ИНФОКОММУНИКАЦИОННОЕ ПРОСТРАНСТВО СИБИРИ И ДАЛЬНЕГО ВОСТОКА

ИНФОСФЕРА

№71 2016

Архитектурные представления программных систем инфокоммуникационных предприятий

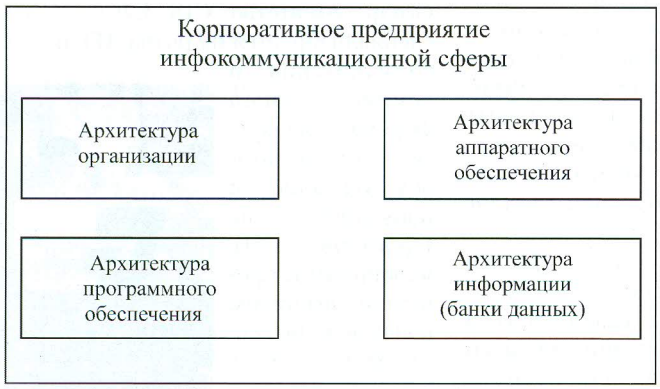
Назаров С.В., главный специалист ЗАО «МНИТИ», д.т.н., профессор, nazarov@mniti.ru

Вилкова Н.Н., президент АРПАТ, генеральный директор ЗАО «МНИТИ», к.т.н.; mniti@mniti.ru

1.Понятие архитектуры предприятия

Эффективная работа современных инфокоммуникационных предприятий различного профиля (сферы связи, телекоммуникаций, радио, телевизионного, спутникового и кабельного вещания) основана на современных информационных технологиях. Показательным примером может служить автоматизированная система расчетов (АСР) – универсальная конвергентная система биллинга Platex® высшего функционального уровня. Система позволяет тарифицировать широкий комплекс услуг, создавать единые отчеты и финансовые документы по их использованию, управлять телекоммуникационным оборудованием, вести лицевые (балансовые) счета абонентов, вести межоператорские расчеты. АСР Platex® совместима с продуктами компании «1С», интегрируется с различными платежными системами, соответствует всем требованиям государственных регулирующих органов, сертифицирована для применения в сетях электросвязи емкостью до 1,5 млн. абонентов.

Любое инфокоммуникационное предприятие характеризуется определенной архитектурой. Будем обозначать ее как А. Архитектура предприятия – сложное понятие, определяющее миссию, стратегию и цели предприятия, организационную структуру, бизнес-архитектуру и многое другое. Основные составляющие архитектуры предприятия можно представить, как показано на рис. 1.



К сожалению, в отрасли IT-технологий не существует соглашения о значении каждого из этих терминов или их отношениях друг к другу, в результате чего одни и те же понятия могут иметь разные значения, а два или более терминов могут обозначать одно и то же. Целью данной работы является уточнение понятий и составляющих архитектуры предприятия, попытка их формализации и акцентирование внимания читателя на важности архитектурного представления и последующей разработки программных систем в общей архитектуре предприятия как основы его эффективного функционирования.

2.Теоретико-множественное представление архитектуры

программных систем в архитектуре предприятия

Рассмотрим элементы, определяющие архитектуру предприятия. К ним относятся (рис. 1):

a) архитектура программного обеспечения A_S (Software), тема данной статьи;

b) архитектура аппаратного обеспечения A_H (Hardware), которое включает такие элементы, как связная, телекоммуникационная, вещательная, компьютерная и др. аппаратура;

c) архитектура организации A_O (Organization) включает элементы, имеющие отношение к бизнес-процессам, структурам организации, ролям и ответственности, а также основные области специализации организации;

d) архитектура информации A_I (Information), включающая структуры, которые упорядочивают информацию.

