Тема 1.2. Требования к архитектуре компьютерной системы автоматизации. Открытость архитектуры.

Требования к архитектуре компьютерной системы автоматизации

Жизненный цикл КСА состоит из следующих фаз:

- разработка архитектуры и эскизное проектирование;
- проектирование и изготовление системы;
- монтаж и пуско-наладка;
- эксплуатация системы;
- обслуживание;
- реконфигурация, модернизация, разборка, утилизация.

Таким образом, разработка архитектуры (концепции построения системы) является первым этапом реализации КСА.

Архитектура КСА *– это абстрактное ее представление, которое включает в себя идеализированные модели компонентов системы, а также модели взаимодействий между компонентами.* Элементы архитектуры находятся во взаимосвязи, образуя единую автоматизированную систему и обеспечивая решение поставленной задачи автоматизации на архитектурном уровне. В то же время архитектура оставляет достаточно свободы для выбора конкретных технических решений. Поэтому правильно спроектированная архитектура допускает множество технических реализаций путем выбора различных компонентов архитектуры и методов взаимодействия между ними.

Элементами архитектуры являются модели (абстракции) датчиков, устройств ввода-вывода, измерительных преобразователей, ПЛК, компьютеров, интерфейсов, протоколов, промышленных сетей, исполнительных устройств, драйверов, каналов передачи информации.

Проектировщика архитектуры называют **архитектором.** Главным требованием к нему является знание предметной области (принципов функционирования объекта автоматизации) и знание технических характеристик аппаратных и программных средств, используемых для построения системы.

Архитектура системы может быть различной в зависимости от решаемой задачи автоматизации. **Задачами автоматизации могут быть:**

- мониторинг (продолжительные измерение и контроль с архивированием полученной информации);
- автоматическое управление (в системе с обратной связью или без нее);
- диспетчерское управление (управление с помощью человека-диспетчера, который взаимодействует с системой через человеко-машинный интерфейс);
- обеспечение безопасности.

В случае большого расстояния между КСА и объектом автоматизации говорят о **задаче телемеханики** (дистанционные измерение, управление, сигнализация)**,** однако, учитывая распространение современных каналов удаленной передачи данных, такую задачу редко выделяют как отдельную.

При построении архитектуры должны быть заложены следующие **свойства будущей системы автоматизации**:

- слабая связанность элементов архитектуры между собой (т.е. декомпозицию системы на части следует производить так, чтобы поток информации через связи был минимален и через них не замыкались контуры автоматического регулирования);
- тестируемость (возможность установления факта правильного функционирования);
- диагностируемость (возможность нахождения неисправной части системы);

- ремонтопригодность (возможность восстановления работоспособности за минимальное время при экономически оправданной стоимости ремонта);
- надежность (например, путем резервирования);
- простота обслуживания и эксплуатации (минимальные требования к квалификации и дополнительному обучению эксплуатирующего персонала);
- безопасность (соответствие требованиям промышленной безопасности и технике безопасности);
- защищенность системы от вандалов и неквалифицированных пользователей;
- экономичность (экономическая эффективность в процессе функционирования);
- модифицируемость (возможность перенастройки для работы с другими технологическими процессами);
- функциональная расширяемость (возможность ввода в систему дополнительных функциональных возможностей, не предусмотренных в техническом задании);
- наращиваемость (возможность увеличения размера автоматизированной системы при увеличении размера объекта автоматизации);
- открытость (см. далее);
- максимальная длительность жизненного цикла системы без существенного морального старения, достигаемая путем периодического обновления аппаратных и программных компонентов, а также путем выбора долгоживущих промышленных стандартов;
- минимальное время на монтаж и пуско-наладку (развертывание) системы.

С целью обеспечения вышеперечисленных свойств КСА, а также решения ею поставленных задач проектирование любой КСА следует начинать с **декомпозиции** (деления на части) системы на подсистемы. Декомпозиция может быть **функциональной** (алгоритмической) или **объектной (топологической, территориальной).**

При объектной декомпозиции используются распределенные системы управления, когда каждый объект автоматизации оборудуется локальным технологическим контроллером, решающим задачи в пределах этого объекта. При функциональной декомпозиции систему автоматизации делят на части, группируя сходные функции, и для каждой группы функций используют отдельный контролер. Оба вида декомпозиции могут быть использованы совместно. Выбор способов декомпозиции является творческим процессом и во многом определяет эффективность будущей системы.

