Шаблон описания архитектуры программного обеспечения

Шаблон составлен на основе шаблона архитектурного описания и книги «Архитектура программных систем» Эоина Вудса и Ника Розански, материалов The TOGAF® Standard, Version 9.2. и дополнен примерами представлений в формате - *.fm[1]

Шаблон поможет:

- собрать и систематизировать информацию об архитектуре, основных функциях и качественных характеристиках информационной системы;
- спроектировать решение с учетом требований заинтересованных лиц и обеспечить функциональную полноту и надежность системы;
- снизить вероятность сбоев, нарушений безопасности и рисков незапланированных затрат при развитии и эволюции системы.

Как начать работу:

- Скопировать шаблон в Google Диск: выберите меню Файл - Создать копию.
- Скачать шаблон в виде локальной копии: выберите меню Файл - Скачать.

Актуальная версия шаблона на Google Диске.

Перевод и дополнения шаблона: Юрченко Анна, 25 лет в ИТ, автор телеграм-канала IT Service Management for you об управлении ИТ, архитектуре, как создать лучший клиентский сервис и сделать ИТ силой бизнеса. Как показать ценность ИТ заказчику.

Предложения по развитию шаблона добавляйте в виде комментариев к файлу или пишите мне @itsm_lady

Если вам полезен шаблон, пройдите опрос из 2 вопросов: ваша отрасль и размер компании — перейти в опрос.

[1] Формат *.fm - формат функциональных моделей для АИАС "А-СТЕК"

История версий шаблона описания архитектуры решения:

Версия	Дата	Автор	Комментарии
1	23.12.2021	Anna Y	Перевод шаблона архитектурного описания Эоина Вудса и Ника Розански, дополнение материалами из книги «Архитектура программных систем» Ника Розански и Эоина Вудса, The TOGAF® Standard, Version 9.2., и шаблонами представлений в в формате - *.fm.

Вопросы и предложения:

п/п	Предложение	Что делаем?

Имя системы

Описание архитектуры программного обеспечения

Версия: номер версии

Дата: __.__.____

1. История версий

Версия	Дата	Автор	Комментарии

2. Оглавление

Оглавление пустое, так как стили абзацев, выбранные для отображения в оглавлении, не используются в документе.

Оглавление пустое, так как стили абзацев, выбранные для отображения в оглавлении, не используются в документе.

3. Инструкции

Шаблон предоставляет схему документа с описанием архитектуры программного обеспечения на основе представлений предложенных в книге «Архитектура программных систем, 2-е издание» Ника Розански и Эоина Вудса (Addison Wesley, 2011).

С книгой можно ознакомиться в библиотеке O'Reilly. Список литературы и ссылок документа см. в разделе Ссылки и литература.

Шаблон содержит типовое архитектурное описание. Если в шаблоне есть разделы, не относящиеся к вашему проекту, следует их исключить. Если есть темы, которые важны для вашей системы, но не включены в шаблон, следует добавить необходимые разделы.

Для получения дополнительной информации об архитектурном подходе, лежащем в основе шаблона, представленных концепциях и методах, используемых для создания эффективных архитектурных описаний, посетите веб-сайт www.viewpoints-and-perspectives.info

Об авторах:

Ник Розански - функциональный архитектор департамента ИТ фронт-офиса крупного британского банка. Он контролирует системный ландшафт всего департамента, а также обеспечивает архитектурное руководство и поддержку ключевых систем и проектов. Он работает в сфере ИТ с 1980 года в нескольких крупных и небольших компаниях, занимающихся системной интеграцией, включая Logica, Capgemini и Sybase.

Эоин Вудс - ведущий системный архитектор в технологической группе по акциям крупного европейского инвестиционного банка, отвечающий за архитектуру и проектирование ключевых систем организации. Он работает в области разработки программного обеспечения с 1990 года и за это время успел поработать в ряде технологических компаний, консультационных фирм и фирм, предоставляющих финансовые услуги.

4. Введение

4.1.Цель и сфера применения

Объясните цель и объем документа.

Рекомендуется избегать представления большого количества материалов, доступных в другом месте, также может быть полезно выполнить некоторые или все из следующих действий:

- Обобщите контекст, цели и задачи проекта
- Подтвердите объем и исключения
- Представьте обзор целей и драйверов, требований и т.д.
- Запишите важные принятые решения и их обоснование
- Представьте альтернативы и причины их отклонения

4.2. Аудитория

Определите заинтересованных сторон в архитектуре системы. В данном разделе необходимо перечислить заинтересованные стороны, не детализировать.

Заинтересованным лицом в архитектуре системы является человек, группа, или организация, заинтересованные в реализации системы.

Заинтересованные стороны попадают в одну из следующих групп:

- Пользователи системы, чей интерес в основном сосредоточен вокруг функционала, однако их может интересовать безопасность и производительность.
- Заинтересованные стороны, принимающие участие в проектировании и внедрении системы, например, разработчики, тестировщики.
- Заинтересованные стороны, на которых влияют архитектурные решения, например, те, кто будет использовать систему, администрировать, поддерживать.
- Заинтересованные стороны, которые влияют на форму и конечный успех разработки, например те, кто за это будет платить.
- Заинтересованные стороны, обладающие специальными знаниями в области бизнеса или технологий и могут проанализировать и оценить соответствие архитектуры требованиям, например, эксперты по применяемым в решении технологиям.
- Заинтересованные стороны, которые включены по организационным или политическим причинам, например, сотрудники службы контроля качества, безопасности, контроля соблюдения нормативных требований, управления архитектурой.

4.3.Статус

Опишите текущий статус архитектуры и архитектурного описания.

Архитектура только в стадии проектирования? Идет реализация решения? В производстве? Вы также можете описать планы по развитию документа архитектурного описания (например, он будет переиздан после комментариев, полученных от заинтересованных сторон).

4.4.Архитектурный подход к проектированию

Опишите архитектурный подход, используемый для описания и разработки содержания документа (например, объясните точки зрения (viewpoints), представления (views) и перспективы (perspectives)). При необходимости объясните, какие архитектурные представления вы используете и почему они используются.

5. Глоссарий

Определите термины и сокращения, которые могут быть незнакомы целевой аудитории. Они могут включать как коммерческие термины, так и термины, связанные с технологиями / архитектурой.

Рекомендуем создать отдельный глоссарий на уровне компании с основными архитектурными терминами, а в документе определять специфические термины связанные с проектируемой системой.

Термин	Определение
срок	определение термина

6. Заинтересованные стороны системы и требования

6.1.Заинтересованные стороны

Определите каждую из ключевых заинтересованных сторон и группы заинтересованных сторон, объяснив их интересы, потребности и опасения по поводу системы.

Можете распределить заинтересованных лиц по уровням важности, для определения стратегии взаимодействия. Это может быть сделано как на представленной модели.