Таким образом, архитектуру предприятия можно представить четверкой вида

$$A = \langle A_S, A_H, A_O, A_I \rangle. \quad (1)$$

Наиболее тесно с архитектурой программного обеспечения связан такой элемент архитектуры предприятия как архитектура бизнеса. Основу этого представления архитектуры бизнеса составляет перечень задач, решаемых предприятием в процессе его функционирования. Будем считать, что это некоторое множество задач

$$Z=\{z_1, z_2,...z_n\}. \quad (2)$$

Каждая задача zi определяет некоторый бизнес-процесс b_i. Совокупность бизнес-процессов обозначим множеством

$$B=\{b_1, b_2,...,b_n\}. \quad (3)$$

В общем случае бизнес-процессы определенным образом связанным между собой, т.е. для любого бизнес-процесса b_i существует определенное его отображение в другие бизнес процессы

$$\Gamma b_i \in B \times B, \quad (4)$$

где B×B – декартово произведение множества B. Возможно, что для некоторых b_j отображение $\Gamma b_j = \varnothing$, что говорит о том, что бизнес-процесс является конечным в некоторой цепочке бизнес-процессов. Однако, при этом обязательно выполнение условия

$$\Gamma^{-1} b_j \neq \varnothing. \quad (5)$$

В противном случае, бизнес-процесс является изолированным (считается, что это невозможно). Таким образом, бизнес-архитектуру предприятия можно представить в виде двойки

$$AB = \langle B, G_B \subset B \times B \rangle, \quad (6)$$

где G_B – представление технологии функционирования предприятия в виде графа бизнес-процессов.

Выполнение бизнес-процессов поддерживается IT-технологиями, реализованными в интегрированных или специализированных программных средствах (пакетах). Множество этих программ обозначим как $P=\{p_1, p_2,..., p_m\}$. Как правило, $m \ll n$ за счет использования интегрированных пакетов. Использование программных пакетов в бизнес-процессах можно представить подграфом двудольного графа с вершинами двух типов B и P в виде тройки

$$A_{it} = <B, P, G_{bp} \subset B \times P>, (7)$$

которую будем называть архитектурой IT-технологий.

Каждый программный пакет pi в пределах бизнес-процесса bi обрабатывает некоторый набор данных d_{ij}. В целом, данные предприятия могут быть представлены в виде базы данных $D=\{d_{ij}\}$, i=1,..,n, j=1,...,m, и по аналогии с (7) использование данных в бизнес-процессах можно представить в виде тройки

$$A_d = <P, D, G_d \subset P \times D>. (8)$$

Однако, представление архитектуры ИТ и архитектуры данных, использованных в этих технологиях, будет неполным в отрыве от технических средств хранения, передачи данных и процессов их обработки. Эти технические средства – неотъемлемая часть корпоративной информационной системы (КИС). Последнюю можно рассматривать как множество аппаратных элементов (компьютеров, серверов, каналов связи, средств коммутации и т.д.), объединенных в определенную структуру (локальную или территориально распределенную), которую можно представить архитектурой аппаратных компонентов КИС в следующем виде

$$A_{КИС} = <K, G_k \subset K \times K>, (9)$$

где $K = \{k_1, k_2,.., k_n\}$ – множество аппаратных элементов КИС, K×K – декартово произведение множества K, G – граф структуры КИС. Говоря об архитектуре КИС, необходимо учитывать распределение программных средств P и данных D по элементам КИС. Такое распределение можно представить следующими выражениями:

$$Rp = <P, K, G_p \subset P \times K> ,(10)$$
$$Rd = <D, K, G_d \subset D \times K>, (11)$$

где G_p – двудольный граф, отражающий распределение программ по аппаратным элементам КИС или G_p = P -> K. G_d – двудольный граф, отражающий распределение данных по аппаратным элементам КИС или G_d = D -> K.

Таким образом, архитектуру КИС можно представить следующим выражением

$$A_{КИС} = <K, P, D, G_k, G_p, G_d>. (12)$$

3. Почему важна архитектура программной системы

Наверное, следующее высказывание подтвердит значимость архитектуры ПС [1]. Архитектура – это то, за что увольняют системного архитектора и руководителя проекта. Можно заметить, что проекты на самом деле проваливаются из-за неправильной организации процесса разработки. Не отрицая этого, нужно сказать, что известно гораздо больше успешных систем, построенных в "кривом процессе" разработки, чем успешных систем с "кривой архитектурой". Остановимся на вопросе, почему важна архитектура программной системы с технической точки зрения. В этом контексте следует привести три основных фактора, которые отмечает Л. Басс [2].