Независимо от метода декомпозиции, основным ее результатом должно быть представление системы в виде набора слабо связанных частей. Слабая связь между частями системы означает отсутствие между ними обратных связей или малость модуля петлевого усиления при наличии таких связей, а также отсутствие интенсивного обмена информацией.

Отдельного внимания заслуживает **свойство открытости** компьютерных систем автоматизации, поскольку именно оно является основой для обеспечения остальных свойств системы.

Открытость как одно из основополагающих свойств архитектуры системы

Длительное время до появления ПТК, как это уже упоминалось в предыдущей лекции, рост размерности системы сопровождался значительным ростом ее сложности и, соответственно, стоимости, поскольку сложные системы изготавливались на заказ и содержали единичные дорогостоящие элементы. Применение систем отдельных производителей, разрабатываемых по их внутренним стандартам, делало заказчика навсегда привязанным к конкретному разработчику.

Решением данной проблемы стало деление системы на **модули** таким образом, чтобы каждый из них становился коммерчески эффективным изделием и мог изготавливаться несколькими конкурирующими производителями в больших количествах. Возникшая при этом **проблема аппаратной и программной совместимости модулей** (т.е. достижение совместимости интерфейсов, конструктивов и выполняемых функций модулей) решается разработкой соответствующих **международных стандартов**.

Таким образом, ***открытой архитектурой*** можно считать *модульную систему, допускающую замену любого модуля на аналогичный модуль другого производителя, имеющийся в свободной продаже по конкурентоспособным ценам, а интеграция системы с другими системами (в том числе с пользователем) легко выполнима благодаря соответствию элементов системы общепринятым стандартам*.

Открытость можно рассматривать на разных уровнях иерархии программного и аппаратного обеспечения системы или ее составных частей. Открытыми, например, могут быть:

- физические интерфейсы, протоколы обмена, методы контроля ошибок, системы адресации, форматы данных, типы организации сети, интерфейсы между программами, диапазоны изменения аналоговых сигналов;
- пользовательские интерфейсы, языки программирования контроллеров, управляющие команды модулей ввода-вывода, языки управления базами данных, операционные системы, средства связи аппаратуры с программным обеспечением;
- конструкционные элементы (шкафы, стойки, корпуса, разъемы, крепежные элементы);
- системы, включающие в себя перечисленные выше элементы.

Как следует из определения**, необходимыми условиями открытости являются**:

- модульность;
- соответствие стандартам (необязательно официальным, но обязательно общепринятым и легко доступным по цене, компенсирующей только затраты на его разработку, поддержку и распространение);
- наличие в свободной продаже аналогичных систем других производителей (подсистем, модулей) по конкурентоспособным ценам.

Требование **модульности** вытекает из требования возможности замены части системы (т.е. модуля) аналогичными изделиями других производителей. Для этого система должна состоять из модулей.

Соответствие стандартам необходимо для обеспечения совместимости.

Наличие в свободной продаже и конкурентоспособность цен являются требованиями, вытекающими из практического аспекта: без выполнения этого условия открытая система может существовать только «на бумаге».

Понятие открытости достаточно многогранно и не стандартизовано. Поэтому практически можно говорить только о **степени открытости** системы, указывая, что именно понимается под открытостью в каждом конкретном случае. Степень открытости можно оценить количеством реализованных признаков открытости.

Идеальным примером открытой системы является офисный компьютер – множество производителей изготавливают взаимозаменяемые программные и аппаратные компоненты, которые можно собрать в систему или нарастить функциональность существующей. Отсутствуют производители-монополисты.

Необходимо отметить, что открытость элементов системы и, соответственно, самой системы не подразумевает **открытости исходного кода** готовых проприетарных продуктов, что существенно снижает надежность системы вследствие потенциальной возможности появления в ней дополнительных ошибок, внесенных во время модификации и компиляции.