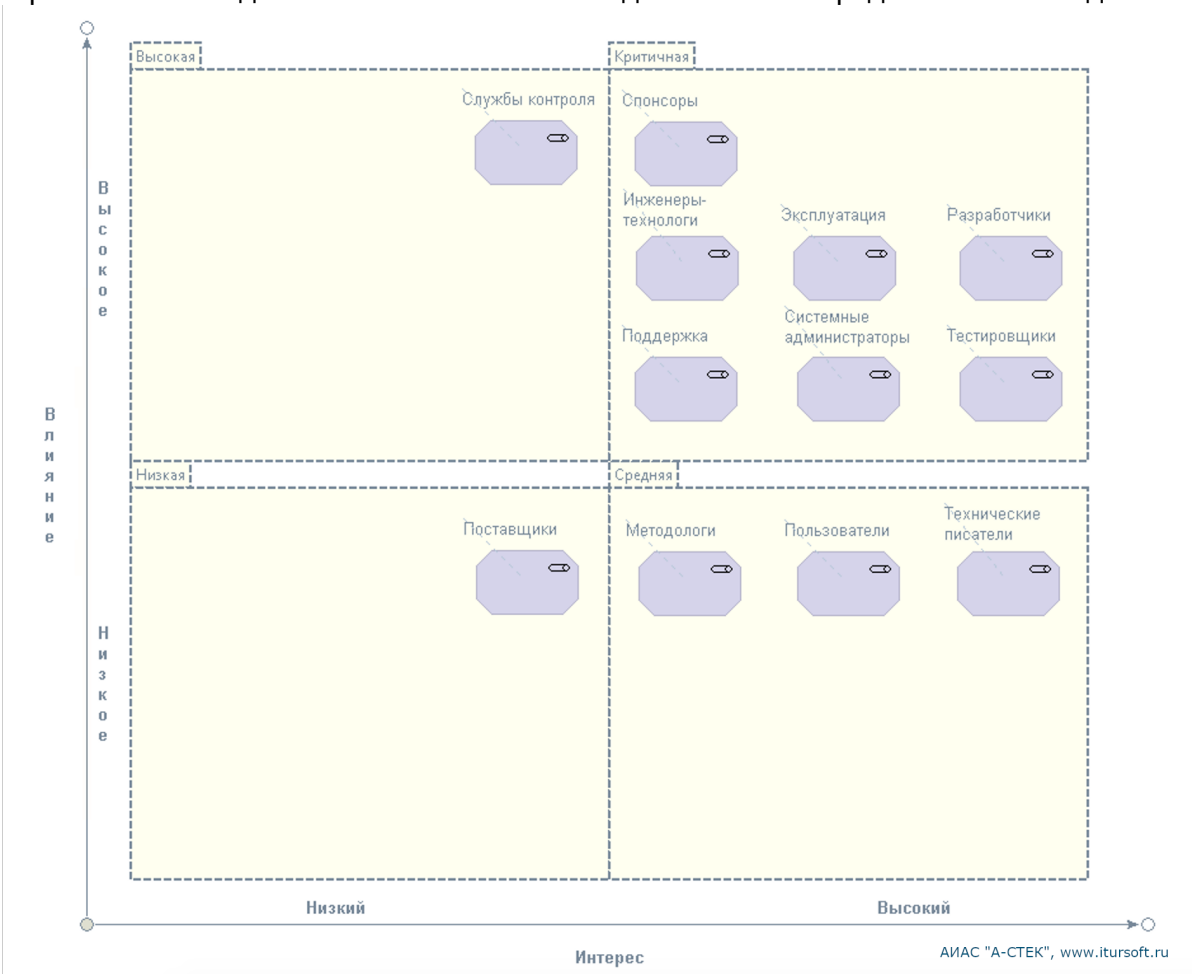


Рисунок 1. Представление заинтересованных сторон

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

Заинтересованная сторона - это любой, кто заинтересован в системе, задокументированной в AD. Рассмотрим следующие группы заинтересованных сторон.

- Спонсоры, которые платят за систему.

Спонсоры наблюдают за закупкой или затратами на систему. Спонсоры обычно включают высшее руководство, которое обеспечивает или санкционирует финансирование разработки, отдел закупок и юридический отдел, которые представляют коммерческие интересы пользователей в переговорах со сторонними поставщиками.

В этой группе также может быть представительство инвестора, если для финансирования проекта требуются особые внешние инвестиции. С точки зрения старшинства покупатели обычно являются вашими наиболее важными заинтересованными сторонами.

Заботы спонсоров обычно связаны с такими вопросами, как соответствие стратегическим целям, окупаемость инвестиций, а также затраты, сроки, планы и ресурсы, необходимые для создания и эксплуатации системы. Их цели обычно заключаются в соотношении цены и качества и эффективного расходования ресурсов во время доставки и эксплуатации.

- Службы контроля, которые проверяют на соответствие.

Подразделения контроля следят за соответствием системы стандартам и правовым нормам. Оценщики могут быть из собственных отделов внутреннего контроля качества или соответствия организации, либо они могут быть внешними юридическими лицами.

Оценщики озабочены тестированием (чтобы продемонстрировать соответствие требованиям) и формальным, очевидным соответствием.

- Технические писатели и методологи, которые создают документы и проводят обучение.

Методологи и тренеры объясняют и обучают системе других заинтересованных лиц, проводят обучение для сотрудников службы поддержки, разработчиков, специалистов по обслуживанию.Технические писатели создают руководства для пользователей и администраторов продукта или системы. В случае внешнего продукта отдел маркетинга должен подготовить материалы для потенциальных клиентов о его ключевых характеристиках, сильных сторонах и преимуществах.

Интересы этой группы заключаются в понимании причин и деталей архитектуры, а также в объяснении их технической и непрофессиональной аудитории.

- Разработчики, которые создают систему.

Разработчики создают и развертывают систему на основе спецификаций.. Разработчики должны понимать общую архитектуру, но также имеют требования, связанные с особенностями разработки, такими как стандарты сборки, выбор платформы, языка и инструментов, а также другие вопросы, такие как ремонтопригодность, гибкость и сохранение знаний с течением времени.

В эту категорию также входят менеджеры по развитию, которые планируют деятельность по разработке и возглавляют команды, выполняющие эту работу.

- Специалисты эксплуатации, которые развивают и исправляют систему.

Специалисты эксплуатации управляют развитием системы после ее ввода в эксплуатацию. Интересы группы связаны с документацией по разработке, инструментарием (средства для мониторинга операционной системы), средой отладки, контролем производственных изменений и сохранением знаний с течением времени.

- Инженеры-технологи, кто отвечает за среду развертывания.

Инженеры-технологи проектируют, развертывают и управляют аппаратными и программными средами, в которых система будет построена, протестирована и запущена. Они несут ответственность за серверные и настольные компьютеры; сетевую инфраструктуру; специальные устройства, такие как поисковые системы или шлюзы связи; периферийные и портативные устройства, за программное обеспечение инфраструктуры, такое как операционные системы, промежуточное ПО для обмена сообщениями и реляционные базы данных.

Большинство крупных организаций запускают свои вычислительные среды в центрах обработки данных, которые обеспечивают безопасную, защищенную и высокодоступную среду со сложными функциями, такими как резервные источники питания, избыточная передача данных с высокой пропускной способностью и средства контроля окружающей среды, такие как кондиционирование и системы пожаротушения. Помимо основных производственных сред, центры обработки данных обычно также содержат среды для аварийного восстановления, приемочного тестирования и, в некоторых случаях, разработки. Они представляют собой значительный финансовый актив, и инженеры-технологи тщательно и строго контролируют их.

Инженеры-технологи вносят значительный вклад в архитектуру развертывания любой системы, и их одобрение обычно требуется перед установкой какого-либо компонента в центр обработки данных.

- Поставщики, которые обеспечивают части системы.

Поставщики создают и / или поставляют оборудование, программное обеспечение или инфраструктуру, на которых будет работать система, или, возможно, они предоставляют специализированный персонал для разработки или эксплуатации системы.

Поставщики - это особый класс заинтересованных сторон: они обычно не участвуют в создании, эксплуатации или использовании системы, но могут налагать ограничения из-за ограничений или требований к поставляемой ими продукции.

Например, программное приложение может предписывать выполнение определенной версии операционной системы, или может работать только на определенных конфигурациях оборудования, или может накладывать ограничения на количество одновременных подключений или максимальный размер данных. Важно, чтобы вы учли такие ограничения при проектировании архитектуры.

- Поддержка, которая помогает пользоваться системой.

Персонал технической поддержки оказывает поддержку пользователям продукта или системы во время ее работы. Обеспокоенность сотрудников службы поддержки связана с наличием информации, необходимой для решения проблем с пользователями.

- Системные администраторы, кто содержит систему в рабочем состоянии.

Системные администраторы запускают систему после ее развертывания. В крупномасштабных коммерческих средах системные администраторы играют ключевую роль, поскольку работа системы имеет важное значение для непрерывности бизнеса. В некоторых сценариях, таких как продукты, предназначенные для внутреннего рынка ПК, системные администраторы также могут быть пользователями.

Системные администраторы могут сосредоточиться на широком спектре проблем, таких как системный мониторинг и управление, непрерывность бизнеса, аварийное восстановление, доступность, отказоустойчивость и масштабируемость.

- Тестировщики, которые проверяют, что это работает.

Тестировщики систематически тестируют систему, чтобы установить, подходит ли она для развертывания и использования. Хотя разработчики также проводят тестирование, тестировщики должны быть независимыми и не иметь такого же чувства собственности на реализацию системы. Это, наряду с их специальными знаниями и опытом, означает, что они могут выполнять более тщательную и объективную работу по оценке системы, чем другие заинтересованные стороны. Тестировщики занимаются установлением требований, разработкой тестов, чтобы доказать, что требования были выполнены, и построением систем, на которых они будут запускать свои тесты.

- Пользователи, которые должны использовать систему напрямую.

Пользователи определяют функциональность системы и в конечном итоге будут ей пользоваться. В случае внутренних систем пользователи - это внутренний персонал, который может иметь дело с клиентами или выполнять функции поддержки. В случае программного продукта конечными покупателями продукта являются пользователи; здесь необходимо, чтобы кто-то представлял их интересы, например, путем тестирования рынка.

Беспокойство пользователей сосредоточено вокруг функциональности; однако у них также есть операционные проблемы, такие как производительность и безопасность.

- Сотрудники отдела информационной безопасности.
- И, конечно же, архитектор также является заинтересованным лицом в архитектурном описании.

6.2.Обзор требований

Обобщите ключевые функциональные и нефункциональные требования к системе.

Функциональные требования определяют, что система должна делать (например, обновлять имя и адрес клиента). Нефункциональные требования определяют, как

система должна вести себя во время работы или развития (например, она должна отвечать на запросы в течение трех секунд при заданной нагрузке; она должна быть доступна 99,99% времени; она должна быть возможность расширения системы для удовлетворения определенных типов новых требований без необходимости проведения серьезной модернизации).

Избегайте вдаваться в детали, которые представлены в другом месте; ссылайтесь на внешние источники, такие как документы с требованиями, SLA, существующие системы и т. д., где это возможно. Требования должны быть пронумерованы, чтобы вы могли однозначно ссылаться на них в другом месте.