3.1. Взаимодействие между заинтересованными лицами. Программная архитектура – это универсальная абстракция системы, на основе которой все или почти все заинтересованные в системе лица могут искать взаимопонимания, вести переговоры, находить компромиссы. В этом плане архитектура – это общий язык, на котором можно сформулировать, обсудить и решить те или иные проблемы на уровне доступном для комплексного изучения человеком.

3.2. Начальные проектные решения. Программная архитектура содержит сведения о том, какие решения были приняты на ранних этапах разработки системы. Значимость подобного рода сведений не ограничивается удельным весом этих этапов относительно оставшихся операций разработки и сопровождения. Именно в это время впервые появляется возможность анализа проектных решений, определяющих дальнейшую разработку системы.

3.3. Переносимая абстракция системы. Программная архитектура – это относительно небольшая вполне доступная для человеческого восприятия модель структурирования системы и взаимодействия ее компонентов. Помимо прочего эта модель обладает свойством переносимости из системы в систему. Например, ее можно применять в отношении других систем, для которых характерны примерно те же требования к атрибутам качества и функциональным возможностям, и тем самым дать начало полноценному многократному применению.

4. Как появляется архитектура.

Кто и что влияет на архитектуру

Любая архитектура является результатом принятия ряда экономических и технических решений [2]. Эти факторы играют свою роль в процессе проектирования, а их конкретная реализация обусловливается средой, в которой архитектура должна работать. В форме требований формулируются некоторые, но далеко не все желаемые свойства системы. С другой стороны, не все требования напрямую затрагивают эти свойства. Спецификация требований – это только начало. Существует масса других ограничений, которые необходимо учитывать при выборе варианта архитектуры программной системы.

4.1. Влияние на архитектуру оказывают заинтересованные в системе лица и организации. Они включают среди прочих, заказчика, конечных пользователей, разработчиков, руководителей проекта, специалистов по сопровождению и маркетингу. У разных заинтересованных лиц есть свои представления о свойствах системы. В частности, они высказывают пожелания относительно поведения системы при прогоне, производительности на тех или иных аппаратных средствах, настраиваемости, переносимости на другие платформы, быстрого выхода на рынок и низких затрат на разработку, высокой заработной платы программистов соответствующих специальностей и т. д.

Именно из этих свойств складывается архитектура. От этих ее свойств зависит, понравится ли система заинтересованным лицам.

4.2. Влияние на архитектуру оказывает компания-разработчик. Воздействие компании-разработчика не исчерпывается доработкой совместно с заказчиком требований, предъявляемых к системе. На архитектуру оказывает влияние структура и характер организации компании-разработчика.

4.3. Влияние на архитектуру оказывают опыт и привычки архитекторов. Если у архитектора есть опыт применения того или иного архитектурного решения, то, скорее всего, он будет его использовать в последующей работе. С другой стороны, если предыдущие попытки применения того или иного решения были неудачны, архитектора будет трудно убедить использовать это решение вновь.

4.4. Влияние на архитектуру оказывает техническая база. Подготовка и опыт архитектора, в частности проявляется в его работе с технической базой. Определенное воздействие на архитектуру оказывают существующие в настоящее время инструментальные и технические средства.

4.5. На архитектуру оказывает влияние ее окружение. Система размещается в некотором окружении, и это окружение оказывает влияние на архитектуру – это миссия бизнеса, которую будет поддерживать архитектура; заинтересованные в системе лица; внутренние стандарты организации и внешние технические ограничения.

