Дисциплина:
Проектирование и архитектура программных систем

Лабораторная 2
Тема: Границы применения и область архитектурного проектирования программного обеспечения

Преподаватель:

Смирнов Константин Алексеевич

+7-921-301-64-21

1. Характеристики современных архитектур программного обеспечения

2. Требования, необходимые для создания аналитической архитектуры программного обеспечения

3. Основные объекты аналитической архитектуры программного обеспечения

4. Связи между объектами архитектуры

5. Влияние внешних событий и атрибутов на архитектуру программного обеспечения

6. Необходимый и достаточный набор и объем документации для создания и развития архитектуры программного обеспечения

Введение

При стремлении применить набор теоретических знаний и лучшие практики архитектурного проектирования, с целью воплощения в жизнь рациональной архитектуры программного продукта, необходимо помнить о том, что на любую задачу нет единого истинного ответа. Его истинность, на текущий момент времени, определяется только степенью нашей общей осведомленности о нем.

Ситуация с выбором и разработкой архитектуры программного продукта еще более запутанная, так как это не просто решение, а их взаимозависимая последовательность. Каждое решение, принимаемое в ходе процессов архитектурного проектирования, формирует будущий образ архитектуры и, соответственно, программного продукта.

От того, насколько адекватными и обоснованными будут наши (как проектировщиков и разработчиков) решения, зависит то, насколько *архитектура* будет соответствовать тому представлению, которое вкладывают в неё будущие пользователи и хозяева, а так же способности удовлетворять их потребности и запросы.

Выбор того или иного решения обуславливается множеством разнообразных факторов. Например:

•Понятность и прозрачность требований к архитектуре программного продукта;

•Желаемые характеристики архитектуры;

•Внешнее окружение архитектуры программного продукта:

•Внутренние, используемые архитектурные компоненты и связи между ними;

Все эти отдельные компоненты единого архитектурного "пирога", от гармоничного ***сочетания*** и процентной доли которых в общем "деле", будет зависеть конечный успех программного продукта.

1. Характеристики современных архитектур программного обеспечения

При рассмотрении характеристик архитектур программных продуктов необходимо однозначно понять, что архитектурное проектирование это не просто ***артефакт*** или результат деятельности определенного процесса, но это еще и подход к работе, определяющий её качество, профессиональный образ действий и мыслей.

С одной стороны, многие наши коллеги говорят о том, что если речь идет о разработке небольшого приложения, то не стоит начинать ***НИОКР***, для проектирования архитектуры этого приложения, а разрабатывать "*по* месту" и "времени". Такой подход, безусловно, имеет право на существование, но есть и ***альтернативные*** точки зрения.

В условиях жесткой необходимости и ограниченности ресурсов применять архитектурный подход не всегда целесообразно, но есть риск того, что со временем такой рабочий принцип будет распространяться и на более крупные приложения, стирая тонкую грань понятия "крупности" и "малости" программы. Подобных ситуаций следует избегать.

Убеждение состоит в том, что продумывание и проектирование программного обеспечения, каким бы малым оно не было, должно быть всегда (это, кстати, подкрепляется принципами программной инженерии, один из постулатов которого гласит о том, что написанное однажды должно использоваться многократно. Каждая ситуация, происходящая в профессиональной деятельности, уникальна *по* своей природе, а задачи, возникающие в процессе работы, должны решаться в соответствии с ресурсными возможностями.

Когда говорится об архитектуре программного продукта, интересует, прежде всего, отдаленный, перспективный взгляд на объекты и связи, формирующие информационную систему. Не стоит с самого начала пытаться сфокусироваться на всех деталях будущей системы. **Для начала стоит выделить основное и уделить пристально внимание общей логике архитектурных процессов, типов и видов используемых данных.**

Если со старта процесса архитектурного проектирования уделить слишком много внимания проработке деталей будущего приложения, но не достаточно скрупулёзно проработать логику взаимодействия его элементов, *архитектор* рискует потерять образ архитектуры и упустить *контроль* над управлением программным продуктом и его развивающейся сложностью, запутавшись в незначимых деталях.

Архитектура должна однозначно определять структуру разрабатываемого информационного продукта. Одна из основных ассоциаций, приходящих на ум при слове *архитектура* – структура. Параллели, постоянно проводимые между областью информационных технологий и строительством, имеют право на существование и для направления архитектурного проектирования. Если вы попросите коллегу поверхностно описать архитектуру программного обеспечения, с которым он работает, то, в 8 случаев из 10, он продемонстрирует схему/диаграмму/модель, на которой будут изображены структурные артефакты системы (объекты и связи). Структурные характеристики архитектуры проявляются многими способами в различных ситуациях. Структурный элемент может быть целой подсистемой, процессом, библиотекой, базой данных, вычислительным узлом, системой в традиционном смысле, готовым продуктом и так далее.

Вместе с определением структурных элементов каждая *архитектура* определяет взаимодействия между ними. Именно характер и тип определенных связей обеспечивает функциональность проектируемой информационной системы.

Несмотря на то, что *архитектура* определяет структуру и логику взаимодействия, она не является доминирующим фактором. Область применения, на которую распространяется влияние архитектуры программного обеспечения, ограничивается значимыми элементами, использование которых имеет длительное и достаточно сильное воздействие на автоматизируемые бизнес процессы.

К примеру, можно перечислить следующие элементы:

•главные структурные элементы, задействованные в бизнес процессах;

•элементы, влияющие на "архитектурное" поведение информационных систем;

•элементы, обеспечивающие нефункциональные характеристики архитектуры программного обеспечения, такие как:

- надежность;
- безопасность;
- масштабируемость;

В большинстве случаев, проработанная *архитектура* программного продукта не должна иметь сильное *отношение* или быть зависима от не значимых деталей, но необходимо помнить о том, что *программный продукт* подвержен постоянным изменениям и тот элемент, который был сегодня не важным, завтра может перейти в разряд более критичных.

Признаком успешной архитектуры, не подвергающейся постоянным переработкам и доработкам, является относительная ***стабильность***. Если *архитектура* требует постоянного пересмотра, при относительно небольших изменениях "внешнего" или "внутреннего окружения", то это плохой признак. *Архитектура* должна гармонизировать все потребности заинтересованных лиц к программному продукту.

Цель создания архитектурного программного продукта – удовлетворение комплекса разноречивых потребностей группы наиболее важных заинтересованных лиц. Очень часто выполнить все пожелания к проектируемой системе довольно затруднительно.