См. Подробная пошаговая инструкция с примерами, чтобы правильно собрать требования заказчика и результат соответствовал ожиданиям.

Ссылка	Требование Описание

6.3.Сценарии системы

Перечислите и кратко опишите наиболее важные сценарии, которые важны для ключевых заинтересованных сторон и / или могут быть использованы для иллюстрации способности системы удовлетворять наиболее важные требования.

Сценарий описывает ситуацию, с которой система может столкнуться в своей производственной среде, а также необходимые реакции системы. Вы должны учитывать как функциональные сценарии (действия, которые система должна делать обычно в ответ на внешнее событие или входные данные), так и сценарии нефункциональных требований системы (то, как система должна реагировать на изменения в своей среде, такие как увеличение рабочей нагрузки).

В большинстве случаев сценарии занимают значительный объем места, и часто целесообразно записывать их в отдельный документ, чтобы архитектурное описание не становилось слишком большим.

Сценарии - полезный архитектурный прием, который используют в разных ситуациях, включая общение с заинтересованными сторонами, оценку и анализ архитектуры, в качестве спецификаций для тестирования в реальных условиях.

Качества хорошего сценария:

- Достоверность: Сценарий должен описывать реалистичную ситуацию, которая может произойти, и должен включать достаточно реалистичных деталей, чтобы читатель мог принять сценарий за действительную ситуацию, с которой может столкнуться система.

- Ценность: сценарий должен быть полезен в архитектурном процессе, будь то объяснение архитектуры заинтересованным сторонам, убеждение эксперта в правильности архитектуры или иллюстрация работы архитектуры для команды разработчиков. Легко увлечься определением сценариев, которые на самом деле не касаются проблем заинтересованных сторон, поэтому учитывайте это при создании новых сценариев.

- Конкретность: Хороший сценарий достаточно конкретен и точно описывает определенную ситуацию, а не пытается обобщить поведение системы на целый класс ситуаций. Опасность при попытке обобщить сценарии за пределы конкретных ситуаций заключается в том, что становится трудно описать их кратко, поэтому их трудно использовать, так как в них рассматривается много различных специфических ситуаций, каждая из которых имеет свои вариации.

- Точность: Определение сценария должно быть достаточно точным, чтобы предполагаемому пользователю сценария было совершенно ясно, какую ситуацию описывает сценарий и какой ответ требуется от системы.

- Понятность: Как и все архитектурные документы, сценарии должны быть понятны тем заинтересованным сторонам, которые должны их использовать. Это означает, что они должны быть написаны четко, с использованием общепонятных терминов и без сокращений и жаргона, которые заинтересованные стороны, скорее всего, не поймут или сочтут запутанными.

6.3.1. Функциональные сценарии

Функциональные сценарии моделируют вещи, которые система должна выполнять в ответ на внешний стимул (например, событие или вход).

Функциональные сценарии почти всегда определяются в терминах последовательности внешних событий (получаемых из сценария использования системы), на которые система должна реагировать определенным образом. Примерами могут служить инициирование пользователями транзакций, поступление данных через внешние интерфейсы, наступление временных событий (таких как конец дня) и т.д.

Функциональные сценарии часто документируются в виде сценариев использования. Эта техника была первоначально сформулирована Иваром Якобсоном, а диаграмма сценариев использования впоследствии стала частью UML. В сценарии использования система рассматривается как "черный ящик", описывающий, что она должна делать, чтобы выполнить задачу для определенного субъекта. Хотя они широко используются для создания функциональных спецификаций, создание сценариев качества системы с помощью сценариев использования может быть затруднено.

Ссылка на сценарий	Короткий уникальный идентификатор и краткое уникальное и описательное название.
Обзор	Краткое описание того, что демонстрирует сценарий.
Состояние системы	Состояние системы до реализации сценария (если это важно). Обычно это объяснение любой информации, которая уже должна быть сохранена в системе, чтобы сценарий имел смысл.
Окружение системы	Любые существенные наблюдения о среде, в которой работает система.
Внешний стимул	Определение того, что вызывает возникновение сценария, например, поступление данных на интерфейс, ввод данных пользователем, течение времени или любое другое событие, имеющее значение для системы.
Требуемый ответ системы	Объяснение, с точки зрения внешнего наблюдателя, как система должна реагировать на сценарий.

6.3.2.Нефункциональные сценарии

Нефункциональные сценарии (сценарии качества) системы моделируют, как система должна реагировать на изменение среды (например, увеличение рабочей нагрузки или нарушение безопасности).

Сценарии качества системы определяются в терминах того, как система должна реагировать на изменения в окружающей среде, чтобы проявить одно или несколько свойств качества. Сценарии качества включают безопасность, производительность, доступность и развитие. Примеры сценариев качества системы включают способность системы быть модифицированной для обеспечения новой функции, справляться с

определенным типом пиковой нагрузки, защищать критическую информацию, даже если часть инфраструктуры безопасности скомпрометирована, и так далее.

Ссылка на сценарий	Короткий уникальный идентификатор и краткое уникальное и описательное название.
Обзор	Краткое описание того, что демонстрирует сценарий.
Окружение системы	Любые существенные наблюдения о среде, в которой работает система, например, недоступность внешних систем, особенности поведения инфраструктуры, ограничения по времени и т.д.
Изменения окружающей среды	Определение того, что вызывает возникновение сценария.
Требуемое поведение системы	Описание того, как система должна реагировать на изменения окружающей среды

7. Архитектурные факторы

7.1. Цели

Перечислите основные архитектурные цели и бизнес-факторы проекта.

Цель - это то, чего хочет достичь проект (например, упростить процессы управления клиентами), в то время как движущей силой бизнеса является некоторая внешняя сила, которая формирует проект (например, уровень жалоб клиентов увеличивается).

Задайте себе следующие вопросы: Какие ключевые цели вы ставите перед собой как архитектор? Вы стремитесь повторно использовать существующее программное обеспечение? Или разработать систему с минимальными затратами? Или вы стремитесь к очень высокой надежности? Или … ?

7.2.Ограничения

Перечислите основные архитектурные ограничения, которые должны соблюдаться в проекте.

Ограничение - это то, что ограничивает ваш архитектурный выбор: например, проект должен быть завершен к определенному срок, должен быть реализован на определенном стеке технологий или соответствовать определенной операционной модели.

Вы также должны ссылаться на любые конкретные стандарты или правила, регулирующие архитектуру.

7.3.Архитектурные принципы

Объясните принципы архитектурного проектирования, которые сформировали архитектуру.

Принцип - это фундаментальное изложение убеждения, подхода или намерения, которым руководствуется определение архитектуры. Это может относиться к текущим обстоятельствам или желаемому будущему состоянию. Хороший принцип конструктивен, аргументирован, четко сформулирован, проверяем и значим.

Каждый принцип должен быть обоснован логическим обоснованием и может быть дополнен некоторыми последствиями. Например, принцип использования открытых стандартов может иметь в качестве обоснования стремление к функциональной совместимости и, как следствие, необходимость оценки и согласования соответствующих стандартов, которые применяются к каждому компоненту.

В крупных организациях рекомендуем перечень принципов определять на уровне компании и закреплять внутренним нормативным документом, например, таким как Техническая политика. С подходом к формированию и примерами принципов TOGAF можно ознакомиться на сайте Open Group.

Ссылка на принцип	(уникальный номер)
Название	Должно отражать суть правила, а также быть легко запоминающимся. Конкретные технологические платформы не должны упоминаться в названии или утверждении принципа. Избегайте двусмысленности.
Заявление	Должно лаконично и недвусмысленно передавать основное правило. По большей части, заявления о принципах управления информацией схожи в разных организациях. Очень важно, чтобы формулировка принципов была однозначной.

Обоснование	Следует подчеркнуть преимущества соблюдения принципа для бизнеса, используя деловую терминологию. Укажите на сходство архитектурных принципов с принципами, регулирующими бизнес-операции. Опишите ситуации, в которых один принцип будет иметь приоритет или больший вес, чем другой, при принятии решения.
Воздействие	Должны быть указаны требования, как для бизнеса, так и для ИТ, для реализации принципа - с точки зрения ресурсов, затрат и действий/задач. Может быть, что текущие системы, стандарты или практики будут противоречить принципу после его принятия. Необходимо четко указать влияние на бизнес и последствия принятия принципа. Читатель должен легко понять ответ на вопрос: "Как это повлияет на меня?".
Дальнейшая информация	

8. Архитектурные представления

8.1.Представление контекста

Контекстное представление системы описывает отношения, зависимости и взаимодействия между системой и ее средой (людьми, системами и внешними объектами, с которыми она взаимодействует).

8.1.1.Контекстная диаграмма

Используйте контекстную диаграмму (и вспомогательное объяснение), чтобы объяснить среду, в которой работает система, и внешние объекты, с которыми она взаимодействует. Кратко опишите каждый из внешних объектов и важные взаимодействия.

