Экосистема открытых компонентов

Совместные инновации в библейских технологиях

https://opencomponents.io

Набросок/план/проект 4 – 20 Июля 2021

Резюме для руководства: Данный документ предполагает, что развитие библейских технологий приобретет пользу от изменения модели производства, в которой каждое приложение представляет собой индивидуальное творение гильдии мастеров, в ту модель, где библейские технологии в основном состоят из взаимозаменяемых компонентов. В документе утверждается, что преимущества такого перехода будут равны преимуществам производства взаимозаменяемых деталей. Ключевой функционал будет легче повторно использовать и поддерживать, больше разработчиков из разных регионов со всего мира смогут продуктивно взаимодействовать, а также значительно возрастет общий творческий и инновационный потенциал глобальной сети разработчиков библейских технологий. Разработка и намеренное взращивание глобальной совместной экосистемы библейских технологий, состоящих из взаимозаменяемых деталей (“открытых компонентов”), создадут контекст для нелинейной модели инованции, максимизируя “конструктивную конкуренцию” (и избегая “деструктивную конкуренцию”). А это, в свою очередь, приведет к тому, что больше людей смогут создавать, распространять и использовать библейские переводы и библейских контент на любом языке, через любые технологии и необходимый формат.

Введение

Производство взаимозамеющих деталей является одним из самых главных факторов, который способствует Промышленной Революции, но его очень легко упустить из виду. До изобретения Эли Уитни в начале 1800-х средства изобретения идентичных компонентов для мушкетов, производство каждого инструмента и машин было, по сути, уникальной работой ремесленника[1]. Сложные инструменты долгое время были единственными в своем роде, так как техника производства не была способна массово производить идентичные компоненты, и, в итоге, такие инструменты могли быть отремонтированы или улучшены только искусными мастерами. Но как только производство взаимозаменяемых деталей стало стандартным средством, произошло несколько важных событий в мире производства: стало возможным быстро ремонтировать и заменять детали, планка входа в производство существенно снизилась, что позволило многим инженерам и изобретателям вовлечься в производство, выросла общая продуктивность и появились многие новые изобретения, которые стали возможными только в таких условиях.[2]

Взаимозаменяемость “изменила промышленную революцию и, тем самым, изменился весь мир. Каждое новое изобретение, появившееся в результате промышленной революции, выиграло от взаимозаменяемости: паровой двигателей, швейные машины, телеграфы и многое другое”. Тезис данного документа заключается в том, что развитие библейских технологий также сможет извлечет пользу от изменения модели “промышленного процесса”, в которой каждое приложение представляет собой, по сути, индивидуальное творение гильдии мастеров, в модель, где библейские технологии в основном состоят из “взаимозаменяемых компонентов”. Первая часть документа предоставляет обширный обзор Экосистемы Открытых Компонентов. Вторая часть документа создана для разработчиков программного обеспечения. Это переход от теоретического обзора идей к практическим реалиям Открытых Компонентов и создания программного обеспечения как части экосистемы. Технические детали Открытых Компонентов, как например обучающие программы, документация и образцы кода, не включены в этот документ, но доступны онлайн: https://opencomponents.io.

Часть 1: Обзор Экосистемы Открытых Компонентов.

Для понимания смысла экосистемы открытых компонентов мы начнем с того, что рассмотрим цели библейских технологий в свете определенных стратегических обсуждений. Затем мы рассмотрим разные модели развития для достижения целей, которым мы следуем в описании открытых компонентов. В конце, мы обсудим преимущества и недостатки данного подхода.

Переосмысление развития библейских технологий в 21-м веке

Быстрый рост вселенской церкви в последние десятилетия существенно изменил контекст развития и использования библейских технологий. Изменение контекста указывает на возможность оглянуться назад и пересмотреть развитие библейских технологий. Мы рассмотрим определенные стратегические обсуждения, которые влияют на развитие библейских технологий и различные подходы для достижения установленных целей.

---

[1] Как и в случае со многими изобретениями, хотя именно Уитни часто приписывают внедрение производства взаимозаменяемых деталей, в развитие нововведения были вовлечены и многие другие перед тем, как оно, в конце концов, стало массовым.

[2] Как таковое, начало производства взаимозаменяемых деталей открыло огромное количество последовательных возможностей, о которых нельзя было и подумать в других условиях. Как концепция конной повозки стала возможной с изобретением колеса, также и массовое производство идентичных инструментов стало возможным с появлением производства взаимозаменяемых деталей. (Читайте больше в 1-й гл. *Хорошие Идеи* (Джосон), о последовательных возможностях.)

Стратегические решения, влияющие на развитие библейских технологий

В целом, библейские технологии предназначены для достижения объединения следующих целей:

- **Производство библейских переводов** – Черновик перевода, проверка, форматирование и публикация переводов (в текстовом, аудио и/или видео форматах) на многих языках, а также ревизия переводов при необходимости через определенное время.
- **Производство библейского контента** – Создание библейских ресурсов, интегрированных в переводы Библии, включая учебные ресурсы, подстрочные переводы, лексические и энциклопедические ресурсы, обучающие ресурсы и т. д.
- **Распространение библейских переводов и контента** – Широкое (пере)распространение переводов Библии и интегрированного с ними библейского контента.
- **Использование библейских переводов и контента** – Чтение, слушание, просматривание и глубокое изучение, используя библейские переводы и библейские ресурсы, интегрированные с ними.

В этом разделе мы рассмотрим факторы, которые влияют на эффективное развитие библейских переводов.

1. Как глобальный контекст и его сложность влияют на развитие и использование библейских технологий?

Сегодня вселенская церковь живет и постоянно распространяется через тысячи групп людей, разговаривающих на разных языках, и через разнообразный культурный контекст. Это вызвало соответствующий рост в деятельности библейского перевода при помощи увеличения числа новый переводчиков, которые переводят Библию на свои языки. Но перед неожиданным наплывом новых участников переводов, разработчики библейских технологий смогли сделать достаточно точные предположения о своих пользователях. В целом, пользователи имели больше общего, чем различий, и их технологические потребности были известными и в основном одинаковыми.

Однако сегодня мир стал еще сложнее, поэтому необходимо пересмотреть стандартные предположения о пользователях библейских технологий. Те пользователи, которых мы знаем, довольно часто используют разнообразные модели технологий, и их потребности в библейских технологиях очень часто конфликтуют друг с другом. Потребности одной группы пользователей часто бывают совершенно противоположными в отношении другой группы пользователей — и это касается только известных нам нужд. Например, одним нужно онлайн-приложение, которое одновременно облегчает управление пользователями и ролями, упрощая совместную работу и обмен информацией. А другие нуждаются в приложении, которое никогда не будет использоваться онлайн, в котором нельзя будет определить пользователя (даже псевдоним), его будет сложно обнаружить, и оно не должно оставлять следов после его удаления. Несмотря на то, что технологии создаются для разных групп пользователей, становится все сложнее отвечать всем их нуждам, особенно когда разработчики не говорят на языке пользователей, имеют разную культуру, часовой пояс, технологические реалии и предположения о том, как должны работать и быть использованы библейские технологии.

Некоторые широко известные библейские разработки имеют большое количество пользователей. Разработчики этих технологий получили большое признание за разработку приложений, которые оказались несомненно эффективными и полезными для многих

пользователей. Но успех таких разработок не должен затемнять огромное количество нужд в библейских технологиях, которые не были еще затронуты ими. И проблема эта кроется не в технологиях или в потенциальных пользователях, которых еще не открыли для себя или не успели попробовать эти нововведения. Но дело в том, что одно приложение не может покрыть широкий спектр различных технологических потребностей вселенской церкви. Если глобальный контекст изменится от однородного и предсказуемого к такому, в котором много противоречивых и неизвестных факторов (и которые влияют друг на друга непредсказуемым образом), то каким образом это может повлиять на библейские технологии?[4] Как минимум, это изменение покажет значительно возросшую потребность в разнообразных и инновационных библейских технологиях, которые позволят вселенской церкви создавать переводы Библии и библейский контент на всех языках, а также распространять и использовать этот контент в различных форматах и на различных технологиях.

2. Как формирование чувства собственности влияет на библейские технологии?

Пользовались ли вы сайтом или приложением, которое было создано людьми не из вашей культуры и языка? Когда содержимым сайта является руководство по эксплуатации пылесоса со странным пользовательским интерфейсом и странным шрифтом, которые являются простой банальностью. Но если представить, что содержимое сайта это Писание, а технологии — способы, благодаря которым пользователь может изучать и понимать истину, тогда это становиться важным. Возможно, в данном случае, средство передачи информации не является самим сообщением, но непреднамеренное ощущение чужеродности самого дизайна технологии может подорвать доверие пользователя и даже подорвать Евангельское послание, причем пользователь не будет этого осознавать. Для некоторых способность доверять технологии (и, надо полагать, контенту, который она обслуживает) подрывается, когда обновления технологии в одностороннем порядке производятся безликими организациями в другой части мира, с которыми у них нет никаких отношений.

Дело не в том, чтобы нынешние представители технологий прилагали неадекватные усилия со своей стороны. Но дело в том, что “формирование чувства собственности” намного важнее в продвижении библейских технологий, чем то, что нам было известно до этого дня.[5] Успешное распространение библейских технологий среди всех народов и языков может подразумевать вовлечение и снаряжение технологов в этих народах для того, чтобы они сами определяли свои потребности, используя те технологии, которыми они владеют и развивают, и используя процессы,

---

[4] Разница между таким контекстами может быть хорошо проиллюстрирована на примере контраста автомобиля Феррари (сложная конструкция) и потоком машин на дороге (комплексная ситуация). Каким бы сложным не был механизм автомобиля, профессиональный механик сможет разобрать и совершенно в точности заново собрать автомобиль. Особенная функция каждой детали и ее взаимосвязь со всем механизмом могут быть известны и контролируемы. Однако движение машин является диспозиционным (предсказать можно только общую ситуацию), так как движение состоит из элементов (напр., ответственность водителей, состояние дорог, погода, происшествия на дорогах и т.д.), которые взаимодействуют непредсказуемым образом, так что движение невозможно контролировать, им можно только управлять. Многие из предположений и решений, которые работают в сложном контексте, обычно мешают быть эффективнвми в контексте, который стал комплексным. Подробнее об этом контрасте см. в статье Сноуден и Бун, "Рамки".