Достижение компромиссных решений, которые будут устраивать большинство заинтересованных лиц, это один из основных аспектов успешного архитектурного проектирования.

Как правило, существуют следующие группы ключевых пользователей программного продукта, с перечисленными пожеланиями к нему:

- **конечный пользователь** - заинтересован в интуитивно понятном интерфейсе и предсказуемом и логичном поведении системы, высокой производительности, надежности, удобстве использования, доступности;
- **системный администратор** - заинтересован в интуитивно понятном и логичном функционале, легком управлении и доступных инструментах мониторинга за рабочим "поведением" программного продукта;
- **маркетолог/Специалист по продвижению продукта** - заинтересован в конкурентоспособных функциях программного продукта, его быстром выводе на рынок, позиционировании и выделении среди других продуктов;
- **специалист по сопровождению** - заинтересован в ясном, однозначном и документируемом принципе создания программного продукта, а также в легкости, с которой можно поддерживать текущее состояние и вносить изменения;
- **разработчик** - заинтересован в понятных и простых требованиях, непротиворечивом принципе проектирования;
- **клиент** - заинтересован в адекватной цене разрабатываемого продукта, стабильности и т.д.;

Многие требования целевых групп, представленные выше, являются нефункциональными *по* своему характеру, так как они напрямую не влияют на функциональность системы (легкость сопровождения). Это приводит к тому, что подобные запросы пользователей и формируют свойства и ограничения системы, выраженные в том, как продукт должен выполнять возложенные на него функции, а не в сути этих функций.

В подобных требованиях заинтересован, прежде всего, ***архитектор* системы, отвечающий за создание архитектуры и *группа* разработчиков**, на которой будет лежать реализация этой архитектуры.

Процесс архитектурного проектирования очень похож на мыслительный процесс распутывания узла преступления. *Архитектор*, выполняет формулирование логических следствий из имеющейся у него информации. Обоснование принимаемых решений важно для того, чтобы *активность* архитектурного проектирования и ее результат соответствовали ожиданиям от разрабатываемой информационной системы. При этом надо документировать решения, которые привели к созданию конечного вида архитектуры, с соответствующим логическим обоснованием.

В дальнейшем сформированный пакет документации послужит основой для нормативной документации, регламентов, позволяющих обслуживать и развивать сформированный *программный продукт*, а также будет ядром процессов рефакторинга или реинжиниринга, если в них будет необходимость.

Без надлежаще оформленных регламентов, инструкций и других документов, процессы поддержки и совершенствования архитектуры можно будет рассматривать, как активности создания нового продукта, с соответствующе необходимым бюджетом и трудозатратами, выделенными на это.

Подобные риски можно локализовать и тем самым повысить качество архитектурного проектирования.

Половинчатым решением (не предусматривающим отказ от документирования) подобной задачи будет использование конкретного архитектурного стиля или набора стилей.

Архитектурный стиль можно рассматривать как набор шаблонов, надо сказать относительно сложных и при этом комплексных шаблонов, который предусматривает применение выводов, полученных на основе накопленного опыта принятия успешных и неуспешных решений.

Применение шаблона позволяет выработать общее решение группы задач/проблем в определенной ситуации, которое способствует повышению или стабильной эффективности уже созданных процессов. При этом, в том случае, если шаблоны не используются, это не значит, что *программный продукт* не будет иметь архитектуру.

Если в процессе архитектурного проектирования не было подготовлено документов, то, в дальнейшем, ***программный продукт*** лишается очень ценного инструмента развития. Документированные архитектуры имеют тенденцию быть более продуманными, следовательно, более эффективными. Процесс осознанной фиксации архитектурных решений в процессе проектирования, приводит к всестороннему обдумыванию будущей архитектуры.

2. Требования, необходимые для создания аналитической архитектуры программного обеспечения

Обсуждая необходимые характеристики программного обеспечения, которым следует уделить особое внимание, при разработке архитектуры программного продукта, следует отметить, что к проектированию информационной системы, обладающей той или иной характеристикой, приводит набор требований, предъявляемых к конечному продукту.

Под *требованиями к программному обеспечению* принято понимать *совокупность утверждений относительно элементов, характеристик или качеств программной системы, подлежащей реализации.*

Таким образом, требования представляют собой связующее звено между разнообразными характеристиками и конкретной реализацией ожидаемого программного продукта. Важно, чтобы требования были сформулированы с такой подробностью и степенью детализации, которая необходима на конкретной стадии жизненного *цикла* программного продукта с целью получения результата заданного качества.

В классической литературе можно найти разнообразную таксономию (классификацию *по* различным типам, видам, группам) требований, но традиционно выделяют две основополагающие группы:

•функциональные требования;

- эта группа требований формулирует характеристики программного продукта к процессу взаимодействия между информационной системой и пользователями, в котором достигаются бизнес цели и задачи;

•не функциональные требования;

- это вид требований, который позволяет заложить системный базис информационного продукта, на котором станет возможным "вырастить" оптимальную для конкретных условий архитектуру программного продукта;
- **Простые в понимании:**

набор требований должен быть прост и доступен понимаю каждого сотрудника, задействованного в работе над ними на любом этапе жизненного цикла разрабатываемого программного продукта. Если идею объяснить тяжело, есть большая вероятность того, что кто-то приняв молчаливое согласие, впоследствии начнет делать работу не так, как было задумано.

•Простые в использовании

требования должно быть легко и просто использовать. Если требования не разложить до нужного уровня детализации, после чего их использование не требует избыточного внимания и умственных затрат, тогда мы рискуем, что архитектура разрабатываемого приложения будет создана не в спроектированном нами виде. Различия между теоретическими схемами и реальной архитектурой в дальнейшем приведет к недопониманию между сотрудниками, развивающими продукт, и пользователями системы.

•Соответствующие реальным ожиданиям пользователей

набор требований должен обеспечивать приемлемые заказчиком атрибуты и функциональность системы. Если система должна обеспечивать оперативность онлайн работы пользователей при условии одновременной работы нескольких тысяч сотрудников, то это действительно надо. В случае, если какой-то атрибут системы будет функционировать не в соответствии с требованиями пользователей, то архитектуру программного продукта нельзя будет считать рабочей.