Контекстная диаграмма обычно представлена как простое высокоуровневое изображение, показывающее границы системы и прилегающие к ней внешние объекты. Внешними объектами обычно являются другие системы, но они также могут быть физическими устройствами, внешними организациями «черного ящика» или более детализированными программными компонентами. Взаимодействия могут быть потоками данных (интерфейсами) или потоками управления (например, вызовом службы или открытой функции).

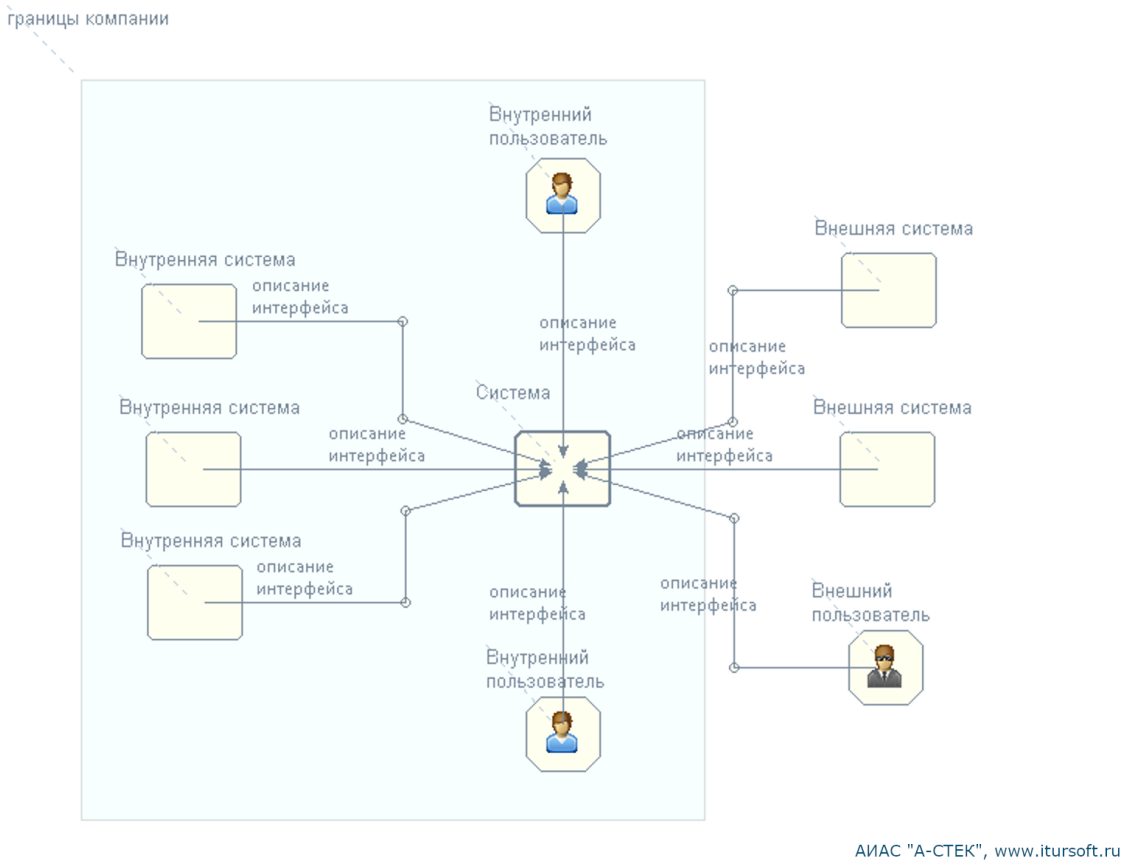


Рисунок 2. Контекстная диаграмма

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

Обычно вы представляете саму систему на схеме в виде единого блока или компонента. Внешние сущности также обычно представлены как отдельные блоки или компоненты, поскольку вы часто не знаете (или не заботитесь) об их внутренней реализации.

Пример контекстной диаграммы приведен выше. Здесь используется «нейтральное» обозначение прямоугольников и линий; вместо этого вы можете использовать более формальный язык моделирования, например, UML.

8.1.2.Сценарии взаимодействия

Если у вас сложное взаимодействие между вашей системой и внешними объектами, рассмотрите возможность моделирования некоторых ожидаемых последовательностей взаимодействия с использованием сценариев взаимодействия. Это может помочь выявить неявные требования и ограничения (например, порядок, объем или временные ограничения) и помочь обеспечить дополнительный, более подробный уровень проверки.

Вы можете фиксировать последовательности взаимодействий с помощью диаграмм последовательностей UML или маркированных списков взаимодействий.

8.2.Функциональное представление

Функциональное представление системы определяет архитектурно значимые функциональные элементы системы, обязанности каждого, интерфейсы, которые они предлагают, и зависимости между элементами.

Разместите здесь функциональную модель (например, схему компонентов UML) и объясните ее содержание в подразделах ниже. Функциональный элемент - это четко определенная часть системы, которая имеет определенные функциональные задачи и предоставляет интерфейсы, которые соединяют ее с другими функциональными элементами.

Сосредоточьтесь на важных функциональных элементах архитектуры. Здесь не нужно моделировать базовую инфраструктуру, если она не выполняет функционально значимую цель (например, шина сообщений, которая связывает элементы системы и преобразует данные, которыми обмениваются между ними).

Если ваша архитектура функционально сложна, вы можете смоделировать ее на высоком уровне, а затем разложить некоторые элементы на дополнительные подмодели (функциональная декомпозиция).

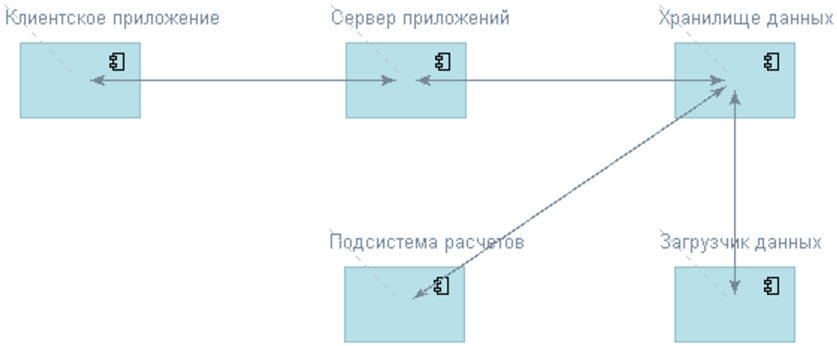


Рисунок 3. Функциональная модель

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

8.2.1.Функциональные элементы

Определите роль и интерфейсы, предлагаемые и / или требуемые каждым функциональным элементом. В качестве альтернативы, если вы используете подход к моделированию, такой как UML, вы можете сохранить основные описания в репозитории моделей UML и суммировать здесь информацию со ссылкой на модель (модели).

Если вы использовали функциональную декомпозицию в предыдущем разделе, вы можете структурировать этот раздел в соответствии с вашей функциональной иерархией.

Имя элемента	
Ответственность	
Интерфейсы - входящие	
Интерфейсы - исходящие	

8.2.2.Функциональные сценарии

Используйте одну или несколько диаграмм взаимодействия, чтобы объяснить, как функциональные элементы взаимодействуют через свои интерфейсы, чтобы соответствовать некоторым ключевым функциональным сценариям системы.

8.2.3.Общесистемная обработка событий

Определите, как будет обрабатываться любая общесистемная обработка (например, если у вас есть система, ориентированная на сообщения, как вы будете бороться с ошибками доставки сообщений в системе).

8.3.Представления информации

Информационные представления системы определяет структуру хранимой в системе информации (например, базы данных и схемы сообщений) и то, как будут решаться связанные аспекты, такие как владение информацией, потоки, обмен, задержки и сроки хранения.

8.3.1.Структура данных

Определите или укажите любые архитектурно значимые структуры данных для постоянных и временных данных, такие как представления модели данных или схемы сообщений.

На этом уровне вы должны поддерживать небольшое количество объектов - не более 20, если возможно. Необязательно быть на 100% нормализованным - для ясности допустимо наличие, например, некоторых отношений «многие ко многим». Не пытайтесь проиллюстрировать здесь каждую сущность и взаимосвязь, иначе ваши читатели потеряются в деталях.

Также может быть полезно логически сгруппировать вместе объекты, которые каким-то образом связаны семантически - например, все данные, связанные с именем и адресом клиента. Это может помочь вашим читателям понять элементы данных и отношения между ними.

Здесь можете использовать нотацию Archimate и представление Conceptual Data Architecture, диаграммы классов UML, возможна вторая может быть слишком детализированным уровнем детализации для архитектурного описания.

