data sources is a time-consuming aspect of the requirement engineering process, especially when data is absent. We work in a team with diverse expertise arranged in two groups, the AI engineer in one and the other group involves business managers, domain experts, and scientific writers. While input from such diverse expertise could offer valuable help, knowledge transfer can be ad-hoc and may lack structure. Provoking valuable feedback from domain experts in an input that can directly specify the data discovery is challenging.

Misstep A.1. Knowledge transfer works both ways and is *Mutual* for both AI engineers and domain experts. While domain experts have valuable insights, it is up to the AI engineer to help domain experts convey their insights. Our industrial partner said: "*We do not know how to help you get the relevant resources since we don't know what it is that you need*". In knowledge transfer, it is unclear how much input is sufficiently collected from domain experts to cover all the requirements. In our case, we went straight into modeling the AI and later found the missing data source, which caused the evaluation slowdown.

Avoidance Strategy A.1. For domain experts, understanding the inner workings of AI algorithms should not be the focus of the collaboration since the common consensus is to leave the detail to the experts. However, we found that giving a short workshop on a high-level understanding of AI pipeline improved data source retrieval. In the workshop, we covered feature engineering strategies on how to take current data into a processed form that can be used by the model, and raising intuition behind AI algorithms to help understand AI's limitations. We communicated to the domain experts that AI is only powerful as the data wrangling steps, "*Without data sources and a proper preprocessing of that data source, we cannot fully leverage AI.*" Since domain experts know their data, the feature engineering step was more apt to their expertise. One domain expert pointed out, "*Our X dataset can tell us when clients are likely to ramp up the manufacturing process. It can help us reduce false positives.*"

Discussion A.1. For companies new to AI, direct knowledge transfer between the different bodies of expertise and AI engi-

neers is challenging due to the disparity between expectations and outcomes of AI. Nevertheless, we strongly suggest AI engineers invest time in teaching domain experts in the whole AI development process. From our experience, the more we involve domain experts in many aspects of AI development the better they can provide feedback.

A.2) Expectation Requirement.

Challenge A.2. It is easier to satisfy stakeholders by making requirements and expectations explicit. However, requirements are never complete in an evolving environment (e.g., anomaly detection) and AI must evolve with it. There are always uncertainties about how to label the data when data behavior changes, i.e., often very frequently, evaluate the prediction and integrate AI into the final legacy systems. While these are necessary, discussing these concerns earlier in requirement engineering is very challenging.

Misstep A.2. A common misconception is that AI is another technology that should be independent of the requirement engineering in legacy systems. In our case, due to challenges in requirement engineering, one mistake we committed was building AI immediately and incrementally improving the model results upon domain feedbacks and justifications. However, incremental changes may go on for days or months and still may not yield output in a usable form. Hence, delays may cause stakeholders to lose interests/commitment.

Avoidance Strategy A.2. An AI engineer's responsibility is to raise awareness of what it takes to integrate AI into the end product during the initial requirement engineering process. For example, evaluation requirements are never complete in anomaly detection due to missing ground truth, especially in the evolving system. It is impossible to give a success metric (i.e., accuracy) to indicate the completion of AI. Our industrial partner said: "*Similar to our safety-critical software, which accumulated decade-old knowledge receiving feedback from stakeholders, this anomaly detection is an ongoing effort to help them regulate their environmental laws.*" As many practitioners may view AI as a magic bullet (i.e., magically providing solutions), clarity on its expectations is necessary early on to avoid dissatisfaction. For example, anomaly detection often cannot be evaluated during training due to missing labels. AI engineers must help practitioners understand the concept of adaptive learning. It deals with limited labels at runtime by adapting to incoming data (i.e., human labeling) to make further adjustments to its AI parameter.

Discussion A.2. The AI model we built is an assistance to help these domain experts improve at anomaly detection but not a complete anomaly detection solution that should be trusted blindly. Our industrial partners' software is critical and will always require verification by domain experts (e.g., environmental scientists). Hence, it reduces the existing overhead of domain experts, but it does not create new overheads. While initial fine-tuning takes time, this tuning may go a long way to help detect new types of anomalies that may come in the future. It is crucial that AI engineers and industrial collaborators have a clear understanding of these expectations early on.

B. AI Adoption in traditional non-AI software

Implementing an AI application is tricky from an end-user adoption perspective. This section demonstrates better adoption stems from enhancing user navigation and interpretation.

B.1) Change Management in AI.

