https://doi.org/10.26160/2474-5901-2026-53-170-175

ИНТЕГРАЦИЯ ГЕНЕРАТИВНОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ИНЖЕНЕРНОЕ ОБРАЗОВАНИЕ

Хабаров С.П.
Санкт-Петербургский государственный лесотехнический университет имени С.М. Кирова, Санкт-Петербург, Россия

Ключевые слова: генеративный искусственный интеллект, промпт-инжиниринг инженерное образование, RAG-архитектура, оценка компетенций, цифровые двойники.
Аннотация. В статье обсуждается вопрос использования генеративного искусственного интеллекта в инженерном образовании с учётом цифровизации промышленности. Предложена методика, основанная на промпт-инжиниринге и RAG-архитектуре, для подготовки специалистов по цифровым двойникам и промышленным системам. Главное в ней – сместить оценку с готового кода на процесс принятия инженерных решений. Это позволяет сохранить качество подготовки и объективность контроля, когда студенты активно пользуются искусственным интеллектом.

INTEGRATION OF GENERATIVE ARTIFICIAL INTELLIGENCE INTO ENGINEERING EDUCATION

Khabarov S.P.
Saint-Petersburg State Forest Technical University, Saint-Petersburg, Russia

Keywords: generative artificial intelligence, prompt engineering, engineering education, RAG architecture, competency assessment, digital twins.
Abstract. This article examines the use of generative artificial intelligence in engineering education in the context of industrial digitalization. A methodology based on prompt engineering and RAG architecture is proposed for training specialists in digital twins and industrial systems. The core idea is to shift assessment from the final code to the engineering decision-making process. This preserves the quality of training and the validity of assessment when students actively use artificial intelligence.

Введение

Машиностроительные предприятия сегодня активно меняются. В цехах появляются станки с ЧПУ, промышленные роботы, системы сбора данных с датчиков вибрации, температуры, нагрузки. Инженер теперь не только читает чертежи, но и настраивает протоколы обмена (Modbus, OPC UA), пишет скрипты для визуализации, разбирается в телеметрии. Грубо говоря, современный инженер должен быть наполовину программистом.

Генеративные нейросети (ChatGPT и др.) стали доступны массовому пользователю. Это даёт очевидное преимущество: студент может быстро получить работающий код для типовых инженерных задач. Например, для панели мониторинга или опроса датчиков. Однако эта простота создаёт скрытую проблему. Выполнив задание с использованием искусственного интеллекта (ИИ), студент часто минует этап глубокого понимания: принципов обмена данными, допустимых задержек, поведения системы при сбоях. В

машиностроении это особенно опасно. Ошибка в логике управления может привести к поломке инструмента, браку деталей или даже к травме.

Традиционная система контроля с проверкой финального кода или отчёта, в новых условиях работает плохо. Нейросеть генерирует внешне правильное решение, но это не доказывает, что студент понимает, почему оно устроено именно так. Как отмечают Лю и др. в своём обзоре методов промпт-инжиниринга [1], качество работы с ИИ напрямую зависит от того, насколько полно человек формулирует задачу, контекст и ограничения. Если этого не делать, ИИ выдаёт ответ, но осмысленности за ним не стоит.

Поэтому нужен другой подход. Оценивать надо не результат, а процесс: как студент ставит задачу ИИ, какие уточнения делает, проверяет ли ответы по документации, замечает ли ошибки модели. Цель статьи – предложить такую методику для инженерных специальностей (с упором на машиностроение), которая сохраняет пользу от ИИ как инструмента, но не позволяет подменять им инженерное мышление.

1. Промпт-инжиниринг в инженерном образовании

На практике качество работы с ИИ напрямую зависит от того, насколько конкретно студент описал задачу, какие детали и ограничения указал. Наиболее полезными на практике оказались три подхода: Role Prompting (модель ИИ работает в заданной роли), Few-Shot Prompting (обучение на примерах реальных данных) и Iterative Refinement (пошаговое уточнение, как при наладке оборудования).

Техника Role Prompting [2] состоит в том, что модели назначается конкретная профессиональная роль. Это заставляет модель использовать правильную терминологию и учитывать отраслевой контекст (табл.1).

Табл. 1. Профессиональные роли ИИ в инженерной подготовке

Роль	Контекст использования	Целевая компетенция
Инженер-наладчик ЧПУ	Написать макрос для обмера детали после обработки	Понимание G-кодов, логики обхода датчиков
Специалист по SCADA	Спроектировать экран диспетчера для линии сборки	Учёт критических параметров, приоритетов тревог
Диагност по вибрации	Предложить алгоритм обнаружения биения шпинделя	Знание пороговых значений, спектрального анализа
Нормоконтролёр	Проверить фрагмент кода на соответствие ГОСТ	Умение работать с нормативной документацией

В качестве иллюстрации рассмотрим пример из практики. В теме «Разработка панели оператора для фрезерного станка с ЧПУ» студенты

инициируют диалог с моделью ИИ, которому присвоена роль «Инженер АСУ ТП». Промпт формулируется примерно так:

«Ты – инженер АСУ ТП на механообрабатывающем участке. У меня есть датчики вибрации шпинделя и температуры подшипников. Нужно сделать веб-интерфейс для мастера. Особенности: связь по радиоканалу с задержками до 1 секунды, возможны обрывы. Как организовать отображение и архивацию данных?»

