

Что такое архитектура программного обеспечения?

Питер Илес (Peter Eeles)

15.02.2006

старший разработчик архитектуры информационных технологий
IBM

Из журнала The Rational Edge: данное введение в относительно новую дисциплину Архитектура программного обеспечения - первая из четырех статей цикла, посвященного "разработке архитектуры" в целом. Автор начинает с определения основных понятий дисциплины и далее анализирует вклад, который вносит хорошо разработанная архитектура в окружение, на котором она разворачивается.



Ни у кого не вызывает сомнений, что наш мир все более

зависит от программного обеспечения. Программы - это основной элемент повсеместно используемых сотовых телефонов, а также комплексных систем управления воздушным движением. По сути, многих новшеств, которые мы теперь воспринимаем как данность -- в том числе таких организаций, как eBay или Amazon -- просто не существовало бы, если бы они не были построены на программном обеспечении. Даже традиционные организации в финансовом, общественном и торговом секторах в значительной мере зависят от программного обеспечения. На сегодняшний день трудно найти организацию, которая хоть каким-либо образом не была вовлечена в сферу программного обеспечения.

Чтобы такие новшества и организации продолжали существовать, программное обеспечение, на котором они строят свою деятельность, должно обеспечивать необходимую производительность, иметь хорошее качество, быть доступным при необходимости и доставляться по приемлемой цене.

На все эти параметры оказывает влияние архитектура программного обеспечения, которая является темой данной статьи. В статье рассматриваются "преимущественно-программные системы", которые IEEE определяет следующим образом:

Преимущественно-программная система - это любая система, в которой программное обеспечение оказывает значительное влияние на проект, конструкцию, развертывание и развитие всей системы. [из IEEE 1471. См. далее раздел "Архитектура определяет".]

В данной статье термин "архитектура", если нет особых пояснений, - синоним термина "архитектура программного обеспечения". Хотя в статье основной упор делается на преимущественно-программных системах, важно иметь в виду, что такая система не может работать без аппаратного обеспечения, поэтому некоторые характеристики, такие как надежность или производительность, достигаются сочетанием программного и аппаратного обеспечения. Поэтому в общем решении нельзя игнорировать аппаратный аспект. Более подробно это рассматривается далее в данной статье.

Архитектура определяет

Когда речь заходит об "архитектуре", обычно не возникает недостатка в определениях. Есть даже Web-сайты, которые собирают такие определения.¹ В данной статье мы используем определение стандарта IEEE Std 1472000, и IEEE 1471 Рекомендуемые методы описания архитектуры преимущественно-программных систем IEEE.² Вот это определение, в котором жирным шрифтом выделены основные характеристики.

Архитектура - это базовая **организация системы**, воплощенная в ее **компонентах**, их **отношениях** между собой и с **окружением**, а также принципы, определяющие проектирование и развитие системы. [IEEE 1471]

В этом стандарте также определяются следующие термины, связанные с данным определением:

Система - это набор компонентов, объединенных для выполнения определенной функции или набора функций. Термин "система" охватывает отдельные приложения, системы в традиционном смысле, подсистемы, системы систем, линейки продуктов, семейства продуктов, целые корпорации и другие агрегации, имеющие отношение к данной теме. Система существует для выполнения одной или более **миссий** в своем окружении. [IEEE 1471]

Окружение, или контекст, определяет ход и обстоятельства экономических, эксплуатационных, политических и других влияний на систему. [IEEE 1471]

Миссия - это применение или действие, для которого одно или несколько **заинтересованных лиц** планируют использовать систему в соответствии с некоторым набором условий. [IEEE 1471]

Заинтересованное лицо - это физическое лицо, группа или организация (или ее категории), которые заинтересованы в системе или имеют связанные с ней задачи. [IEEE 1471]