Challenge B.1. Stakeholders often need help understanding what comes after the model implementation. In our case, we needed to integrate the AI pipeline first to acquire feedback (e.g., correct anomalies) from domain experts to adjust our model. This differs from the traditional AI pipeline, where labels are available, and the model can be trained before integration. In practice, this stage of integration can be time-consuming as domain experts may have little time and may not see how AI could be utilized in practice.

Misstep B.1. In our collaboration, we decided every development decision from AI research perspectives, such as selecting which model to use, understanding the data source, preprocessing data, and evaluating the model. When we were ready to integrate our AI for evaluation in run-time, we found that domain experts did not know what it took to incorporate it into their system. We frequently had back-and-forth meetings, where we asked "*We shared out code, and AI is ready for integration to obtain feedback on the results*", and domain experts' reply would be "*Due to X release in our software, we did not have time to work on it*". It took us a significant amount of time to push the integration and finally obtain feedback.

Avoidance Strategy B.1. While integration slowdowns are related to social and business issues, from our experience, AI engineers can still impact change management. Due to domain experts' limited time, we took the initiative to build a simple prototype to help domain experts see the end product. For example, Figure 3-C displays the general statistics about the anomaly type and the number of anomalies detected for each product. Furthermore, as shown in Figure 3-D, we implemented the table that shows the original dataset with severe anomalies highlighted in grey. For example, the severe anomaly is indicated by a 0.125 value reported by the user, which was lower than the general report amounts, which is, on average, 1.041. Although this interface shows the anomalies detected for pattern 1 (e.g., anomalous amount), the interface for other patterns is similar. As a design decision, if we detect pattern 1, we only show the result for pattern 1. This design decision keeps the user focused on analyzing one anomaly at a time. We also added functionality where end-users can export the table as an excel spreadsheet. This step is crucial since users come from various non-software backgrounds, and excel is the go-to option for data analytic for traditional companies. Therefore, users can quickly obtain a glimpse of the anomalies and export the dataset into the dedicated analytic tool for detailed validation. When our industrial partner witnessed the prototype, they were able to give more suggestions. One domain expert said: "*This really helps to see how we can integrate AI. We now have some idea as to what we can reuse.*" This strategy is incorporated in Model Interpretation component shown in Figure 2.

C

EXPORT

Product	Total Anomalies Detected
Product A	2

D

Product	Location1	Start Date	End Date	outliers	Amount	Relative Change	Absolute Change	Mean	Median	Max	Min
Product A	...	9/17/2019 12:00	9/17/2019 12:00	Normal	0.75	0.4	0.5	1.041	1.25	1.375	0.125
Product A	...	9/24/2019 12:00	9/24/2019 12:00	Normal	0.75	0.4	0.5	1.041	1.25	1.375	0.125
Product A	...	10/15/2019 12:00	10/15/2019 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125
Product A	...	12/10/2019 12:00	12/10/2019 12:00	Normal	0.75	0.4	0.5	1.041	1.25	1.375	0.125
Product A	...	1/28/2020 12:00	1/28/2020 12:00	Severe_Outlier	0.125	0.9	1.125	1.041	1.25	1.375	0.125
Product A	...	1/28/2020 12:00	1/28/2020 12:00	Moderate_Outli…	1.375	0.1	0.125	1.041	1.25	1.375	0.125
Product A	...	2/4/2020 12:00	2/4/2020 12:00	Normal	1.125	0.1	0.125	1.041	1.25	1.375	0.125
Product A	...	3/10/2020 12:00	3/10/2020 12:00	Normal	1	0.2	0.25	1.041	1.25	1.375	0.125
Product A	...	5/5/2020 12:00	5/5/2020 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125
Product A	...	6/9/2020 12:00	6/9/2020 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125
Product A	...	6/30/2020 12:00	6/30/2020 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125
Product A	...	8/11/2020 12:00	8/11/2020 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125
Product A	...	9/15/2020 12:00	9/15/2020 12:00	Normal	1	0.2	0.25	1.041	1.25	1.375	0.125
Product A	...	10/13/2020 12:00	10/13/2020 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125
Product A	...	12/1/2020 12:00	12/1/2020 12:00	Normal	1.25	0	0	1.041	1.25	1.375	0.125

Fig. 3: Dashboard of anomaly detection results. We aggregate the dataset at a record level according to user-parameterization in Section III-A. Interface C shows the statistics for the anomalous amount entry for product A. Interface D shows the amount entry for product A. Grey highlights the severely anomalous entry.