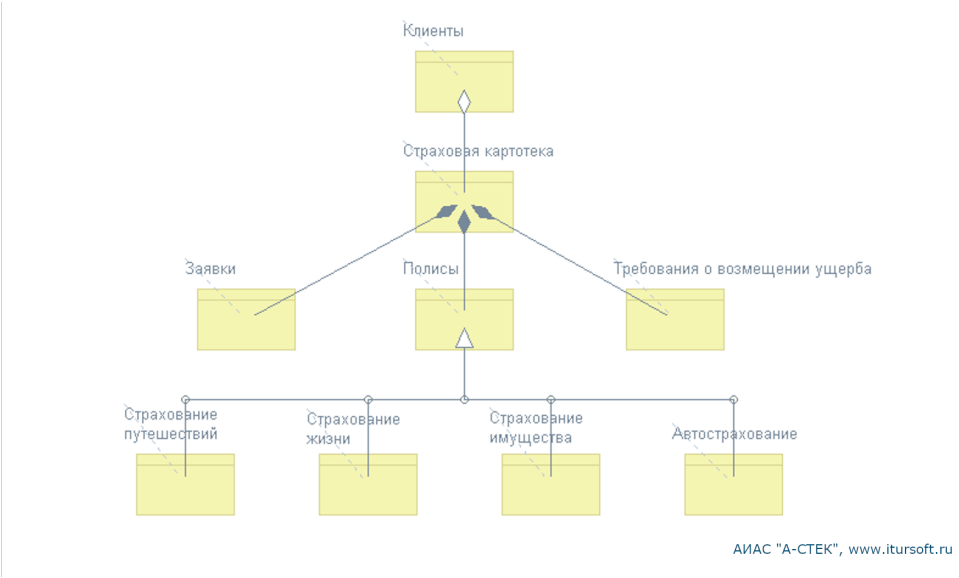


Рисунок 4. Концептуальная архитектура данных

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

8.3.2.Потоки данных

Если это не ясно из диаграмм взаимодействия функционального представления, определите, как данные проходят через систему от одного компонента к другому и к внешним компонентам.

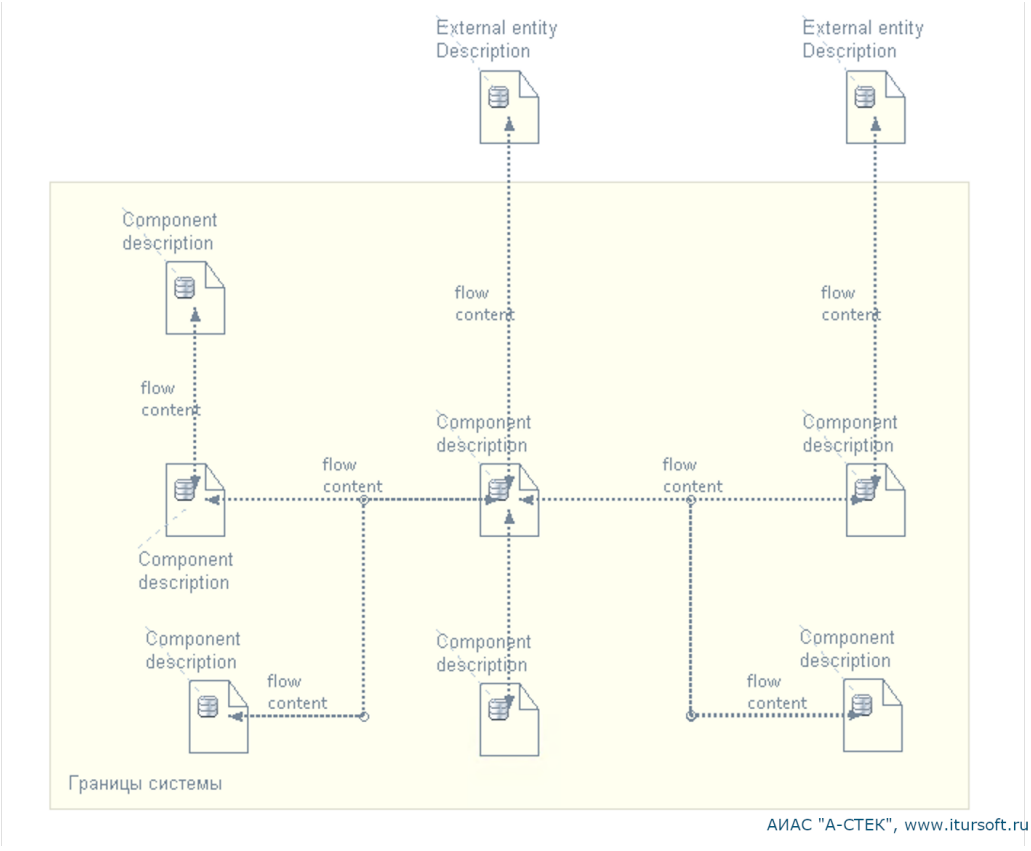


Рисунок 5. Потоки данных системы

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

Как и в случае с диаграммой структуры данных, сохраняйте простоту и сосредоточьтесь не более чем на 10-15 ключевых функциональных элементах. Выше показан пример с использованием диаграммы потока данных.

8.3.3.Право владения данными

Если данные принадлежат более чем одному объекту или части системы, определите, кому принадлежат какие части данных, и объясните, как будут решаться возникающие проблемы.

В приведенном ниже примере можно увидеть, что существуют проблемы с объектом 4, который может быть обновлен системой D, которая не является владельцем. В архитектурном описании должно быть объяснено, как это несоответствие будет устранено.

Сущность	Система A	Система B	Система C	Система D
Сущность 1	ВЛАДЕЛЕЦ	читать копировать	читать	читать
Сущность 2	читать	ВЛАДЕЛЕЦ	-	читать
Сущность 3	-	читатель	ВЛАДЕЛЕЦ	читать
Сущность 4	ВЛАДЕЛЕЦ	-	-	читать изменять удалять

8.3.4.Жизненный цикл информации

Если ключевые объекты имеют сложные жизненные циклы, смоделируйте то, как их состояние изменяется с течением времени.

Сосредоточьтесь на нескольких ключевых объектах, чьи переходы помогают осветить ключевые особенности архитектуры, а не просто созданы / обновлены / уничтожены.

Существует два распространенных метода моделирования жизненных циклов информации, историй жизни сущностей и диаграмм перехода состояний. Оба полезны; выберите один стиль и придерживайтесь его.

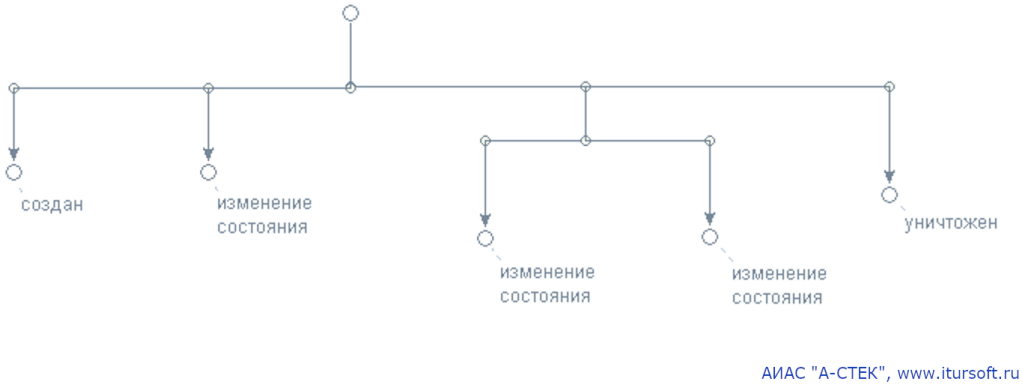


Рисунок 6. История жизни сущности

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

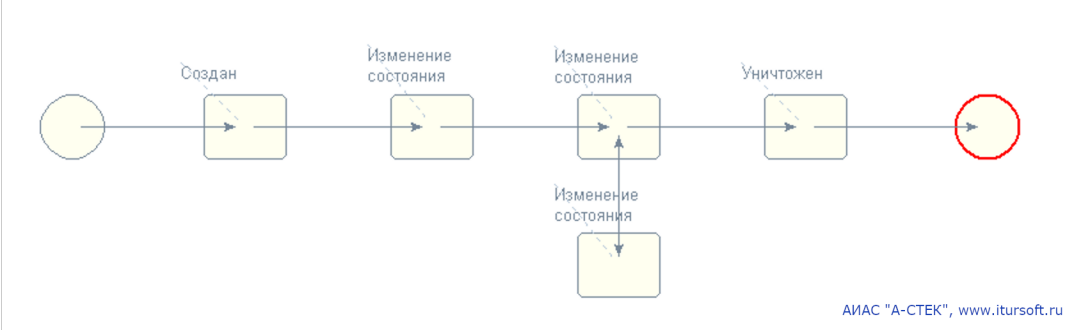


Рисунок 7. Изменение статусов сущности

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

8.3.5.Своевременность, латентность и возраст данных

Если информацию необходимо копировать в системе или регулярно обновлять, объясните, как будут выполняться требования к своевременности и латентности.

Если ваша информация хранится в едином хранилище данных и доступ к ней всегда осуществляется синхронно в режиме реального времени, своевременность, задержка и возраст могут не представлять существенной проблемы. К сожалению, многие системы работают иначе, и неизбежно, что в некоторых сценариях будет использоваться устаревшая или неактуальная информация, хотя бы на несколько минут.