Мы видим, что в определениях часто употребляется термин "компонент". Тем не менее, большая часть определений архитектуры не определяет термина "компонент", и IEEE 1471 - не исключение, поскольку намеренно оставляет это понятие неопределенным, чтобы оно соответствовало множеству толкований, возможных в отрасли. Компонент может быть логическим или физическим, технологически-независимым или технологически-связанным, крупно- или мелкогранулированным. В этой статье я использую определение компонента по спецификации UML 2.0 в достаточно широком смысле, что позволяет охватить различные элементы архитектуры, которые могут нам встретиться, в том числе объекты, технологические компоненты (такие как корпоративные компоненты JavaBean), сервисы, программные модули, традиционные системы, архивы приложений и т. д. Вот определение термина "компонент" по спецификации UML 2.0:

[Компонентом называется] модульная часть системы, которая инкапсулирует ее содержимое; воплощение компонента является замещаемым в его окружении. Поведение компонента определяется в терминах предоставляемого и требуемого интерфейсов. Таким образом, компонент используется в качестве типа, соответствие которого описывается этими двумя интерфейсами, предоставляемым и требуемым (объединяя как статическую, так и динамическую их семантику).³

Приведенные здесь определения охватывают множество различных понятий, которые подробно обсуждаются далее в этой статье. Хотя в отрасли не существует общепризнанного определения понятия "архитектура", стоит рассмотреть некоторые другие определения, чтобы можно было обратить внимание на сходство между ними. Рассмотрим следующие определения, в которых, как и раньше, выделены жирным шрифтом основные характеристики.

Архитектура - это **набор значимых решений** по поводу организации системы программного обеспечения, набор **структурных элементов** и их интерфейсов, при помощи которых компонуется система, вместе с их **поведением**, определяемым во взаимодействии между этими элементами, **компоновка** элементов в постепенно укрупняющиеся подсистемы, а также **стиль архитектуры** который направляет эту организацию -- элементы и их интерфейсы, взаимодействия и компоновку. [Крачтен (Kruchten)]⁴

Архитектура программы или компьютерной системы - это структура или структуры системы, которые включают элементы программы, видимые извне свойства этих **элементов** и **связи** между ними. [Басс (Bass) и др.]⁵

[Архитектура - это] структура организации и связанное с ней поведение системы. Архитектуру можно рекурсивно разобрать на части, взаимодействующие посредством интерфейсов, связи, которые соединяют части, и условия сборки частей. Части, которые взаимодействуют через интерфейсы, включают классы, компоненты и подсистемы. [UML 1.5]⁶

Архитектура программного обеспечения системы или набора систем состоит из всех важных проектных решений по поводу структур программы и взаимодействий между этими структурами, которые составляют системы. Проектные решения обеспечивают желаемый набор свойств, которые должна поддерживать система, чтобы быть успешной. Проектные решения предоставляют концептуальную основу для разработки системы, ее поддержки и обслуживания. [Мак-Говерн (McGovern)]⁷

Хотя определения несколько отличаются, мы можем видеть немалую степень сходства. Например, большинство определений указывают на то, что архитектура связана со структурой и поведением, а также только со значимыми решениями, может соответствовать некоторому архитектурному стилю, на нее влияют заинтересованные в ней лица и ее окружение, она воплощает решения на основе логического обоснования. Эти и другие темы рассматриваются ниже.

Архитектура определяет структуру

Если вы попросите описать "архитектуру", то девять человек из десяти обязательно упомянут о структуре. Это понятие в английском языке часто связывается со строительством зданий или некоторых других инженерных сооружений (engineering structure), например, мостов. Хотя существуют и другие характеристики этих элементов, такие как поведение, соответствие цели и даже эстетика, но именно термин "структура (structure)" наиболее известен и чаще всего упоминается.

Не удивляйтесь тому, что если вы попросите какого-либо человека описать архитектуру программного обеспечения, с которым он работает, то он, скорее всего, покажет схему, на которой будут изображены структурные аспекты системы - будь то архитектурные уровни, компоненты или распределенные узлы. Действительно, структура является важнейшей характеристикой архитектуры. Структурные аспекты архитектуры проявляются многими способами, и в результате большинство определений архитектуры сознательно оставляют неопределенными. Структурный элемент может быть подсистемой, процессом, библиотекой, базой данных, вычислительным узлом, системой в традиционном смысле, готовым продуктом и так далее.

