УДК 658.5
DOI 10.17513/snt.40423

МЕТОД ПРОЕКТИРОВАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ПРОЦЕССОМ ПАЛЕТИЗАЦИИ КОРОБОК НА ОСНОВЕ КОНЕЧНОГО АВТОМАТА И РЕЛЯЦИОННОЙ МОДЕЛИ ДАННЫХ

[1]Холопов В.А., [1]Клягин М.М., [1]Огорельцев Р.М., [2]Мишина В.А.

[1]ФГБОУ «МИРЭА – Российский технологический университет», Москва, e-mail: mmklyagin@gmail.com;
[2]ООО «ВЕКАС», Москва

В статье предложен метод представления и исполнения логики системы управления технологическим процессом палетизации коробок в виде конечного автомата, реализованного средствами реляционной модели данных и языка SQL. Целью исследования является разработка инженерного подхода, обеспечивающего формализованное, масштабируемое и конфигурируемое описание дискретной логики управления, пригодное для интеграции с цифровыми производственными платформами. Каждое состояние автомата реализуется в виде таблицы базы данных с управляющими сигналами и флагом активности, а переходы задаются через условия и исполняются с помощью SQL-триггеров. Такая архитектура позволяет управлять логикой процесса непосредственно в структуре данных, без привязки к программному коду контроллера, обеспечивая прозрачность, адаптируемость и возможность верификации поведения системы. Предложенный подход успешно описывает как штатные, так и аварийные сценарии палетизации, включая взаимодействие роботов, конвейера и системы машинного зрения. Реализация была протестирована на производственном стенде и подтвердила устойчивость и гибкость модели. Сравнение с традиционными средствами проектирования логики показало, что применение реляционной модели и SQL-автомата упрощает модификацию, поддержку и интеграцию системы управления в цифровую инфраструктуру предприятия.

Ключевые слова: система управления, конечный автомат, реляционная модель данных, палетизация

METHOD FOR DESIGNING A CONTROL SYSTEM FOR THE BOX PALLETIZING PROCESS BASED ON FINITE STATE MACHINE AND RELATIONAL DATA MODEL

[1]Kholopov V.A., [1]Klyagin M.M., [1]Ogorelcev R.M., [2]Mishina V.A.

MIREA – Russian Technological University, Moscow, e-mail: mmklyagin@gmail.com;
VEKAS LLC, Moscow

The paper proposes a method for representing and executing the control logic of a box palletizing process in the form of a finite state machine implemented using a relational data model and the SQL language. The aim of the study is to develop an engineering approach that enables a formalized, scalable, and configurable description of discrete control logic suitable for integration with digital manufacturing platforms. Each state of the finite state machine is implemented as a database table containing control signals and an activity flag, while transitions are defined by conditions and executed using SQL triggers. This architecture allows process logic to be managed directly within the data structure, without being tied to controller code, thus ensuring transparency, adaptability, and verifiability of system behavior. The proposed approach successfully describes both normal and fault scenarios in the palletizing process, including the interaction of robots, conveyor systems, and machine vision. The implementation was tested on a production stand and confirmed the model's robustness and flexibility. A comparison with traditional logic design tools demonstrated that the use of a relational model and SQL-based automaton simplifies modification, maintenance, and integration of the control system into the enterprise's digital infrastructure.

Keywords: control system, finite state machine, automation, relational data model, palletizing

Введение

Палетизация коробок является типичным дискретным процессом, широко применяемым в логистике и упаковочном производстве. Этот процесс требует высокой точности, устойчивости к ошибкам и способности к конфигурации под различные шаблоны укладки [1]. Автоматизация палетизации обеспечивает повышение производительности, снижение трудозатрат и снижение количества ошибок при укладке [2]. Современные промышленные системы палетизации, как правило, реализуются с использованием робототехнических комплексов, управляемых программируемыми логическими контроллерами, а логика операций описывается при помощи языков стандарта МЭК 61131-3 – таких как Ladder Diagram, Sequential Function Chart и Structured Text. Несмотря на надёжность этих технологий, они имеют ряд ограничений, особенно на этапах проектирования и сопровождения систем управления.

Ключевым недостатком традиционных решений является жёсткая привязка управляющей логики к среде программирования и конкретному оборудованию. В подавляющем большинстве случаев поведение системы формализуется в виде процедурного кода, что затрудняет визуализацию, верификацию и адаптацию логики при изменении производственных условий. Кроме того, подобная архитектура затрудняет интеграцию с внешними цифровыми системами и плохо масштабируется при добавлении новых состояний или вариантов исполнения.

Различные авторы, как отечественные, так и зарубежные, предпринимали попытки формализовать логику палетизации через модель конечного автомата. Так, в работе X. Chen и его коллег описывается проектирование системы управления палетайзером с помощью программного комплекса RobotStudio. В статье показано, как графические интерфейсы и моделирование могут быть использованы для настройки логики укладки, однако реализация конечного автомата остаётся жёстко связанной с программным обеспечением конкретной платформы [3].

Более универсальные модели, основанные на конечных автоматах, рассматриваются в работах Fragapane G. и соавторов в контексте робототехники, а также в исследованиях Канахина В.С. и Гудинова В.Н. для пищевой промышленности [4; 5]. Однако реализация логики в этих случаях тоже не выходит за пределы кода или структурных схем, что ограничивает гибкость проектирования.