Свойства открытых систем

Открытые системы (системы с открытой архитектурой) обладают следующими свойствами:

- модульность;
- платформенная независимость;
- взаимозаменяемость с компонентами других производителей;
- интероперабельность (возможность совместной работы) с компонентами других производителей;
- масштабируемость.

Модульность *— это способность аппаратного или программного обеспечения к модификации путем добавления, удаления или замены отдельных модулей (компонентов системы) без воздействия на оставшуюся ее часть.* Модульность обеспечивается декомпозицией (объектной или функциональной) на этапе проектирования архитектуры системы. Главным достижением в направлении развития модульности программного обеспечения АСУТП является выделение в нем независимых подсистем: программы в ПЛК, ОРС сервера, баз данных, операторского интерфейса и алгоритмической части, реализуемой на языках стандарта IEC 61131-3, а также деление на серверную и клиентскую части.

Платформенная независимость *– возможность выполнения программ на разных аппаратно-программных платформах, обеспечивающая независимость от поставщика этих платформ.* **Преимущества платформенной независимости** следующие:

- расширение выбора оборудования путем увеличения числа поставщиков;
- независимость от поставщика аппаратного и программного обеспечения.

Платформенную независимость обеспечивают применение различных операционных систем (Windows, FreeBSD, OpenBSD, Linux, MacOS и т.д.), языков программирования со стандартными компиляторами (C++, Java, Delphi, языки стандарта IEC 61131-3 и др.), СУБД и языка запросов SQL (доступ к базе данных с помощью SQL осуществим независимо от программно-аппаратной платформы, на которой она находятся), а также применение интранет-технологий, когда передача информации к рабочей станции осуществляется с помощью языка xml, а ее представление пользователю выполняется с помощью любого веб-браузера. Веб-браузер позволяет в качестве рабочей станции КСА использовать компьютер и операционную систему любого производителя из имеющихся в свободной продаже.

Взаимозаменяемость — это возможность замены любого модуля (компонента) системы на аналогичный компонент другого производителя, имеющийся в свободной продаже, и возможность обратной замены. Это свойство позволяет ускорить замену отказавшего модуля, улучшить качество уже работающей системы, исключить ценовую зависимость от поставщика.

Интероперабельность (аппаратно-программная совместимость) *— это способность открытых систем использовать программы, выполняющиеся одновременно на различных платформах в общей сети, с возможностью обмена информацией между ними.* Иначе говоря, программные компоненты системы, расположенные на разных аппаратных платформах в общей сети, должны быть способны работать как часть единой системы. Открытая интероперабельная система должна обладать способностью коммуникации и с другими уровнями АСУ предприятия, обеспечивая одновременно безопасность поступающей извне информации.

Методами обеспечения интероперабельности Windows и Unix платформ может быть применение стандарта CORBA (Common Object Request Broker Architecture), а также DCOM или SOAP.

Масштабируемость — это возможность применения одного и того же аппаратного и программного обеспечения (баз данных, пользовательских интерфейсов, средств коммуникации) для систем разного размера (больших и малых). Для обеспечения

масштабируемости достаточно, чтобы программное обеспечение больших и малых систем было совместимо по операторскому интерфейсу, языкам программирования, а также интерфейсу с аппаратными средствами и не требовало дополнительного обучения персонала. Масштабируемая система должна обеспечивать возможность простого наращивания функциональных возможностей и размеров путем включения новых компонентов как в аппаратную, так и программную часть системы без модификации старых, опробованных программных и аппаратных модулей.

До появления открытых систем обеспечение масштабируемости достигалось путем проектирования системы с большим запасом по габаритам, количеству слотов, интерфейсов. Наращиваемость открытой системы подразумевает иной путь, не требующий запаса ресурсов (и связанных с ним избыточных финансовых вложений). В частности, система, обладающая свойством платформенной независимости и интероперабельности, уже является расширяемой, поскольку она позволяет добавлять новое оборудование или заменять старое новыми модификациями, в том числе оборудованием других производителей.