Многие определения архитектуры признают также не только сами структурные элементы, но и композиции из структурных элементов, их связи (и любые соединительные звенья, необходимые для поддержки этих отношений), интерфейсы и разбиение. И вновь, каждый из этих элементов может быть представлен разными способами. Например, соединительное звено может представлять собой сокет, быть синхронным или асинхронным, быть связанным с конкретным протоколом и так далее.

Пример некоторых структурных элементов показан на рисунке 1. На рисунке изображена диаграмма классов UML, содержащая некоторые структурные элементы, которые представляют систему обработки заказов. Мы видим три класса -- OrderEntry, CustomerManagement и AccountManagement. Класс OrderEntry зависит от класса CustomerManagement и от класса AccountManagement.

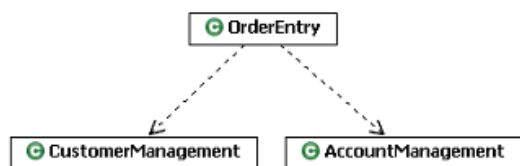


Рисунок 1: диаграмма класса UML class, демонстрирующая структурные элементы

Архитектура определяет поведение

Наряду с определением структурных элементов любая архитектура определяет взаимодействия между этими структурными элементами. Это такие взаимодействия, которые обеспечивают желаемое поведение системы. На рисунке 2 представлена диаграмма сценария UML, которая показывает несколько взаимодействий, которые в сумме позволяют системе поддерживать создание заказа в системе обработки заказов. Мы видим здесь пять взаимодействий. Сначала, деятель Sales Clerk создает заказ при помощи экземпляра класса OrderEntry. Экземпляр класса OrderEntry получает сведения о клиенте при помощи экземпляра класса CustomerManagement. Затем экземпляр класса OrderEntry использует экземпляр класса AccountManagement для того, чтобы создать заказ, внести в него элементы заказа, а затем разместить заказ.

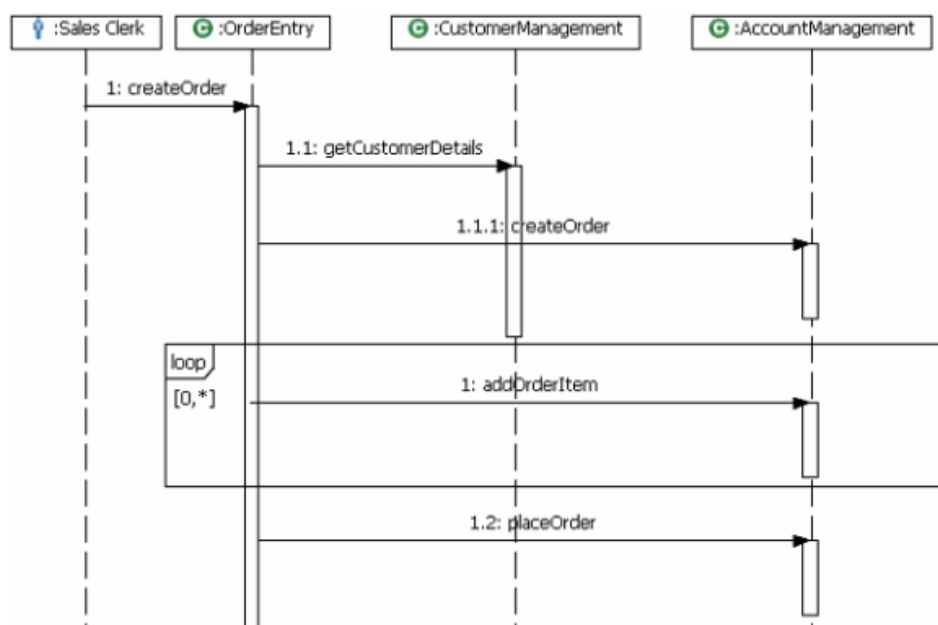


Рисунок 2: диаграмма сценария UML, показывающая элементы поведения

Следует отметить, что рисунок два согласуется с рисунком 1 так, что мы можем извлечь зависимости, показанные на рисунке 1, из взаимодействий, определенных на рисунке 2. Например, экземпляр класса OrderEntry в процессе выполнения зависит от экземпляра класса CustomerManagement, как показывают взаимодействия на рисунке 2. Эта

зависимость отражается в отношении зависимости между соответствующими классами OrderEntry и CustomerManagement, как показано на рисунке 1.