Вы должны определить ключевые моменты, в которых могут возникнуть несоответствия, основанные на времени, и с помощью заинтересованных сторон разработать стратегии их устранения, например, следующие.

- Помечать важные элементы данных датой и временем "последнего обновления".

- Определить "временные окна" для важных элементов данных.

- Предупреждать пользователей, когда информация может быть устаревшей.

- Скрывать или отбрасывать информацию, которая может быть слишком старой.

- Снизить задержку за счет более быстрых интерфейсов или прямого доступа к источникам данных.

8.3.6.Архив и хранение информации

Объясните, как система будет выполнять требования к архивации и хранению.

Во многих системах информация удаляется все реже; она может храниться по юридическим причинам или для исторического анализа. Хотя дисковые хранилища сейчас относительно недороги, управление большими базами данных - сложный процесс, и даже дисковые архитектуры предприятий не могут расширяться бесконечно, поэтому рано или поздно ваша информация вырастет до такой степени, что хранить ее всю в сети будет нежелательно. Тогда вам придется архивировать старую, менее полезную информацию на другой носитель, например, на автономное хранилище большой емкости.

Вы должны тщательно определить объем информации для архивирования. Очевидно, что это не может быть информация, которая все еще необходима для поддержки какой-либо производственной деятельности, а также информация, которая может быть полезна для регулярного анализа. Обычно информация отбирается на основе возраста в сочетании с бизнес-правилами для определения ее полезности.

Стратегия архивирования может оказать значительное влияние на архитектуру.

- Архивирование больших объемов информации может сделать некоторые системы полностью или частично недоступными в течение значительного периода времени.

- При определении размера физического диска необходимо учитывать продолжительность времени, в течение которого будет храниться информация.

- Вам может понадобиться определить процессы, которые перемещают производственную информацию на архивные носители.

- Возможно, потребуется предпринять действия для обеспечения целостности и согласованности производственного и архивного хранилищ.

- Если архивное хранилище удалено, это может повлиять на сетевую инфраструктуру.

С самого начала проектируйте архитектуру таким образом, чтобы архивирование было естественной частью жизненного цикла информации.

8.4.Параллельное представление

Параллельное представление системы определяет набор работающих элементов во время выполнения (таких как процессы операционной системы), в которые упакованы функциональные элементы системы.

Если параллельная структура сложна или неочевидна из информации в других представлениях, определите, как функциональные элементы будут упакованы в процессы и потоки, и объясните, как они безопасно и надежно взаимодействуют, используя подходящие механизмы межпроцессного взаимодействия. Этого можно достичь с помощью модели UML (с использованием стереотипов), с помощью специального языка моделирования параллельного представления или путем создания неформальной нотации для данной ситуации.

Пояснения:

Исторически сложилось так, что информационные системы проектировались для работы с небольшим или отсутствующим одновременным использованием, работая в пакетном режиме на больших центральных компьютерах. Однако ряд факторов (включая распределенные системы, растущие рабочие нагрузки и дешевое многопроцессорное оборудование) привели к тому, что современные информационные системы часто практически не имеют пакетной обработки и по своей сути являются параллельными.

В отличие от них, системы управления всегда были по своей природе параллельными и событийными, учитывая их необходимость реагировать на внешние события для выполнения операций управления. Поэтому вполне естественно, что по мере того, как информационные системы становятся все более параллельными и событийными, они начинают приобретать ряд характеристик, традиционно ассоциируемых с системами управления. Для того чтобы справиться с этим параллелизмом, разработчики информационных систем переняли и адаптировали проверенные методы из сферы систем управления. Многие из этих методов составляют основу точки зрения о параллельности.

Параллельное представление используется для описания параллельности системы и структуры и ограничений, связанных с состоянием. Это включает определение частей системы, которые выполняются одновременно и как это контролируется (например, определение того, как функциональные элементы системы упаковываются в процессы операционной системы и как процессы координируют свое выполнение). Для этого необходимо создать модель процесса и модель состояния: Модель процесса показывает планируемую структуру процессов, потоков и межпроцессного взаимодействия; модель состояния описывает набор состояний, в которых могут находиться элементы времени выполнения, и допустимые переходы между этими состояниями.

После создания моделей процессов и состояний вы можете использовать ряд техник анализа, чтобы убедиться в правильности запланированной схемы параллельности. Использование таких методов обычно является частью создания параллельного представления (Concurrency view).

Стоит отметить, что не все информационные системы получают реальную пользу от параллельного представления. Некоторые информационные системы имеют мало параллельности. Другие, демонстрируя параллельное поведение, используют средства базовых структур и контейнеров (например, серверов приложений и баз данных), чтобы скрыть используемую модель параллельности.

8.4.1.Параллельная модель

Смоделируйте процессы, группы процессов и потоки, а также каналы межпроцессного взаимодействия между ними.

Вы также можете смоделировать механизмы, используемые для защиты целостности данных и других ресурсов, совместно используемых параллельными исполнительными модулями, такими как мьютексы или семафоры.

Вы можете использовать модель компонентов UML для графического представления информации, создавая соответствующие стереотипы для компонентов.

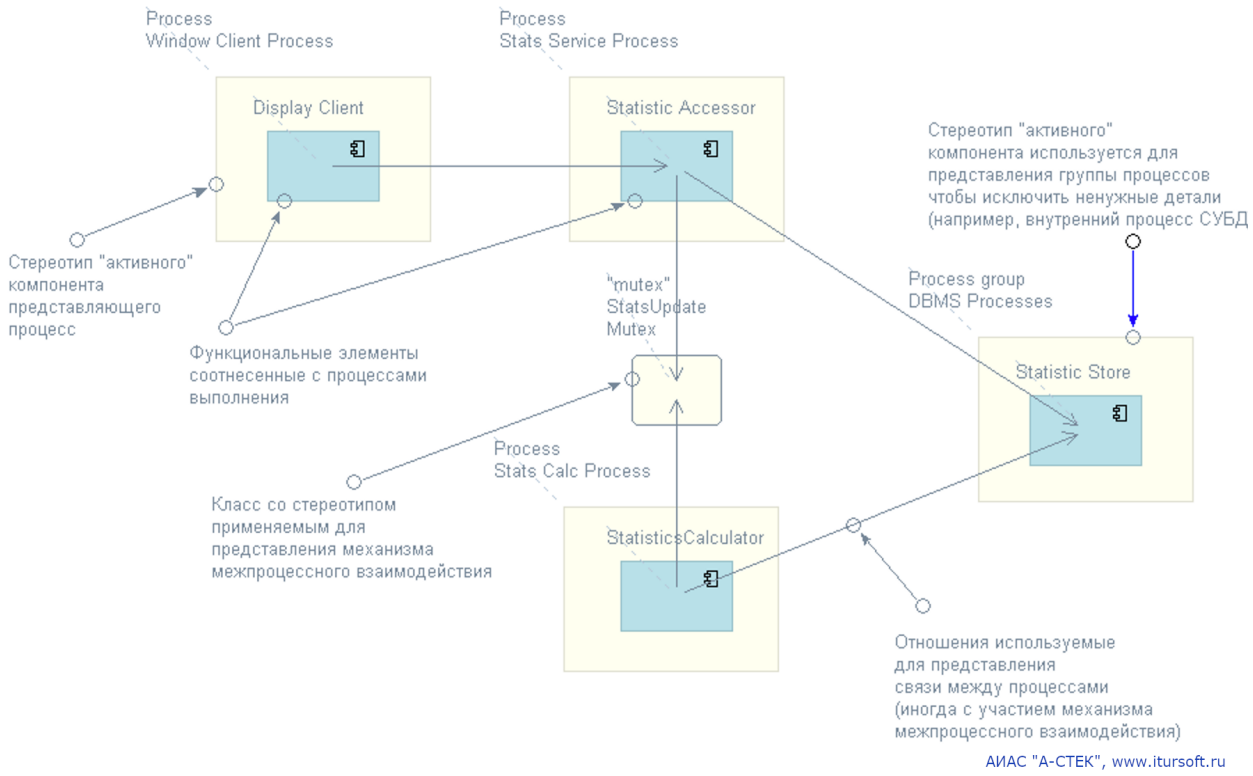


Рисунок 8. Параллельная модель

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

8.4.2.Модель состояния

Смоделируйте состояния, в которых могут находиться элементы среды выполнения системы, переходы между этими состояниями и события, которые управляют этими переходами.

Состояние - это идентифицированное, именуемое стабильное состояние, которое возникает во время работы системы. Событие - это то, что происходит, вызывая переход элемента из одного состояния в другое. Действия также могут быть связаны с переходами, так что, пока элемент меняет состояние, действие выполняется.

Сосредоточьтесь на нескольких ключевых элементах, состояния и переходы которых помогают осветить ключевые особенности архитектуры.