•Обеспечивающие комфортный процесс разработки

структура требований должна быть не только простой и адекватной ожиданиям заказчиков, но и обеспечивать прозрачную и понятную последовательность реализуемых компонент, чтобы разработчики архитектуры могли чувствовать себя комфортно в процессе реализации архитектуры программного продукта.

•Команда должна быть согласна с архитектурой

команда, трудящаяся над реализацией архитектуры и будущего программного продукта, должна быть согласна с ними. Если в команде есть явный противник выбранного способа реализации (скажем, не согласен с выбранной технологий реализации базы данных), то, как правило, если не локализовать его влияние на процесс разработки, сделав при этом адептом общего "пути", результат может быть провальным. Есть множество техник и приемов, позволяющих добиться нужного эффекта (смена команды, смена технологии, тренинги на командообразование, воспитательные беседы и т.д.), но это только альтернативы, которые должны применяться в случае, если Вы заранее не смогли обеспечить общее согласие всех членов команды проектирования и разработки программного продукта.

•Общие принципы для всех профессиональных активностей, с которыми работаешь в рамках конкретного продукта и конкретной технологии

если принято какое-то решение, то оно должно быть использовано везде и всеми. Это рабочий принцип применим не только к формулировке требований, но и к большинству процессов домена информационных технологий. Если где-то использование принятого решения вызывает затруднение, то необходимо выявить, почему это так! В случае, если ситуация действительно представляет собой исключение из общего правила, то правило необходимо дополнить и переработать таким образом, чтобы оно покрывало новое ответвление.

Тем самым мы придем к универсализации рабочих принципов и можем постепенно добиться создания полноценной организационной системы архитектурного проектирования, но важно помнить, что в любом деле очень важно чувство меры. Система должна быть гибкой и адаптивной, но при этом жесткой. Это позволит выдержать множество проверок, в виде разнообразных поступающих задач по созданию архитектуры программных продуктов. Закон/принцип должен быть. Если он не будет исполняться – будет плохо всем.

•Требования должны учитывать организационную структуру, в которой будет применяться программный продукт;

При разработке всех типов требований к архитектуре программного продукта ***необходимо учитывать организационную и функциональную структуру компании/подразделений***, для которых разрабатывается программный продукт. Архитектура создаваемого продукта, как минимум, не должна противоречить существующему в компании/ях укладу дел, структуре взаимодействия при реализации процессов разработки программного обеспечения, рабочим практикам и т.д.

Если в автоматизированных процессах создаваемого программного продукта будут задействованы разные подразделения или, компании, то сферы ответственности и зоны влияния модулей информационного продукта должны быть четко, ясно, понятно и однозначно очерчены. В противном случае рискует быть ситуация, при которой задачи, в реализации которых будут задействованы несколько исполнителей, "повиснут" не имея решения, и, как следствие, результат активностей не будет достигнут.

•Сроки реализации требуемого функционала должны учитываться при разработке требований;

Иногда, крайне редко, случается так, что на реализацию поставленной задачи отводится срок, которого недостаточно на предварительный сбор и *анализ* требований, в этих случаях необходимо….

Текущие реалии таковы, что на реализацию задач отводится все меньше и меньше времени. Срок реализации - это один из ресурсов, необходимых для процесса архитектурного проектирования и создания качественной архитектуры программного продукта. Период времени от замысла до воплощения его в коде должен быть реалистичен, соотносясь с качеством, диктуемым рынком и имеющимися ресурсами. Пытаться добиваться безупречного решения, в условиях, когда оно не требуется, это, конечно, очень амбициозно, но при этом довольно глупо, безрассудно и непрофессионально. Зрелый процесс архитектурного проектирования должен учитывать различные способы проектирования и предлагать, в зависимости от ресурсной составляющей процесса, различные результаты.

Анализируя имеющиеся требования, необходимые для разработки архитектуры программного продукта, мы считаем целесообразным привести несколько рабочих принципов, которые могут способствовать формированию рабочего видения и эффективного подходов к проектированию:

- *Обработка проектируемых элементов, связей, алгоритмов работы должна представлять собой набор правил (а лучше правило) преобразования и представления данных*. Если на последующих стадиях реализации продукта или в процессе управления изменениями поступили требования, которые не вписываются в структуру правил, то необходимо переделать правила, расширив их необходимым образом или перепроектировать систему, с учетом измененных к ней требований;
- **Проектируемая архитектура** программного продукта должна представлять собой набор относительно небольших модулей, компетенции по работе с которыми должны быть разделены между членами команды проектировщиков. Проектирование каждого модуля и их единое представление должно выполняться в соответствии с стандартами, лучшими практиками, рекомендациями отрасли программной инженерии. Таким образом, можно обеспечить преемственность задач и обеспечить их дальнейшее сопровождение и развитие (не отрицая, а дополняя активность документирования). Важно, чтобы процесс передачи компетенций и наследования знаний постоянно контролировался со стороны менеджмента;
- Обдуманно и очень осторожно относиться к технологии "Copy&Paste". Во многих случаях этот подход к работе (с учетом применения его к лучшим и отработанным решениям) лучшее решение, но применять его везде – губительно. Дальнейшее сопровождение/изменение/развитие (сделаем особенное ударение на часть изменение) решений, сделанных по подобной технологии, порождает большое количество ошибок.
- Слишком сложные решения, для того, чтобы разобраться в которых требуется большое количество времени и сил приводит к дальнейшей не реализуемости архитектур. Приведем эмпирическую метрику и скажем о том, что в высокоуровневой схеме архитектуры должен уметь разобраться каждый член команды проектировщиков, самостоятельно, не более чем за 15 минут.

3. Основные объекты аналитической архитектуры программного обеспечения

Современные архитектуры программных продуктов базируются на основе компонентной модели разработок. Под компонентом мы будем понимать *модуль* системы или отдельный *программный продукт*, назначение которого состоит в обработке и инкапсуляции его содержимого. Результаты деятельности "черного ящика" должны быть гармонизированы с остальными частями программного продукта. Таким образом, поведение компонента, как основного объекта архитектуры программного продукта, определяется тремя основными группами требований:

1. Требованиями к внешнему интерфейсу, через который осуществляется взаимодействие с остальными частями архитектуры;
2. Требования к внутреннему интерфейсу/структуре данных, которые определяют характеристики компонента, его преимущества и недостатки;
3. Требования к функционалу, интегрирующему внешнее и внутреннее поведение компонента и преобразующему данные в единый формат на основе которого становится возможным взаимодействие между модулями архитектуры программного продукта;

Объект, как элемент архитектуры программного обеспечения, должен поддерживать статические (элементы структуры) и динамические (бизнес процессы) связи компонентов и, в зависимости от типа связи, поддерживать определенную, спроектированную для конкретных целей, функциональность.