Архитектура концентрируется на значимых элементах

Хотя архитектура определяет структуру и поведение, она определяет не *все* в структуре и не *все* в поведении. Она занимается только такими элементами, которые оцениваются как значимые. Значимые элементы - это 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. Как правило, архитектура не имеет отношения к гранулярным деталям этих элементов. Архитектурную значимость можно также назвать экономической значимостью, поскольку главный признак, по которому некоторые элементы оцениваются выше остальных - это стоимость создания и стоимость изменения.

Поскольку архитектура фокусируется только на значимых элементах, она предлагает нам конкретную перспективу оцениваемой системы - перспективу, которая наиболее значима для разработчика архитектуры.⁸ В этом смысле архитектура - это некоторое обобщение системы, помогающее разработчику архитектуры управлять сложностью.

Стоит также отметить, что набор значимых элементов не является статичным и может измениться с течением времени. Он может измениться при уточнении результата требований, идентификации рисков, создании исполняемой программы. Однако относительная стабильность архитектуры несмотря на изменения, в некоторой степени является признаком хорошей архитектуры, хорошо отлаженного процесса разработки и хорошего разработчика. Если архитектура требует постоянного пересмотра при относительно небольших изменениях, это плохой признак. Тем не менее, если архитектура относительно стабильна, это утверждение справедливо.

Архитектура уравнивает потребности заинтересованных лиц

В конечном итоге 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. Можно либо уменьшить границы функции, чтобы она соответствовала расписанию, либо предоставить полную функциональность, но за более продолжительный отрезок времени. Аналогично, различные заинтересованные лица могут иметь противоречивые потребности и здесь должно быть достигнуто определенное равновесие. Поэтому принятие компромиссных решений является необходимым аспектом процесса разработки архитектуры, а преодоление трудностей - неотъемлемой чертой разработчика.

Просто для того, чтобы получить представление о близкой задаче, рассмотрим следующие потребности нескольких заинтересованных лиц:

- Конечный пользователь заинтересован в интуитивно понятном и корректном поведении, производительности, надежности, удобстве использования, доступности и безопасности;
- Системный администратор заинтересован в интуитивно понятном поведении, управлении и инструментах мониторинга;
- Специалист по маркетингу заинтересован в конкурентноспособных функциях, времени до выхода программы, позиционировании среди других продуктов и в стоимости;
- Клиент заинтересован в цене, стабильности и возможности планировать;
- Разработчик заинтересован в понятных требованиях и простом и непротиворечивом принципе проектирования;
- Руководитель проекта заинтересован в предсказуемости хода проектирования, планировании, продуктивном использовании ресурсов и бюджета;
- Специалист по сопровождению заинтересован в понятном, непротиворечивом и документируемом принципе проекта, а также в легкости, с которой можно вносить изменения.

Как видно из списка, еще одна проблема разработчика - это то, что заинтересованные лица заинтересованы не только в том, чтобы система обеспечивала необходимую функциональность. Многие пункты из списка интересов являются нефункциональными по характеру, так как они не влияют на функциональность системы (например, интерес по отношению к цене и планированию). Тем не менее, такие интересы формулируют свойства или ограничения системы. Нефункциональные требования очень часто являются самыми значимыми требованиями, поскольку в них заинтересован разработчик архитектуры.

Архитектура воплощает решения на основе логического обоснования

Важный аспект архитектуры - это не только конечный результат, то есть сама архитектура, но и ее логическое обоснование. Таким образом, важно обеспечить документирование решений, которые привели к созданию этой архитектуры, и логические обоснования таких решений.