Discussion B.1. While high prediction and accuracy are the ambitions of AI engineers, the story-telling aspect of AI is important for its integration as we are building AI for end-users. Based on our experience, companies new to AI may have challenges from AI implementation to integration. Namely, due to a lack of resources or existing infrastructure, integrating AI into existing software is time-consuming. Hence, AI engineers should invest time in prototyping the user interface to demonstrate theories into practice. Based on our experience, Dash Plotly [3] was the most straightforward framework to prototype the reporting component of our anomaly results. Dash Plotly ships with many out-of-the-box components that are easy to prototype. Many frameworks and libraries help integrate the user interface and AI-based input verification framework. Another advantage was that the API is abstracted in Python, which can integrate with many popular Python-based AI/ML frameworks and libraries (e.g., Pytorch and Scikit-learn). Finally, we find it vital to use tools and analytics that are more familiar to a broader range of audiences. Hence, we implemented our AI application as an upstream component to quickly summarize anomalies and serialize results to any downstream analytics tool of the user's choice. For example, domain experts wanted data serialization into Excel to perform additional validation.

B.2) End-User Navigation.

Challenge B.2. Our stakeholders come from various manufacturing domains with unique operational behaviors. Each manufacturer has multiple facilities that change the outcome of the dataset. There are also time constraints in the datasets. Some stakeholders care about yearly anomalies, while others care about weekly criteria. To cater to diverse stakeholders,

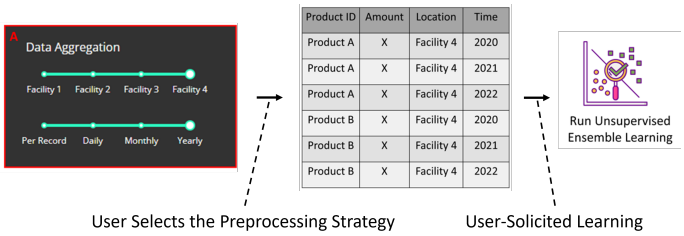


Fig. 4: Interface A allows users to select aggregation strategy. The user defines what the model should learn. The preprocessing is based on (1) Facility 4 and (2) Yearly.

the AI pipeline, pre-processing, and feature engineering details must also vary. However, it is challenging to derive all rules in the pipeline to cater diverse stakeholders.

Misstep B.1. Many existing AI pipelines in the wild generally stay consistent in the pipeline step, i.e., preprocessing. Once built, the model remains constant, and its source of variation is its ability to intake new dataset examples to learn the new pattern. Our first iteration of the model did not consider pipeline adaptation due to a lack of understanding of user behavior. We chose two stakeholder datasets (e.g., out of 100+) for our initial evaluation to analyze with the domain experts and applied a similar AI pipeline. However, the preprocessing for one company could not work for another due to differences in behavior. Our model needed to adapt in this step from stakeholder guidance.

Avoidance Strategy B.2. The issue with our current pipeline is that our preprocessing must adapt to stakeholders' behavior. For example, if the three anomaly types are only relevant monthly due to the way stakeholders enter the chemical usage, then it may not make sense to study daily. Only they know

how to preprocess the dataset. As a solution, we implemented a user-enabled parameterization strategy where users can change preprocessing step in the pipeline. One domain expert said: "*The parameterization allows the adoption by closely aligning the navigation from the end-user perspective*". We implemented various ways for user interactions, allowing better control over the model input. Figure 4 - A, Component 1 (i.e., Model specification & Construction) allows users to decide how to aggregate the dataset before executing the anomaly detection model. As suggested by domain experts, we provided two aggregation parameters based on facility location and time. This strategy is incorporated in Model Specification component shown in Figure 2

Discussion B.2. From our experience, user-enabled parameterization mutually benefits both domains, i.e., AI engineers and stakeholders. For example, domain experts, who are more familiar with their data, can become directly involved in specifying the model's outcome, increasing trusts in the model. It also simplifies the implementation from the application designers' perspective. For example, it would be impossible to cater to every client since there is tremendous end-user behavior. We allow users to decide the preprocessing that are advantageous for their environment. Hence, researchers engaging in AI integration in the industry should consider: 1) it is crucial to look for opportunities to parameterize the model based on domain experts' input, and 2) these parameterized configurations should be impactful, such as to allow domain experts to make operational decisions.

B.3) Explainability of AI.

Challenge B.3. AI consists of many complex parameters hidden away for ease of use for end users. However, abstraction may hinder user understanding and cause a loss of trust in the AI output. Finding a balance between AI interpretability and abstraction is challenging.

