Платформа (объектный фреймворк) промышленной автоматизации ERSY Control и автоматизация крупных объектов агропромышленного комплекса.

Базис Инжиниринг Групп (ООО «ПроектМаркет», Ermakov Systima), Ермаков И. Е.

В работе представлен опыт коллектива по созданию собственной программной платформы для промышленной автоматизации ERSY Control, выполненой как объектно-ориентированный фреймворк на базе языка программирования общего назначения Component Pascal (Oberon-L) и визуальной исполняющей среды BlackBox Component Builder [1, 2], исполняющейся на базе ОС GNU/Linux или MS Windows.

Платформа создавалась непосредственно в процессе создания АСУТП для объектов агропромышленного комплекса – зерносушильных комплексов и мощностей хранения – холдинга «МираТорг» (Рис. 1, Рис. 2). На части объектов система внедрялась как первая, на других происходило импортозамещение унаследованной АСУТП на базе Siemens Simatic WinCC. Для сбора данных на нижнем уровне приходилось использовать разное предопределённое для нас оборудование: Siemens Simatic S7-300, Omron NX, ОВЕН.

Рис. 1. Мощности хранения зерна

Поскольку важнейшим критерием для нас изначально была наработка повторно используемой базы компонентов, то это обусловило архитектуру: использование ПЛК и модулей только для ввода-вывода и базовых защитных правил, размещение основной управляющей логики на встраиваемом сервере (устанавливаемом на DIN-рейку, достаточно скромном по параметрам — от 1Гб ОЗУ и т.п.) с ОС GNU/Linux. Понимая ограничения данного решения в части требований жёсткого реального времени (как и допустимость его применения для нашего класса объектов), мы ведём работу над собственным ПЛК и системой распределённого ввода-вывода, оснащённого нашей исполняющей системой уже в режиме «без ОС общего назначения».

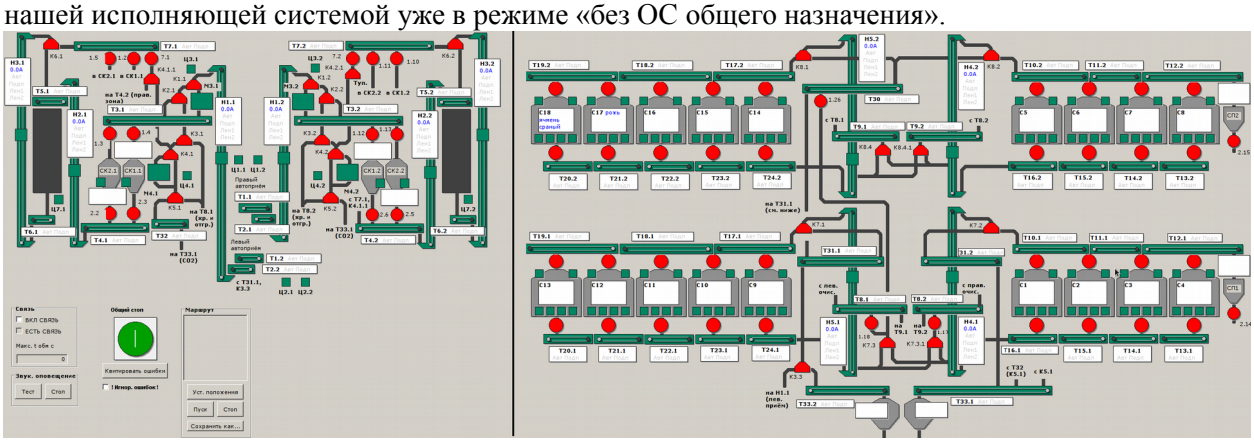

Рис. 2. Общая мнемосхема мощностей хранения зерна на 180 тыс. тонн (офлайн-состояние)

Чем обусловлен выбор в пользу инструментов программирования общего назначения? В информационных технологиях имеется общая закономерность: если мы используем любой предметно-ориентированный инструмент для решения задачи, якобы «исключающий программирование» – например, среду визуальных МЭК-языков (которые на сегодняшний день явно устарели как в аспекте возможностей абстракции, принятых для программной инженении, так и в плане возможностей формализации и верификации систем управления) или SCADA-платформу – то на каком-то уровне сложности задачи её легче или вообще необходимо решать с помощью программирования на языке общего назначения. Отсюда, конечно же, наличие во всех подобных инструментах скриптовых языков. Но в этом другая проблема: такие специальные скриптовые языки (как и часто используемые VB Script или JavaScript) плохо спроектированы, не имеют нормальной безопасности в виде статической типизации и системы модулей – и для программиста с опытом из общей сферы ИТ разрабатывать что-то на них – всё равно что по известной метафоре «кушать кактус».