Каждый реализованный ***объект*** должен удовлетворять требованиям, предъявляемым к архитектуре программного обеспечения, быть в состоянии поддерживать заданный уровень качества и надежности программного продукта. Особенно важно понимание того, что уровень производительности объекта будет определяться производительностью его самого слабого звена, если способ реализации одного из требований к компоненту будет предполагать дополнительные ресурсные *затраты* (время на обработку событий, правила, процедуры и алгоритмы обработки, и т.д.), то важно четко определить к какому компоненту разработка будет относиться.

Таким образом, ***значимость*** компонента в архитектуре может изменяться, но вместе с этим будет меняться и необходимая, для его функционирования, ресурсная составляющая, что может влиять на дальнейшее развитие архитектуры.

Адекватная ***архитектура*** отличается тем, что она в состоянии балансировать и распределять ***значимость*** компонентов (а соответственно и их загрузку) равномерным "слоем" между всеми составляющими её объектами. Это позволяет снизить перегруженность и *значимость* отдельных объектов и повысить ***универсальность*** архитектуры в целом и её возможную *производительность*, но так как процесс развития архитектуры процесс динамический, архитектурное проектирование должно следовать этим принципам на протяжении всего жизненного ***цикла*** программного продукта.

Развитие каждого объекта, являющегося частью архитектуры должно идти в соответствии с развитием архитектуры. Такой принцип позволит избежать появления "узких горлышек" и создать оптимальные, для конкретных условий и организаций, объекты и архитектуру программных продуктов

4. Связи между объектами архитектуры

Для осознания возможностей, целей и потенциальных задач применения компонентной архитектуры программного обеспечения, проведем аналогию между архитектурами в строительном проектировании и в архитектурном проектировании программных продуктов.

Автор современного витка строительства серийных сооружений, французский ***архитектор*** (оставивший свой "след" и в Москве) Ле Корбюзье, оказал существенное влияние на разработки и массовое применение **унифицированных строительных блоков**.

Разрабатывая архитектуру зданий и сооружений, француз своей основной целью видел создание не просто отдельного здания (компонента), а пространственной композиции **(*архитектура*)**, воспринимая строительные блоки примерно так же, как его предшественники-зодчие воспринимали кирпичи.

Результат своей работы Корбюзье представлял в виде архитектурно - пространственного ландшафта, основанного на повсеместном и многоразовом применении блоков, узлов и других составляющих, позволяющих возводить здания и связывать их в единый комплекс, довольно быстро и качественно.

Подобный подход был революцией в строительстве 30-хх годов прошлого века. Отметим, идея была, действительно, новой и эволюционной, и проекты, реализованные самим Корбюзье, впечатляли. Дальнейшее развитие этой технологии, без надлежащего контроля, привело к тренду создания советских новостроек конца 50-х и продолжается сейчас в массовом строительстве.

В области информационных технологий ситуация не сильно отличается. Архитектурное проектирование, во многом, является вполне персонализированным ремеслом, влияние отдельной личности в котором можно минимизировать, но исключить вряд ли. В соответствии с развивающейся областью программной инженерии, **компоненты являются строительными блоками**, с помощью которых создаются архитектуры программных продуктов.

Применение готовых или почти готовых и проверенных ***компонент*** смещает основной фокус внимания с объектов на аспект их связи\интеграции. Это предъявляет дополнительные требования к программным архитектурам, которые должны быть совместимы и максимально сочетаемы друг с другом.

Если переложить на бизнес язык понятие компонента, то в общем смысле он представляет собой "часть" конкретного бизнес процесса, а ***связь*** между ними приводит процесс в действие и обеспечивает достижение поставленного результата.

Оптимально разработанные связи будут способствовать не только достижению конечного результата, как суммы результатов отдельных компонентов, но и могут "привнести" бонусы, выраженные в виде дополнительной функциональности, быстродействии, надежности и других характеристиках, которые могут быть достигнуты за счет эффекта **эмерджентности**.

Под **эмерджентностью** мы будем понимать дополнительные свойства, приобретаемые набором компонент, соединенных в единую систему, за счет использования связей между ними и, как возможное следствие, преобразование потенциально не задействованных ранее свойств в явные результативные эффекты.

От того, насколько успешно разнородные компоненты будут интегрированы в единую и монолитную структуру в процессе архитектурного проектирования, зависит кпд и возврат инвестиций от создаваемой архитектуры.

5. Влияние внешних событий и атрибутов на архитектуру программного обеспечения

Создание единой архитектуры изменяет окружение не только с программной точки зрения. Она привносит дополнительные черты и характеристики. Со временем, при условии использования и развития архитектуры на всех иерархических бизнес уровнях компании, облик организации изменяется:

•Вырабатывается подход многократного использования активов;

•Изменяется подход к ведению бизнеса;

•Изменяется среда (бизнес, организационная, профессиональная) в терминах навыков, доступных пользователям архитектуры;

•Повышается обоснованность принимаемых решений;

•Вырабатывается единый рабочий стиль;

Единство, в первую очередь, определяется как унифицированная рабочая философия конкретной компании, пронизывающая её "насквозь", от исполнителей бизнес процессов до топ-менеджеров, формирующих стратегию компании. Подобная философия должна представлять собой best practice организации. В нем должен быть агрегирован и представлен наиболее эффективный, с разных точек зрения, опыт, методологии, технологии работы, доказавшие свою результативность на практике, в разнообразных рабочих ситуациях.

В случае применения подобных подходов, окружение, оказывающее влияние на архитектуру (это называют "архитектурой в контексте"), будет четко и однозначно не только определять границы использования архитектуры, но и повысит качество работы пользователей и уровень фирменной/корпоративной культуры, развитие которой является обязательным для развития организации в целом.