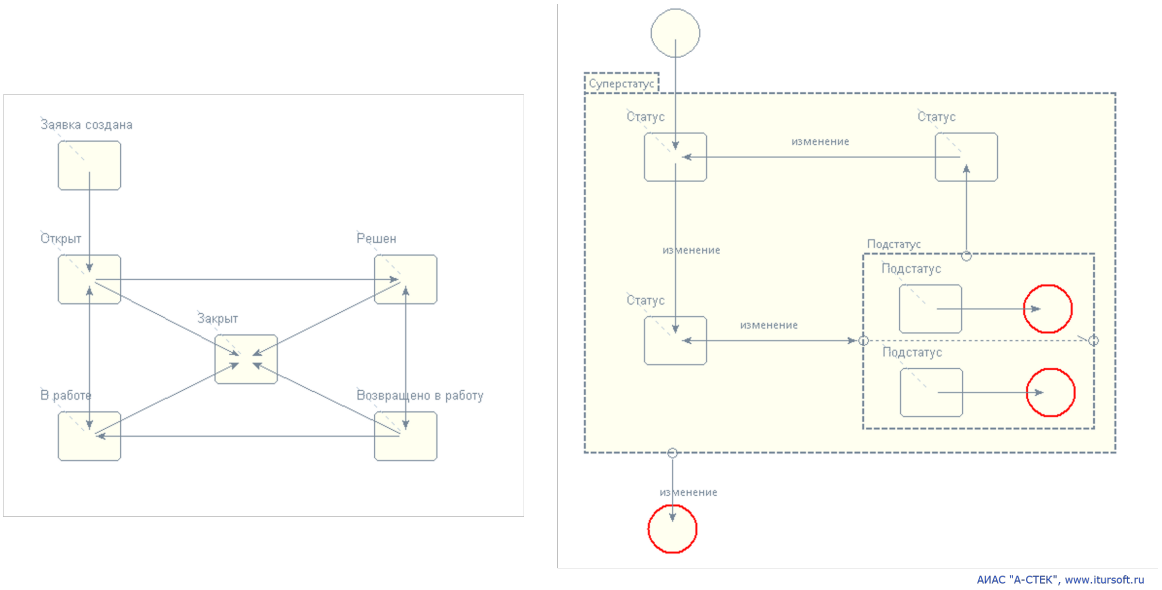


Рисунок 9. Модели статусов

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

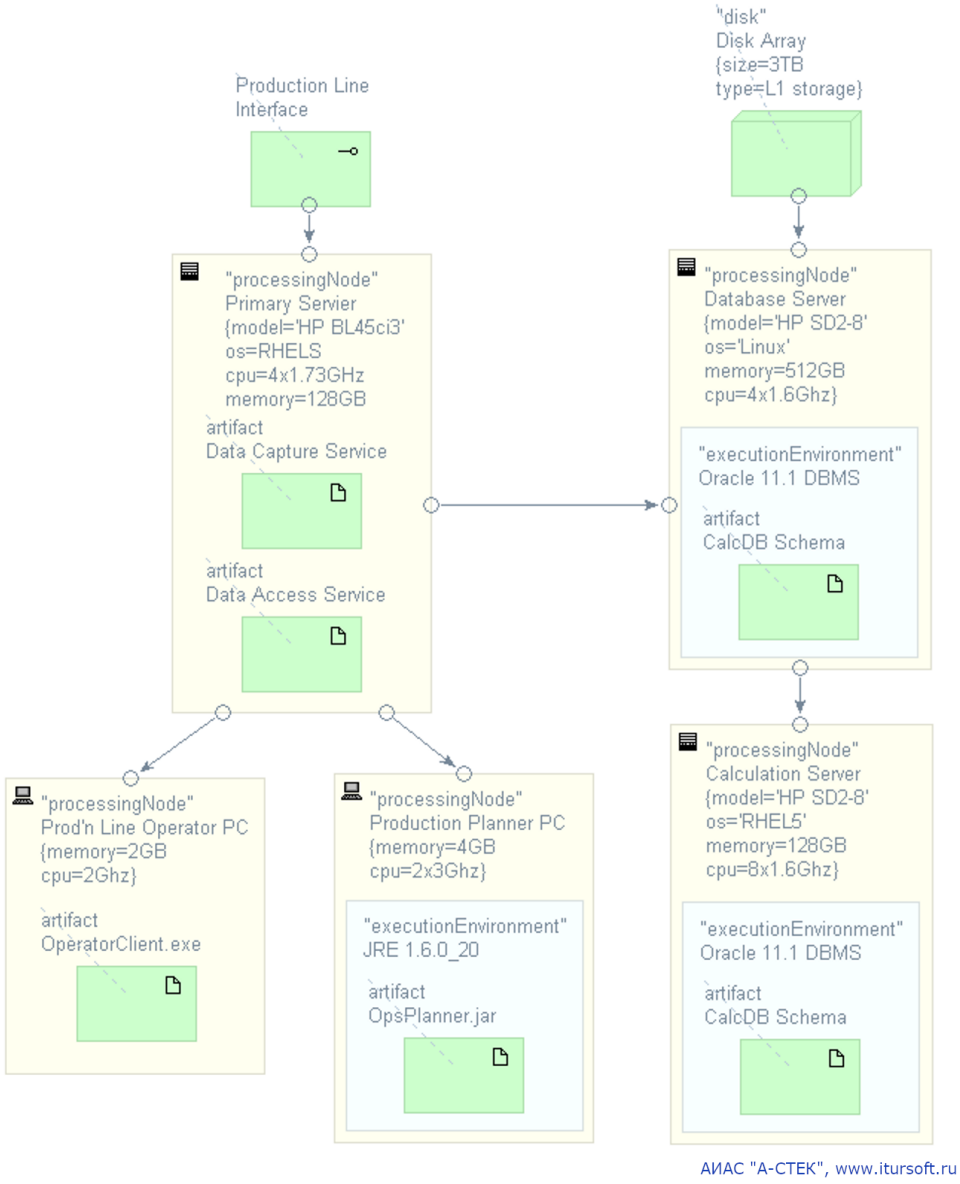
8.5.Представление развертывания

Представление «Развертывание» системы определяет важные характеристики операционной среды развертывания системы. Это представление включает в себя подробную информацию об узлах обработки, которые требуются системе для ее установки (т. е. ее исполняющая платформа), зависимости программного обеспечения на каждом узле (например, требуемые библиотеки) и сведения о базовой сети, которая потребуется системе.

8.5.1.Модель платформы во время исполнения (runtime)

Показать рабочую платформу системы (определение узлов, связей и сопоставление функциональных элементов или процессов с узлами).

Здесь вы можете использовать диаграмму развертывания UML или более простую диаграмму в виде прямоугольников и линий.



Фигура 9. Модель развертывания

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

Часто бывает полезно явно сопоставить функциональные элементы с узлами, на которых они будут работать, особенно если модель развертывания сложна или сопоставления неочевидны.

Функциональный элемент	Узлы развертывания
элемент	узел (ы)
элемент	узел (ы)

8.5.2.Программные зависимости

Определите программное обеспечение, которое потребуется на различных типах узлов в модели платформы во время исполнения, чтобы поддерживать систему (например, требования к операционной системе, системному программному обеспечению или библиотеке). Если версии известны, вы должны указать их. Четко укажите все известные зависимости версий (например, для компонента A требуется как минимум версия X компонента B).

Обычно это можно представить в виде таблицы.

8.5.3.Сетевая модель

Если сетевые требования являются сложными, включите сетевую модель, которая иллюстрирует узлы, каналы и сетевое оборудование, которые требуются системе, делая четкие требования к качеству обслуживания.