[5] Формирование "чувства собственности" не используется здесь с точки зрения намеренного манипулирования действиями пользователя в интересах корпорации, но подразумевается, что разработка библейской технологии направляет поведение пользователя, независимо от того, осознает он это или нет. Если это так, то из этого следует, что разработчики, намеревающиеся распространить библейские технологии в глобальном масштабе, должны учитывать психологию владения как часть функции и развития технологий.

которыми они сами управляют, тем самым увеличивая доверие как к разработыным ими технологиям, так и к библейскому контенту, который представлен благодаря этим технологиям.

3. Как растущие технические способности церкви глобального юга могут способствовать удовлетворению их собственных потребностей в библейских технологиях?

Вселенская церковь продолжает расти, и также продолжает расти количество профессиональных технологов (разработчиков), которые мотивированы использовать технологии в служении церкви. Разработчики со всего мира признают, что сегодняшние библейские технологии хороши, но они не соответствуют критериям или потребностям их региона. Следовательно, они разрабатывают свои библейские инструменты, которые подходят их группе людей.

Мы можем извлечь пользу из рассмотрения важного принципа по оказанию помощи. Одно из главных рассуждений относительно эффективности усилий по оказанию помощи и развитию заключается в том, что те люди, которые испытывают нужду также имеют право на удовлетворение их нужды. Таким образом, если цель — это “покончить с бедностью”, то более эффективным способом является обеспечение прав бедных на удовлетворение собственных потребностей, чем полагаться на технократический подход, в котором только у “экспертов” есть право “покончить с бедностью” вместо тех, кто живет в бедности. Таким же образом, если наша цель — “покончить с библейской технологической бедностью”, то очень важно изучить, как можно снарядить нуждающихся, чтобы они сами могли удовлетворять свои нужды, избегая технократических подходов в развитии библейских технолоний, могущих непреднамеренно ограничить их эффективность.[6]

Так как сегодня разработчики по всему миру все чаще сами решают, какие им нужны библейские технологии, то это предоставляет возможность целенаправлено оснащать вселенскую церковь для достижения этой цели. Если будет соглашение, что каждый должен иметь возможность удовлетворить свои собственные потребности в библейских технологиях, то это указывает на ценность экосистемы библейских технологий, которая предрасполагает и приглашает к сотрудничеству и обмену, чтобы в результате не получился фрагментарный (хоть и благонамеренный) подход.

4. Как мы можем развивать оптимальные условия для иноваций в библейских технологиях?

Существует одно очень важное упущение в развитие иноваций: это нельзя спланировать или предсказать. Если новшество можео заранее предсказать, тогда “это не будет новшеством. Если о нововведении уже всем известно, значит оно не нуждается в изобретении. Из этого следует, что путь инноваций это всегда неожиданность.”[7] Самый лучший способ предрасположить появленние иноваций —это развивать условия, которые создают благоприятную обстановку для иноваций. Попытки создать иновации без оценки и оптимизации условий для новшеств сравнимо с началом посева без вскапывания и удобрения земли —земля может произрастить что-то, но истинный потенциал не будет реализован. В связи с этим, “количество иноваций зависит от условий, в которых находятся предприниматели.”[8] Мы предлагаем факторы для оптимизации условий для иноваций:

● **Сниженная планка для входа** – Сделайте так, чтобы новым участникам библейских технологий было как можно проще начать работу. “Чем выше цена входа, тем меньше будет достойных внимания идей ... Именно поэтому сниженная цена входа критически важна для всего, что имеет критическую важность”.[9]

● **Посев на технологических полях** – Предоставляте как можно больше технологий в повторно используемых форматах, чтобы минимизировать ненужное дублирование и предоставить

максимальную функциональность для наибольшего числа участников за минимально короткое время. “Главное предсказание заключается в том, что чем больше у нас сегодня технологий, тем быстрее будут появлятся инновации”.[10]

- **Максимальный поток идей** – Большинсво инноваций происходит благодаря новаторским рекомбнациям существующих идей и технологий. Отсюда следует, что уменьшение трения в потоке идей и повторного использования технологий увеличит инновационный потенциал вовлеченного сообщества. Наиболее эффекстивные “модели инноваций … создают открытое окружение, где поток идей льется по некотролируемым каналам. В контролируемой среде, где естественное движение идей сильно сдерживается, они задыхаются”.[11]

5. Насколько это возможно “революционно внедрять иновации” без преднамеренной “разрушительной конкуренции” с другими приложениями (или разработчиками)?

В разработке программного обеспечения сотрудничество часто находится в противоречии с инновациями. Более того, определение приоритета одной ценности производит разрушительный эффект на другую. Если сотрудничество цениться больше, то инновации могут быть сокращены и ограничены до такой степени, чтобы не нарушить работу ни одного из участников. Если инновации ценятся выше, то сотрудничество оказывается под угрозой из-за подрывных инноваций, которые могут легко превратиться в пагубную форму конкуренции - особенно когда пользователи начинают переходить на приложение, которое совершило инновационный прорыв. Стоит понимать, что мы не считаем, что конкурнеция вредна. Наоборот, конструктивная конкуренция может оказать эффективное влияние на рост инновация и решение трудных проблем.

Примером такой формы “конструктивной конкуренции” может быть хакатон, в котором несколько команд разработчиков соревнуются в нахождении наилучшего решения по общей технологической задаче и в котором каждый сможет применить в своих технологиях наулучшее решение победителей.[12]

---

[6] Истерли утверждает, что история помощи и материального развития в целом следовала технократической идее "сосредоточиться на развитии, а не на правах тех, кто должен быть развит" (Тирания, стр. 49). Он отмечает, что акцент на материальном развитии был сосредоточен на том, "что мы должны сделать, чтобы покончить с глобальной бедностью?", игнорируя при этом неравные права черных и белых, неравные права на Западе и в других странах" (стр. 95).

[7] Easterly, *Tyranny*, стр. 299.

[8] Wu, *The Master Switch,* стр. 146.

[9] Wu, *The Master Switch,* стр. 122.

[10] Easterly, *Tyranny,* стр. 284.

[11] Johnson, *Good Ideas,* § Conclusion.

[12] Более органичным примером является прогрессия инноваций в интерфейсах редакторов перевода Библии, которые появились благодаря тому, что разработчики ранних открытых компонентов учились на работе других, чтобы развивать новые технологии. Ранний компонент usfm-js способствовал появлению нового способа разбора USFM в usfm-grammar, оба из которых послужили толчком к появлению Proskomma, преобразующего USFM в граф.

Когда команды разработчиков находятся (или считают, что находятся) в условиях конкуренции за ограниченное финансирование, это может привести к менее альтруистической форме

конкуренции. В таких условиях команды могут быть менее склонны делиться своими технологическими инновациями с другими, чтобы сохранить конкурентное преимущество. Хотя для такого рода "разрушительной конкуренции" есть свое время и место, мы полагаем, что она менее чем идеальна для развития технологий, связанных с переводом, распространением и использованием переводов Библии на тысячи языков.

Путь вперед

Предлагаемое решение проблемы, которую мы рассмотрели выше, заключается в намеренном и совместном развитии открытой экосистемы разработки библейских технологий. Этот подход признает сложность потребности в библейских технологиях во всем мире и решает ее с помощью модульного подхода, который может быть легко реконфигурирован любым способом по мере необходимости. Этот подход согласуется с утверждением, что "масштабные социальные изменения [например, перевод и распространение Библии на всех языках и среди всех народов] лучше проходят благодаря межсекторной координации, а не изолированному вмешательству отдельных организаций".[13] В нем признается, что те, кто лучше всего способен удовлетворить потребности в библейских технологиях, как правило, находятся ближе всего к этим потребностям, особенно когда они связаны с другими людьми, работающими вместе с ними в рамках совместной сети. Это обеспечивает оптимальный контекст для инноваций и позволяет участникам инновационной деятельности действовать сообща, не вступая в деструктивную конкуренцию друг с другом. Для того чтобы лучше понять экосистемный подход в разработке программного обеспечения, мы сначала сравним его с централизованным и фрагментированным подходами.

Три подхода в развитии библейских технологий

Какие подходы могут быть использованы нами в развитии ПО для достижения целей в библейских технологиях?[14] Существует, по крайней мере, три подхода в разработке программного обеспечения, с помощью которых потенциально могут быть достигнуты цели библейских технологий. Будет полезно рассмотреть каждый из этих подходов в свете некоторых факторов, которые были упомянуты выше, а именно:

- **Чувство собственности**: *децентрализованное* (несколько приложений, принадлежащих нескольким командам разработчиков, отвечают разным нуждам) против *централизованного* (одно приложение, принадлежащее одной команде разработчиков).

- **Сотрудничество**: *органичное* (быстрое, по мере необходимости, относительно простое) против *формального* (возможно, но требуется определенная официальная документация и накладные расходы).

---

[13] Кания и Крамер, "Коллективное воздействие". Далее авторы отмечают, что "некоммерческий сектор чаще всего использует подход, который мы называем изолированным воздействием. Это подход, ориентированный на поиск и финансирование решения, воплощенного в рамках одной организации... существует мало доказательств того, что изолированные инициативы являются лучшим способом решения многих социальных проблем в современном сложном и взаимозависимом мире. Ни одна организация не несет ответственности за любую крупную социальную проблему, и ни одна организация не может ее решить".

[14] Объем данной статьи ограничен разработкой ПО, хотя многие из описанных здесь принципов могут быть применимы к аппаратным средствам библейских технологий, например, специализированным аудиоплеерам для Писания, устройствам потокового вещания и т.д.

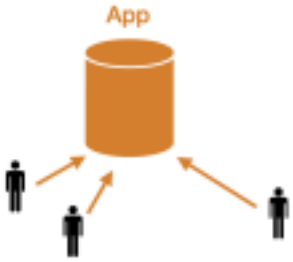
- **Инновации**: *нелинейные* (без ограничений, потенциально расходящиеся, «нестандартные») против *ограниченных* (ограничены, чтобы избежать разрушительных инноваций, линейные и

инкрементные, «стандартные»).

- **Адаптируемость**: *адаптируемость любым человеком* (неограниченная, любой человек с техническими возможностями может адаптировать технологию под свои нужды, существует множество версий/вариантов) против *управляемой адаптивности* (очень ограниченная адаптация, возможна только для владельца технологии, существует ограниченное количество различных версий).

- **Портативность**: *оптимальная возможность повторного использования* (элементы технологии могут быть повторно использованы в нескольких различных приложениях) против *ограниченной возможности повторного использования* (технология и функциональность обычно существуют только в первоначальном приложении, для которого она была разработана).