Достоинства и недостатки открытых систем

До появления компонентов открытых систем создание оборудования требовало разработки специализированных печатных плат, что было чрезмерно дорого и долго. Кроме того, некоторые необходимые функции при этом не могли быть реализованы никогда из-за жестких ограничений на сроки создания системы, поэтому основным преимуществом систем с открытой архитектурой является низкая стоимость их жизненного цикла.

Достоинствами применения открытых систем являются:

- пониженные вложения на проектирование системы и предпроектные изыскания благодаря наличию на рынке большого выбора готовых компонентов открытых систем;
- упрощение процесса интеграции — открытость подразумевает возможность простой интеграции разнородных систем;
- экономия финансовых средств благодаря низкой стоимости жизненного цикла (в основном вследствие конкуренции независимых производителей и отсутствия диктата цен монопольным поставщиком);
- увеличенное время безотказной работы благодаря выбору наиболее надежных модулей из имеющихся на рынке;
- минимизированное время вынужденного простоя благодаря большому выбору взаимозаменяемых модулей всегда можно найти поставщика, имеющего нужные модули на складе;
- минимальные усилия на ввод в действие как аппаратуры, так и программного обеспечения благодаря устранению затрат времени на дополнительное обучение как монтажной организации, так и эксплуатирующего персонала;
- простое изменение конфигурации системы для работы с новыми технологическими процессами — вытекает из свойств модульности и расширяемости открытых систем;
- минимальный объем дополнительного обучения персонала и, как следствие, простота обслуживания;
- применение новейших технологий и технических решений благодаря широкому выбору наилучших решений и специализации производителей;
- увеличение времени жизни системы благодаря взаимозаменяемости отработавшего ресурс и нового оборудования, а также возможности наращивания функциональных возможностей.

Недостатки открытых систем видны не сразу. И все же они имеются:

- при создании автоматизированной системы на базе открытых решений ответственность за работоспособность системы в целом ложится на систем-

ного интегратора, а не на производителя системы. Поэтому при появлении в системе невоспроизводимых отказов некому предъявить претензии, поскольку поставщиков много, а системный интегратор отвечает только за монтаж и пусконаладку системы;

- универсальность всегда находится в противоречии с простотой. Универсальные протоколы, интерфейсы, сети и программное обеспечение, чтобы быть универсальными, должны быть достаточно сложными, следовательно, дорогими и ненадежными. Хотя снижение надежности, вызванное сложностью, компенсируется повышением надежности благодаря большому тиражу и, следовательно, продолжением отладки после начала продаж;
- эффект снижения надежности программного обеспечения, части которого пишутся разными производителями. Когда ПО пишется внутри одной фирмы, можно предвидеть почти все ситуации, которые могут возникнуть на границе между ПО и пользователем или аппаратурой. Если же в этом участвуют несколько разных команд в разных фирмах, между которыми нет взаимодействия, то становится непонятно, кто отвечает за надежность всего комплекса. Надежность и безопасность открытых систем остаются темами, требующими решения;
- иногда к признакам открытости относят открытость исходных кодов. Однако наличие открытых кодов снижает надежность программной системы, поскольку нарушается принцип инкапсуляции, необходимость которого обоснована в идеологии объектно-ориентированного программирования;
- как и любая стандартизация, открытость накладывает ограничения на диапазон возможных технических решений, затрудняя творчество и снижая вероятность появления новых и плодотворных технических решений.

Отметим, что проблема надежности относится не ко всем компонентам открытых систем. Например, такие компоненты, как базы данных, компьютеры или сети Ethernet, обладают высокой надежностью благодаря огромному тиражу и, как следствие, качественной валидации этих компонентов и оптимизации процессов изготовления. Кроме того, выше перечислены только факторы, понижающие надежность открытых систем. Однако одновременно имеются факторы, которые ее повышают, — это увеличенный тираж модулей открытых систем по сравнению с низким тиражом полностью заказных систем. Поэтому вывод о надежности открытой системы может быть как положительным, так и отрицательным, в зависимости от конкретного состава ее элементов.

Средства достижения открытости

Открытость систем, как уже было сказано, достигается применением взаимозаменяемых аппаратных и программных средств, производимыми независимыми производителями и удовлетворяющих требованиям общепринятых стандартов. Вкратце рассмотрим основные средства.