Факторы "контекста", которые являются внешними событиями и атрибутами *по* отношению к архитектуре программного обеспечения, это:

- **Миссия бизнеса, которую будет поддерживать архитектура программного продукта**

Под миссией принято понимать то, что организация может предоставить для общества, в котором она осуществляет свою деятельность, в обмен на получаемую от него выгоду. Архитектура программных продуктов оказывает косвенное влияние на миссию компании. Во – первых, оно выражено в виде оптимизации той части процессов компании, которые направлены на повышение эффективности взаимодействия с членами общества представляющими различные социальные слои; во-вторых, в эффективном потреблении и результативном многократном использовании разнообразных внешних ресурсов, экономия которых это основная задача, стоящая перед современным обществом.

•Цели и задачи бизнеса

Цель бизнеса - это формулировка желаемого результата деятельности в терминах конкретной бизнес области. Цель представляет собой набор заданных условий, выполнение которых необходимо. Цель представляется в виде задач, которые играют роль последовательно расположенных ступенек, движение по которым должно привести к цели.

Цель влияет и определяет задачи, а, исходя из набора конечных задач, выполняется формулировка рамок требований и соответствующих характеристик, определяющих как процесс архитектурного проектирования, так и саму архитектуру программного продукта.

•Заинтересованные в успешном функционировании системы лица/стороны

Заинтересованных лиц принято называть стэйкхолдеры, вернее, более правильно говорить не о заинтересованных лицах, а о сторонах, представляющих целевую аудиторию сотрудников, в интересах которых, чтобы продукт обладал набором определенных, очень часто противоречивых, качеств/характеристик (Ключевые пользователи, администраторы приложений, тестировщики и т.д.). От того, какими будут эти требования (как функциональные, так и не функциональные) зависит будущий программный продукт, а соответственно и его архитектура.

- **Внутренние технические ограничения**

Существуют внутренние технические ограничения, которые связаны с конкретной технологией создания, сопровождения, развития программного продукта (язык программирования, его новизна и темпы развития, применяемая технология хранения данных, и т.п.), игнорировать и не учитывать влияние которых не представляется возможным.

Архитектура программного продукта - это достаточно абстрактное понятие, но ряд требований должен быть учтен обязательно, иначе архитектура рискует прийти в устаревшее и не актуальное рыночным потребностям состояние, так и не увидев "свет";

•Внешние ограничения

Из внешних ограничений, формирующих рамки требований к архитектуре и архитектурному проектированию программного продукта нужно выделить те, которые предъявляются не столько к определенному продукту, а сколько к бизнес домену, в котором информационная система выполняет свое функционирование.

Для примера можно привести:

- Необходимость взаимодействовать с внешней системой;
- Соответствие внешним регулятивным нормам (законам и постановлениям правительства, отраслевым стандартам);

Во многом подобные требования и ограничения определяют структуру не только архитектуры программных продуктов, но и тип, регламент бизнес процессов будущего продукта.

6. Необходимый и достаточный набор и объем документации для создания и развития архитектуры программного обеспечения

Количество, масштаб проектов, комплекс и компоненты разрабатываемой архитектуры и самих программ, видение их дальнейшего развития - это важнейшие факторы, определяющие формирование, структуру и содержание документации, поддерживающей весь ***жизненный цикл*** программных продуктов.

Оценки перечисленных факторов, определяющих архитектуру информационной системы, должны быть достаточно подробно проанализированы и, *по* необходимости, пересмотрены. В результате выполненного анализа процессы, необходимые для реализации архитектуры могут быть скорректированы, но важно в ходе выполненной активности установить рамки и объем документации, необходимой для создания архитектуры программного продукта.

Для реализации части требований может потребоваться пересмотр выделенных для нее ресурсов с целью обеспечения результата при выполнении фаз проекта (проектирование, разработка, тестирование) и т.д. Из этого следует, объем и количество документации в целом и стадии архитектурного проектирования в частности должны быть согласованы с масштабом ***работ*** и выделенных на них ресурсы. В подобных ситуациях целесообразно использовать адекватные техники, методы, показатели и ***размерность.***

Формирование адекватных, конкретной ситуации требований к архитектуре программного продукта должно соответствовать принятому жизненному циклу требований.

От согласованного масштаба программного продукта зависят:

•Ресурсы для документирования

Целесообразно создавать в реальных проектах только те шаблоны документов, использование и развитие которых экономически целесообразно;

•Масштаб проекта и спецификация требований

Данные факторы оказывают влияние на состав, содержание и объем документации, необходимой участникам проекта. Каждый из членов рабочих коллективов в определенной степени задействован в процессах разработки и управления требованиями к связанной с ними документации. Ответственным исполнителям отдельных стадий и этапов работ необходимо выработать профессиональные подходы для изложения в создаваемых документах истинных потребностей в функциональности архитектуры программного продукта.

В качестве примера можно привести следующие целевые группы пользователей и их потребности, выраженные в соответствующих данных, которым должны удовлетворять создаваемые документы:

- **Разработчики:**

Формируют представление о сложности, размере, характеристиках создаваемой системы;

•**Аналитики:**

получают информацию, необходимую для создания различных видов спецификаций требований к программному продукту, валидируют соответствие системы требованиям существующих законов и постановлений;

•**Руководство проекта:**

получают основание для расчета содержания работ над спецификациями и необходимых ресурсах;

•**Тестировщики:**

создают планы тестирования, варианты испытаний, процедуры проверок;

•**Системные администраторы**:

получают представление о функциональности каждой составной части продукта;

•**Целевые бизнес пользователи:**

формируют конечное представление о программном продукте, изучают его;

• **Специалисты, ответственные за обучение персонала:**

Получают спецификации требований и документацию для разработки обучающих материалов.

На сегодняшний день многополярность сферы информационных технологий и отсутствие катализаторов стандартизации в направлении развития коммерческих программных приложений привело к ситуации, когда размер программных продуктов принято выражать различными размерностями одних и тех же единиц, или показателями, которые не согласуются друг с другом. Подобная тенденция породила многозначность числовых значений, которые описывают масштаб программ. Они разняться в зависимости от автора и типа публикации, но при этом такой подход имеет преимущество при определенных целях проектирования и создания программных продуктов.

Единицы измерения размера программ - один из показателей, который определяет оценку объективно необходимого объема документации. Необходимо учитывать процесс изменения, трансформации программ, в зависимости от которого могут меняться подходы к измерению их масштаба и взаимосвязанные с ними единицы измерения. Принято выделять 2 группы единиц измерения:

•Первая группа – функциональная;

Характеризует размер документации, которая разрабатывается и поддерживается специалистами. Она должна отражать сложность, трудоемкость и длительность создания архитектуры программного продукта, его компонентов и их функциональности;

•Вторая группа – системная;

Она отражает размер создаваемых систем и поддерживающих их данных, характеризующих объем памяти, производительность, необходимые для рабочего функционирования и исполнения программы в соответствии с его назначением и определенными атрибутами;

Подобное деление на группы позволяет комплексно и достаточно объективно отразить размер программных продуктов и документов под разными точками зрения на систему.