Централизованное (одно приложение)



Централизованное развитие: одно приложение пытается решить все нужды пользователя

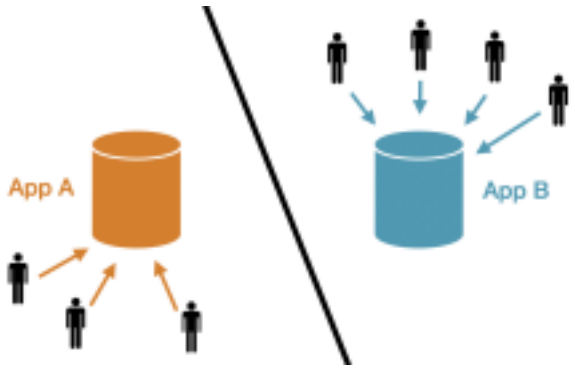
В этом подходе, все разработки сфокусированы на одном приложении с целью сократить дублирование и оптимизацию разработки. Хотя такой подход может привести к быстрому результу, он имеет негативный эффект в виде значительного сокращения числа технологов и инноваторов, которые могут внести значительный вклад в расширение библейских технологий.[15] Количество нужд в библейских технологиях огромен и продолжае расти —расширение на тысячи языки и десятки технологий. Растущее число потребностей, а также растущее число библейских технологов и разработчиков программного обеспечения, уже работающих в глобальном масштабе над решением этих потребностей, говорит о том, что представление о том, что одно приложение способно выполнить все, что необходимо глобальной церкви, становится все менее правдоподобным.[16] Поэтому в данной работе мы сосредоточим свое внимание на двух оставшихся моделях развития: Фрагментированной и Экосистемной.

---

[15] By отмечает, что существует "неоспоримая эффективность", которая присуща централизованным моделям развития. Он отмечает, " однако минус таких хорошо работающих машин заключается в том, что они инициируют созидательное разрушение, которое революционизирует отрасли и, в конечном счете, умножает производительность и ценность. А когда информация является конечным товаром, эффект мультипликатора неисчислимо велик" (The Master Switch, 195, курсив добавлен). Необходимость умножения производительности и ценности в отношении перевода, проверки, распространения и использования конечной информации - Священного Писания - на каждом языке предполагает, что централизованный подход может непреднамеренно пожертвовать долгосрочными инновационными прорывами ради краткосрочного повышения эффективности.

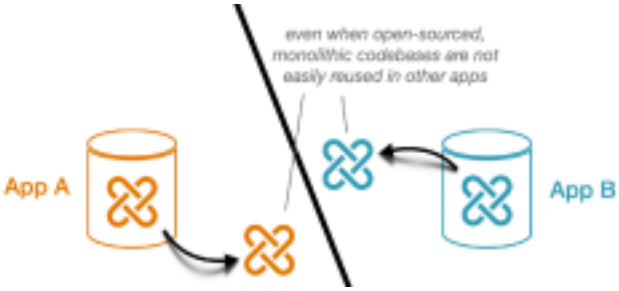
[16] Число команд, занимающихся разработкой библейских технологий, растет во всем мире захватывающими темпами. Неполный список библейских технологий, разрабатываемых командами по всему миру, которые известны авторам данной статьи, включает: пакет перевода Библии Autographa (компания Bridge Connectivity Solutions (BCS) в Индии), Vachan Online (изучение Библии, BCS в Индии), FABIST (перевод Библии, 222 Ministries для иранских языков), V-Draft (перевод Библии, GeCraft в Центральной Азии), VietBible (изучение Библии, Вьетнам), SABDA (изучение Библии, Индонезия), Bible Study App (GeCraft в Центральной Азии).

Фрагментированная система (разные приложения)



Фрагментированная разработка: несколько сторон создают централизованные приложения

При фрагментированном подходе в разработке программного обеспечения независимые приложения порождают конкуренцию за желаемую функциональность и наибольшую пользовательскую базу. Это стандарт де-факто в большинстве областей разработки программного обеспечения, особенно там, где желательно увеличение доли рынка для компании за счет прямой конкуренции. Также этот подход лучше всего отражает настоящую среду библейских технологий, несмотря на то, что такая конкуренция обычно считается вредной для миссионерства.



Фрагментированная разработка: неоптимальное сотрудничество и инновации

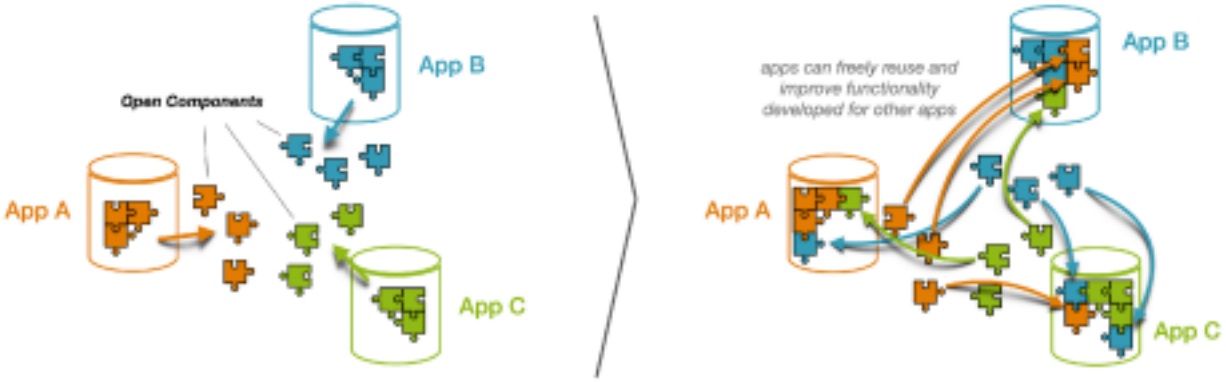
Многие приложения Библии выполняют в целом схожие функции, но даже когда разработчики хотят сотрудничать друг с другом, преодолевая организационные границы и кодовые базы, им не хватает эффективных средств для этого (см. диаграмму выше). Кроме того, поскольку технологии не могут быть легко повторно использованы (из-за структуры кодовой базы, лицензии или того и другого), каждое приложение в фрагментированной системе, в конечном итоге, должно воссоздать функциональность другого приложения, чтобы не потерять свои преимущества перед конкурентами или способность обслуживать свою пользовательскую базу. Результатом этого является повсеместное увеличение ненужного дублирования и затрат, а также ослабление сотрудничества и инноваций.[17]

17В фрагментированной системе нередко крупные компании пытаются вернуть все к централизованной системе с разрешением только одного приложения. Обоснованием для этого обычно служит сокращение дублирования и затрат, а также оптимизация усилий по разработке. Эти аргументы не лишены оснований, поскольку по сравнению с фрагментированной системой, состоящей из множества независимых приложений, в централизованной меньше расходов. Вместо того чтобы "десять компаний соревновались в разработке лучшего телефона - изобретая колесо каждый раз - ресурсы общества могут быть синхронизированы в стремлении к общей цели. Нет дублирования исследований, когда множество лабораторий гоняются за одним и тем же изобретением" (Wu, The Master Switch, 110). Однако централизованные системы, как известно, ограничены в своих инновациях. Ву утверждает, что они ограничены серьезным "генетическим недостатком", который заключается в том, что "они не могут создавать технологии, которые могут, при малейшей возможности, угрожать ... системе". Централизованные системы "практически ограничены изобретениями для поддержания жизнедеятельности; о разрушительных технологиях, тех, которые могут даже бросить тень неопределенности на бизнес-модель, [просто не может быть и речи]" (Wu, The Master Switch, 107). Ву отмечает, что такие практичные изобретения, как автоответчик, были придуманы в Bell Labs в самом начале, но были прекращены руководителями, которые боялись малейшего нарушения бизнес-модели AT&T. Многие из таких полезных изобретений появились на рынке лишь спустя десятилетия, после того как монополия была ликвидирована.

Экосистема (совместные инновации)

Экосистема: множество игроков совместными усилиями создают эффективную среду для быстрого внедрения инноваций

В данном документе мы выступаем за децентрализованную модель в развитии библейских технологий. В некоторых отношениях этот подход похож на фрагментированный подход, но есть одно исключение: разработчики библейских технологий не сфокусированы только на разработке своего приложения, но **они создают для того, чтобы также укреплять экосистему библейских технологий, частью которой они являются.**



Экосистема: состоит из компонентов с открытым исходным кодом, предназначенных для повторного использования

Создавая модульное программное обеспечение с открытым исходным кодом, которое можно повторно использовать в других приложениях, разработчики по всему миру могут более эффективно внедрять инновации в рамках модели обширного сотрудничества, которая максимизирует эффективность библейской технологии при максимальной конструктивной конкуренции (и минимизации деструктивной конкуренции).
Результатом этой децентрализованной и совместной модели развития технологии является экосистема открытых компонентов.

Понятие Открытых Компонентов

Открытые компоненты соответствуют трем критериям технологических разработок: расширяемость, портативность и открытый источник.