Студент получает развёрнутый ответ, но затем должен критически его разобрать: подходят ли предложенные протоколы для реального станка, как часто опрашивать датчики, что делать при пропадании связи. Главное здесь – сместить внимание с синтаксиса кода на инженерную надёжность.

При использовании техники Few-Shot Prompting [3], студент дает модели несколько реальных примеров данных. Например, JSON с показаниями контроллера или фрагмент лога ЧПУ. Посмотрев на эти образцы, модель сама «понимает» нужный формат и дальше обрабатывает похожие данные по тому же принципу. Это помогает делать проект аккуратным и сразу подгонять его под принятые на предприятии стандарты. В машиностроении такой подход удобен, когда нужно написать скрипты для обработки однотипных данных с нескольких станков, не объяснять же каждый раз с нуля.

Техника Iterative Refinement [4] лучше всего показывает, умеет ли студент не просто задать вопрос, а довести решение до ума, как при реальной наладке оборудования. Смысл простой. Студент не принимает первый ответ ИИ за окончательный, а проводит цикл: запрос → анализ ответа → поиск слабых мест → уточняющий запрос. И так несколько раз, пока решение не станет рабочим и безопасным. На рисунке 1 показана схема такого взаимодействия.

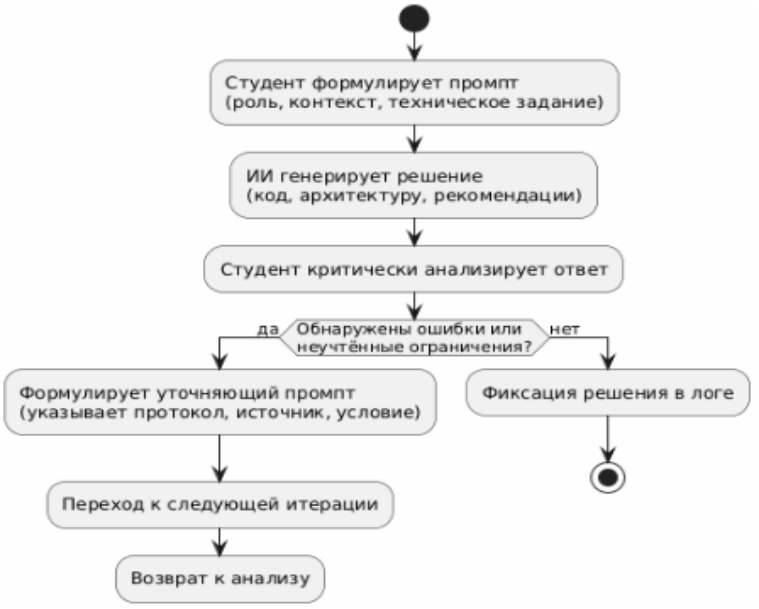

Рис. 1. Итеративный промптинг при решении инженерной задачи

Рассмотрим, как это выглядит на конкретном примере. Допустим, что студент должен разработать модуль визуализации для линии сборки.

Итерация 1. Студент просит ИИ сгенерировать базовую структуру панели отображения данных с датчиков. Получает вариант – вроде работает, но ничего не сказано про связь.

Итерация 2. Студент уточняет: «Что будет, если данные передаются по узкому радиоканалу? Какие проблемы с производительностью?» ИИ выдаёт предупреждения о задержках и предлагает буферизацию.

Итерация 3. Студент просит реализовать оптимизацию: обработку обрывов связи, кэширование, сжатие данных. ИИ генерирует уже более надёжное решение.

На каждом шаге студент вынужден вскрывать неявные допущения: какую пропускную способность ИИ «предположил», какой тип датчика, присутствует ли резервирование. Такой режим превращает модель ИИ из «генератора ответов» в «оппонента-консультанта». Это и есть главное отличие предлагаемого подхода от бездумного использования ИИ.

2. Типичные ошибки и система оценки

Наблюдая за тем, как студенты на самом деле работают с ИИ, можно выделить несколько типичных ошибок. В таблице 2 они перечислены вместе с вариантами их исправления.