Misstep B.3. We used unsupervised ensemble learning to tackle the absence of ground-truth. Since different models have different advantages, we wanted to run many distinct ones to flag many suspicious anomalies. We initially abstracted many details of our ensemble model to ease the interpretation of the AI output. We only revealed the final anomaly score for end users and described a high anomaly score as being more anomalous. However, we found that having a metric without explanation caused a lot of suspicion for domain experts. When results did not agree with their opinion, they did not have good intuition behind the output to give feedback. One domain expert said: "*How do we justify the given anomaly score?*". Explaining the weighted average output of the ensemble model was a huge challenge for end-user interpretation.

Avoidance Strategy B.3. Most AI model uses feature importance to explain the model result. Unfortunately, feature importance is unattainable for unsupervised learning, where ground truth is absent. To allow the interpretability of the ensemble model, we took on a different approach. We explain individual anomaly scores returned by distinct models and how much they contribute to the final score. Namely, if the isolation forest model contributed more to the score than the local outlier factor, we described the result for anomalous report frequency as, "*31 days is the number of days between two report dates. Historically, users reported one day apart; 31 days is a rare frequency.*" If the local outlier factor score contributed to the final score, we would append the following, "*31 days is also anomalous based on its neighboring pattern of report frequency*". Describing the model based on the intuition of the outlier algorithm helped users understand the output. Namely, domain experts could either agree or disagree with the result without needing clarification about the ambiguous metric. Model importance is incorporated in *Model Interpretation* component shown in Figure 2.

Discussion B.3. In our learning, the ensemble model can flag many patterns that are less intuitive for end-users. Domain experts initially disagree with the flagged anomaly, even if it is truly anomalous. Hence, we provided how our model flagged them as an anomaly, using the model importance factors. Since different models look for different abnormalities in the data, explaining the anomaly based on model contributions improved interpretability.

C. Ongoing Challenge in Verification of User-centric AI-based Models

Integrating an AI application is challenging from an AI evaluation perspective. As ground truth is absent initially, anomaly detection requires working with domain experts to evaluate accuracy of the model.

C.1) Bias in Human-in-the-loop.

Challenge C.1. The limitation of an outlier detection algorithm is bounded by accuracy issues. In our case, we find that none of the manufacturing stakeholders in the industry have considered quality control at this level in the business process. Hence, no prior dataset exists to help us evaluate our result. Our evaluations require verification with stakeholders involved in the manufacturing process, which can be time-consuming.

Misstep C.1. In the safety-critical system, domain experts must always need to verify the results. While domain experts' opinions may be the only source of validity in anomaly evaluation, feedback can sometimes be challenging to integrate into the AI pipeline. During our collaboration, we found that feedback on anomalies could change depending on circumstances. For example, one domain expert said "*This X value is not flagged as an anomaly but should be considered anomalous because it seems anomalous compared to its nearby values.*" Another domain expert said: "*I don't think it's an anomaly because it is pretty common if you look at other data points.*" Justifications can sometimes be purely from a data distribution, which the model has already learned, instead of the actual source of the cause. We find that sometimes justifications can be uncertain and challenging to adopt.

Avoidance Strategy C.1. To avoid bias in the anomaly justifications, we can consult two groups of domain experts, our industrial partner, and the stakeholder domain expert. The industrial domain experts first derive justifications for anomalous or normal data and indicate new anomaly suggestions. Our industrial domain experts then consult the stakeholder domain

experts to clarify the justifications. Hence, experts may suggest multiple views on the anomalies. Finally, in human-in-the-loop justifications, it is very important to distinguish what has been learned by the model from the actual source of the anomalies. For example, saying that the anomaly is flagged as data does not align with general distribution is still a symptom of the anomaly. It needs to explain why it is an anomaly from a business domain perspective.

Discussion C.1. Most work in anomaly detections focuses on the algorithm of the model given ground truth. They often take for granted the labeled dataset, which is hard to obtain. Unlike traditional software settings, such as software testing, our user-centric system does not have immediately visible downstream errors. Hence, these errors, i.e., anomalies, need to be manually derived. While there are works in requirement engineering that try to automate collaborations in data labeling, it is domain-specific and difficult to utilize in our scenario [4], [5], where there are many complex depths in the anomalies. Hence, we use a more hands-on approach and rely on multi-experts opinions to suggest multiple views on the anomalies to reduce bias. However, it may be difficult to incorporate justifications into the model.

C.1) Complexity of Anomalies in Evolving System.

Challenge C.2. While unsupervised learning gives a quick analysis of anomalies in the data, there is usually a discrepancy between algorithm-assumed and real-world anomalies, which requires numerous adaptations. The challenge remains in explaining what has been learned by the model and distinguishing whether anomalies align with domain experts' views.