С проблемой «уже проще программировать, чем конфигурировать» в задачах агропромышленного комплекса мы столкнулись сразу же. Например, потому, что в отличие от задач, подобных диспетчеризации котельных и ЖКХ, управление сыпучим производством требует интерактивного взаимодействия и активного принятия решений оператором – по перестроению маршрутов и т. п. Нам приходилось видеть попытки инженеров АСУТП реализовывать поддержку маршрутов и их защиты на основе, например, МЭК-языков. Обычно это приводило к попытке явно согласовать с заказчиком ограниченное подмножество возможных маршрутов и реализовать это путём многократно дублирующихся МЭК-описаний («метод копировать-вставить»). В то же время поддержка полноценной гибкой логики в подобных задачах (автоматический поиск всех возможных путей от А до Б, активация маршрута и его контроль) требует достаточно простого программирования алгоритмов из области теории графов – прекрасно программируется на объектно-ориентированном языке общего назначения и превращается в универсальный компонент, пригодный для любого объекта сыпучих призводств.

В силу данного опыта мы выработали своё понимание, как должен развиваться прикладной инструментарий, подобный инструментарию промышленной автоматизации. Первый шаг – это выбор качественного, простого, надёжного и эффективно компилируемого в машинный код языка программирования общего назначения. Для языка обязательно наличие строгой статической типизации, системы модулей, современной компонентной модели (ООП, динамическая загрузка и выгрузка модулей, метапрограммирование). О выборе языка Component Pascal (Oberon-L) мы скажем ниже.

На основе такого языка разрабатывается объектно-ориентированный фреймворк, содержащий абстракции, позволяющие выразительно описывать понятия задачи и их отношения, переносить знания инженера в код. В частности, наш фреймворк ERSY Control позволяет описывать онтологии (в простом смысле этого слова): виды и подвиды тегов, перечислимые множества и подмножества их значений, классификацию нештатных ситуаций, а далее на основе этого создавать хорошо структурированные, обобщённые алгоритмы, которые можно повторно использовать на разных объектах, накапливая корпоративную библиотеку решений и описывая только отличающиеся особенности нижнего уровня (точный состав датчиков и сигналов обратной связи в шкафах, типы пуска – прямой, УПП или «звезда-треугольник» и т. п.).

Далее над таким фреймворком могут быть сделаны инструменты визуального конфигурирования и автоматического порождения кода. Например, удобные формы табличного описания состава сигналов и структуры объекта. Но важно, что под этими визуальными инструментами находится не «чёрный ящик» с непонятным устройством, а порождаемый набор сущностей фреймворка, описанных на языке программирования общего уровня. Это принципиально влияет на надёжность, контролируемость и расширяемость решений.

Другим составным компонентом решения является гибкая графическая среда, позволяющая удобное расширение своими графическими элементами, визуальное редактирование экранов интерфейса и их исполнение на базе компактного приложения (10-20 Мб), пригодного для запуска и на базе ОС GNU/Linux, и MS Windows (которую в современной ситуации следует расценивать как опасный рудимент в отрасли АСУТП, подлежащий замещению) и реализующего работу с графикой и документами своими средствами — то есть **не использующего** никаких «жирных» и опасных для ответственных применений графических и мультимедийных платформ (веб-браузера, Flash, Silverlight и т. п., что достаточно часто задействуется в современных SCADA-платформах – например, у Siemens или Schneider Electric, но в рамках ответственного подхода эти технологии не должно были выйти за пределы бытового применения). Основой для графической среды АСУТП стала среда разработки и исполнения BlackBox Component Builder и её документная графическая система BlackBox Component Framework, которая была доработана и расширена для удобной разработки человеко-машинного интерфейса АСУТП.

Мы знаем, что наш подход «язык общего уровня – объектный фреймворк – инструментарий» не уникален, и что решения подобного рода есть. Однако в качестве базовых языков программирования выбирается либо Java, либо C#, либо C++. Все они являются избыточно сложными по сравнению с выбранным нами языком семейства Оберон (и, в общем, особенно после признания индустрией наследника Оберона – языка Google Golang – их сложность может расцениваться как устаревшая). C++ (в отличие от Java и C#) не является безопасным компонентным языком (не обеспечивает герметичность указателей и контроль границ массивов, не поддерживает динамическую загрузку и выгрузку модулей и метапрограммирование). Java и C#, обладая такими свойствами, в отличие от Оберон-языков находятся достаточно далеко от уровня

машины, вместо прямой и прозрачной трансляции в машинный код используют промежуточные представления и сложные, не верифицируемые Just-In-Time-кодогенераторы, а самое главное – не могут использоваться без сборщика мусора, что ограничивает их применение только операторским интерфейсом (и то под вопросом).