Промышленные сети и протоколы. Наиболее распространенными в России являются сети Modbus, Profibus, CAN, Ethernet.

Физические интерфейсы. Наибольшая часть средств промышленной автоматизации, представленных на российском рынке, имеет интерфейсы RS-232, RS-485, RS-422, CAN, Ethernet, USB. Большое значение для повышения степени открытости имеют преобразователи интерфейсов и межсетевые шлюзы, которые позволяют объединять в единую систему несовместимое по интерфейсам и протоколам оборудование.

Программные интерфейсы. Для взаимодействия открытых систем на программном уровне наибольшее распространение получила DCOM-технология фирмы Microsoft, ставшая промышленным стандартом OPC (OLE for Process Control), который пришел на смену устаревшей технологии DDE (Dynamic Data Exchange). Стандарт OPC

обеспечил возможность применения оборудования различных производителей практически с любыми SCADA, имеющимися на рынке, поскольку большинство из них поддерживает стандарт OPC.

Аналогичная задача может быть решена также с помощью технологии Jini фирмы SUN и CORBA фирмы OMG, однако воплощение в международный стандарт OPC получила только технология DCOM, ориентированная на Windows-платформы.

Интерфейс пользователя. Интерфейс между SCADA (системами сбора данных и диспетчерского управления) и пользователем в настоящее время выполняется примерно одними и теми же визуальными средствами, которые стали стандартом де-факто: кнопки пуск/стоп, цифровое табло, линейный или радиальный индикатор уровня, цветовая сигнализация, окна с текстовыми сообщениями, окна ввода данных, графики (тренды) и т.п. Такой интерфейс легко осваивается операторами АСУТП. Кроме того, многие SCADA-системы последних лет поддерживают веб-технологию, когда пользовательский интерфейс SCADA выполняется в виде веб-страницы и располагается на сервере локальной сети. При этом любой пользователь, обладающий достаточными правами доступа, с помощью стандартного веб-браузера может следить за технологическим процессом и управлять им. Такой подход является значительным прогрессом в направлении открытости SCADA-пакетов, поскольку предоставляет пользователю широкий выбор веб-браузеров и обеспечивает применение практически любой аппаратно-программной платформы для обмена данными со SCADA.

Языки программирования средств управления. Программирование контроллеров поддерживается тремя международными стандартами: стандартом МЭК 61131-3 на языки программирования и стандартами МЭК 61499 и МЭК 61804 на функциональные блоки. Стандарты поддерживаются большинством производителей программного обеспечения. Примером могут быть системы ISaGRAF фирмы ICS Triplex и CoDeSys фирмы 3S. Поддержку открытости обеспечивают также конверторы блоков UML (Unifid Modeling Language) в функциональные блоки стандарта МЭК 61499, а также UML в xml (extended Markup Language).

Программная совместимость. Благодаря технологиям OLE (Object Linking and Embedding, связывание и внедрение объекта), DCOM (Distributed COM – распределённая COM) стандарта COM и технологии CORBA (Common Object Request Broker Architecture – общая архитектура брокера объектных запросов) программы могут обращаться к компонентам друг друга и обмениваться данными в т.ч. и по сети, что делает возможным построение распределенных полностью совместимых приложений.

Для Windows-приложений технология .NET позволяет создавать виртуальные машины, обеспечивающие взаимодействие приложений, написанных на разных языках программирования (C, C++, VB.NET, Delphi, JScript .NET и др.).

Совместимость баз данных с использующими их программами обеспечивает широко распространенный язык запросов SQL, соответствующий международному стандарту и поддерживаемый несколькими СУБД (системами управления базами данных), например Informix, Sybase, Ingres, MS SQL Server. Интерфейс ODBC (Open Data Base Connectivity) позволяет подключать различные СУБД, что повышает степень ее открытости.

Обеспечение возможности программирования на языке Visual Basic, а также возможность встраивания ActiveX и COM объектов сторонних производителей позволяет применить принцип повторного использования программного кода, написанного для других приложений, а также адаптировать программные продукты к аппаратуре, не поддерживающей стандарт OPC, что актуально для, например, для SCADA систем.