Misstep C.2. Based on our domain experts' feedback, some detected anomalies were common behavior shown for changes in manufacturing decisions. One domain expert said: "*a sudden decrease in chemical usage is common for companies trying out a new chemical product before phasing out its usage.*" Hence, there is a discrepancy between real-life anomalies and algorithmically-assumed anomalies. Detecting such context-related issues is often undermined in research, which only endeavors to improve synthetic accuracy. The real challenge remains understanding what has been learned by AI in a way understandable by the end-users and revisiting the requirement engineering process to adjust the model.

Avoidance Strategy C.2. Due to the existence of incorrectly flagged anomalies related to context-specific behavior, we used an example-driven feedback strategy. We present the result of each anomaly to domain experts, and they will justify whether it is true or false. Using this, our domain experts helped us curate a new dataset to add the missing context to the learning process. However, while example-driven evaluation is practical, it may not cover all potential context-dependent anomalies. One domain expert said: "*Some incorrectly flagged behaviors frequently appear within the company.*" Hence, we needed a way to ensure all examples of anomalies we validate cover an entire range of possible contexts. To tackle this issue, we decided to use an external post-processing strategy to categorize context-related behaviors. Hence, we clustered the time series of the anomalous chemical usages using a dynamic-time-warping algorithm (DTW) [6]. Other clustering techniques may work, but one advantage of DTW is that it can cluster time series of different lengths [6]. We then conducted a large-scale analysis by presenting the clustering result (e.g., similar behavior) to our domain experts to determine which behaviors are normal and should be filtered out in the output. We then preserved those time-series behaviors as negative examples, indicating potential false positives. In the future, when we run our anomaly detection, we could use the similarity score in the DTW algorithm to automatically filter out the candidates.

Discussion C.2. Our ensemble learning utilized isolation forest, which assumes no distribution and distance metric, and DBSCAN, which utilizes distance metric and its variant local outlier factor. The downside of using these models is that they cannot extract sequential information from the dataset. While we could have used deep learning (e.g., LSTM) to extract sequential information, it was impractical as datasets were too diverse and had a minimal history (e.g., some may only have 5 data points). To mitigate this, we used dynamic time warping to cluster time series to expose frequently occurring behaviors and present them to domain experts to filter out context-dependent anomalies that are not anomalies. Moreover, while a constraint-based input management system may seem probable for a particular type of dataset (e.g., value reported is greater than X), we find that it is not as easy to decide on such a level of threshold for our use cases. Since parameters may change over time; hence we rely on a continuous feedback loop by the practitioner.

C.3) Scaling Human-in-the-loop in User-Centric System

Challenge C.3. In a user-intensive system, there is no single oracle to evaluate the AI model, as anomalies are context-specific to different end-users. Hence, deriving the ground truths requires manually working with numerous stakeholders, which is time-consuming.

Misstep C.3. While human-in-the-loop is necessary for anomaly detection with non-existent labels, it is nonetheless time-consuming. Our industrial domain experts host one-to-one meetings with manufacturing companies to discuss requirement changes in the business. However, anomaly validation for 100+ client is time-consuming. Initially, we worked with a single stakeholder to verify the anomalies. However, given anomaly validation's time-consuming nature, it may need to scale better when dealing with 100+ stakeholders.

Avoidance Insight C.3. Our legacy system possesses a report generation module. Many clients employ this module for generating a monthly report to maintain compliance with government environmental protocol. Hence, we serialized our output from the AI module to the existing report generation module to send detected anomalies to the users as a report. The report module has a scheduler to send environmental reports to its companies automatically. In the report, users can provide false positive and negative feedback.

Discussion C.3. While many researchers utilize AI to achieve state-of-art performance metrics, there needs to be more discussion on obtaining ground truths for evaluation, especially