5. Архитектурные образцы, эталонные модели и эталонные варианты архитектуры

Между простейшими "набросками" архитектур и комплексными архитектурами, укомплектованными всей необходимой информацией о системах, существуют многочисленные переходные этапы. Каждый из этих этапов представляет собой результат принятия ряда архитектурных решений и архитектурных альтернатив. Некоторые из них сами по себе имеют определенную ценность. В [2] отмечается три промежуточных этапа.

5.1. Архитектурный образец (шаблон) – это описание типов элементов и отношений и изложение ряда ограничений на их использование. Образец нужно рассматривать как совокупность ограничений, накладываемых на архитектуру. У образцов есть очень полезный аспект – они демонстрируют известные атрибуты качества. Именно поэтому архитекторы выбирают

образцы, исходя из определенных соображений, и во многих случаях выбор архитектурного образца является первым существенным решением архитектора.

5.2. Эталонная модель – это разделение между отдельными блоками функциональных возможностей и потоков данных. По сути, эталонной моделью является стандартная декомпозиция известной проблемы на части, которые взаимодействуя, способны ее разрешить.

5.3. Эталонная архитектура. Это эталонная модель, отображенная на программные элементы, которые сообща реализуют функциональность, определенную в эталонной модели, и потоки данных между ними. В то время как эталонная модель обеспечивает разделение функций, эталонная архитектура отображает эти функции на декомпозицию системы. Соответствие может быть как однозначным, так и неоднозначным.

6. Что определяет и на что влияет выбранная архитектура

Свойства программного проекта во многом определяются выбранным архитектурным образцом. Те образцы, которые в наибольшей степени подходят для решения конкретной задачи, повышают качество реализации конечного проектного решения. Так что же определяет и на что влияет выбранная архитектура программного обеспечения?

6.1. Архитектура определяет структуру. Структурные аспекты архитектуры проявляются многими способами. Структурный элемент может быть подсистемой, процессом, библиотекой, базой данных, вычислительным узлом, системой в традиционном смысле, готовым продуктом и т.д. Многие определения архитектуры признают не только сами структурные элементы, но и композиции из структурных элементов, их связи, любые соединительные звенья, необходимые для поддержки этих отношений, интерфейсы и разбиения.

6.2. Архитектура определяет поведение. Наряду с определением структурных элементов любая архитектура определяет взаимодействие между этими структурными элементами. Это такие взаимодействия, которые обеспечивают желаемое поведение системы.

6.3. Архитектура концентрируется на значимых элементах. Хотя архитектура определяет структуру и поведение, она занимается только такими элементами, которые оцениваются как значимые. Значимые элементы – это такие элементы, которые имеют продолжительное и устойчивое действие, например, главные структурные элементы, элементы, связанные с основным поведением, и элементы, которые определяют значимые свойства, такие как надежность и масштабируемость. Набор значимых элементов не является статичным, он может изменяться с течением времени. Однако относительная стабильность архитектуры является признаком хорошей архитектуры, хорошо отлаженного процесса разработки и хорошего разработчика.

6.4. Архитектура уравновешивает потребности заинтересованных лиц. Архитектура создается для удовлетворения комплекса потребностей заинтересованного лица. Однако часто невозможно выполнить все выраженные пожелания. Например, заинтересованное лицо может попросить, чтобы некоторая функциональность укладывалась в определенный временной промежуток, но эти две потребности (функциональность и промежуток времени) являются взаимоисключающими. Принятие компромиссных решений является необходимым аспектом процесса разработки.

6.5. Архитектура воплощает решение на основе логического обоснования. Важнейший аспект архитектуры – ее логическое обоснование. Важно обеспечить документирование решений, которые привели к созданию этой архитектуры, и логическое обоснование таких решений. Эта информация является значимой для многих заинтересованных лиц, особенно для тех, кто должен обслуживать систему.

6.6. Архитектура может соответствовать некоторому архитектурному стилю. Архитектурный стиль определяет семейство систем в терминах шаблона организации структуры. Большинство архитектур построено на основе систем, которые используют сходные наборы интересов. Сходство может быть определено как архитектурный стиль, который можно рассматривать как особый вид шаблона.