Исследования о встраивании логики конечного автомата непосредственно в реляционную базу данных предприняты в работе Рыбанова А.А. и других. Авторы демонстрируют, как бизнес-процесс «управление заказами» может быть реализован как конечный автомат внутри PostgreSQ с использованием SQL-функций, агрегатов и триггеров. Логика переходов реализована в виде SQL-функции с множеством ветвлений, агрегатная функция вычисляет финальное состояние по истории событий, а триггер гарантирует, что последовательность действий не нарушает бизнес-правила. Однако реализация остаётся жёстко закодированной внутри функций, что ограничивает возможность масштабирования и перенастройки. Все состояния и переходы фиксированы в коде, а не в данных, что снижает адаптивность и препятствует многократному использованию архитектуры для других процессов [6].

Одновременно с этим развивается направление цифрового производства, где всё большую роль играют цифровые двойники и реляционные модели данных. В таких системах требуется формализованное, адаптируемое и проверяемое представление логики процессов. В работах Mikkelsen H. и соавторов, а также Panushev I. и других показан потенциал SQL и реляционных моделей в управлении цифровыми копиями производственных объектов, однако использование SQL для непосредственного проектирования логики управления в них не рассматривается [7; 8].

На международном уровне тенденция к унификации инженерных данных реализована в стандарте AutomationML, который обеспечивает обмен структурированной информацией между CAD/CAE-системами, программными средствами проектирования программируемого логического контроллера (ПЛК), моделирования и планирования производственных процессов. AutomationML представляет поведение систем, включая конечные автоматы, с использованием формализованных XML-структур: Computer Aided Engineering Exchange (CAEX) для топологии, PLCOpen XML для логики и COLLADA для кинематики. Несмотря на то, что предложенный в настоящем исследовании метод не реализован в формате AutomationML, он во многом соответствует его идеологии. Реализация логики в виде таблиц состояний и переходов, отделение логики от программного кода, поддержка масштабирования и возможность повторного использования – все эти характеристики создают предпосылки для потенциальной совместимости с AutomationML и интеграции с другими средствами цифрового проектирования [9].

На фоне вышеописанных ограничений особенно актуальным становится подход, в котором логика управления технологическим процессом описывается через структуру конечного автомата, реализованную средствами реляционной базы данных и языка SQL. Такой подход был предложен авторами ранее [10] и показал свою эффективность при моделировании процессов сборки. В предлагаемой архитектуре каждое состояние и переход формализуются в виде записей в таблицах базы данных, а логика исполнения задаётся через SQL-триггеры, что обеспечивает реализацию конечного автомата непосредственно в системе управления базами данных (СУБД) без необходимости внешнего программного кода.

С позиции проектирования систем управления, предложенный подход даёт ряд значительных преимуществ. Он обеспечивает формализацию логики на этапе проектирования – до её реализации – что соответ-

ствует современным требованиям к управлению жизненным циклом автоматизированных систем. Представление логики в виде таблиц состояний и переходов позволяет наглядно описывать поведение системы, упрощая верификацию проекта. Благодаря единому хранилищу логики в базе данных исключается несоответствие между проектной документацией и реализацией. Изменение логики осуществляется путём модификации данных, а не кода, что позволяет легко адаптировать систему под новые требования без перепрограммирования. Более того, такая архитектура создаёт условия для масштабируемости и повторного использования проектных решений: модели на основе конечного автомата могут быть параметризованы, сохранены как шаблоны и перенесены на другие технологические процессы.

Таким образом, предложенный метод сочетает преимущества формальной модели конечного автомата, гибкости SQL и наглядности реляционной структуры. Он оптимизирует процесс проектирования систем управления, делает его более открытым, адаптируемым и интеграционно ориентированным для современных цифровых производств. Настоящая работа посвящена применению этого метода к задаче управления технологическим процессом палетизации коробок. Предлагаемый метод позволяет формализовать процесс палетизации как конечный автомат, описанный средствами SQL и реляционной модели.

Целью исследования является разработка метода проектирования системы управления технологическим процессом палетизации коробок на основе модели конечного автомата, реализованной средствами реляционной базы данных и языка SQL.

Исследование направлено на решение следующих задач:

1. Формализовать процесс палетизации как дискретную систему, состоящую из конечного набора состояний и условий переходов между ними.

2. Разработать реляционную структуру данных, позволяющую описывать состояния и переходы в терминах таблиц базы данных.

3. Реализовать управляющую логику на уровне СУБД, обеспечив автоматическое выполнение переходов посредством триггеров.

4. Обеспечить возможность модификации и расширения логики без необходимости внесения изменений в программный код.

5. Оценить применимость предложенного подхода к проектированию систем управления дискретными производственными процессами в контексте гибких цифровых производств и совместимости с международными стандартами, включая AutomationML.

Материалы и методы исследования

Объектом исследования является дискретный технологический процесс палетизации коробок на роботизированной установке. Управление осуществляется с использованием роботизированных манипуляторов, конвейерной линии и системы машинного зрения, позволяющей проводить контроль качества продукции до её укладки. Модель технологического процесса представлена на рисунке 1, где отображены основные элементы установки: зоны укладки, механизм подачи и камера системы машинного зрения, расположенная в центральной части конвейерной линии. В процессе работы коробки поочерёдно подаются на конвейер, проходят через зону сканирования, проверяются системой машинного зрения и в зависимости от результата либо укладываются на палету, либо удаляются как брак. Каждый этап представляет собой логически завершённую операцию, связанную с конкретным управляющим решением.