С целью уменьшения возникающих неопределенностей и потенциальных методологических ошибок необходимо определить основные понятия и *размерность* определения масштаба программного продукта перед стартом процесса функционального анализа данных и архитектурного проектирования его возможностей.

В самом начале работы над реализацией архитектуры и функциональностью программного продукта важно фиксировать и вести учет всей поступающей информации о необходимых возможностях, входящих в рамки разработки архитектуры программного продукта. Согласованная "плотность" этих рамок, соответствующая целям создания и развития информационной системы, будет определять действительную трудоемкость, *стоимость*, содержание *работ* и комплекс необходимых документов. Все управленческие решения относительно создания и изменения функциональности архитектуры должны быть зафиксированы в управленческих документах, способствующих принятию решений для планирования развития программных продуктов, локации ресурсов и других решениях для развития информационных систем.

В целях успешной разработки согласованного пакета документов, после того как выполнена оценка функциональности программ, целесообразно выполнять циклы поэтапного определения и формирования необходимых спецификаций требований к архитектуре и компонентам информационного продукта.

Первый этап ***работ*** в жизненном цикле программного продукта - это формирование его концепции и набора первичных, высокоуровневых требований к его функциональности, которые в дальнейшем будут разбиты на фактические программные компоненты. ***Разбиение*** должно структурировать и детализировать ***представление*** о продукте, формируя его более конкретный образ с необходимыми связями и нефункциональными характеристиками. Довольно много внимания следует уделить сбору требований о пользовательских возможностях продукта, которые должны быть обеспечены его архитектурой и их причинам.

После того как будет выполнена первичная оценка необходимой функциональности, следует перейти к итерационной, поэтапной разработке функциональных спецификаций требований и сопутствующим им документам. Стоит понимать, что ограничения, связанные с прогнозированием функциональности к программному продукту, которые будут отражены в создаваемых спецификациях, определяются имеющимися у их разработчиков данными, которые будут использоваться для формирования структуры архитектуры, алгоритмов её функционирования и обобщенных рабочих характеристик. Спецификация, как документ, в котором содержатся требования к архитектуре программного продукта, должна быть ясной и понятной. Это позволит снизить риски непонимания между заказчиками и разработчиками информационного продукта в том, как должна быть реализована и каким образом будет функционировать информационная система.

В распространенной практике создания информационных систем существует подход, при котором составление спецификаций выполняется не перед стартом процесса разработки соответствующей функциональности, а параллельно с ним или даже после того, как разработка выполнена.

Существуют разные рабочие ситуации, при которых выполняемое действие обоснованно его условиями, но *опыт* показывает и практика подтверждают, что необходимость фиксирования согласованных требований с нужным уровнем детализации необходимо выполнять, в большинстве случаев, до начала стадии реализации программного продукта. Это позволит избежать множественных ошибок, связанных с противоречием и недостоверностью данных.

В "правильном" процессе проектирования архитектуры для того, чтобы привести достоверную оценку, прогнозирование и обоснование спецификаций необходимы следующие данные:

- Характеристики необходимых ресурсов для документирования, оценки влияния на них функций, различных факторов, критичных для процессов разработки объектов и среды разработки;
- Планы документирования, включая перечни работ, реальные графики проведенных ранее оценок и разработок;
- Цели и содержание работ в процессе создания архитектуры и различных документов для обеспечения необходимого качества программного продукта;
- Структура и содержание комплекта документов, являвшегося результатом выполнения отдельных работ конкретного проекта;

Правила составления спецификации требований, документации отражают подходы, в соответствии с которыми все заинтересованные в архитектуре стороны смогли разобраться в профильных для них документах:

•Разделы, подразделы и отдельные требования должны иметь согласованные названия;

•Нужно использовать средства визуального выделения (различные шрифты, стили и т.д.) последовательно, иерархически и в разумных пределах, применяя соответствующие ГОСТы;

•Каждое требование должно содержать оглавление и алфавитный указатель, чтобы облегчить пользователям поиск необходимой информации;

•Нумеровать все рисунки и таблицы, ссылаясь на них, используя присвоенные номера;

Для верификации и последующего изменения документов, каждое функциональное требование должно быть представлено уникально и неизменно. Подобное правило работы с ними позволит достаточно оперативно находить необходимое требование из запроса на изменение, в хронологии изменений, в перекрестных ссылках или матрице изменений для отслеживания реализации требований.

Для обеспечения подобной системы работы с требованиями, на этапах архитектурного проектирования должны быть определены следующие принципы работы с требованиями, в зависимости от видов процессов проектирования:

- **Первичное архитектурное проектирование**:

Фиксируются и анализируются высокоуровневые требования к назначению, функциональной пригодности, составу и перечню необходимых не функциональных характеристик и первоначальному составу документов;

- **Обзорное рабочее проектирование:**

Правдоподобная оценка масштаба работ по программному продукту, требования к функциональным и не функциональным характеристикам, структуре и содержанию шаблонов документов в жизненном цикле информационной системы с учетом первоначальных ограничений к ресурсам;

- **Подробное детальное проектирование:**

Подробные требования к функциональным и не функциональным качествам программного продукта, с декомпозицией ресурсов, полный состав и содержание документации;

Всеобъемлющая и "одноразовая" фиксация требований к характеристикам разрабатываемой архитектуры свойственна только классическому подходу к проектированию архитектуры программных продуктов - **"водопадному"** процессу разработки информационных систем. Основными потребителями результатов программной инженерии (собственно программных продуктов) являются коммерческие организации, в которых изменения требований к программному обеспечению происходят синхронно с изменениями бизнес среды, то есть постоянно. Такая *интеграция* коммерческих организаций и информационных продуктов диктует свои правила развития программной инженерии, что соответствующим образом трансформирует составляющие её процессы, в которые входит и *активность* документирования.

В пользу **итерационной фиксации** требований можно отнести и тот факт, что в начале процесса работы над ними, **ожидания заказчиков и представления разработчиков** о деталях назначения программного продукта, его функциях и возможностях реализации при доступных ресурсах, как правило, не совпадают.