6.7. Архитектура определяет ограничения реализации. Реализация отражает архитектуру только в том случае, если она соответствует изложенным в ней проектным решениям. Таким образом, реализацию следует разделить на ряд установленных архитектурой элементов. Взаимодействие между этими элементами должно производиться в установленном архитектурой порядке. Решения о распределении ресурсов также накладывают на реализацию определенные ограничения.

6.8. Архитектура способствует или сдерживает реализацию атрибутов качества системы. Способность или неспособность системы реализовать предполагаемые атрибуты качества в значительной степени обусловлена архитектурой. При этом важно понимать, что сама по себе архитектура не гарантирует ни функциональности, ни качества. На качество системы влияют все решения. Можно ли, не дожидаясь окончания разработки и размещения системы, утверждать, что те или иные архитектурные решения приняты верно? Будь ответ на этот вопрос отрицательным, задача выбора архитектуры потеряла бы всякий смысл – ее можно было бы выбрать произвольно!

6.9. Архитектура облегчает анализ изменений и их организацию. В продолжение своего жизненного цикла программные системы претерпевают изменения – локальные, нелокальные и архитектурные. Для внесения локальных изменений достаточно откорректировать соответствующий элемент. Нелокальное изменение требует корректировки нескольких элементов. Однако базовые архитектурные принципы при этом остаются неизменными. Для принятия решений о необходимости внесения изменений, оценки последствий изменений и расстановки приоритетов относительно требуемых изменений необходим анализ взаимоотношений, производительности и поведения программных элементов системы. Все эти задачи должны быть прописаны в должностной инструкции архитектора.

6.10. Архитектура облегчает эволюционное макетирование. Любую архитектуру можно проанализировать и смоделировать в качестве макета. Процесс разработки от этого выигрывает в двух отношениях. Система становится исполняемой на раннем этапе жизненного цикла продукта. Ее точность повышается по мере замены элементов макета полноценными версиями программного обеспечения. Из того, что система довольно рано становится исполняемой, следует, что потенциальные проблемы, связанные с производительностью, выявляются на ранних этапах ее жизненного цикла. Каждый из этих факторов снижает риск разработки.

6.11. Архитектура позволяет более точно рассчитать стоимость и сроки создания системы. Рассчитав стоимость и составив график, руководитель может, во-первых, получить необходимые ресурсы, а во-вторых, узнать, удовлетворительно ли состояние проекта. Расчеты стоимости на основе блоков системы оказываются точнее тех, которые базируются на общих знаниях о системе.

6.12. Архитектура оказывает влияние на окружение. Создание архитектуры изменяет окружение с технологической точки зрения – может также изменить среду в терминах навыков, доступных в пределах организации. Результирующая система представляет собой сочетание программного и аппаратного обеспечения, информационных ресурсов и сотрудников. Системное проектирование особенно заинтересовано в том, чтобы рассматривать программное, аппаратное обеспечение и людей как равноценные понятия, избегая, таким образом, неверных заключений, в которых, например, аппаратные устройства рассматриваются как элементы второго сорта по сравнению с программными или наоборот.

6.13. Архитектура оказывает влияние на структуру и задачи компании-разработчика. Архитектура обусловливает структуру системы, в частности набор блоков программ, которые надлежит реализовать, а потом интегрировать в рамках системы. Группы разработчиков укомплектовываются именно по блокам; операции в рамках процессов разработки, тестирования и интеграции также выполняются в отношении блоков. Согласно

графикам и бюджетам ресурсы выделяются частями на отдельные блоки. Если компания наработала опыт конструирования семейств сходных систем, она будет вкладывать средства в повышение профессионального уровня разработчиков. Следовательно, группы встраиваются в структуру организации.

Архитектура способна оказывать воздействие на задачи компании-разработчика. Сконструированная на основе определенной архитектуры успешная система предоставляет компании возможность укрепиться в данном сегменте рынка.