В Европе и Швейцарии (по месту рождения) языки и ОС семейства Оберон активно используются для промышленных задач десятилетиями [3]. Российское Оберон-сообщество развивается активно с 2005 года, вокруг проектов OberonCore[1] и Информатика-21[2]. За этот период возник ряд российских применений Оберон-систем в ответственных задачах: АЭС [4], беспилотные летательные аппараты [5], системы энергораспределения [6], наши задачи в агропромышленном комплексе.

Российское Оберон-сообщество за эти годы сформировало сильную эскпертную группу по проблематике ответственных ИТ-систем и общей проблеме качества в цифровой индустрии. Силами этой группы в октябре 2018 года была проведена конференция «Оберон-технологии, образование и проблема качества в цифровой индустрии» [7], с участием Юрга Гуткнехта – почётного профессора Цюрихской технологической школы, экс-президента Швейцарского общества информатики, соавтора Никлауса Вирта по созданию языка и ОС Оберон. Если первый день конференции был посвящён применениям и развитию Оберон-технологий, то второй – общей проблеме качества в цифровой индустрии, её преодолению путём накопления и внедрения хороших практик, изменений в образовании и развития специализированных компактных предметно-ориентированных инструментов, создаваемых на базе качественных языков программирования общего назначения и фреймворков на них.

Нам представляется, что в связи с ростом сложности, развитием массовости (Internet of Things) и интеграцией в открытые сети проблема качества, надёжности и безопасности систем промышленной автоматизации встаёт в полный рост. Последние два года мы видим серию отчётов о критических уязвимостях в дорогих и «жирных» системах от ведущих поставщиков (часто такие системы разработаны их производителями, изначально специализированными на электронике, путём «индусской» аутсорс-разработки, без грамотного проектирования программной архитектуры – со всеми вытекающими последствиями).

Сейчас особо ценно сопоставление двух миров: промышленной автоматизации и ответственных встроенных (например, бортовых) систем управления. Во второй сфере накоплен большой опыт разработки на основе и общих, и специализированных технологий программирования, но с использованием и средств обеспечения надёжности (развитые системы строгой типизации, как в языке Ada, формальной верификации – как в её подмножестве SPARK), и средств хорошей абстракции и повторного использования кода. Примером интегрированной технологии разработки ответственного бортового ПО, построенной поверх надёжного и простого языка программирования общего назначения Модула-2 (предшественник Оберона), является технология разработки бортового ПО спутников связи и ГЛОНАСС, разработанная под руководством Колташёва А. А. в ИСС им. Решетнева [8].

Однако сфера промышленной автоматизации тоже может поделиться своим опытом. Опытом инструментов, позволяющих без программирования, высокоуровневым визуальным конфигурированием, собирать системы автоматизации по принципу «конструктора». Если убрать из этого подхода непрозрачные корпоративные среды и опереть его на открытые фреймворки на качественном языке программирования общего назначения – то это и будет являться, по нашему мнению, перспективным направлением развития инструментов промышленной автоматизации, способным решить текущий комплекс проблем с избыточной сложностью, низкой надёжностью, слабым повторным использованием и уязвимостями для вторжений. А также решить проблему импортозамещения и технологической независимости для государств в 21 веке.

1. Проект OberonCore – центр российского сообщества по Оберон-технологиям и системе Component Pascal/BlackBox [Электронный ресурс] Режим доступа: http://oberoncore.ru/

2. Международный общественный научно-образовательный проект «Информатика-21» [Электронный ресурс] Режим доступа: http://www.inr.ac.ru/~info21

3. Применения Оберон-систем [Электронный ресурс] Режим доступа: https://oberoncore.ru/wiki/применения

4. Внедрение решений на базе Оберона на Ростовской АЭС Д. В. Дагаевым [Электронный ресурс] Режим доступа: http://www.inr.ac.ru/~info21/oberon_innovation/oberonRostovAES.htm

5. Оберон в беспилотных летательных аппаратах [Электронный ресурс] Режим доступа: http://www.inr.ac.ru/~info21/oberon_innovation/oberonBPLA.htm

6. Оберон в энергораспределении [Электронный ресурс] http://www.inr.ac.ru/~info21/oberon_innovation/oberonKaliningrad.htm

7. Конференция и заседание экспертной группы «Оберон-технологии, образование и проблема качества в цифровой индустри» [Электронный ресурс] Режим доступа: https://oberoncore.ru/oberonconf

8. Антамошкин А. Н., Колташев А. А. Технологические аспекты создания бортового программного обеспечения спутников связи. // Вестник СибГАУ им. акам. М. Ф. Решетнева. Вып. 6 - Красноярск: СибГАУ, 2005.