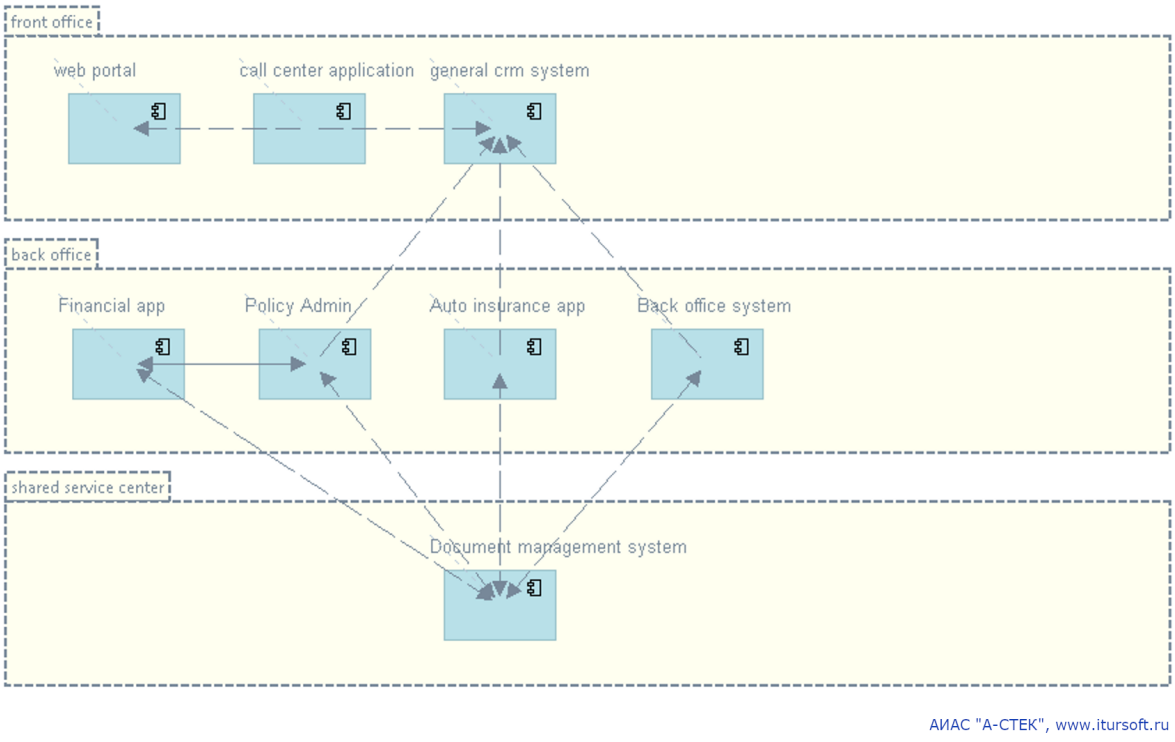
8.6.Представление разработки

Представление разработки системы определяет любые ограничения процесса разработки программного обеспечения, которые требуются архитектурой. Они включают в себя организацию модулей системы, общую обработку, которую должны реализовать все модули, любую требуемую стандартизацию проектирования, кодирования и тестирования, а также организацию кодирования системы.

Большая часть информации в этом представлении обычно представлена на итоговом уровне, а более подробная информация доступна в других документах, ориентированных на разработчиков, таких как документ стандартов разработки. Однако на этом этапе вам все равно может потребоваться записать некоторые архитектурно значимые решения, например, относительно выбора библиотек или фреймворков, или подхода и инструментов для развертывания программного обеспечения или управления конфигурацией.

8.6.1.Представление модуля

Используйте модель, которая определяет модули кода, которые будут созданы, и зависимости между ними. Представление структуры модуля можно выполнить в диаграмме archimate или диаграмме пакетов UML.



Фигура 10. Диаграмма структуры модуля

Загрузить с Google Диска представленные в документе модели в формате *.fm для АИАС “А-СТЕК”

8.6.2.Общее проектирование

Определите общее проектирование (например, ведение журнала, безопасность, отслеживание и т.д.) которое должно выполняться стандартным способом во всей системе, и то, как это должно выполняться (например, с помощью шаблона проектирования или ссылки на библиотеку кода или образец).

8.6.3.Стандарты проектирования, кодирования и тестирования

Определите все стандарты, которым необходимо следовать при проектировании, кодировании и модульном тестировании, возможно, со ссылкой на внешний документ.

8.6.4.Организация кодирования

Определите структуру кодирования (то есть, как исходный код будет храниться в виде иерархии каталогов и как он будет встроен в поставляемое программное обеспечение). Определите иерархию каталогов, инструменты сборки и инструменты доставки (например, инструменты тестирования или непрерывной интеграции), которые будут использоваться для доставки программного обеспечения для тестирования и производства.

8.7.Операционное представление

Операционное представление определяет, как система будет установлена в производственной среде, как данные и пользователи будут перенесены в нее и как она будет настраиваться, управляться, отслеживаться, контролироваться и поддерживаться. Цель информации в этом представлении - показать, как следует создавать и поддерживать операционную среду, а не определять подробные инструкции или процедуры.

8.7.1.Установка и миграция

Определите шаги, необходимые для установки системы.

Если требуется параллельная работа старых и новых систем, объясните, как это будет происходить, не нарушая работу существующих систем, и укажите требуемые переходные состояния.

8.7.2.Управление эксплуатационной конфигурацией

Определите основные группы элементов операционной конфигурации и общие наборы значений для них (например, наборы для пакетной обработки и ночные обработчики) и объясните, как эти группы будут управляться в производственной среде.

8.7.3.Системное администрирование

Объясните требования, которые система предъявляет к системным администраторам (как в обычных, так и в исключительных ситуациях), а также средства, которые система будет предоставлять или использовать в операционной среде.

8.7.4.Предоставление поддержки

Определите группы, участвующие в обеспечении поддержки системы, а также роли и обязанности каждой (включая процедуры эскалации, если это необходимо).

9. Качества системы

В этом разделе объясняется, как представленная архитектура соответствует каждому из требуемых свойств качества системы.

Хотя большая часть этой информации будет неотъемлемой частью представлений, описанных в предыдущей главе, часто бывает полезно выделить ее отдельно. В частности, если такое качественное свойство, как безопасность или производительность, зависит от функций, задокументированных в нескольких различных представлениях, вам следует объяснить это здесь. Например, масштабируемость может зависеть от оптимизации модели данных (задокументированной в информационном представлении) вместе с компонентами балансировки нагрузки (задокументированными в представлении развертывания).

9.1.Производительность и масштабируемость

По каждому из основных требований к производительности и масштабируемости объясните, как система будет соответствовать этому требованию. Обратитесь к практическому тестированию и работе по моделированию производительности, выполненной в рамках применения этой точки зрения.

Требование	Как будет обеспечено
среднее время отклика пользователя должно быть XX под нагрузкой YY	см. таблицу моделирования производительности

9.2.Безопасность

Для каждого из основных требований безопасности объясните, как система будет соответствовать этому требованию. Определите (или укажите) модель угроз, политику безопасности и схему безопасности, которые использовались как часть применения этой перспективы.

Требование	Как будет обеспечено
все пользователи должны пройти аутентификацию, прежде чем им будет разрешен доступ к системе.	доступ ко всем экранам осуществляется через стандартный экран входа в систему с паролями, синхронизированными за ночь с центральной службой LDAP

9.3.Доступность и устойчивость

Объясните требования доступности и устойчивости.

Определите графики доступности для системы.

Объясните, как система будет соответствовать требованиям, ссылаясь на практические работы по тестированию, моделированию и проектированию, которые были выполнены в рамках исследования этой точки зрения.

Требование	Как будет обеспечено
не должно быть единой точки отказа	все узлы развертывания сгруппированы или с балансировкой нагрузки; где узлы сгруппированы, отказ компонента обнаруживается автоматически, и пассивный узел запускается автоматически

9.4.Эволюция

Объясните требования эволюции.

Определите эволюционные измерения, относящиеся к системе.

Объясните, как система будет соответствовать требованиям, принимая во внимание вероятность возникновения каждого типа эволюции (объясняя, как были получены вероятности).

Требование	Как будет обеспечено
должна быть возможность добавлять дополнительные входные каналы без необходимости перепроектировать основную систему	компоненты входного канала слабо связаны с модулями центральной обработки через стандартизированный абстрактный интерфейс

9.5.Другие качества

9.5.1.Доступность

Объясните, как система отвечает требованиям доступности (если таковые имеются).

9.5.2.Интернационализация

Объясните, как система отвечает требованиям интернационализации (или локализации) (если таковые имеются).

9.5.3.Место расположения

Объясните, как система отвечает требованиям для различных географических местоположений (мест), в котором она может быть установлена (если таковые имеются).

9.5.4.Нормативные требования

Объясните, насколько система соответствует нормативным требованиям (если таковые имеются).

9.5.5.Удобство использования

Объясните, как система соответствует любым требованиям к удобству использования (если таковые имеются).

10.Приложения

10.1.Приложение: Решения и альтернативы

Объясните основные принятые архитектурные проектные решения, их обоснование и набор альтернатив, рассматриваемых для каждого из них.

10.2.Приложение: вопросы и ответы

По мере того, как вы разрабатываете свою архитектуру и объясняете ее людям, одни и те же вопросы, вероятно, будут задаваться неоднократно. В некоторых случаях это сигнал к улучшению одной конкретной части описания архитектуры, но в других случаях нет единого места, где можно было бы адресовать запрос, поэтому ответьте на эти вопросы здесь.

Также иногда возникают вопросы, которые вы хотите, чтобы читатели задавали себе, чтобы убедиться, что они поняли архитектуру и ее значение, и хороший способ поощрить это также - задать соответствующий вопрос и ответить в этом разделе.

10.3.Приложение: Ссылки и литература

Перечислите ссылки, которые вы используете по всему тексту, чтобы их можно было процитировать и найти. Например:

1. Книга “Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives”, Second Edition в библиотеке O'Reilly
2. Оригинал шаблона Architecture Description на сайте Розански и Вудс
3. Архитектурные принципы The TOGAF® Standard, Version 9.2 на сайте Open Group
4. Программный продукт для моделирования и управления архитектурой АИАС “А-СТЕК”, включен в реестр российского ПО.

10.4.Перевод и дополнение шаблона

Перевод и дополнения шаблона: Юрченко Анна, автор телеграм-канала IT Service Management for you об управлении ИТ, архитектуре, как создать лучший клиентский сервис и сделать ИТ силой бизнеса. Как показать ценность ИТ заказчику.

Предложения по развитию шаблона добавляйте в виде комментариев к файлу или пишите @itsm_lady