- **Расширяемая** структура предназначена для расширения функциональности за счет включения кода, разработанного третьей стороной, чаще всего в виде плагина. В связи с этим другим разработчикам предлагается принести свой код и "поиграть в песочнице", предоставляемой базовой технологией.
- **Портативная** структура предназначена идти в другом направлении, предоставляя функциональность, которая может быть включена в другие технологии, обычно в виде библиотек или простых приложений. Разработчикам предлагается "создать свою собственную песочницу", включив переносимые компоненты в свои собственные приложения.
- **Технология с открытым исходным кодом** состоит из исходного кода, который доступен по открытой лицензии, так что разработчики могут перепрофилировать, расширять, улучшать, распространять и иным образом повторно использовать технологию без ограничений.[18]

Рассматривая взаимодействие этих факторов, мы можем увидеть, как различные технологии и инструменты обеспечивают многогранную функциональность, как показано на диаграмме ниже:

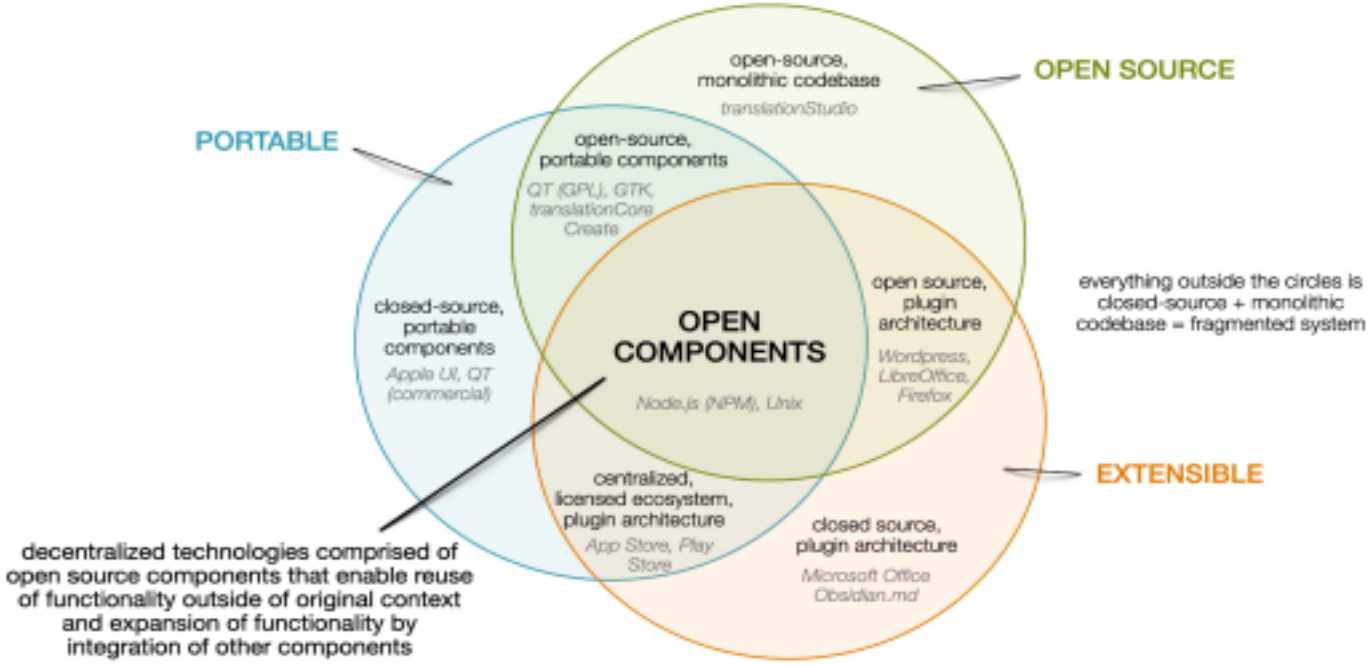


Диаграмма Экосистемы открытых компонентов

---

[18] Связь между открытым исходным кодом и потенциалом нелинейных инноваций хорошо прописана. Одним из классических примеров является 10-дневная конференция по программированию MATLAB, в ходе которой участники должны представить алгоритмы для решения проблемы. Конкурс носит игровой характер, и весь исходный код каждого участника открыт для других участников, чтобы они могли увидеть, повторно использовать и подправить его, чтобы создать лучшее решение проблемы. Хау отмечает, что "экстраординарный аспект MATLAB... *в котором вся интеллектуальная собственность выбрасывается на публичную площадку* для многоразового использованния по своему усмотрению, оказывается безумно эффективным методом решения проблем... В среднем... лучший алгоритм в конце конкурса превосходит лучший алгоритм первого дня в тысячу раз" (Howe, Crowdsourcing, выделено автором).

Открытые компоненты существуют в пересечении всех трех этих критериев: это совокупность децентрализованных технологий, состоящих из компонентов с открытым исходным кодом (Открытый источник), которые позволяют использовать функциональность вне их

первоначального контекста (Портативность), а также расширять функциональность путем интеграции других компонентов (Расширяемость). “Компонент” относится к переносному кодовому модулю, который обеспечивает разделение проблем вокруг конкретного набора задач (например, конкретного элемента пользовательского интерфейса, функции без заголовка, такой как токенизатор, и т.д.). “Открытый компонент” - это компонент, выпущенный под лицензией с открытым исходным кодом, которая обеспечивает максимальную возможность повторного использования (например, MIT, AGPL). “Экосистема открытых компонентов” состоит из компонентов с открытым исходным кодом, технологий, которые объединяют эти компоненты для решения различных проблем, и людей, вовлеченных в эту технологию.[19] В этой экосистеме новая технология **проектируется и создается с учетом двух параллельных перспектив:** удовлетворения непосредственной потребности (с точки зрения "приложения") и создания открытых компонентов, которые могут быть повторно использованы другими технологиями в экосистеме для удовлетворения той же потребности (с точки зрения "экосистемы").

Открытые компоненты - это **больше, чем модульное программное обеспечение.** Многие кодовые базы программного обеспечения разрабатываются по модульному принципу для облегчения обслуживания, но сами модули не предоставляются другим, и кодовая база не расширяется третьими лицами.[20] Переносимый и расширяемый код обязательно является модульным, но не весь модульный код портативен и расширяем. Экосистема открытых компонентов включает в себя технологии, состоящие из модульных кодовых баз, которые также предназначены для улучшения за счет повторного использования других компонентов (расширяемость) и для улучшения других технологий за счет возможности повторного использования созданных новых компонентов (портативность).

Открытые компоненты - это **больше, чем программное обеспечение с открытым исходным кодом.** Выпуск исходного кода под открытой лицензией не создает тем самым открытую экосистему. Открытый лицензионный исходный код является необходимым элементом открытой экосистемы, но не менее важно, чтобы сам код был разработан для экосистемы. Монолитный код - будь то с закрытым или открытым исходным кодом - может быть создан быстрее, но полезен только для приложения, для которого он был разработан.[21] Даже если он доступен по открытой лицензии, то может быть сложным для понимания и громоздким для улучшения или повторного использования. Другие разработчики могут использовать его повторно, развивая и изменяя код в соответствии со своими потребностями (например, добавляя новые функции и исправляя ошибки), и они могут внести эти улучшения обратно в существующий проект. Именно тогда, когда оба проекта должны расходиться, чтобы соответствовать их соответствующим сценариям использования, ограничения монолитного приложения с открытым исходным кодом становятся помехой.Хотя они могут иметь до 99% одного и того же кода, будущие исправления ошибок и патчи для одного проекта настолько утомительны для применения в другом, что они обычно не происходят - несмотря на благие намерения. По этой причине разработчики часто избегают сотрудничества с монолитными приложениями, даже если они с открытым исходным кодом.

[19] Экосистема открытых компонентов является примером компонуемости - принципа проектирования систем, который касается взаимосвязи компонентов. Высокая комбинируемость системы "обеспечивает компоненты, которые могут быть выбраны и собраны в различных комбинациях для удовлетворения конкретных требований пользователя". В информационных системах существенными характеристиками, делающими компонент компонуемым, являются его самодостаточность (модульность, т.е. он может быть внедрен отдельно и может взаимодействовать с другими компонентами, но зависимые компоненты заменяемы) и отсутствие статичности (он рассматривает каждый запрос как независимую транзакцию, не связанную ни с каким предыдущим запросом). См. "Composability" (https://en.wikipedia.org/wiki/Composability).

[20] Модульный код отличается от монолитного кода по некоторым важным параметрам. Слоотвег в книге "Монолитность" отмечает, что модульный код "редко нуждается в изменениях - как только он написан, имеет четко определенный набор событий и методов и свободен от ошибок, он больше не нуждается в изменениях. Если приложение хочет начать использовать данные по-другому, это не требует изменений в компоненте; данные по-прежнему имеют тот же формат, и приложение может просто обрабатывать их по-другому". Кроме

того, стоимость модульной кодовой базы может быть выше на начальном этапе, но это "единовременные затраты. После первого проекта вы уже знаете, где найти большинство необходимых модулей, и создание модульного базиса не будет дороже, чем монолитного". Кроме того, модульный код обычно легче понять и поддерживать, и он позволяет относительно легко заменять компоненты без необходимости значительного рефакторинга кодовой базы.

[21] Многие - хотя и не все - технологии в фрагментированной системе состоят из монолитных кодовых баз, что означает, что код тесно связан (сильно зависит от другого кода) и содержит много предположений о том, как код взаимодействует внутри. Подробнее о монолитном коде и связности см. в Lin, "Monolithic" и Beard, "Microservice Architecture".

Открытые компоненты **не ограничиваются элементами пользовательского интерфейса.** Открытые компоненты могут предоставлять функции фронтенда, бэкенда, обработки данных, на стороне сервера, на стороне клиента и т.д. Как мы увидим ниже, бэкенд-компоненты, не включающие элементы пользовательского интерфейса, значительно выигрывают от предоставления визуальной песочницы, в которой можно изучить функциональность компонента.

Преимущества

Экосистема открытых компонентов обеспечивает ряд важных **преимуществ** по сравнению с традиционными моделями разработки программного обеспечения. Эти преимущества существуют на уровне написания и внедрения программного кода (преимущества на уровне кода), а также на уровне экосистемы в целом (преимущества на уровне экосистемы).

Преимущества на уровне кода

- **Легче разрабатывать** – Разделение проблем в этом подходе позволяет разработчику сосредоточиться на одной функции в данном компоненте, что облегчает разработку.[22]
- **Легче создавать прототипы** – В надежной экосистеме приложение может быть быстро собрано из существующих компонентов, чтобы обеспечить основу для разработки новых компонентов прототипов. Учитывая простоту включения/отключения компонентов в такой системе, несколько прототипов можно тестировать независимо или параллельно. Возможность сосредоточиться на прототипах приводит к более быстрому внедрению инноваций, поскольку нет необходимости наращивать функциональность, предоставляемую существующими компонентами.
- **Легче тестировать** – Таким же образом тестирование компонента становится легче, поскольку разделение проблем позволяет разработчику сосредоточиться на компоненте и быстрее пройти цикл сборки → измерения → обучения.[23]
- **Легче повторно использовать** – Компонент может быть легко использован повторно в других приложениях при наличии хорошей документации и дизайна ввода/вывода. Это обеспечивает быструю функциональность практически без дублирования существующей функциональности.
- **Легче улучшать** – Компонет легче улучшать, поскольку его уникальная направленность и автономная структура упрощают понимание и улучшение кода. По мере использования компонента в различных приложениях, все больше разработчиков могут выявить инновационные модификации и предложить (или написать) улучшения в коде, которые повышают его функциональность.
- **Легче обслуживать** – Таким же образом, по мере использования компонента в других приложениях, все больше разработчиков могут выявить ошибки и отправить исправления для их устранения создателю (тому, кто поддерживает) компонента. Следовательно, долгосрочное обслуживание кодовой базы приложения обходится дешевле, поскольку затраты на ее

поддержание распределяются между экосистемой. По сути, другие разработчики участвуют в техническом обслуживании других приложений, занимаясь поддержкой компонентов, используемых в этих приложениях. *Примечание: это влечет за собой затраты на пересмотр для разработчиков приложений, поскольку им придется потратить время на изучение того, как изменилась функциональность обновленных компонентов, и определить, нужно ли/когда/как их интегрировать.*

• **Легче поддерживать** – Срок службы компонента может легко пережить полезность приложения, для которого он был разработан. Это особенно важно в таких быстро меняющихся областях, как компьютерная языковая технология, где приложения устаревают гораздо быстрее, чем ключевые функции, для которых они были разработаны. Время существования технологии, реализованной в виде открытого компонента, не зависит от относительно короткого срока использования отдельного приложения, что позволяет поддерживать ее полезность в экосистеме дольше, чем это было бы возможно.