Для формализации логики управления применяется модель детерминированного конечного автомата [11]:

$$A = (Q, \Sigma, \delta, q_0, F),$$

где Q – множество состояний (например, ожидание старта цикла, укладка);

Σ – множество входных условий и событий (например, результат работы системы машинного зрения, сигналы датчиков);

δ – функция переходов;

q_0 – начальное состояние;

$F \subseteq Q$ – допускающие состояния автомата.

Конечный автомат для системы управления палетизацией коробок описывается следующим образом:

$$A = \left\{q_0, q_1, q_{\_err1,}, q_2, q_{\_err2,}, q_3, q_{\_err3,}, q_4, q_{\_err4,}\ q_5, q_{\_err5,}\ q_6, q_{\_err6,}\right\}, \delta, q_0, \{q_0\}$$

Множество входных условий и событий будет отражено на графе конечного автомата.

Множество состояний включает рабочие и аварийные этапы, множество входных символов соответствует сигналам от датчиков и результатам обработки коробки, функция переходов описывает условия перехода между состояниями, начальным состоянием автомата является ожидание запуска, а допускающими считаются состояния, при которых система может вернуться к началу нового цикла.

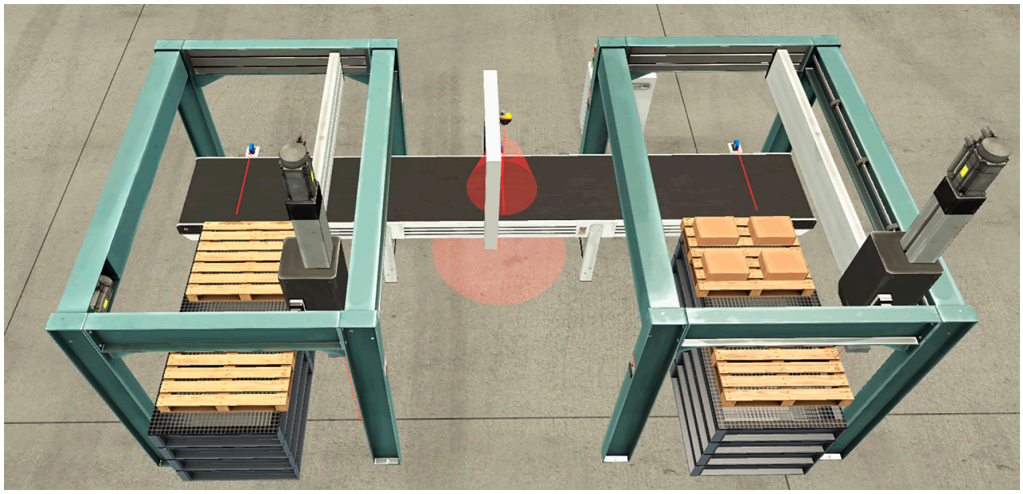

Рис. 1. Модель технологического процесса палетизации коробок
Источник: составлено авторами

Далее представлено описание всех состояний конечного автомата:

1. Q0 – ожидание старта цикла.
2. Q1 – укладка коробки с палеты роботом № 1 в зону выгрузки конвейера.
3. Q2 – доставка коробки в зону сканирования.
4. Q3 – сканирование коробки системой машинного зрения.
5. Q4 – доставка коробки в зону укладки.
6. Q5 – укладка корректной коробки роботом № 2 на палету.
7. Q6 – удаление бракованной коробки роботом № 2.
8. Q_err1 – ошибка, связанная с укладкой коробки (коробка не оказалась на конвейере или ошибка работы робота).
9. Q_err2 – застревание или исчезновение коробки с конвейера.
10. Q_err3 – ошибки, связанные с работой системы машинного зрения (не распознало коробку за отведенное время или проблемы с камерой).
11. Q_err4 – застревание или исчезновение коробки с конвейера.
12. Q_err5 – ошибка, связанная с укладкой коробки (коробка не оказалась на конвейере или ошибка работы робота).
13. Q_err6 – ошибка, связанная с укладкой коробки (коробка не оказалась на конвейере или ошибка работы робота).

В предложенной архитектуре каждое состояние автомата реализовано в виде отдельной таблицы базы данных, содержащей управляющие сигналы, которые система должна активировать при входе в состояние, флаг активности, определяющий текущее состояние автомата, и метку времени. Каждое состояние описывается строго отдельно, что упрощает сопровождение, визуализацию и масштабирование модели.

Результаты исследования и их обсуждение

Исходной логикой проектирования послужила блок-схема технологического процесса палетизации (рис. 2), в которой визуально представлены ключевые стадии обработки коробки: от подачи и сканирования до принятия решения и выполнения укладки или удаления. Эта схема легла в основу последующей формализации логики автомата и трансформации в реляционную структуру базы данных.

Переходы между состояниями оформляются как таблицы, где фиксируются условия, при выполнении которых автомат переходит из одного состояния в другое. Эти условия задаются в виде булевых и числовых параметров, таких как результат анализа системы машинного зрения или наличие коробки в определённой зоне. Название таблицы перехода (например, transition_Q3_Q4) определяет направление перехода, что исключает необходимость хранения избыточной информации о начальном и конечном состоянии внутри таблицы. В момент, когда создаётся запись об активном состоянии (например, state_Q3), SQL-триггер проверяет наличие допустимого перехода с подходящими условиями. При их выполнении создаётся новая запись в целевом состоянии (например, state_Q4), активируя соответствующее управляющее воздействие и фиксируя момент перехода. Все ранее активные записи в исходном состоянии деактивируются, что обеспечивает уникальность текущего состояния автомата.