6.14. Архитектура помогает в процессе обучения. Архитектура, содержащая описание взаимодействия элементов в рамках требуемого поведения, может послужить своеобразной инструкцией, вводящей новых участников проекта в курс дела. Этим дополнительно подтверждается утверждение о том, что одной из основных функций программной архитектуры является организация и содействие общению между представителями различных заинтересованных групп.

6.15. Архитектура представлена в каждой системе. Каждая система имеет архитектуру, даже если эта архитектура формально не документирована или система слишком проста. Документирование архитектуры представляет собой очень ценное средство. Документированные архитектуры имеют тенденцию быть более продуманными а следовательно, более эффективными, чем недокументированные, поскольку процесс записи архитектуры естественным образом ведет к всестороннему обдумыванию.

7. Архитектурные структуры и представления

Рассматривая представления архитектуры, часто употребляют связанные между собой понятия структуры (structure) и представления (view). Представление – это отображение ряда связанных архитектурных элементов в том виде, в котором ими оперируют заинтересованные в системе лица. В нем фиксируется отображение совокупности элементов и установленных между ними связей. Структура же – это собственно ряд элементов, существующих в раках программного или аппаратного обеспечения. В частности модульная структура представляет собой набор модулей системы с указанием их организации. Модульное представление есть отображение этой структуры, документируемое и применяемое теми или иными заинтересованными лицами.

По классификации, предложенной в [2], архитектурные структуры подразделяются на три общие группы, в каждую из которых включаются элементы определенного характера.

Модульные структуры. Элементами таких структур являются модули – блоки реализации. Модули предполагают рассмотрение системы с точки зрения программного кода. Модульные структуры позволяют отвечать на такие вопросы, как: “Какие функциональные требования выполняет данный модуль? К каким программным элементам он может обращаться? Какое программное обеспечение он фактически использует? Между какими модулями установлены отношения обобщения или специализации (например, наследования)?

Структуры “компонент и соединитель”. В данном случае элементами являются компоненты и соединители периода прогона (выполнения). Среди вопросов, на которые отвечают структуры “компонент и соединитель”, такие, например, как: “Каковы основные исполнительные компоненты и как происходит их взаимодействие? Каковы основные совместно используемые хранилища данных? Какие части системы воспроизводятся? Каким образом по системе проходят данные?

Структуры распределения. Структуры распределения демонстрируют связь между программными элементами, с одной стороны, и элементами одной или нескольких внешних сред, в которых данное программное обеспечение создается и исполняется, – с другой. Они отвечают на вопросы: “На каком процессоре исполняется данный программный элемент? В каких файлах каждый элемент хранится в ходе разработки, тестирования и конструирования системы? Каким образом программные элементы распределяются между группами разработчиков?”

Перечисленные три структуры соответствуют трем универсальным типам решений, применяемым в ходе архитектурного проектирования:

a) как следует структурировать совокупность блоков кода (модулей) системы?

b) каким образом следует структурировать совокупность элементов системы, обладающих поведением (компоненты) и демонстрирующих взаимодействие (соединители) в период прогона?

c) как установить связи между системой и непрограммными структурами среды (например, с процессорами, файловыми системами, сетями, группами разработчиков и т. д.)?

Наиболее распространенные и полезные программные структуры можно разделить на три группы: модульные, компонент и соединитель и распределение [2].

Модульные структуры делятся на следующие разновидности.

7.1. Декомпозиция. В качестве блоков выступают модули, между которыми установлены отношения “является подмодулем...”. Таким образом, крупные модули в рекурсивном порядке разлагаются на меньшие, и этот процесс завершается только тогда, когда меньшие модули становятся вполне понятными. Структура декомпозиции в значительной степени обеспечивает модифицируемость системы – при этом складывается ситуация, когда наиболее вероятные изменения приходятся на долю нескольких небольших модулей.