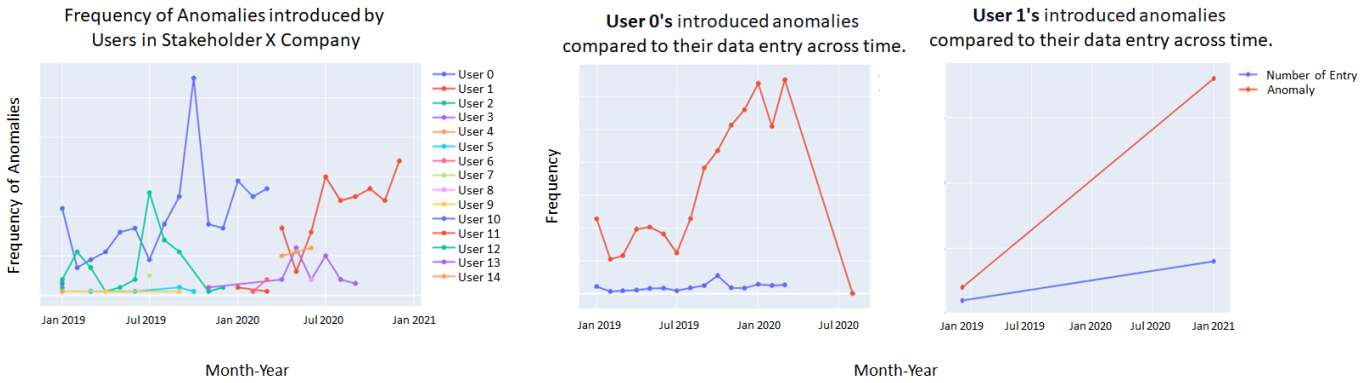


Fig. 5: Evolution of Anomalous Usage Amounts Reported by Users.

in settings with diverse user behaviors. Since we cannot avoid human validation, our solution was to use an existing function that is already prominently used, the report generation module. If the user needs to generate the report, then providing additional information on flagged anomalies should be in their interest and should have minimal impact on their workflow. This was made possible with the scalability infrastructure that our industrial software already possessed in its system. For researchers working on AI integration, similar scalability advantages will be available in their industrial partner, which can significantly enhance the process of acquiring AI feedback.

C.4) Root Cause Analysis in User-Centric Sytems.

Challenge C.4. As discussed in II, our software system collects manufacturing data from stakeholders. As data are manually inserted into the system by domain experts, mistakes are highly likely to occur during data entry. While anomaly detection is the first step toward identifying mistakes in user data, our goal is to understand the root cause of anomalies to obtain correct anomalies from normal behaviors. However, root cause analysis is difficult because stakeholders have diverse operational behaviors. Even within the manufacturing stakeholders, many users can introduce anomalies.

Misstep C.4. As our industrial partners' software is safety-critical, their concern was to reduce false negatives at the expense of allowing false positives (e.g., detecting all potential anomalies). We then needed to consult multiple domain experts, i.e., our industrial domain experts and their stakeholders, to find the correct cause-effect relationship to distinguish noise from anomalies. However, in a user-centric system, stakeholder feedbacks are very difficult to receive due to wide range of users who can introduce anomalies within the company, which causes a huge delay in our model evaluation.

Avoidance Insight C.4. While it is straightforward to leave the cause-and-effect analysis of anomalies to the stakeholders, providing them with the ability to understand anomalies may be easier for stakeholders to analyze results or even speed up feedback. In our case, improving the feedback loop is important to tune/improve the model. Hence, we hypothesized that stakeholders would be greatly interested in details about the User who introduced the anomalies over time. Hence, as shown in the left plot of Figure 5, we analyzed the number of anomalies introduced by different users each month. We can see that anomalies are introduced in different periods by different users. Given this frequency, as shown in the right plot of Figure 5, we compared the correlation between the introduced anomalies and the total number of reported records (e.g., all data entered by the users) for User 0 and User 1. A larger positive correlation indicates that a user may introduce more anomalies, i.e., make more mistakes, as they enter more inputs into the system. This can give us insights into users who require proper training to circumvent further anomalies. In the analysis, we find that User 1 has a high positive correlation, and stakeholders can discuss the anomalies with user 1. Our industrial partner possesses a generalizable report generation module. Many clients frequently employ this module for generating a monthly report to maintain compliance with government environmental protocol. Hence, we serialized our output from the AI module to the existing report generation module to send detected anomalies to the users as a report. The report module has a scheduler to send environmental reports to its companies automatically. In the report, users can provide false positive and negative feedback. This strategy is Incorporated in model interpretation in Figure 2.

Discussion C.4. Researchers working on anomaly detection in the user-provided dataset may benefit from looking at anomalies at the user level. Such information is readily available and leveraged to study the root cause of anomalies. Exposing users with a larger positive correlation with a higher effect size should indicate taking action (i.e., giving proper training to the user) to avoid future misreports.