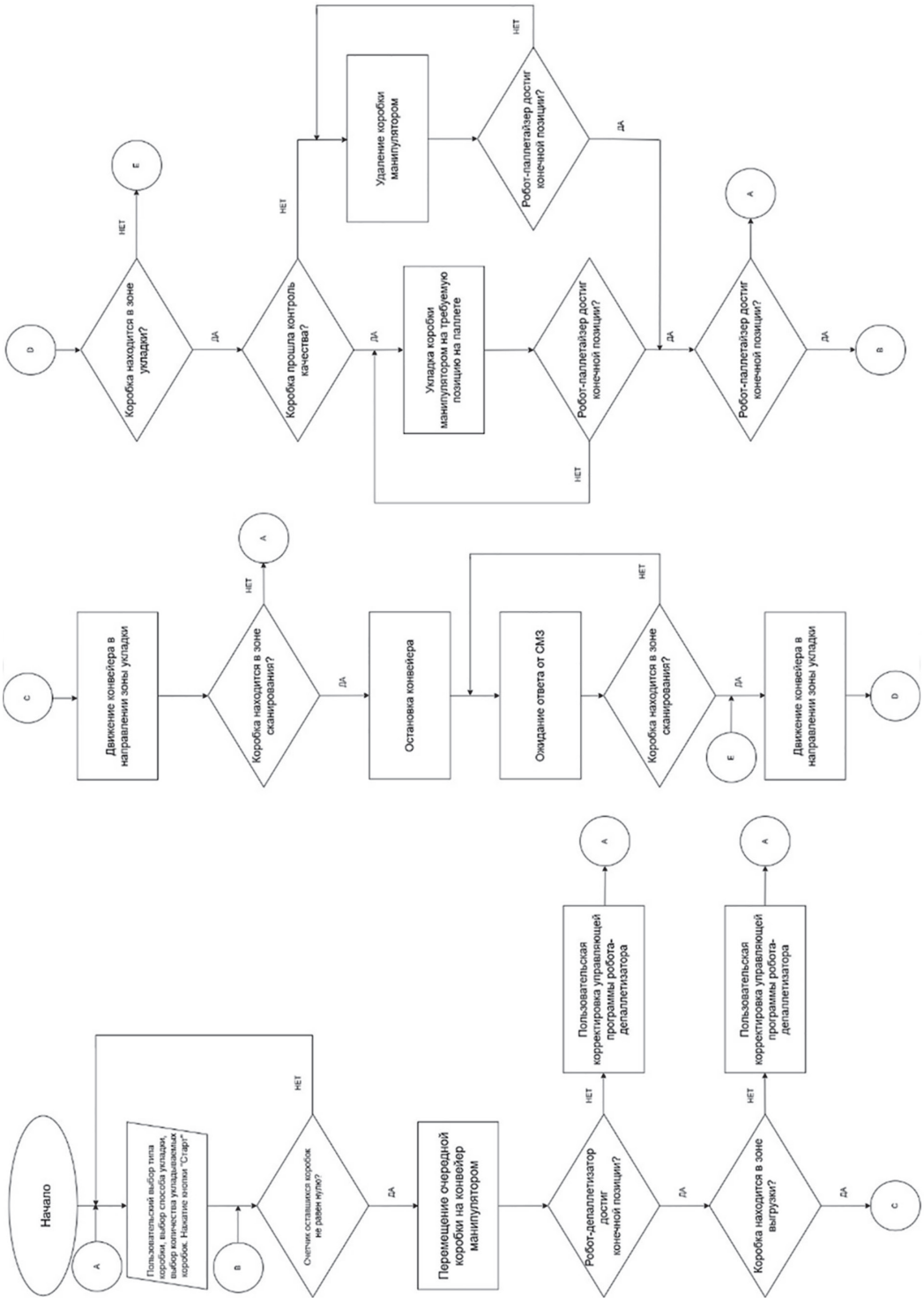

Рис. 2. Обобщенная блок-схема технологического процесса палетизации коробок
Источник: составлено авторами

Реализация переходов полностью основана на SQL-триггерах, которые работают на вставку новых записей в таблицы состояний и не изменяют саму строку, вызвавшую триггер. Это обеспечивает корректную работу в рамках ограничений системы управления базами данных MySQL. Деактивация предыдущих состояний реализована прямо в триггере за счёт команды UPDATE, не нарушающей ограничения языка SQL, поскольку воздействие осуществляется на другие строки таблицы. Таким образом, конечный автомат работает полностью автономно: при любом успешном переходе всегда существует одно активное состояние.

Непрерывно производится логирование изменений состояния: все таблицы содержат поле timestamp, отражающее момент входа в состояние, а таблицы переходов – поле last_triggered, регистрирующее момент активации перехода. Эти данные могут быть использованы для анализа поведения системы, отладки и визуализации истории исполнения.

Структура системы ориентирована на интеграцию с внешними цифровыми платформами. Благодаря тому, что вся логика описана в виде данных, цифровой двойник может считывать активные состояния, отслеживать выполнение автоматных переходов, реагировать на события или даже управлять автоматом через интерфейс к базе данных [12]. Такая совместимость соответствует идеологии цифрового производства и принципам стандарта AutomationML, обеспечивая возможность формализованного описания поведения системы и обмена данными между инженерными инструментами [13].