---

[22] Как указано ниже в разделе "недостатки", разработка компонентов может потребовать больше времени, по крайней мере, на начальном этапе.

[23] Подробнее о бережливом развитии см. в книге Ries, Lean Startup.

Преимущества на уровне экосистемы

- **Обеспечивает нелинейные инновации без разрушительной конкуренции** – Благодаря тому, что все инновации доступны по лицензиям с открытым исходным кодом, а также разработаны как компоненты, которые могут быть повторно использованы в других приложениях, инновации самого разрушительного характера не нарушают работу других членов экосистемы. Если новаторское изобретение другого разработчика окажется полезным для пользователей другого приложения в экосистеме, оно может быть включено в это приложение с минимальными усилиями. В результате контекст разработки в значительной степени предрасположен как к нелинейным инновациям, так и к конструктивному сотрудничеству.
- **Обеспечивает более экономичную разработку технологий** – Как упоминалось выше, возможность включения компонентов, созданных одной командой разработчиков, в приложение другой значительно сокращает количество ресурсов, затрачиваемых на воссоздание существующих технологий.
- **Обеспечивает эффективное участие новых представителей в глобальной сети разработчиков библейских технологий** – Экосистема не только открыта для новых участников, но и активно приветствует их, поскольку чем больше участников в сети идей, тем больше инновационный потенциал экосистемы в целом. Более того, чем больше разработчиков технологий присоединяются к экосистеме из церковных сообществ, участвующих в переводе Библии и богословском образовании, тем более эффективной становится вся сеть в распространении полезных библейских технологий в каждой группе людей и на каждом языке.
- **Обеспечивает эффективное участие разработчиков с частичной занятостью и волонтеров** – Одна из самых больших проблем, с которой сталкиваются разработчики, не имеющие возможности уделять много времени разработке библейской технологии, заключается в том, что они тратят большую часть времени, на то, чтобы узнать, что изменилось в исходном коде с момента их последней возможности помочь. В экосистеме открытых компонентов разработчик, работающий неполное время, может оказать значимую услугу всей экосистеме, разработав и поддерживая даже один компонент, от которого зависят другие приложения.
- **Обеспечивает повторное использование функционала в различных областях библейских технологий** – Часто различные приложения сосредоточены либо на создании переводов Библии, либо на их распространении. Однако существует определенная степень совпадения технологических потребностей обоих видов приложений (например, токенизация текстовых

строк на разных языках, отображение подстрочных переводов и т.д.). При реализации в виде открытого компонента одна и та же функциональная потребность может быть удовлетворена в различных технологических областях. Междоменные связи являются очень плодородными источниками инноваций и креативности - чем больше они катализируются, тем более вероятны нелинейные и дивергентные инновации.

- **Позволяет коммерческим компаниям участвовать в экосистеме** – В целом, мир библейских технологий разделен между некоммерческим сектором (часто намеревающимся обслуживать многие (или все) из 7000+ языков, на которых говорят в мире) и коммерческим сектором (обычно сосредоточенным на относительно небольшом количестве основных языков). Разница в направлениях, а также растущая тенденция в некоммерческих библейских технологиях к использованию технологий с открытым исходным кодом могут затруднить сотрудничество представителей разных секторов. Экосистема открытых компонентов предоставляет разработчикам из обоих сторон значимый способ совместной работы над технологиями, которые служат обоим сторонам.[24]

---

[24] Кания и Крамер отмечают важность значимого сотрудничества между представителями некоммерческой стороны и теми, кто находится за его пределами, для решения сложных социальных проблем: "Социальные проблемы возникают в результате взаимодействия правительственной и коммерческой деятельности, а не только из-за поведения организаций социального типа. В результате сложные проблемы могут быть решены только межсекторными коалициями, в которые вовлечены те, кто не входит в некоммерческий сектор" ("Коллективное воздействие").

Недостатки

Экосистема открытых компонентов не лишена **недостатков**, включая следующие:

- **Требуется развитие с целью обслуживания экосистемы** – Разработчикам в экосистеме открытых компонентов необходимо скорректировать свою практику разработки с учетом параметров и конвенций экосистемы (т.е. портативность, расширяемости, открытый исходный код), вместо того, чтобы просто сосредоточиться на своем приложении и решить непосредственную задачу. Это может оказаться сложной задачей для разработчиков, привыкших рассматривать только контекст своего приложения, а также для приложений, состоящих из монолитного кода, который не доступен по лицензии с открытым исходным кодом.
- **Требуется больше времени на разработку библиотек компонентов, чем это необходимо в противоположном случае** – Написание кода для традиционного приложения часто требует меньше времени, чем написание кода, который может быть использован во многих приложениях экосистемы. Разработка технологий, служащих экосистеме, требует краткосрочных временных инвестиций (замедление в некоторой степени предоставления функциональности собственному приложению) для долгосрочного увеличения ценности экосистемы (предоставление функциональности в качестве компонента, который могут использовать все другие приложения в экосистеме). *Примечание: наблюдения показывают, что по мере того, как разработчики привыкают к созданию компонентов вместо традиционного кода, их эффективность возрастает настолько, что разница во времени может стать незначительной в некоторых случаях и даже привести к большей эффективности в других.*
- **Как правило, требует использования одного языка программирования** – Экосистема открытых компонентов эффективна в той мере, в какой приложения создаются с использованием одного и того же языка программирования. Например, компонент, созданный на C#/.NET, не сразу пригодится в приложении, написанном на JavaScript, и наоборот. В этом наблюдении есть три важных момента. Во-первых, это не означает, что экосистема открытых компонентов ограничена одним языком программирования - это не так. Во-вторых, благодаря использованию кросс-компиляции и транспонирования, компоненты, написанные на одном языке программирования, могут быть использованы в других технологических стеках

(например, Rust → JavaScript; многие языки программирования → Web Assembly). В-третьих, даже если компоненты нельзя напрямую повторно использовать в другой технологии, отдельные компоненты, которые поддерживают разграничение задач, могут быть относительно легко перенесены на различные языки программирования/среды. Таким образом, один и тот же компонент может быть воссоздан на нескольких языках программирования, что улучшает портативность функциональности компонента в рамках всей экосистемы.

Размышления о привлечении финансирования

Экосистема открытых компонентов обеспечивает надежные и экономически эффективные условия для разработки библейских технологий. Во-первых, это **минимизирует количество потраченных средств** на воссоздание функций, которые уже существуют в других приложениях. Создавая функциональность в виде компонентов, которые можно повторно использовать в различных технологиях, команды разработчиков тратят меньше времени на воссоздание того, что уже существует. Во-вторых, экосистема открытых компонентов позволяет **максимально использовать ресурсы для решения новых задач и изобретения новых технологий**. Создавая контекст для всемирного сотрудничества, растущая сеть разработчиков технологий может сотрудничать друг с другом с наименьшими трудностями, исследовать новые возможности и внедрять инновации для решения нужд вселенской церкви в библейских технологиях.

В дополнение к этому, одним из самых больших преимуществ экосистемы открытых компонентов является то, что она **помогает устранить противоречия между командами разработчиков, когда они добиваются финансирования** для библейских технологий. Традиционная модель финансирования технологий, как правило, стимулирует либо централизованную (одно приложение), либо фрагментарную (несколько приложений) модель разработки технологий. По сути, разработчики технологий борются за финансирование, демонстрируя функциональность своих собственных приложений, косвенно противопоставляя их другим "конкурентам". Это, мягко говоря, делает значимое сотрудничество и обмен между командами разработчиков сложными и противоречивыми. С другой стороны, экосистема открытых компонентов может полностью изменить картину финансирования и избежать этих конфликтов. Однако это возможно только в том случае, если те, кто финансирует библейские технологии, понимают, как работает экосистема, и соответствующим образом скорректируют свое финансирование для развитии технологий. Для этого мы предлагаем следующие варианты.

Понимание экосистемы

Доноры часто привыкли финансировать организацию для создания конкретного приложения. Важно, чтобы те, кто обеспечивает развитие технологий, понимали тесную взаимосвязь экосистемы открытых компонентов и то, что *функциональность используется во многих приложениях.* Следовательно, донорам будет полезно понимать функциональность, которую они финансируют, а не только приложение, для которого эта функциональность разрабатывается.

Не все участники экосистемы открытых компонентов обязательно будут иметь приложение, демонстрирующее их работу. Вместо этого некоторые команды разработчиков могут создавать компоненты, которые используются в нескольких приложениях, созданных другими командами разработчиков. Важно, чтобы финансирующие организации признавали эту динамику и понимали, чем финансирование технологической экосистемы отличается от простого финансирования разработки приложения. Вся экосистема будет укрепляться, когда доноры признают огромную ценность финансирования тех, кто работает за кулисами, создавая технологии, от которых зависит вся экосистема.

Стимулирование создания экосистемы

С пониманием экосистемы открытых компонентов тесно связана возможность, которую имеют

спонсоры для стимулирования ее создания. В зависимости от того, как финансирующие организации будут это делать, они смогут быстро расширить сеть участников и повысить инновационный потенциал экосистемы в целом. Практически это может произойти следующим образом:

- **Финансирование "компонентизации" существующих технологий** – Некоторые функции, необходимые всей экосистеме, в настоящее время существуют только в одном приложении. Спонсоры могли бы значительно укрепить экосистему, предоставив ресурсы для адаптации технологии в открытый компонент, который затем будет полезен во всей экосистеме.
- **Финансирование создание новых компонентов** – Финансирующие организации могут поощрять команды разработчиков создавать новые технологии в виде открытых компонентов, которые приносят пользу всей экосистеме, а не только их собственному приложению.