Чтобы снизить подобные риски непонимания и несовпадений желаемых возможностей и реализуемого функционала, необходимо однозначно, в момент возникновения или трансляции ожиданий, фиксировать их, анализировать возможности реализации, с учетом рационального расходования имеющихся ресурсов, и согласовывать с заинтересованными сторонами не желаемый образ решения/функционала, а его **реальные варианты реализации**. Если некоторые характеристики противоречивы или принципиально нереализуемы, заказчик должен как можно раньше узнать об этом для корректировки своих ожиданий или локации больших ресурсов на соответствующие активности. Не сбалансированные требования и ограниченные ресурсы будут со временем являться **"узким"** горлышком архитектуры или программного продукта. В зависимости от сложности и комплексности программного продукта в результате активностей сбора, анализа требований и проектирования архитектуры должна стать подробная документация (спецификации, регламенты, инструкции и т.д.) согласованная *по* своим свойствам, структуре и содержанию с заинтересованными лицами.

Разработанного пакета документов должно быть достаточно для поддержки жизненного *цикла* программного продукта, начиная с момента разработки и вплоть до стадии вывода из эксплуатации.

Сложность и ***значимость*** продукта будет определяться выделенным для его создания ресурсом, "глубиной" (качество) и "шириной" (рамки функциональности продукта) проработки требований к характеристикам его архитектуры, качеству создаваемых документов.

Чем сложнее продукт, тем больше времени и трудозатрат потребуется для детального анализа, разработки документов, проектирования архитектуры и последующей оптимизации созданных артефактов программного продукта.

Если информационная система не отличается высокой сложностью, то в некоторых случаях многими шагами стадий анализа и проектирования пренебрегают, с целью рационального и адекватного использования выделенных на проект ресурсов.

При выявлении высокоуровневых требований к функционалу будущего программного продукта, какой бы сложности он не был, на начальных стадиях работы над ним, не всегда возможно спрогнозировать необходимый ***ресурс.***

Недостаточный для реализации архитектуры и информационной системы ***ресурс*** может привести к снижению требований к некоторым, наиболее трудозатратным при реализации, характеристикам, конструктивным возможностям программы или документации.

Поэтому, в практике управления созданием программных продуктов, а точнее в части процессов управления взаимоотношениями с исполнителями, требования к документам фиксируются в договорах и прописываются в соответствующей документации, в соответствии с которой разработчик должен отчитываться перед заказчиком, ***по* ходу завершения этапов** создания архитектуры и самого программного продукта.

Но, после того, как конкретный этап завершен возникают изменения требований и функциональности. Их последующая реализация должна инициировать не только изменение конкретного функционала, но и тех частей архитектуры, которые связаны с измененной частью, то есть являются *по* отношению к нему "серыми" или "белыми" компонентами. Такие активности **принято называть анализом влияния**.

Анализ* влияния** должен запускаться на всех этапах жизненного ***цикла при возникновении активностей изменения и конфигурационном управлении.

В подобных случаях должны адекватным образом изменяться соответствующие документы ***по* согласованию между заказчиком и разработчиком**.

Актуальность и своевременность изменений связана с **процессами мониторинга и управления** процессами жизненного *цикла* программного продукта.

Требования к функциональным и не функциональным характеристикам программного продукта, закрепленные в технической документации и утвержденные после стадии предварительного проектирования, будут в дальнейшем использоваться ***для оценки качества подготавливаемой документации*** при её сопоставлении с реализацией требований в процессах верификации, валидации и возможной сертификации программного продукта.

Для тех программных продуктов, *значимость* которых для заказчиков высока или их инновационность и новизна является основным фактором при решении об их создании, то уточнение, *детализация*, *дополнение* требований к качеству документирования при выполнении детального проектирования, с точки зрения повышения эффективности значения **коэффициента - качество/*затраты*** ресурсов неизбежны.

Для заказчика программного продукта и групп заинтересованных пользователей имеют ***значимость*** следующие результаты процессов разработки и использования конкретного приложения:

- Функциональная эффективность системы;
- Пригодность качественных характеристик;
- Спрос на результаты деятельности программного продукта, не только конечных пользователей, но и смежных информационных систем;
- Конкурентоспособность по отношению к другим, аналогичным по функциям, программным продуктам, с учетом его общего качества и стоимости;

Последний фактор определяет необходимость в уточнении требований к отдельным характеристикам не только для их адекватной реализации группой разработки, а так же для оценки интегрального качества архитектуры и функциональности программного продукта, при его выводе на рынок информационных систем.

Как правило, заказчики и исполнители в начале работы над созданием информационной системы устанавливают требования ко всем характеристикам и содержанию документов программного продукта без учета затрат на их создание. ***Детальный анализ*** влияния характеристик отдельного компонента или функциональности на структуру документа, описывающего его, выполняется позднее.

Это приводит к тому, что документация разрабатывается с неудовлетворительным качеством *по* следующим причинам:

- перекос по выделенным ресурсам в сторону менее значимых документов, не представляющим высокой ценности для последующих процессов;
- несбалансированные значения требований к отдельным, взаимосвязанным характеристикам и документам;

В отдельных проектах это может привести к рискам повышения стоимости, снижения конкурентоспособности создаваемого программного продукта из-за недостаточного уровня отдельных показателей качества и документирования его атрибутов качества.

Полезность и значимость отдельных документов имеют различную степень, в зависимости от **стадии жизненного *цикла*** информационной системы. Этот фактор приводит к их несопоставимости между собой в определенный момент времени.

Каждый тип документа должен выбираться и согласовываться с заказчиком при поэтапном анализе значимости документа на определенной стадии жизненного *цикла* программного продукта.

Для того чтобы это стало осуществимо, необходим учет относительного влияния определенной характеристики и документа на дальнейшую пригодность в процессах реализации и использования программного продукта с учетом необходимых для их разработки ресурсов. Иначе возможна ситуация когда выделенные ресурсы будут не рационально расходованы на создание функциональности или требований, *значимость* которых оценена не адекватно их реальной стоимости.

Эффективное управление документацией программного продукта в процессах архитектурного проектирования может быть выстроено при условии обоснованного выделения и обобщения величины полезности различных характеристик и соответствующих документов в обобщенный интегральный показатель, представляющий количественную метрику, отражающую их совокупное влияние на их дальнейшую пригодность.