Предложенный метод сочетает строгость математической модели конечного автомата и гибкость реляционной модели данных. Он обеспечивает масштабируемость, повторное использование логики, верифицируемость, что делает его эффективным для задач цифрового инжиниринга и проектирования интеллектуальных систем управления.

На основании разработанной структуры управления был реализован конечный автомат, отображающий последовательность этапов процесса палетизации коробок. Общая логика работы системы представлена в виде графа состояний, приведённого на рисунке 3.

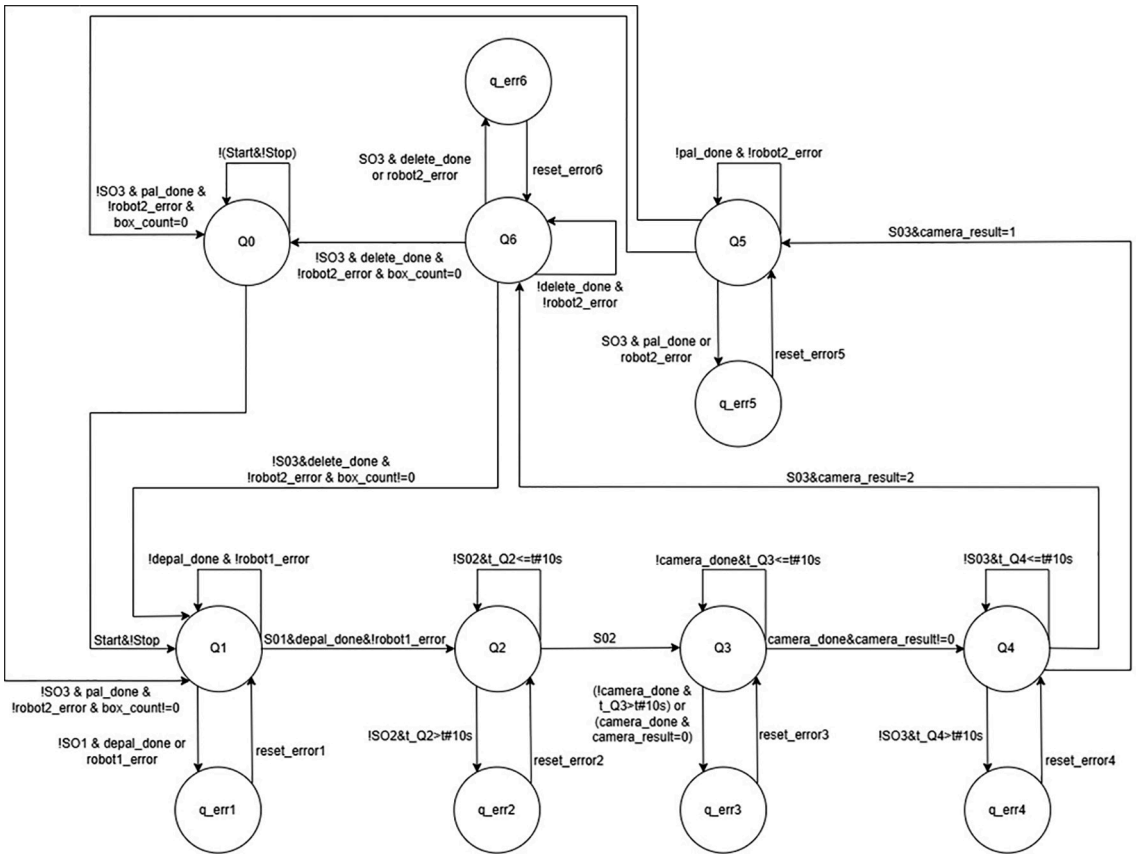

Рис. 3. Граф конечного автомата палетизации коробок
Источник: составлено авторами

В графе описаны как основные рабочие этапы (подача, сканирование, укладка), так и ветви аварийной обработки (ошибки датчиков, отказ системы машинного зрения, остановка робота). Такое представление позволяет наглядно визуализировать поведение системы и служит основой для построения табличной модели автомата в реляционной базе данных [14].Основное внимание уделяется созданию формальной, наглядной и масштабируемой структуры логики управления, представленной в виде таблиц состояний и переходов, с автоматическим исполнением переходов при помощи встроенных SQL-триггеров.

Результатом работы является универсальный подход, позволяющий проектировать и реализовывать системы дискретного управления с высокой степенью адаптивности, повторного использования и интеграции с внешними ИТ-средами. На примере процесса палетизации демонстрируется то, как описанная модель может быть практически применена для создания управляемой структуры с чётко определёнными логическими переходами и прозрачной архитектурой.

Описание переходов состояний:

1. $Q_0 \rightarrow Q_0$ – не поступило команды на старт цикла (Start) при неактивной кнопке остановки (!Stop).

2. $Q_0 \rightarrow Q_1$ – поступила команда на старт цикла (Start) при неактивной кнопке остановки (!Stop).

3. $Q_1 \rightarrow Q_1$ – не завершена укладка коробки на конвейер (!depal_done) при отсутствии ошибок по роботу № 1 (!robot1_error).

4. $Q_1 \rightarrow Q_{err1}$ – поступил сигнал о завершении выкладки коробки на конвейер (depal_done) при отсутствии сигнала от датчика наличия коробки (!SO1) или ошибка по роботу № 1 (robot1_error).

5. $Q_{err1} \rightarrow Q_1$ – возврат из ошибки сигналом от пользователя об устранении причин ошибки (reset_error1).