Эта информация является значимой для многих заинтересованных лиц, особенно для тех, кто должен обслуживать систему. Она часто имеет ценность для разработчика архитектуры, когда ему нужно пересмотреть логические обоснования принятых решений, чтобы избежать ненужного повторения своих действий. Например, эта информация используется при пересмотре архитектуры, а разработчику нужно объяснить принятые ранее решения.

Архитектура может соответствовать некоторому архитектурному стилю

Большинство архитектур построены на основе систем, которые используют сходные наборы интересов. Сходство может быть определено как архитектурный стиль, который можно рассматривать как особый вид шаблона, хотя этот шаблон часто является сложным и составным (когда одновременно применяются несколько шаблонов). Как и шаблон,

архитектурный стиль представляет собой кодификацию опыта, и для разработчиков архитектур было бы неплохо ждать случая, чтобы снова использовать этот опыт. Примеры архитектурных стилей включают распределенный стиль, стиль "каналы и фильтры", стиль с централизованной обработкой данных, стиль, построенный на правилах и так далее. Конкретная система может демонстрировать более одного архитектурного стиля. Вот как описывают архитектурный стиль Шоу и Гарлан (Show and Garlan):

*[Архитектурный стиль] определяет семейство систем в терминах шаблона организации структуры. Точнее, архитектурный стиль определяет номенклатуру компонентов и типов соединительных звеньев, а также набор условий, в соответствии с которыми они могут соединяться.*⁹

Еще одно определение в терминах UML:

*[Шаблон] - это общее решение общей проблемы в данном контексте.*¹⁰

Кроме повторного использования опыта, применение архитектурного стиля (или шаблона) несколько облегчает жизнь разработчиков, поскольку стиль обычно документирован в терминах рационального обоснования его использования (следовательно, меньше придется обдумывать) и в терминах его структуры и поведения (следовательно, придется выработать меньше документации по архитектуре, поскольку вместо этого мы можем просто обратиться к стилю).

На архитектуру оказывает влияние ее окружение

Система размещается в некотором окружении, и это окружение оказывает влияние на архитектуру. Иногда это называют "архитектурой в контексте". В основном, окружение определяет границы, в которых должна работать система, а это, в свою очередь, влияет на архитектуру. Факторы окружения, оказывающие влияние на архитектуру - это миссия бизнеса, которую будет поддерживать архитектура, заинтересованные в системе лица, внутренние технические ограничения (например, требование соответствовать стандартам организации) и внешние технические ограничения (такие как необходимость взаимодействовать с внешней системой или соответствовать внешним регулятивным нормам).

И наоборот, как это ярко описано у Басса, Клементса и Кацмана (Bass, Clements и Kazman),¹¹ архитектура тоже оказывает влияние на свое окружение. Создание архитектуры изменяет окружение не только с технологической точки зрения - оно может, например, приносить в организацию многократно используемые активы - создание архитектуры может также изменить среду в терминах навыков, доступных в пределах организации.

Когда речь заходит о преимущественно-программных системах, то следует учитывать конкретные аспекты окружения, как было рассмотрено ранее в этой статье. Чтобы программа приносила пользу, она должна работать. Для того чтобы программа работала, ее запускают на некотором аппаратном обеспечении. Поэтому результирующая система представляет собой сочетание программного и аппаратного обеспечения, и именно это

сочетание позволяет добиться таких характеристик, как надежность и производительность. Программное обеспечение не может достичь этих характеристик без аппаратного обеспечения, на котором оно выполняется.

Стандарт IEEE Std 12207-1995, IEEE Standard for Information Technology -- Software Life Cycle Processes (Стандарт IEEE по информационным технологиям -- Процессы жизненного цикла приложения) определяет систему иначе, чем ранее упомянутое определение стандарта IEE 1471 (которое концентрируется на преимущественно-программных системах), но при этом согласуется с определением системы в области системного проектирования:

[Системой называется] интегрированный комплекс, состоящий из одного или более процессов, аппаратных устройств, программ, средств и людей, предоставляющий возможность удовлетворить данную потребность или условие. [IEEE 12207]¹²

Техническое описание "A configuration of the Rational Unified Process® for Systems Engineering (RUP SE)" содержит похожее определение.