V. Future Works & Final Discussions

While AI is a natural solution to the problem we face, we are in no way proposing a new solution to AI but instead giving an experience report on challenges we encountered when integrating AI into non-AI software. Not all companies have the scalability and resource capacity of large-scale companies like Facebook and Google. Despite this, many companies can still leverage AI and obtain a cost-efficient return on investment given the proper execution and delivery. We aim to provide such insights to companies in similar situations and help them integrate and adopt AI successfully. From our experience, there still needs clarity between AI implementation and actual adoption. Implementing AI is easy, as many readily available

API libraries exist nowadays, allowing quick prototyping. Much of the work in AI comes post-mortem; how can we validate the result so that users can trust them and use them in their operations? Moreover, sometimes research's purpose is only to improve the technique relative to some metric, which is becoming a blinding aim for many AI papers. There are tremendous other challenges that easily tip the scale for failure in AI integration; 1) how to work with domain experts effectively at every step and 2) applicable human-computer interaction for the business-to-business user-intensive system. Our ongoing challenge is the validation of our AI model. We are at a prototyping stage with a few dedicated clients with whom we have worked closely over the years. We plan to make this component available to all our users. Finally, we intend to conduct surveys to collect insights into improving anomaly validation, which would be invaluable for AI engineers.

VI. Related Works

We divide the related work into two categories: 1) user input verification in databases and 2) integrating AI-based software capabilities into existing systems.

A. User Input Verification in Databases

Verifying user inputs is one of the essential verification goals in user-centric software systems, as mistakes or anomalies in user input may result in harmful consequences for software systems. There exist diverse domains of input verification studies, from errors in clinical systems [7], cross-site scripting vulnerabilities [8], to errors in spread sheets [9], [10]. In such verification methods, there are two primary locations of the verification strategy in user inputs data: pre-sanitizing the data before insertion into the database, and post-mortem analysis, where anomalies are detected and analyzed for their root cause. Prior studies apply data sanitization techniques are used to remove illegal characters and invalid values during a user input operation [8], [11], [12]. While data sanitization may help reduce errors, many errors can bypass this stage and insert into software storage. Hence, more intelligence softwares harness the power of AI/ML to verify the user input after being inserted into storage. AI-based capability software is becoming widespread in software industries due to its ease of accessibility and automatically; AI-based software (e.g., computer vision) [13]–[16] or AI for software (e.g., harnessing AI) [17]–[19]. While many existing studies focus on the methodology for designing software with AI-based input verification capabilities, very few studies address the even more challenging nature of integrating AI-based capability into existing non-AI infrastructure.

B. Integrating AI-based Capabilities into Existing Software.

Many existing studies focus on a new methodology for designing a software system with AI-based capabilities, educational workflow [20], workflow management [21], requirement elicitation [18] and clinical science [22]. However, our work reports our experience of integrating the AI-based input verification capabilities into an existing software system that did not consider AI capabilities in its infrastructure. The most relevant to our research is work come from a study by [23], who describe the experience report of integrating autonomic computing capabilities to reduce human intervention on performance configuration tuning. Another relevant work by [24] reports experience on adopting a defect prediction model in practice. Our work is different from the studies mentioned above. Our work does not focus on the methodology but the software process of integrating AI into non-AI infrastructure. Moreover, our work is an experience report in AI-based user input verification/testing, which was never studied in any prior researches. In particular, we focus on the notable challenges we faced while integrating AI-based user input verification into industrial software. From the challenges we faced during requirement soliciting and data wrangling, facilitating the integration of the AI-based input verification into non-AI infrastructure. Finally, we discuss the challenging nature of evaluating the user-centric AI model.

VII. Conclusion

In user-centric reporting software systems, there are thousands of records added daily by diverse clients. In these systems, checking for the quality of user input is of utmost importance as users rely on these reporting systems for mission-critical tasks. Unfortunately, manual verification is costly and almost impossible as there are tremendous user inputs. Hence, harnessing AI in automatic user verification becomes immensely helpful. Despite the promise and perils of AI/ML, it is challenging to add its capabilities to the existing software from an end-user adoption and navigation perspective. Users may not know how to navigate the AI capabilities as the new framework poses additional challenges. Moreover, since there may be numerous end-users with diverse user behavior, it also poses challenges for developers of AI frameworks to consider a common solution for various stakeholders. This paper documents our experience of successfully adding AI into large-scale user-centric reporting software. In particular, we focus on three stages of challenges when integrating the AI component into a non-AI system: 1) Challenges in collaborative requirement engineering, involving many development teams of various backgrounds, 2) Challenges in AI adoption in user-centric reporting systems with stakeholders with diverse operational behavior and 3) Challenge in AI verification, in user-centric reporting systems with stakeholders with diverse patterns of operational behavior and company-dependent anomalies. Our experience can help software practitioners who also want to integrate AI capabilities into existing non-AI software.

VIII. Acknowledgement

We want to thank ERA Environmental Management Solutions (ERA) for providing access to the enterprise systems that we used in our case study. The findings and opinions expressed in this paper are those of the authors and do not necessarily represent or reflect those of ERA and/or its subsidiaries and affiliation. Our results do not in any way reflect the quality of ERA's products.