6. $Q_1 \rightarrow Q_2$ – поступил сигнал о завершении укладки коробки на конвейер (depal_done) при наличии сигнала от датчика наличия коробки (SO1) без ошибок по роботу № 1 (!robot1_error).

7. $Q_2 \rightarrow Q_2$ – коробка не доехала по конвейеру до зоны сканирования машинным зрением (!SO2), время движения не более 10 секунд (t_Q2<=t#10s).

8. $Q_2 \rightarrow Q_{err2}$ – коробка не доехала по конвейеру до зоны сканирования машинным зрением (!SO2), время движения более 10 секунд (t_Q2>t#10s).

9. $Q_{err2} \rightarrow Q_2$ – возврат из ошибки сигналом от пользователя об устранении причин ошибки (reset_error2).

10. $Q_2 \rightarrow Q_3$ – коробка доехала по конвейеру до зоны сканирования машинным зрением (SO2).

11. $Q_3 \rightarrow Q_3$ – система машинного зрения не завершила обработку (!camera_done), время работы не более 10 секунд (t_Q3<=t#10s).

12. $Q_3 \rightarrow Q_{err3}$ – система машинного зрения не завершила обработку (!camera_done), время работы более 10 секунд (t_Q3>t#10s), или система машинного зрения завершила обработку (camera_done), но не распознала объект (camera_result=0).

13. $Q_{err3} \rightarrow Q_3$ – возврат из ошибки сигналом от пользователя об устранении причин ошибки (reset_error3).

14. $Q_3 \rightarrow Q_4$ – система машинного зрения завершила работу (camera_done), объект распознан (camera_result!=0).

15. $Q_4 \rightarrow Q_4$ – коробка не доехала по конвейеру до зоны укладки (!SO3), время движения не более 10 секунд (t_Q4<=t#10s).

16. $Q_4 \rightarrow Q_{err4}$ – коробка не доехала по конвейеру до зоны сканирования машинным зрением (!SO3), время движения более 10 секунд (t_Q4>t#10s).

17. $Q_{err4} \rightarrow Q_4$ – возврат из ошибки сигналом от пользователя об устранении причин ошибки (reset_error4).

18. $Q_4 \rightarrow Q_5$ – коробка доехала по конвейеру до зоны сканирования машинным зрением (SO3), и она пригодна для погрузки на палету (camera_result=1).

19. $Q_4 \rightarrow Q_6$ – коробка доехала по конвейеру до зоны сканирования машинным зрением (SO3), и она непригодна для погрузки на палету (camera_result=2).

20. $Q_5 \rightarrow Q_5$ – не завершена укладка коробки на палету (!pal_done) при отсутствии ошибок по роботу № 2 (!robot2_error).

21. $Q_5 \rightarrow Q_{err5}$ – поступил сигнал о завершении укладки коробки на палету (pal_done) при наличии сигнала от датчика наличия коробки (SO3) или ошибка по роботу № 2 (robot2_error).

22. $Q_{err5} \rightarrow Q_5$ – возврат из ошибки сигналом от пользователя об устранении причин ошибки (reset_error5).

23. $Q_5 \rightarrow Q_1$ – отсутствует коробка на конвейере (!SO3), укладка коробки на палету окончена (pal_done), отсутствуют ошибки по роботу № 2 (!robot2_error), на разгружаемой палете еще остались коробки (box_count!=0).

24. $Q_5 \rightarrow Q_0$ – отсутствует коробка на конвейере (!SO3), укладка коробки на палету окончена (pal_done), отсутствуют ошибки по роботу (!robot2_error), на разгружаемой палете не осталось коробок (box_count=0).

25. $Q_6 \rightarrow Q_6$ – не завершено удаление коробки (!pal_done) при отсутствии ошибок по роботу № 2 (!robot2_error).

26. $Q_6 \rightarrow Q_{err6}$ – поступил сигнал о завершении удаления коробки (delete_done) при наличии сигнала от датчика наличия коробки (SO3) или ошибка по роботу № 2 (robot2_error).

27. $Q_{err6} \rightarrow Q_6$ – возврат из ошибки сигналом от пользователя об устранении причин ошибки (reset_error6);

28. $Q_6 \rightarrow Q_1$ – отсутствует коробка на конвейере (!SO3), удаление коробки завершено (delete_done), отсутствуют ошибки по роботу № 2 (!robot2_error), на разгружаемой палете остались коробки (box_count!=0).

29. $Q_6 \rightarrow Q_0$ – отсутствует коробка на конвейере (!SO3), удаление коробки завершено (delete_done), отсутствуют ошибки по роботу № 2 (!robot2_error), на разгружаемой палете не осталось коробок (box_count=0).

Реализация автомата выполнена средствами СУБД MySQL, где каждое состояние представлено отдельной таблицей с флагом активности и управляющими параметрами. Переходы формализованы как отдельные таблицы с условиями и логикой срабатывания. Управление переходами осуществляется через SQL-триггеры, автоматически проверяющие соблюдение условий и переключающие активное состояние.

Ниже показана структура таблиц: пример для состояния Q3 (сканирование коробки), состояния Q4 (транспортировка в зону укладки) и перехода между ними (листинг 1).