7.2. Варианты использования. Блоками этой важной структуры могут быть либо модули, либо процедуры или ресурсы интерфейсов модулей. Между такими блоками устанавливаются отношения использования. Структура использования полезна при конструировании систем, которые легко расширяются дополнительными функциями, либо предлагают возможность быстрого извлечения полезных функциональных подмножеств.

7.3. Многослойные. Если отношения использования в рамках этой структуры находятся под строгим контролем, возникает система слоев – внутренне связанных наборов родственных функций. В рамках строгой многослойной структуры уровень N может обращаться к услугам только в том случае, если они представлены слоем N-1. На практике это правило существует в виде многочисленных вариантов, которые частично снимают приведенное структурное ограничение. Слои во многих случаях проектируются в виде абстракций (виртуальных машин), которые, стараясь обеспечить переносимость, скрывают детали своей реализации нижележащих слоев от вышележащих.

7.4. Класс или обобщение. Блоки модулей в рамках этой структуры называются классами. Отношения между ними строятся по образцам “наследуют от...” и “являются экземпляром...”. Данное представление способствует анализу коллекций сходного поведения или сходных возможностей (например, классов, которые наследуют от других классов) и параметрических различий, фиксация которых производится путем определения подклассов. Структура классов позволяет анализировать вопросы повторного использования и инкрементного введения функциональности.

7.5. Структуры “компонент и соединитель”. Среди структур данного вида выделяются следующие структуры.

a) Процесс или сообщающиеся процессы. Подобно другим структурам “компонент и соединитель”, эта является ортогональной по отношению к модульным структурам, поскольку она связана с динамическими аспектами исполняемой системы. В качестве блоков в данном случае выступают процессы или потоки, связь между которыми устанавливается путем передачи данных, синхронизации и/или операций исключения.

b) Параллелизм. Данная структура позволяет архитекторам выявлять перспективы параллелизма и локализовать возможности состязаний за ресурсы. В качестве блоков выступают компоненты, а соединители играют роль “логических потоков”. Логическим потоком называется такая последовательность вычислений, которую впоследствии, в ходе процесса проектирования, можно связать с отдельным физическим потоком. Структура параллелизма задействуется на ранних этапах проектирования и способствует выявлению требований к организации параллельного исполнения.

c) Совместно используемые данные или репозиторий. В состав данной структуры входят компоненты и соединители, обеспечивающие создание, хранение и обращение к данным

постоянного хранения. Она наилучшим образом приспособлена к таким ситуациям, когда система структурирована на основе одного или нескольких репозиториев совместно используемых данных.

d) Клиент-сервер. Эта структура предназначена для систем, сконструированных в виде группы взаимодействующих клиентов и серверов. В качестве компонентов в данном случае выступают клиенты и серверы, а соединителями являются протоколы и сообщения, которыми они обмениваются в процессе обеспечения работоспособности системы. Такая структура требуется для разделения задач, физического распределения и выравнивания нагрузок.

7.6. Структуры "распределения". Среди этих структур можно выделить следующие структуры.

a) Размещение. Структура размещение отражает распределение программной системы между элементами аппаратной обработки (компьютерами) и передачи данных. Отношения устанавливаются по распределению и демонстрируют физические устройства, на которых размещаются программные элементы. Возможны также отношения миграции в случае динамического распределения, например, в системах виртуальных машин. Настоящее представление позволяет инженерам анализировать производительность, целостность данных, готовность и безопасность. Все эти характеристики чрезвычайно важны в условиях распределенных и параллельных систем.

b) Реализация. Данная структура демонстрирует отображение программных элементов (обычно модулей) на файловую структуру (структуры) в условиях разработки системы, интеграции и управления конфигурациями. Это крайне важно в контексте разработки и процессов конструирования.

c) Распределение функций. Данная структура обеспечивает распределение обязанностей по реализации и интеграции модулей между соответствующими группами разработчиков. Наличие в составе архитектуры структуры распределения функций делает очевидным, что при принятии соответствующих решений учитывались как архитектурные, так и организационные факторы. Архитектору должно быть известно, какие именно навыки требуются от разных групп разработчиков.