[Системой называется] набор ресурсов, предоставляющий сервисы, которые используются предприятием для выполнения цели или миссии бизнеса. Компоненты системы обычно состоят из аппаратного обеспечения, программного обеспечения, информационных ресурсов и сотрудников¹³.

В области системного проектирования компромисс достигается за счет использования программного обеспечения, аппаратного обеспечения и людей. Например, если ключевым фактором является производительность, то принимается решение реализовать определенные элементы системы в виде аппаратных устройств, а не программ или людей. Еще один пример: чтобы предоставить удобную для покупателей систему, принимается решение предоставить клиентский интерфейс в виде человека, а не программы или оборудования. Более сложные сценарии требуют добиться определенных характеристик системы через сочетание программного, аппаратного обеспечения и человека. (В соответствии с этим, в данном цикле статей в некоторых случаях даются ссылки на элементы, не являющиеся программным обеспечением.)

Интересно отметить, что системное проектирование особенно заинтересованно в том, чтобы рассматривать программное и аппаратное обеспечение (а также людей) как равноценные понятия, избегая таким образом неверных заключений, в которых аппаратные устройства рассматриваются как элементы второго сорта по сравнению с программами, или программное обеспечение рассматривается как второсортное по сравнению с аппаратными устройствами и являющееся простым проводником для обеспечения нужной функции этих устройств.

Архитектура оказывает влияние на структуру коллектива

Архитектура определяет когерентные группировки родственных элементов, которые адресуют данный набор интересов. Например, архитектура системы обработки заказов может иметь определенные группировки элементов для ввода заказов, ведения

счетов, работы с клиентами, выполнения заказа, интеграции с внешними системами, персистентности и безопасности.

Каждая из этих группировок может требовать различных наборов навыков. Поэтому имеет смысл выровнять структуры групп разработчиков программного обеспечения в соответствии с архитектурой после того, как она будет определена. Однако чаще встречается ситуация, когда первоначальная группа разработчиков оказывает влияние на архитектуру, а не наоборот. Это ошибка, которой следует избегать, поскольку в результате обычно получается архитектура, далекая от идеала. "Закон Конвея" утверждает, что "если у вас есть четыре группы, работающих над компилятором, то вы получите четырехпроходный компилятор". Практически мы часто непреднамеренно создаем архитектуры, которые являются отражением организации, создающей архитектуру.

Однако так же справедливо будет сказать, что это несколько идеализированное представление не всегда реально, поскольку, по причинам чисто прагматичным, реальная структура коллектива и доступные навыки представляют весьма ощутимое ограничение того, что могло бы быть, и разработчику следует это учитывать.

Архитектура представлена в каждой системе

Следует также отметить, что каждая система имеет архитектуру, даже если эта архитектура формально не документирована или система слишком проста, и, скажем, состоит из одного элемента. Обычно 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.

И наоборот, если архитектура не документируется, трудно (если не невозможно) доказать, что она соответствует утвержденным требованиям в терминах адресных характеристик, таких как удобство обслуживания, заимствование передового опыта и так далее. Архитектуры, которые не документировались, как, кажется, большинство из тех, что существуют на сегодняшний день, имеют тенденцию быть случайными, а не продуманными.

Архитектура имеет особую область применения

Существует много видов архитектуры, лучше всего известна архитектура, ассоциируемая со строительством зданий и других гражданских инженерных сооружений. Даже в области разработки программного обеспечения мы часто встречаемся с различными формами архитектуры. Например, помимо понятия *архитектура программного обеспечения* мы можем столкнуться с такими понятиями, как *корпоративная архитектура*, *системная архитектура*, *организационная архитектура*, *архитектура информации*, *архитектура аппаратного обеспечения*, *архитектура приложения*, *архитектура инфраструктуры* и так далее. Вы услышите также и другие термины, каждый из которых определяет особую область разработки архитектуры.