Листинг 1. Создание таблиц состояний и переходов.

```
-- Состояние Q3
CREATE TABLE state_Q3 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    camera_trigger BOOLEAN DEFAULT TRUE,
    camera_done BOOLEAN,
    camera_result INT,
    active BOOLEAN DEFAULT FALSE,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
-- Состояние Q4
CREATE TABLE state_Q4 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    conveyor_enable BOOLEAN DEFAULT TRUE,
    active BOOLEAN DEFAULT FALSE,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
-- Переход Q3 → Q4
CREATE TABLE transition_Q3_Q4 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    camera_done BOOLEAN NOT NULL,
    camera_result INT NOT NULL,
    last_triggered DATETIME
);
```

Все таблицы, описывающие состояния конечного автомата, имеют обязательные атрибуты:

1) Id – уникальный идентификатор записи;

2) Active – флаг, указывающий, является ли данная запись текущим активным состоянием автомата;

3) Timestamp – метка времени, фиксирующая момент входа в состояние. Используется для логирования и отладки.

Остальные атрибуты таблиц созданы в зависимости от целевого назначения состояния. Например, таблица state_Q3 имеет атрибут camera_trigger – булевый параметр, определяющий управляющее воздействие – активацию системы машинного зрения. По умолчанию установлен в TRUE, что означает, что при входе в это состояние должно быть инициировано сканирование. Таблица state_Q4 содержит атрибут conveyor_enable – булевый флаг, определяющий управляющее воздействие – включение ленты конвейера.

Таблицы переходов состоят из параметров, реализующих условия на графе конечного автомата.

Автоматизированный переход конечного автомата реализуется через триггер (листинг 2). Триггер trg_transition_Q3_to_Q4 срабатывает каждый раз при вставке новой записи в таблицу state_Q3, то есть при активации состояния сканирования коробки. Он извлекает входные параметры (camera_done, camera_result) и сравнивает их с условиями, заданными в таблице переходов transition_Q3_Q4. Если условия совпадают (например, камера завершила ана-

лиз, а результат обработки – положительный) и новая запись активна (NEW.active = TRUE), происходит следующее:

1) деактивация всех записей в state_Q3, чтобы гарантировать единственность активного состояния;

2) добавление новой активной записи в state_Q4, сигнализирующей о запуске конвейера (через conveyor_enable = TRUE);

3) фиксация времени срабатывания перехода в таблице transition_Q3_Q4 с помощью обновления поля last_triggered.

Листинг 2. Триггер для автоматического перехода состояний.

```
-- Триггер: автоматическое выполнение перехода
DELIMITER $$
CREATE TRIGGER trg_transition_Q3_to_Q4
AFTER INSERT ON state_Q3
FOR EACH ROW
BEGIN
  DECLARE match_count INT;
  SELECT COUNT(*) INTO match_count
  FROM transition_Q3_Q4
  WHERE camera_done = NEW.camera_done
    AND camera_result = NEW.camera_result;
  IF NEW.active = TRUE AND match_count > 0 THEN
    UPDATE state_Q3
    SET active = FALSE
    WHERE active = TRUE;
    INSERT INTO state_Q4 (conveyor_enable, active)
    VALUES (TRUE, TRUE);
    UPDATE transition_Q3_Q4
    SET last_triggered = NOW()
    WHERE camera_done = NEW.camera_done
      AND camera_result = NEW.camera_result;
  END IF;
END$$
DELIMITER ;
```

Сравнение методов проектирования дискретной логики управления

Критерий	Традиционный метод (языки стандарта МЭК 61131-3)	Предложенный метод (конечный автомат + реляционная модель данных)
Поддержка верификации и тестирования	Отладка проекта средствами конкретной среды разработки только непосредственно при разворачивании проекта	Поддержка формальных методов проектирования и верификации
Отделение логики от исполнения	Логика завязана на развертывании на конкретном ПЛК	Полное: логика описана в таблицах и отделена от конкретной платформы
Модифицируемость проектируемой системы управления	Требуется изменение всех компонентов, использующих модифицированный блок	Добавление новых состояний и переходов – без рефакторинга всей логики
Средства трассировки и логирования	Ограниченные – через внешние средства	Встроенные временные метки и состояния в базе данных
Масштабируемость проектируемой системы управления	Централизованная: отсутствуют средства создания логики управления распределенными системами	Централизованная и распределенная
Структурная модель поведения	Зависит от структуры кода; поведение часто распределено по блокам и трудно читается	Формализована как конечный автомат с явными состояниями и переходами
Интеграция с цифровыми платформами (SCADA/ MES/Digital Twin)	Требует адаптеров и шлюзов для обмена данными	Нативное считывание и управление из SCADA, цифровых двойников

Источник: составлено авторами по результатам проведенных экспериментов.

Функционирование описанного автомата было протестировано на производственном стенде предприятия ООО «ВЕКАС», где имитировались рабочие и аварийные сценарии. Автомат демонстрировал устойчивость, корректную реакцию на сигналы, последовательную активацию состояний, а также восстановление из аварий. Временные метки (timestamp, last_triggered) позволили отследить всю цепочку действий и подтвердили надёжность исполнения логики в реальном времени.

Для оценки практической значимости предложенного метода проектирования был проведён сравнительный анализ по ряду критериев. В таблице противопоставлены традиционный подход, основанный на использовании программируемых логических контроллеров и языков стандарта МЭК 61131-3 [15], и предложенная архитектура, реализующая управляющую логику средствами реляционной модели и SQL.

Таким образом, результаты реализации подтверждают применимость реляционного представления конечного автомата как основы для построения гибкой, формализованной и проверяемой системы управления, соответствующей требованиям современных цифровых производств.