8. Отношения между структурами

Из рассмотренного определения архитектуры и архитектурных структур становится понятным, что все программные системы состоят из множества структур. Как правило, структура системы анализируется с точки зрения ее функциональности. При этом часто забывают о других ее свойствах: физическом распределении, взаимодействии процессов и синхронизации и др. Все эти свойства обязательно должны учитываться на уровне архитектуры. Каждая структура содержит метод анализа тех или иных атрибутов качества. К примеру, чтобы создать легко расширяемую или сокращаемую систему, необходимо сконструировать структуру использования. Структура процессов конструируется с целью исключения взаимоблокировки и расширения "узких мест". Структура декомпозиции модулей конструируется в расчете на производство модифицируемых систем и т. д.

Все перечисленные структуры предполагают различные представления и подходы к проектированию программной системы. Элементы одной структуры связаны с элементами прочих, и на эти отношения следует обратить внимание. В частности, модуль в структуре декомпозиции может декларироваться как отдельный компонент, как часть отдельного компонента или несколько компонентов в рамках одной из структур "компонент и соединитель", отражая, таким образом, своего представителя периода прогона.

В тоже время каждая рассмотренная структура снабжает архитектора оригинальным представлением системы и дополнительной базовой точкой проектирования. Заметим, что современные CASE-средства проектирования архитектуры программных систем, например, Rational Software Architect реализуют именно подобный подход при разработке программных систем, позволяя представлять архитектуру системы множеством взаимосвязанных моделей [3].

Структуры – это основные базовые точки конструирования архитектуры. Отдельные структуры отвечают за те или иные атрибуты качества. Они выражают принцип разделения задач при создании архитектуры, а также при ее последующем анализе и изложении для заинтересованных лиц.

8. Заключение

Архитектура инфокоммуникационного предприятия предлагает развернутую и точную модель системы. Методика формирования программной архитектуры такого предприятия предполагают детальный анализ системы перед ее реализацией. Такие методики, как Attribute Driven Design (ADD) [1], гарантируют, что программное обеспечение, реализованное на основе предварительно сформированной архитектуры, будет точно отвечать своему предназначению. Есть немало примеров использования архитектуры в стратегических целях.

Один из самых известных примеров – Common Object Request Broker Architecture (CORBA). Эта архитектура служит для связи унаследованных систем, для интеграции систем, написанных на разных языках, и поддержки взаимодействия между компьютерами с различными аппаратными архитектурами. CORBA позволяет дать новую жизнь унаследованным системам и в то же время быстро интегрировать в них новые приложения, что обеспечивает предприятиям стратегические преимущества перед конкурентами, модифицирующими унаследованные системы. Еще один пример – свободно распространяемая интегрированная среда разработки Eclipse [1].

В целом процесс проектирования архитектуры состоит из систематической декомпозиции элементов верхнего уровня на совокупности более мелких элементов. С помощью подхода ADD архитектор выбирает конкретную декомпозицию, стремясь улучшить определенные свойства конечного продукта.

Процесс создания архитектуры предполагает разработку системы с конкретными свойствами и функциональностью, причем каждое из таких свойств имеет заданный приоритет. Начиная с общей структуры, которая поддерживает всю требуемую функциональность, архитектор методично проводит декомпозицию функциональности и распределяет ее между компонентами. Суть процесса проектирования заключается в декомпозиции системы независимо от способа ее представления: модульный, объектный или компонентный.

Литература

1. Назаров С.В. Архитектура и проектирование программных систем: монография / С.В.Назаров. – 2-е изд., перераб. и доп. – М.: ИНФРА-М, 2016. – 374 с. – (Научная мысль) – www.dx.doi.org/ 10.12737/18292.
2. Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. 2-е изд. – СПб.: Питер, 2006. – 575 с.
3. MDD. Общий обзор и концепция разработки, управляемой моделями. [Электронный ре-сурс]: http://www.ibm.com/developerworks/ru/library/mdd/ch1/ch1.html-#author1