К сожалению, в отрасли не существует соглашения о значении каждого из этих терминов или их отношениях друг к другу, в результате чего одни и те же термины могут иметь разные

значения (омонимы), а два или более терминов могут обозначать одно и то же (синонимы). Однако об области применения некоторых из этих терминов можно сделать вывод на основании рисунка 3. Если вы внимательно изучите рисунок и описание, то определенно найдете элементы, с которыми вы не согласны или элементы, которые в вашей организации используются по-другому. Но в этом и смысл -- показать, что термины используются в отрасли, но по поводу их значений нет согласия.

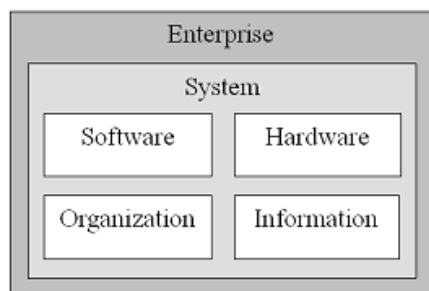


Рисунок 3: области применения различных терминов

Элементы, показанные на рисунке 3:

- Архитектура программного обеспечения (Software), главная тема данной статьи, как было определено выше;
- Архитектура аппаратного обеспечения (Hardware), которое включает такие элементы, как ЦПУ, память, жесткие диски, периферийные устройства, например, принтер, а также элементы, используемые для их соединения;
- Архитектура организации (Organization), которая включает элементы, имеющие отношение к бизнес-процессам, структурам организации, ролям и ответственности, а также основные области специализации организации;
- Структура информации (Information), включающая структуры, которые упорядочивают информацию;
- Архитектура программного обеспечения, архитектура аппаратного обеспечения, архитектура организации и архитектура информации, которые являются подмножеством целостной архитектуры системы (System), рассматривались ранее в данной статье;
- Архитектура корпорации (Enterprise), которая похожа на архитектуру системы тем, что тоже включает элементы, а именно: аппаратное и программное обеспечение и людей. Однако архитектура корпорации более сильно связана с бизнесом, так как она концентрируется на достижении бизнес-целей и занимается такими объектами, как быстрое реагирование на возникающие ситуации и эффективность организации. Архитектура корпорации может выходить за границы компании.

Как и следовало ожидать, существуют корреспондирующие формы разработчик архитектуры (например, разработчик архитектуры программного обеспечения, разработчик архитектуры аппаратного обеспечения и так далее) и разработка архитектуры (например, разработка архитектуры программного обеспечения, разработка архитектуры аппаратного обеспечения и так далее).

И сейчас, когда мы рассмотрели эти определения, многие вопросы остались без ответа. В чем заключается разница между архитектурой корпорации и архитектурой системы? Является ли корпорация системой? Архитектура информации и архитектура данных, которая встречается в некоторых преимущественно-информационных программах - это одно и то же? К сожалению, на эти вопросы нет согласованных ответов.

На сегодня вам следует просто знать, что эти разные термины существуют, но в отрасли нет непротиворечивых определений этих терминов и их взаимосвязей. Следовательно, можно рекомендовать выбирать термины, подходящие для вашей организации, и соответствующим образом их определять. Тогда вы, по крайней мере, добьетесь некоторой непротиворечивости и уменьшите вероятность недопонимания.

Заключение

В этой статье делается упор на определении основных характеристик архитектуры программного обеспечения. Однако на многие вопросы мы не ответили. В чем заключается роль разработчика архитектуры программного обеспечения? Какие основные действия должен выполнять разработчик архитектуры? В чем преимущества "разработки архитектуры"? На эти и другие вопросы мы ответим в следующих статьях этого цикла.