Ресурсы для руководства и инфраструктуры экосистемы

Поскольку это намеренное сотрудничество между многими различными участниками, успех экосистемы открытых компонентов напрямую зависит от хорошей коммуникации и координации между участниками. В определенной степени необходима вспомогательная инфраструктура, обеспечивающая такие вещи, как репозиторий доступных компонентов, пособия, документация для участия, коммуникационные платформы, хакатоны, совместное управление проектами и т. д. Без этого не стоит ожидать эффективного сотрудничества.[25] Таким образом, мы предлагаем учитывать эту потребность и выделять соответствующие ресурсы для обеспечения эффективного руководства и инфраструктуры, чтобы снизить возможные риски и обеспечить важное сотрудничество в экосистеме.[26]

---

[25] Благие намерения слишком легко отходят на второй план, поскольку каждая организация имеет дело с жесткими временными рамками для достижения разрозненных целей и задач - независимо от того, насколько они кажутся похожими друг на друга. "Ожидание, что сотрудничество может происходить без поддерживающей инфраструктуры, является одной из наиболее частых причин его провала" (Кания и Крамер, "Коллективное воздействие").

[26] Это аналогичное изменение, которое требуется для финансирования альянсов коллективного воздействия. "Это требует фундаментальных изменений в том, как спонсоры видят свою роль: от финансирования организаций к руководству долгосрочным процессом социальных изменений. Уже недостаточно финансировать инновационное решение, созданное одной некоммерческой организацией, или наращивать потенциал этой организации. Вместо этого, спонсоры должны помочь создать и поддержать коллективные процессы, системы отчетности оценки и лидерство сообщества, которые позволяют появляться и процветать межсекторным коалициям" (Кания и Крамер, "Коллективное воздействие").

Итог

В первой части этой статьи мы предложили переход в развитии библейских технологий от фрагментарной среды к глобальной совместной экосистеме библейских технологий, состоящей из взаимозаменяемых деталей (открытых компонентов). Мы предположили, что конечный результат этого перехода будет аналогичен производству взаимозаменяемых деталей: будет проще повторно использовать (и поддерживать) ключевых функций , больше разработчиков из разных частей мира смогут эффективно сотрудничать друг с другом, а общий творческий и инновационный потенциал глобальной сети разработчиков библейских технологий значительно возрастет, избегая при этом разрушительной конкуренции. Это, в свою очередь, приведет к тому, что больше людей смогут создавать, распространять и использовать переводы Библии и библейский контент на любом языке, на любой технологии и в любом необходимом формате. Переходим ко второй части статьи, в которой мы рассмотрим практическую реализацию экосистемы открытых компонентов.

Часть 2: На пути к экосистеме открытых компонентов в библейских технологиях

Остальная часть данной статьи написана с целью рассмотрения некоторых технических вопросов, которые могут быть интересными для разработчиков программного обеспечения. Учебные пособия, шаблоны и примеры кода доступны онлайн: https://opencomponents.io.

Экосистема открытых компонентов уже существует: несколько организаций сотрудничают друг с другом для создания многократно используемых компонентов, относящихся к созданию и использованию переводов Библии. В этом разделе мы рассмотрим состояние экосистемы на сегодняшний день, а затем рассмотрим принципы и практики ("простые правила"), которые ведут к созданию здоровой экосистемы библейских технологий.

Открытые компоненты в реальной жизни

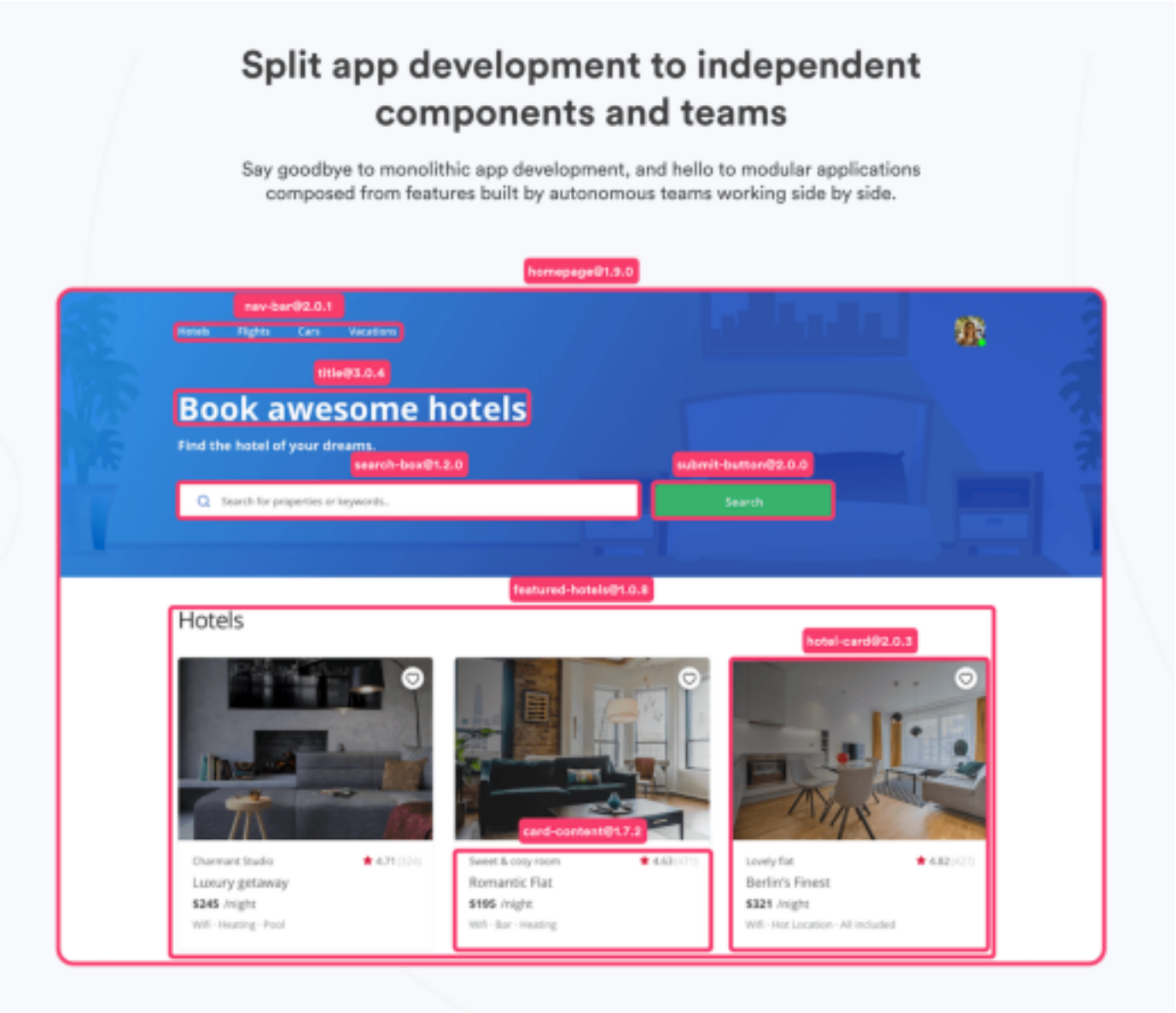
Здесь мы рассмотрим примеры открытых компонентов из промышленности, а также с ранних этапов развития самой экосистемы открытых компонентов.

Примеры из промышленности

Хорошо реализованным примером совместной разработки компонентов в более широкой индустрии разработки ПО является https://bit.dev. Этот сайт напоминает NPM и облегчает разработчикам программного обеспечения описание и обмен компонентами JavaScript..[27] Снимок экрана с их сайта дает наглядное представление об иерархии компонентов, которые были использованы для создания популярного сайта. На сайте представлены руководства по разработке компонентов с веб-ориентированным фокусом, включая такие популярные основы, как React, Vue, Angular, а также Node.js для загружаемых приложений. Хотя не существует скоординированного сегмента рынка, на который нацелен bit.dev, это хороший пример того, как общие компоненты могут быть повторно использованы для различных сегментов. Экосистема открытых компонентов имеет преимущество в том, что как финансирующие, так и реализующие организации хотят активно сотрудничать друг с другом на более узком рынке (библейские технологии).

---

[27] NPM - это онлайн-репозиторий для публикации проектов Node.js с открытым исходным кодом (https://www.npmjs.com/).



Примеры из экосистемы открытых компонентов

На момент написания данной статьи экосистема открытых компонентов находится на ранней стадии развития и быстро развивается. Несколько организаций и команд разработчиков работают над созданием компонентов, которые обеспечивают необходимую функциональность библейских технологий. Большинство из этих открытых компонентов представляют собой библиотеки компонентов React, часто предназначенные для использования как в веб-приложениях, так и во фреймворках на основе JavaScript (например, Electron).[28] Следующий сокращенный список дает представление о широте экосистемы на этой ранней стадии, а также о некоторых основных функциях, предоставляемых в виде компонентов.

[28] Компоненты, созданные как библиотеки компонентов React, дают значительные преимущества, в частности, количество платформ, на которых они могут быть использованы (включая веб-, настольные и мобильные приложения), и простота интеграции компонентов в онлайн "песочницу" для браузерного тестирования вариантов

использования. Примечание: несколько организаций независимо друг от друга выбрали этот стек для разработки библейской технологии.