Табл. 2. Типичные ошибки промптинга и методы коррекции

Тип ошибки	Пример (неправильно)	Коррекция (правильно)
Не указан протокол обмена	«Считай данные с датчика вибрации»	«Считай данные по протоколу Modbus RTU, адрес 0x64»
Отсутствие ролевой привязки	«Напиши интерфейс мониторинга»	«Ты – инженер АСУ ТП. Разработай SCADA-интерфейс для диспетчерской»
Игнорирование среды передачи	«Реализуй веб-сокеты для реального времени»	«Реализуй буферизацию для ограниченной пропускной способности канала»
Нет ссылок на источники	«Допустимое давление – 200 бар»	«Согласно тех. документации, давление 180±10 бар»

В предлагаемом подходе следует оценивать не готовый код, а сам диалог студента с моделью, используя следующие критерии:

– Полнота начального промпта (есть ли роль, контекст, технические ограничения) – от 0 до 2 баллов. Если просто написал «напиши код» – 0.

– Количество содержательных уточнений (итераций) – от 0 до 3 баллов. Важны не просто «спасибо», а реальные вопросы по существу.

– Качество аргументации (ссылки на документацию, ГОСТ, паспорта оборудования) – от 0 до 3 баллов.

– Наличие лога переписки с ИИ – 0 или 2 балла. Без него понять, кто что делал, невозможно.

Максимум – 10 баллов, ниже 6 баллов работа не засчитывается. Практика показывает, что студенты, набравшие меньше шести, обычно просто копировали первый ответ ИИ и не вникали в суть. Те, кто получает 8-10, как правило, могут объяснить, почему выбрали то или иное решение, и заметить ошибки модели.

3. RAG-архитектура для работы с технической документацией

Для снижения риска ошибок из-за «галлюцинаций» ИИ, при проектировании систем целесообразно использование архитектуры RAG (Retrieval-Augmented Generation) [4]. Подход подразумевает настройку ИИ-ассистента на работу с верифицированной базой знаний, включающей документацию производителей, ГОСТ и регламенты безопасности. RAG-подход не обязательно требует разработки сложной инфраструктуры. Для учебных целей целесообразно использование доступных Low-Code решений, куда преподаватель загружает верифицированные материалы. Студент получает ответы, которые привязаны к конкретным разделам документации оборудования, с обязательным указанием источника.

В рамках учебного модуля студентам предлагается сравнительное задание: получить решение через общедоступную модель и через модель с подключённой базой знаний технической документации (RAG). Он должен выявить расхождения и аргументировать выбор решения. Данный подход формирует критически важный навык верификации технической информации и работы с первоисточниками.

4. Этические и нормативные аспекты применения ИИ

Интеграция ИИ в инженерное образование требует нормативного регулирования, особенно в контексте ответственности за безопасность технических систем. Рекомендуется включать в рабочую программу требование обязательного декларирования использования ИИ: студент указывает, какие части кода сгенерированы, а какие разработаны вручную. Для обеспечения прозрачности в курсовом проектировании вводится требование к структуре отчета: студент предоставляет файл с историей промптов, ссылками на коммиты и описанием ручной модификации кода. Несоблюдение этого требования должно рассматриваться как нарушение академической этики.

Необходимо также учитывать риски конфиденциальности: студентам запрещается загрузка реальных данных или схем управления в публичные нейросети. Этический компонент обучения должен включать обсуждение ответственности за последствия внедрения автоматизированных решений в системы управления, где ошибка может привести к аварийным ситуациям.

Заключение

Интеграция генеративного ИИ в инженерное образование требует пересмотра подходов к оцениванию. Предложенная методика, основанная на

промпт-инжиниринге и RAG-архитектуре, позволяет сместить фокус контроля с финального кода на процесс принятия инженерных решений. Задача преподавателя — не запрещать ИИ, а учить работать с ним как с инструментом. Иначе студент будет получать готовые ответы, не понимая их. Преподаватель должен организовать работу, где ИИ выступает инструментом развития, а не подменой мышления. Предложенная методика универсальна и применима в дисциплинах по автоматизации и моделированию. Дальнейшие исследования надо направить на разработку метрик качества инженерных промптов и оценку влияния ИИ-ассистентов на формирование профессиональной ответственности инженера.

Список литературы / References

1. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing // ACM Computing Surveys. 2023, vol. 55, no. 9, pp. 1-35. DOI: 10.1145/3560815.
2. Brown T., Mann B., Ryder N. et al. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems (NeurIPS). 2020, vol. 33, pp. 1877-1901. DOI: 10.48550/arXiv.2005.14165.
3. Madaan A., Tandon N., Gupta P. и др. Self-Refine: Iterative Refinement with Self-Feedback // Advances in Neural Information Processing Systems (NeurIPS). 2023, vol. 36. DOI: 10.48550/arXiv.2302.07918.
4. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems (NeurIPS). 2020, vol. 33, pp. 9459-9474. DOI: 10.48550/arXiv.2005.11401.

Хабаров Сергей Петрович – кандидат технических наук, старший научный сотрудник, доцент, доцент кафедры информационных систем и технологий	**Khabarov Sergei Petrovich**– candidate of technical sciences, senior researcher, associate professor, associate professor of the Department of information systems and technologies
serg.Habarov@mail.ru	

Received 17.04.2026