Таким образом, мы выявили следующие задачи:

- **Анализ комплексной эффективности документации в процессах анализа, разработки требований и последующего архитектурного проектирования;**
- **Оценка системного влияния различных типов документов на программный продукт, с учетом ресурсов на их реализацию;**

Для их решения целесообразно каждому из создаваемых документов присваивать коэффициент значимости или приоритет, в соответствии с которым документ влияет на реализуемую функциональность программного продукта в частности и на архитектуру в целом. Экспертность и *точность* определения оценки вряд ли возможна выше 10%, поэтому количество градаций шкалы оценок не должно превышать 10. Аналогично, *по* соответствующей шкале экспертами может быть обоснованно оценено количество ресурсов, которые необходимы на реализацию конкретного документа. Коэффициент влияния на функциональную пригодность к затратам на его достижение для конкретного документа должен быть рассмотрен как обобщенный уровень приоритета его реализации. Для каждого конкретного случая разработки программного продукта значения приоритетов документов должны быть итерационно адаптированы с учетом назначения и функций определенного документа.

Самый высокий приоритет следует интерпретировать как **обязательную необходимость создания документа** с заданными характеристиками и с должным уровнем качества.

Самый низкий следует воспринимать как **факультативную *значимость*** конкретного требования или документа, который, скорее всего, вряд ли будет реализован.

Промежуточные оценки приоритетов будут представлять относительное влияние конкретных документов на соответствующую функциональность или структуру программного продукта, с учетом доступных для их реализации ресурсов.

Низкие приоритеты требований и документов могут приводить к отсутствию/не полной реализации соответствующих артефактов процессов разработки, которая будет "творчески" оцениваться ответственными исполнителями.

Процесс последовательной обобщенной приоритезации функциональности и документов целесообразно проводить в соответствии со следующими шагами:

- Оценку анализа влияния, требуемой характеристики и документа на функциональность программного продукта (в диапазоне 1 – 10);
- Оценку относительных затрат ресурсов на реализацию требуемого документа (в диапазоне 1 – 10);
- Оценку относительного коэффициента влияния характеристики документа, на функциональность программного продукта с учетом затрат на его реализацию, как интегральный показатель первых двух оценок;

В процессе архитектурного проектирования необходимо **поэтапно уточнять и детализировать требования к структуре, свойствам и характеристикам документации** с использованием первых двух видов коэффициентов. Для крупных программных продуктов, при постепенном уточнении требований к документации, целесообразно использовать обобщенный уровень приоритета. Это позволит понимать актуальный приоритет разработки определенного типа документа. Набор значений обобщенных уровней приоритетов для выбранных атрибутов документов конкретного программного продукта можно разделить на три группы:

- Общий приоритет больше 8 – документ оказывает критическое влияние на функциональность программного продукта, с учетом конкретной стадии его разработки;
- Общий приоритет в интервале между 4 и 8 – документ оказывает полезное, но не критичной влияние на функциональность программного продукта и будет разработан только при наличии необходимых на это ресурсов;
- Общий приоритет менее 4 – в данный конкретный момент документ оказывает второстепенное влияние на функциональность программного продукта и вряд ли будет разработан;

Представленная оценка может применяться в качестве управляющей информации при принятии решений о разработке соответствующей документации на определенной фазе **жизненного *цикла*** программного продукта. Она позволит рациональным образом расходовать выделенный для созданий **программного продукта *ресурс***, экономя его в тех случаях, когда создание определенного типа документации не обоснованно.

Анализ представленных приоритетов позволяет выделить те документы, которые отличаются высокой ресурсной составляющей, необходимой для их разработки, но дальнейшее использование которых не целесообразно, *по* причине их низкого влияния на артефакты программного продукта.

Такие документы следует заранее исключать из рамок разработки программного продукта, спрогнозировав степень их необходимости.

Для пользователей и заинтересованных сторон, ответственных за будущую поддержку и развитие программного продукта наибольшее ***значение*** имеют те документы, которые будут описывать **эффективное применение компонентов** и архитектуры программного продукта. При расчете обобщенных коэффициентов и расстановки приоритетов создания документации это так же необходимо учитывать.

Процесс документирования затруднен, если речь идет о создании относительно нового и инновационного программного продукта, который не имеет аналогов. *По* таким системам, как правило, отсутствуют необходимые статистические данные с разработкой и применением его компонентов. В подобных случаях следует разрабатывать и выполнять специальные мероприятия, которые позволят обеспечить должный уровень документирования и соответствующее ***управление процессами создания документации***. В таких случаях применение обобщенных показателей необоснованно и должно быть **заменено на экспертную оценку профессионалов**, обладающих достаточным опытом и профессиональными навыками в сфере разработки подобного программного обеспечения. Следует отметить, что общие закономерности для направления документирования, как показывает ***опыт***, в применении к отдельным типам документов, сохраняется и в случае работы над уникальными информационными системами.

Выводы

Были рассмотрены аспекты, связанные с характеристиками программного обеспечения и требований к ним, которые позволяют сформировать конкурентоспособную для конкретного окружения, архитектуру программного продукта.

Были определены внешние события, прямо не относящиеся к рамкам создаваемого программного продукта, но которые влияют и формируют образ, как архитектуры, так и самой информационной системы.

Архитектура и архитектурное проектирование - это комплексное понятие, состоящее из *множества*, казалось бы, не совсем связных, частей, которые испытывают на себе внешнее влияние *множества* факторов, но при этом и сами оказывают на них влияние, трансформируя существующие и порождая новые информационные потоки. От того, насколько качественными и соответствующими актуальным и стратегическим потребностям они окажутся, будут судить о качестве породивших их программных продуктов и оптимальности архитектур, находящихся в их базисе.

Наша основная цель состоит в том, чтобы выстроить качественный процесс проектирования с учетом факторов и условий, определяющих его эффективность и *оптимальность* результата, которым должна являться *архитектура* программного продукта.

Задание:

1. Спроектировать архитектуру ПО ВКР, провести ее обоснование.

   (см. лекции 6,7 ПЖЦ 1-й семестр)

1. Использовать графические нотации UML для обоснования архитектуры.

   (см. материалы ПиАПС 1-й семестр)

Оформление : № группы, Ф.И.О. , номер, тема лабораторной работы, основной текст (структурированный, рисунки), выводы.

konst17@mail.ru