Команда компонентного типа

Редактор USFM – *WYSIWYG интерфейс редактора Библии React BCS*

Настройки предпочтений – *хранение пользовательских предпочтений React BCS*

Grapha UI – *UI компоненты React BCS*

Grapha Sync – *пользовательский интерфейс с функцией перенесения (drag-and-drop) для синхронизации файлов React BCS*

Подстрочный редактор – *WYSIWYG редактор совмещения React Clear.Bible*

Подборка библейских ссылок– *Навигация по книге-главе-стиху React GeCraft*

Писание ePub – *экспорт в EPUB Javascript MVH Solutions*

Проскомма – *Механизм выполнения Писания Javascript MVH Solutions*

Графит (Graphite) – *система умных шрифтов для сложных скриптов C++ SIL*

WordMap – *многоязычный подбор слов в соответствии с слиянием Javascript unfoldingWord*

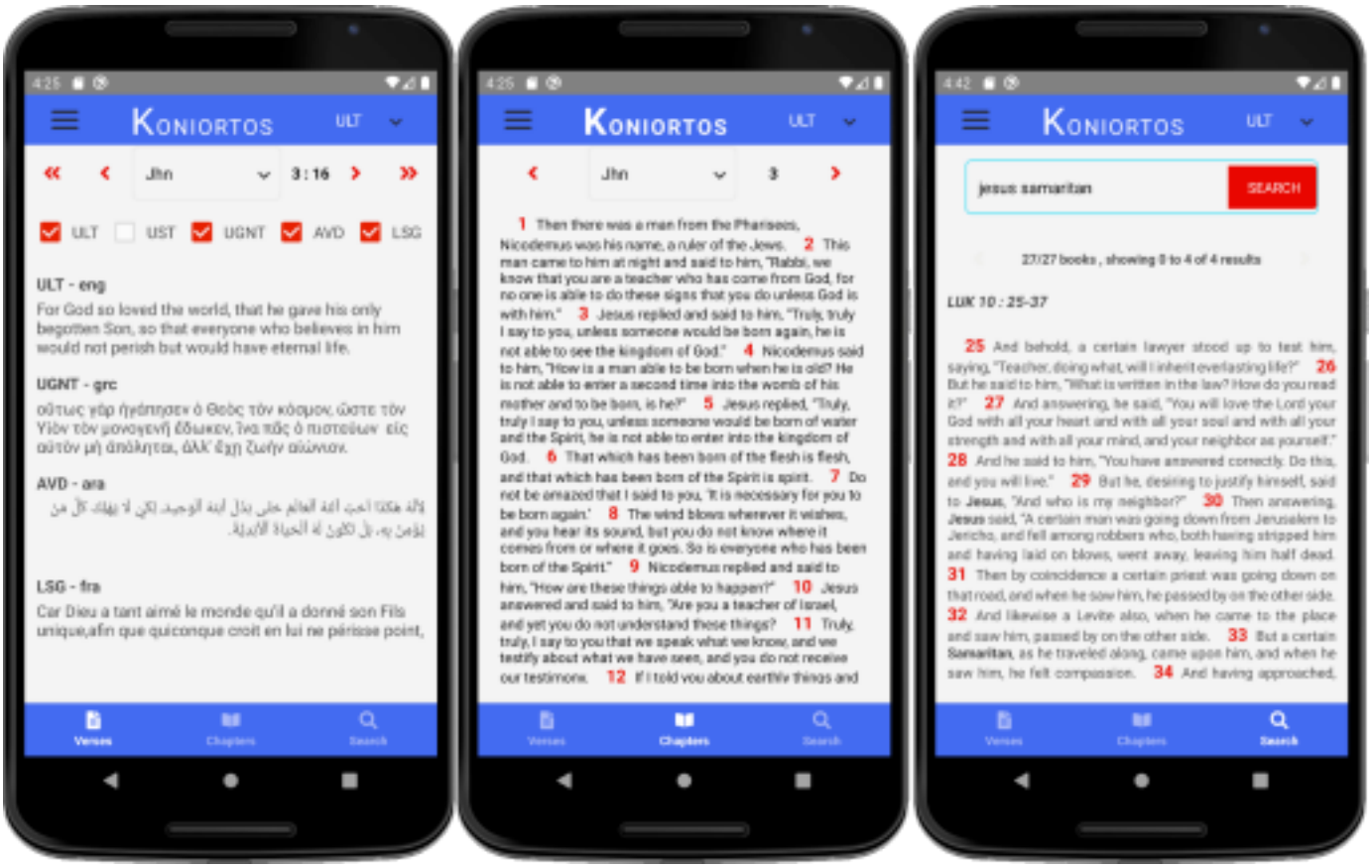
Токенизатор пунктуации строки – *классификатор строковых токенов Javascript unfoldingWord*

Макет рабочего пространства без заголовка – *Схема компонентов пользовательского интерфейса* React unfoldingWord

Валидатор контента – *USFM, Markdown, TSV* Javascript unfoldingWord

Примечание: Текущий список компонентов и команд разработчиков находится на сайте по адресу https://opencomponents.io.

Недавно стажер-разработчик программного обеспечения смог создать многофункциональный прототип мобильного приложения Библии менее чем за 6 недель благодаря существующим компонентам. Не имея опыта работы ни с Библией, ни с разработкой программного обеспечения, Имад разработал Koniortos на основе компонента Proskomma JavaScript и React Native. Поскольку Proskomma выполняет всю сложную работу по обработке текста USFM, а затем представляет его разработчику в виде графика, было потрачено очень мало времени на разработку сложной проблемы, которая уже решена.



Принципы совместной экосистемы библейских технологий

В разделе "Переосмысление развития библейских технологий в 21 веке" мы предположили, что глобальный контекст библейских технологий изменился от "сложного" (много движущихся частей, но все же предсказуемого и структурированного) к "комплексному" (элементы взаимодействуют непредсказуемым образом). Поскольку сеть участников экосистемы открытых компонентов продолжает расти, сложность сети и количество технологий, изобретенных для решения различных задач, также увеличивается. Следовательно, важно определить и сообщить параметры, которые ведут к здоровой экосистеме. В противном случае усложнение взаимодействия между различными технологиями может привести к снижению эффективности и совместимости. Простые правила могут быть использованы в сложных ситуациях для максимизации сотрудничества.[29]

[29] Салл и Эйзенхардт используют концепцию принципов как "простые правила" и утверждают, что "встреча сложности со сложностью может создать больше путаницы, чем разрешить". Далее они спрашивают: "Как люди могут управлять сложностью, присущей современному миру? Наш ответ, основанный на исследованиях и реальных результатах, заключается в том, что простые правила лучше справляются со сложностью, чем сложные решения" (Sull and Eisenhardt, Simple Rules, 12).

Принципы построения экосистемы открытых компонентов:

Принципы, которые касаются практических аспектов разработки программного обеспечения для экосистемы открытых компонентов, можно сформулировать в одном предложении: *сосредоточьтесь на менее сложных инструментах, которые могут быть использованы большим числом разработчиков для решения задач большего числа пользователей.*

1. Сосредоточьтесь на менее сложных инструментах… Здесь мы отмечаем принцип "меньше значит больше": выгоднее разрабатывать специализированные инструменты, которые решают отдельные задачи.

2. … используются многими разработчиками …Сделать это путем создания модульного и многократно используемого кода с открытым исходным кодом, который хорошо задокументирован (и легко тестируется в интерактивном режиме с помощью онлайновой "песочницы").

3. … для решения проблем большего числа пользователей. Отдавайте предпочтение простым интерфейсам, повышающим реконфигурируемость, чтобы максимально улучшить удобство для самого широкого круга пользователей.

Принципы создания открытых компонентов

Чтобы проиллюстрировать отличительные особенности экосистемы открытых компонентов, рассмотрим производство автомобилей. Полностью индивидуальное транспортное средство будет состоять исключительно из невзаимозаменяемых функциональных элементов. Все, от свечей зажигания до дверей и колес, будет сделано на заказ только для этого автомобиля. Такой подход имеет некоторые преимущества (например, чрезвычайно точная настройка конструкции автомобиля по каждой детали). Но любому человеку, кроме первоначального инженера, будет трудно понять, как все компоненты сочетаются в автомобиле. Следовательно, это приведет к появлению автомобиля, который будет обременительным и дорогим для модернизации или ремонта.

Именно поэтому автомобильная промышленность сегодня, как правило, характеризуется взаимозаменяемыми компонентами, которые могут быть индивидуально отремонтированы и модернизированы. Проектирование экосистемы позволяет собирать из одних и тех же компонентов различные транспортные средства для самых разных целей - от мотоциклов до седанов и пикапов. Инноваторы могут совершенствовать отдельные компоненты без необходимости разрабатывать под них целый автомобиль - новый или усовершенствованный компонент может быть интегрирован в автомобиль, состоящий из других предварительно изготовленных компонентов. Аналогичным образом, экосистема открытых компонентов может быть определена тремя принципами, которые отличают ее от других моделей разработки технологий: легкость повторного использования, легкость изменения, легкость обучения.

1. Легко использовать повторно

Основная цель экосистемы - максимизировать количество проектов, которые могут использовать преимущества существующего кода, при этом минимизируя количество дублирующих усилий. В известной книге о хорошем проектировании программного обеспечения "*Прагматичный программист*" говорится о возможности повторного использования:

> Вы пытаетесь создать среду, в которой легче найти и повторно использовать существующий материал, чем писать его самому. Если это нелегко, люди не будут этого делать. А если вы не сможете повторно использовать, вы рискуете дублировать знания..[30]

Это ключевые параметры, которые максимально повышают возможность повторного

использования компонента.

- **Создавайте экосистему:** При правильном разделении задач компонент можно использовать повторно не только в наборе программных инструментов одной организации, но и совместно с несколькими организациями. Больше не нужно копировать и вставлять файлы и строки кода в разные проекты или искать каждый проект, который делает то же самое, чтобы внести исправления и устранить ошибки. Главное преимущество - это не повторять одну и ту же работу. Это относится как к одному приложению, так и к разным приложениям одной команды и даже разных команд. Правильное версионирование модулей с помощью Semantic Versioning может обеспечить обратную совместимость и позволить каждому проекту заблокировать и ограничить непредвиденные изменения.[31]

---

[30] Thomas and Hunt, *Pragmatic*, ch. 10 § “Topic 9: DRY—The Evils of Duplication, Tip 16: Make It Easy to Reuse”, emphasis added.

[31] См. Семантическая версификация (https://semver.org).

- **Делайте что-то одно хорошо:** Маленькие компоненты легче использовать повторно, поэтому идеально ограничить область применения каждого из них. Используйте философию Unix: делайте что-то одно и делайте это хорошо. Сложные модули, которые делают слишком много, могут быть заманчивыми, особенно в отношении связанной функциональности, но за это приходится платить ограничением повторного использования.[32]
- **Стандартизируйте ввод/вывод:** Для максимального повторного использования среди отдельных систем важно стандартизировать ввод и вывод, особенно в отношении форматов данных.[33] Это может быть сложным для разработки нового функционала, поскольку может не быть существующего стандарта. Много времени и усилий может быть потрачено на изучение стандартизации, поэтому будьте осторожны, чтобы не застрять в этом. Вы всегда можете выпустить новую версию модуля с уточненным стандартом ввода/вывода. Каждая итерация должна стремиться к его улучшению.
- **Де-факто stateless:** Модули и компоненты без сохранения состояния (или те, которые максимизируют количество состояния, управляемого изнутри) проще всего использовать повторно. Некоторые функциональные возможности требуют состояния и могут быть абстрагированы так, чтобы код был разбит на части без состояния, а также завернут с необязательным управлением состоянием. Выход из состояния позволяет отделить проблему управления состоянием от приложения и повысить совместимость.
- **Открытый исходный текст:** Все компоненты экосистемы должны быть открыто лицензированы под лицензией с открытым исходным кодом, позволяющей максимально широкое повторное использование компонента (например, MIT Public License или аналогичная).