References

[1] M. Chui, "Ai adoption advances, but foundational barriers remain."

[2] W. E. L. Grimson, M. E. Leventon, O. D. Faugeras, W. Wells, M. Mirmehdi, and B. Thomas, "Computer vision methods for image guided surgery.," in *BMVC*, pp. 1–12, 2000.

[3] P. T. Inc., "Collaborative data science," 2015.

[4] J. S. Grosman, P. H. Furtado, A. M. Rodrigues, G. G. Schardong, S. D. Barbosa, and H. C. Lopes, "Eras: Improving the quality control in the annotation process for natural language processing tasks," *Information Systems*, vol. 93, p. 101553, 2020.

[5] S. Park, A. Y. Wang, B. Kawas, Q. V. Liao, D. Piorkowski, and M. Danilevsky, "Facilitating knowledge sharing from domain experts to data scientists for building nlp models," in *26th International Conference on Intelligent User Interfaces*, pp. 585–596, 2021.

[6] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[7] S. I. Goldberg, A. Niemierko, and A. Turchin, "Analysis of data errors in clinical research databases," in *AMIA annual symposium proceedings*, vol. 2008, p. 242, American Medical Informatics Association, 2008.

[8] F. Duchene, R. Groz, S. Rawat, and J.-L. Richier, "Xss vulnerability detection using model inference assisted evolutionary fuzzing," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pp. 815–817, IEEE, 2012.

[9] S. G. Powell, K. R. Baker, and B. Lawson, "Errors in operational spreadsheets," *Journal of Organizational and End User Computing (JOEUC)*, vol. 21, no. 3, pp. 24–36, 2009.

[10] R. R. Panko, "What we know about spreadsheet errors," *Journal of Organizational and End User Computing (JOEUC)*, vol. 10, no. 2, pp. 15–21, 1998.

[11] M. Bishop, J. Cummins, S. Peisert, A. Singh, B. Bhumiratana, D. Agarwal, D. Frincke, and M. Hogarth, "Relationships and data sanitization: A study in scarlet," in *Proceedings of the 2010 New Security Paradigms Workshop*, pp. 151–164, 2010.

[12] J. C.-W. Lin, J. M.-T. Wu, P. Fournier-Viger, Y. Djenouri, C.-H. Chen, and Y. Zhang, "A sanitization approach to secure shared data in an iot environment," *IEEE Access*, vol. 7, pp. 25359–25368, 2019.

[13] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, *et al.*, "Recipes for building an open-domain chatbot," *arXiv preprint arXiv:2004.13637*, 2020.

[14] Yelp, "How we use deep learning to classify business photos at yelp," 2015.

[15] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, "Ernie 2.0: A continual pre-training framework for language understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 8968–8975, 2020.

[16] Google, "Recent advances in google translate," 2020.

[17] R. DeLine, "Research opportunities for the big data era of software engineering," in *2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering*, pp. 26–29, IEEE, 2015.

[18] S. Sharma and S. Pandey, "Integrating ai techniques in requirements elicitation," in *Proceedings of International Conference on Advancements in Computing & Management (ICACM)*, 2019.

[19] D. Fisher, R. DeLine, M. Czerwinski, and S. Drucker, "Interactions with big data analytics," *interactions*, vol. 19, no. 3, pp. 50–59, 2012.

[20] J. McCardle, "The challenge of integrating ai & smart technology in design education," *International Journal of Technology and Design Education*, vol. 12, no. 1, pp. 59–76, 2002.

[21] P. Kearney *et al.*, "Integrating ai planning techniques with workflow management system," *Knowledge-Based Systems*, vol. 15, no. 5-6, pp. 285–291, 2002.

[22] O. S. Pianykh, G. Langs, M. Dewey, D. R. Enzmann, C. J. Herold, S. O. Schoenberg, and J. A. Brink, "Continuous learning ai in radiology: implementation principles and early applications," *Radiology*, vol. 297, no. 1, pp. 6–14, 2020.

[23] H. Li, T.-H. Chen, A. E. Hassan, M. Nasser, and P. Flora, "Adopting autonomic computing capabilities in existing large-scale systems," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pp. 1–10, IEEE, 2018.

[24] C. Tantithamthavorn and A. E. Hassan, "An experience report on defect modelling in practice: Pitfalls and challenges," in *Proceedings of the 40th International conference on software engineering: Software engineering in practice*, pp. 286–295, 2018.