Благодарности

Содержание этой статьи было взято из книги, которая скоро должна выйти в свет и имеет рабочее название "The Process of Software Architecting" (Процесс разработки архитектуры программного обеспечения). В результате изложенное было прокомментировано многими людьми, которых я хочу поблагодарить, – это Греди Буч (Grady Booch), Дейв Брейнс (Dave Braines), Алан Браун (Alan Brown), Марк Диксон (Mark Dickson), Хольгер Хойсс (Holger Heuss), Келли Хаустон (Kelli Houston), Лун Дон Мин (Luan Doan-Minh), Филипп Кручтен (Philippe Kruchten), Ник Розански (Nick Rozanski), Дейв Уильямс (Dave Williams) и Эойн Вудс (Eoin Woods).

Примечания

¹ Web-сайт Института разработки программного обеспечения (SEI), посвященный архитектуре - определения архитектуры, много хороших примеров. См. <http://www.sei.cmu.edu/architecture/definitions.html>

² Компьютерное общество IEEE, Рекомендуемые IEEE методы описания архитектуры преимущественно-программных систем: стандарт IEEE 1472000. 2000.

³ Object Management Group Inc., UML 2.0 Infrastructure Specification: Document number 03-09-15 (Спецификация инфраструктуры языка UML 2.0: Документ номер 03-09-15). сентябрь 2003 г.

⁴ Philippe Kruchten, *The Rational Unified Process: An Introduction*, Third Edition (Филипп Крачтен, Rational Unified Process: введение, третье издание, издательство Addison-Wesley Professional 2003).

⁵ Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, Second Edition (Лен Басс, Пол Клементс и Рик Кацман, Практическая архитектура программного обеспечения, второе издание), издательство Addison Wesley 2003 год.

⁶ Object Management Group Inc., *Спецификация унифицированного языка моделирования OMG версия 1.5*, Документ номер 03-03-01. март 2003 г.

⁷ James McGovern, et al., *A Practical Guide to Enterprise Architecture* (Джеймс Мак-Говерн и другие, Практическое руководство по архитектуре корпораций). Издательство Prentice Hall 2004 г.

⁸ Роль, которая будет рассмотрена в следующей статье этого цикла.

⁹ Mary Shaw and David Garlan, *Software Architecture -- Perspectives on an Emerging Discipline* (Мэри Шоу и Дэвид Гарлан, Архитектура программного обеспечения - перспективы рождения предмета) Издательство Prentice Hall 1996 г.

¹⁰ Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide* (Грэди Буч, Джеймс Рамбаф и Ивар Джекобсон (Руководство пользователя по унифицированному языку моделирования (UML) издательство Addison Wesley 1999 год.

¹¹ Bass et al.(Басс и другие), Цитируемое произведение.

¹² Компьютерное общество IEEE, Стандарт IEEE по информационным технологиям -- Процессы жизненного цикла программного обеспечения. Стандарт IEEE 12207-1995.

¹³ Murray Cantor, "Rational Unified Process for Systems Engineering." Журнал *The Rational Edge*, август 2003 г. (Мюррей Кантор, "Применение Rational Unified Process в системном проектировании". http://public.dhe.ibm.com/software/dw/rationaledge/aug03/f_rupse_mc.pdf

Об авторе

Питер Илес (Peter Eeles)



Питер Илес (Peter Eeles) работает для групп IBM Rational, IBM Software Group; большая часть его профессиональной деятельности связана с разработкой архитектуры и реализацией крупномасштабных распределенных систем. Он живет в Великобритании и помогает организациям во внедрении таких продуктов, как Rational Unified Process и IBM Software Development Platform. Питер является соавтором книг Building J2EE Applications with the Rational Unified Process (Создание приложений J2EE при помощи Rational Unified Process) (издательство Addison-Wesley, 2002 г.) и Building Business Objects (Создание бизнес-объектов) (издательство John Wiley and Sons, 1998 г.), и членом коллектива авторов книги Software Architectures (Архитектура программного обеспечения) (издательство Springer-Verlag, 1999 г.).

© Copyright IBM Corporation 2006

(www.ibm.com/legal/copytrade.shtml)

Торговые марки

(www.ibm.com/developerworks/ru/ibm/trademarks/)