2. Легко изменить

Вторым ключевым моментом проектирования экосистемы открытых компонентов является то, что компоненты должны быть разработаны с учетом максимальной гибкости для адаптации к изменяющимся потребностям пользователей и контекстуальным факторам. Это означает написание кода, который легко обновлять и заменять, как описано в книге "*Прагматичный программист*":

---

[32] Философию UNIX можно резюмировать так: пишите программы, которые делают одно дело и делают его хорошо, пишите программы, которые работают вместе, и пишите программы для работы с текстовыми потоками, потому что это универсальный интерфейс. (Salus, *UNIX*, 52).

[33] Перспективным решением для создания совместимых форматов данных для библейских проектов является Scripture Burrito (https://docs.burrito.bible/).

> Вещь хорошо спроектирована, если она адаптируется к людям, которые ее используют. Для кода это означает, что он должен адаптироваться через изменения. Поэтому мы придерживаемся принципа: Легче изменить.[34]

[34] Томас и Хант, Прагматик, гл. 2 § "Совет 14: Хороший дизайн легче изменить, чем плохой". Далее авторы отмечают: "Если вы не уверены в том, какую форму примут изменения, вы всегда можете вернуться к главному пути "легко изменить": попытаться сделать то, что вы пишете, заменяемым. Таким образом, что бы ни случилось в будущем, этот фрагмент кода не будет препятствием". Позже в книге они расширяют значение модульности и изменяемости кода в свете будущих сложностей: "Чем больше вам приходится предсказывать, как будет выглядеть будущее, тем больше риск, что вы ошибетесь. Вместо того чтобы тратить усилия на разработку кода для неопределенного будущего, вы всегда можете вернуться к разработке кода, который можно заменить. Сделайте так, чтобы код можно было легко заменить на что-то более подходящее. Если сделать код заменяемым, это также поможет в обеспечении целостности, связности, разъединения и не дублирования, что приведет к улучшению общей конструкции" (гл. 4 § "Тема 27: Не обгоняйте свои фары").

В качестве аналогии вспомните, когда в последний раз у вас спустило колесо на автомобиле. Представьте, что бы вы почувствовали, если бы колесо не было легко снять и было интегрировано с осью. А теперь представьте, что бы вы чувствовали, если бы ваша ось была частью шасси. У вас не было бы возможности заменить шину на обочине дороги. Есть причина, по которой колеса легко заменяются. Аналогичным образом компоненты в экосистеме открытых компонентов должно быть легко изменить за счет реализации как можно большего количества следующих характеристик.

- **Разделение задач:** Самые полезные компоненты экосистемы разработаны таким образом, чтобы их можно было легко заменить для внедрения новой или улучшенной функции. Цель состоит в том, чтобы найти правильный баланс с точки зрения детализации компонента, как например, замена автомобильной шины на обочине дороги, а не замены колеса, оси и ходовой части с помощью эксперта-механика. В большинстве случаев легче ошибиться, если компонент слишком мал, так как всегда можно объединить более мелкие части вместе, чтобы упростить реализацию и совместимость.
- **Функциональная изоляция (обособление):** Это дает руководящие принципы для логического обмена между связанными частями. Это дает многочисленные преимущества в таких областях, как простота тестирования, уменьшение побочных эффектов, четко определенные границы и простота замены.
- **Разделяйте пользовательский интерфейс и бизнес-логику:** Разделение пользовательского интерфейса и бизнес-логики также важно, поскольку это позволяет сделать пользовательский интерфейс и работу с данными взаимозаменяемыми. Иногда вам нравится пользовательский интерфейс, но вы хотите обеспечить собственную функциональность. В других случаях вам нравится функциональность, но вы хотите использовать свой собственный пользовательский интерфейс.[35]

[35] Следует отметить, что во многих случаях разделение задач происходит на этапе создания прототипа (см. ниже). Не всегда оптимально или понятно, как эффективно разделить задачи, пока вы не докажете функциональность. Это может быть итеративный процесс.

- **Минимизировать миграцию данных:** Сведите к минимуму миграцию данных, чтобы облегчить обновление и сократить долгосрочное обслуживание. Избегайте таких вещей, как сохранение промежуточных форматов данных, включая базы данных, и по возможности используйте пользовательские форматы данных. Может возникнуть соблазн хранить или кэшировать промежуточное данных по причине производительности, но за это приходится платить немалую цену. Это может показаться единственным решением, но редко когда оптимизация стоит того, чтобы идти на компромисс в долгосрочной перспективе. Если следовать этому правилу, то в конечном итоге появятся эффективные решения с дополнительным преимуществом в виде минимизации затрат на долгосрочное обслуживание.
- **Максимальное обучение посредством итеративного развития:** Для проектирования и разработки легко изменяемого программного обеспечения может потребоваться время и

множество итераций. Один из подходов к созданию приложений, удобных для использования, и при этом обеспечивающих надлежащее разделение задач, заключается в последовательности Доказательство концепции → Прототип → Минимально жизнеспособный продукт (MVP). Начиная с пробной концепции, вы можете определить видение того, что должно быть создано, но быстро и небрежно. Не нужно тратить время, беспокоясь о том, как сделать все правильно. Цель - сделать все как можно быстрее. Следующий этап - переписывание кода таким образом, чтобы начать модульное разделение задач. Разбейте код на части, которые кажутся разумными. Когда прототип пройдет несколько итераций, структура и дизайн MVP станут более ясными и очевидными.[36]

[36] Подробнее о повышении уровня обучения путем итераций через PoCs, прототипы и MVP см. в Klapp, "Lean Expectations"

3. Легко научиться

Каждый разработчик знает, что документирование кода очень важно, как для собственного использования в будущем, так и для информирования других людей, которым может понадобиться поддерживать код. "*Прагматичный программист*" напрямую затрагивает эту тему в совете 13: **"Встраивайте документацию, а не прикручивайте ее"**:

> Из комментариев в исходном коде легко создать красивую документацию, и **мы рекомендуем добавлять комментарии к модулям и экспортируемым функциям**, чтобы дать другим разработчикам возможность получить преимущество, когда они начнут их использовать.[37]

[37] Томас и Хант, Прагматика, гл. 1 § "Тема 7: Общайтесь".

Если картинка стоит тысячи слов, то **визуальная песочница**, позволяющая разработчику протестировать функциональность компонента в своем веб-браузере, стоит как минимум столько же, сколько традиционная документация! Даже свободные компоненты часто можно обернуть в визуальный слой, который позволяет заинтересованным сторонам, руководителям проектов и разработчикам тестировать собственные данные с помощью компонента и сравнивать плюсы и минусы различных вариантов.

Заключение

Переход к производству сменных деталей позволил случиться промышленной революции и послужил катализатором изобретения практически всего, чем мы пользуемся сегодня - от автомобилей и самолетов до мобильных телефонов и Интернета. Подобным образом, у нас есть возможность разработать новый подход к библейской технологии - подход, построенный на открытых компонентах, которые одновременно поощряют нелинейные инновации и значимое сотрудничество в сети библейских разработчиков. Создавая технологические компоненты, которые легко изучать, повторно использовать и изменять, разработчики создадут благоприятные условия для инноваций и широкого использования библейских технологий. Новые команды разработчиков найдут низкую планку для входа в пространство библейских технологий, поскольку все больше функциональных возможностей становится доступным благодаря растущему числу компонентов. Эти факторы помогут максимизировать поток идей, что приведет к появлению инновационных решений для большего числа людей, вовлеченных в производство, распространение и использование переводов Библии и других библейских ресурсов на любом языке.

Узнать больше об экосистеме открытых компонентов, изучить существующие компоненты, научиться создавать новые компоненты и общаться с другими участниками движения можно по адресу **https://opencomponents.io.**

Ссылки

Beard, Ross. “Microservice Architecture” – “Why a Microservice Architecture Is Important (4 Reasons).” Shadow-Soft, April 9, 2018. https://shadow-soft.com/why-microservice-architecture/.

“Composability.” In Wikipedia, June 26, 2021. https://en.wikipedia.org/wiki/Composability.

English, Trevor. “Interchangeable” – “The History of Interchangeable Parts in the Industrial Revolution,” August 31, 2019. https://interestingengineering.com/the-history-of-interchangeable-parts-in-the-industrial-revolution.

Johnson, Steven. *Good Ideas – Where Good Ideas Come From: The Natural History of Innovation.* Reprint edition. New York: Riverhead Books, 2011.

Jore, Tim. “Established” – “From Unreached to Established.” unfoldingWord, May 16, 2018. https://unfoldingword.org/established.

Kania, John, and Mark Kramer. “Collective Impact.” Stanford Social Innovation Review, 2011. https://ssir.org/articles/entry/collective_impact.

Klapp, Christopher. “Lean Expectations — PoC, Prototype, MVP.” Medium, September 8, 2018. https://medium.com/@klappy/lean-expectations-poc-prototype-mvp-140749383fd4.

Lin, Robert. “Monolithic” – “Monolithic vs Modular.” Medium, November 3, 2016. https://medium.com/@berto168/monolithic-vs-modular-9b6d69684a2c.

Preston-Werner, Tom. “Semantic Versioning 2.0.0.” Semantic Versioning. Accessed June 28, 2021. https://semver.org/.

Ries, Eric. *The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* New York: Crown Business, 2011.

Salus, Peter H. *A Quarter Century of UNIX.* Addison-Wesley Publishing Company, 1994.

Slootweg, Sven. “Monolithic vs. Modular - What’s the Difference?” Gist. Accessed April 6, 2021. https://gist.github.com/joepie91/7f03a733a3a72d2396d6.

Snowden, David J., and Mary E. Boone. “A Leader’s Framework for Decision Making.” Harvard Business Review, November 1, 2007. https://hbr.org/2007/11/a-leaders-framework-for-decision-making.

Sull, Donald, and Kathleen M. Eisenhardt. *Simple Rules: How to Thrive in a Complex World.* Reprint edition. Mariner Books, 2016.

Thomas, David, and Andrew Hunt. *The Pragmatic Programmer: Your Journey to Mastery, 20th Anniversary Edition.* 2nd edition. Addison-Wesley Professional, 2019.

Wu, Tim. *The Master Switch: The Rise and Fall of Information Empires.* Reprint edition. Vintage,

2010. 24