Заключение

Предложенный метод проектирования логики систем управления технологическими процессами с использованием конечного автомата, реализованного средствами реляционной модели и SQL, представляет собой альтернативу традиционным подходам, ориентированным на процедурное кодирование. Он не подменяет классические ПЛК-системы, а расширяет инструментарий разработчика на этапах предварительного проектирования, моделирования и цифровой верификации логики.

Основное преимущество метода заключается в чётком разделении поведения системы и её реализации: логика существует как структура данных, поддающаяся быстрому изменению. Это открывает возможность интеграции с внешними цифровыми системами – от SCADA до цифровых двойников, а также упрощает перенос логики на другие объекты управления.

Поскольку автомат представлен в виде таблиц, его поведение может быть прослежено, зафиксировано, проанализировано и повторно использовано. Такой подход обеспечивает прозрачность, поддержку жизненного цикла системы и адаптацию к условиям гибкого производства. Полученные результаты демонстрируют потенциал реляционной модели как эффективного средства для описания дискретной логики в цифровой среде.

Список литературы

1. Cid N., Rico J.J., Pérez-Orozco R., Larranaga A. Experimental Study of the Performance of a Laboratory-Scale ESP with Biomass Combustion: Discharge Electrode Disposition, Dynamic Control Unit and Aging Effect // Sustainability. 2021. Vol. 13. № 18. DOI: 10.3390/su131810344.

2. Cangea O., Petrescu V.-B. Study and design of an automated pallet system // Romanian Journal of Petroleum & Gas Technology. 2022. Vol. 3. № 74. P. 49-56. DOI: 10.51865/JPGT.2022.02.05.

3. Xiaokang Chen, Yin Zhou, Bin Yang, Xinghua Miao. Designing Control System of Palletizing Robot Based on RobotStudio // Journal of Physics Conference Series. 2022. Vol. 2024(1). № 012039. DOI: 10.1088/1742-6596/2402/1/012039.

4. Giuseppe Fragapane, Rene de Koster, Fabio Sgarbossa, Jan Ola Strandhagen. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda // European Journal of Operational Research. 2021. Vol. 294. № 2. P. 405-426. DOI: 10.1016/j.ejor.2021.01.019.

5. Канахин В.С., Гудинов В.Н. Применение теории конечных автоматов при проектировании дискретных САУ цикловыми технологическими процессами в пищевой промышленности // ИТ. Наука. креатив: Материалы I Международного форума: в 5-ти томах, Омск, 14–16 мая 2024 года. Москва: Колос-с, 2024. С. 191-199. EDN: CIFYVI.

6. Рыбанов А.А., Свиридова О.В., Филиппова Е.М. Автоматный подход к реализации логики автоматизируемых бизнес-процессов в реляционных базах данных // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2023. № 3 (326). С. 40-47. DOI: 10.53598/2410-3225-2023-3-326-40-47.

7. Peter H. Mikkelsen, Cláudio Gomes, Peter G. Larsen. Survey on open-source digital twin frameworks – A case study approach // Software: Practice and Experience. 2024. Vol. 54. № 6. P. 929-960. DOI: 10.1002/spe.3305.

8. Mokhtari K., Panushev I., McArthur J.J. Development of a Cognitive Digital Twin for Building Management and Operations // Frontiers in Built Environment. 2022. Vol. 8. DOI: 10.3389/fbuil.2022.856873.

9. AutomationML: Engineering Data Exchange Format for Use in Industrial Automation Systems Engineering. IEC 62714 Series. URL: https://www.automationml.org (дата обращения: 17.04.2025).

10. Холопо В.А., Клягин М.М., Огорельцев Р.М. Метод моделирования конечных автоматов технологических процессов с применением языка SQL // Информационные и математические технологии в науке и управлении. 2025. № 1(37). С. 92-103. DOI: 10.25729/ESI.2025.37.1.009.

11. Закирзянов И.Т. Построение детерминированных конечных автоматов по примерам поведения с использованием подхода уточнения абстракции по контрпримерам // Научно-технический вестник информационных технологий, механики и оптики. 2020. Т. 20. № 3. С. 394-401. DOI: 10.17586/2226-1494-2020-20-3-394-401.

12. Кузнецов Н.А., Антонов С.В. Использование автоматного подхода для проектирования сетевой информационно-управляющей системы // Вестник компьютерных и информационных технологий. 2024. Т. 21. № 3 (237). С. 37-43. DOI: 10.14489/vkit.2024.03.pp.037-043.

13. Чистяков О.В., Благовещенская М.М., Рылов С.А., Веселов М.В., Благовещенский В.Г., Михайлов А.В. База для описания Цифрового двойника в рамках единой технологической системы управления с использованием формата AUTOMATIONML // Интеллектуальные автоматизированные управляющие системы в биотехнологических процессах : сборник докладов всероссийской научно-практической конференции, Москва, 29 марта 2023 года. Москва: Российский биотехнологический университет; ЗАО «Университетская книга», 2023. С. 336-343. EDN RIJMSR.

14. Клягин М.М., Зайцев И.Ю. Особенности проектирования баз данных производственных процессов // Концепция устройства современного мира в эпоху цифры: сборник научных трудов по материалам Международного научного форума, Москва, 15 декабря 2023 года. М.: Алеф, 2023. С. 93-102. DOI: 10.26118/1300.2023.63.84.006.

15. ГОСТ Р МЭК 61131-3–2016. Программируемые контроллеры. Часть 3. Языки программирования. Введ. 2017-04-01. М.: Стандартинформ, 2017. https://meganorm.ru/Data2/1/4293755/4293755016.pdf (дата обращения: 15.03.2025).