МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)
БГТУ.СМК-Ф-4.2-К5-01

Факультет И «Информационные и управляющие системы»
шифр Наименование

Кафедра И5 «Информационные системы и программная инженерия»
шифр Наименование

Дисциплина Проблемы человеко-машинного взаимодействия

ОТЧЁТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
ПО ТЕМЕ

Особенности процесса разработки программного обеспечения с модульной архитектурой на этапе сопровождения

Выполнил студент группы И9М33

Крылов К.А.
Фамилия И.О.

РУКОВОДИТЕЛЬ

Гущин А.Н. _____
Фамилия И.О. Подпись

Оценка _____

«_____» _____ 2019 г.

САНКТ-ПЕТЕРБУРГ
2019 г.

РЕФЕРАТ

Отчёт о НИР 24 с., 4 рис., 1 табл., 16 источников.

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, МОДУЛЬНАЯ АРХИТЕКТУРА, СОПРОВОЖДЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ЭКОСИСТЕМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ЭВОЛЮЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПОЛЬЗОВАТЕЛЬСКОЕ ВЗАИМОДЕЙСТВИЕ

Цель работы – рассмотрение особенностей процесса разработки программного обеспечения с модульной архитектурой на этапе сопровождения.

В ходе выполнения работы был произведён обзор литературы и введены понятия, связанные с модульной архитектурой, эволюцией программного обеспечения и экосистем программного обеспечения. На основании приведённых понятий была проведена оценка влияния модульного подхода на эволюцию экосистемы программного обеспечения, а также возможные способы улучшения использования подобного подхода.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Ввиду стремительного развития индустрии разработки программного обеспечения, а также усложнения протекающих в ней процессов, существует потребность подходах, способных предоставить разработчикам возможность принимать более эффективные решения на этапах проектирования, разработки, сопровождения и анализа состояния программного обеспечения, а также повышать эффективность разработки и снижать её стоимость. Модульный подход является одним из эффективных подходов к организации разработки программного обеспечения.

Модульный подход к разработке характерен тем, что при разработке программа разбивается на относительно независимые составные части - программные модули. При этом каждый модуль может разрабатываться, программироваться, транслироваться и тестироваться независимо от других. Внутреннее строение модуля для функционирования всей программы, как правило, значения не имеет, а при модификации алгоритма, реализуемого модулем, структура программы не должна меняться [1].

Однако, одного лишь рассмотрения эффективного подхода к разработке не всегда бывает достаточно, так как вследствие нестабильности рынка, появления новых технологий, перемены потребностей пользователей и прочих факторов, даже хорошо сделанное программное обеспечение может не пользоваться популярностью и провалиться на рынке.

В данной работе будет рассмотрен процесс разработки программного обеспечения с модульной архитектурой на этапе сопровождения, однако кроме, собственно, особенностей модульной архитектуры, будут также затронуты особенности внесения изменений в программное обеспечение, а также будет рассмотрена усложнённая модель представления взаимодействия разработчика программного обеспечения с пользователями.

1 МОДУЛЬНАЯ АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Для определения понятия модульной архитектуры программного обеспечения следует определить понятие программного модуля. Программный модуль это программа или отдельная её часть, рассматриваемая как единое целое в контекстах хранения, замены, трансляции, объединения с другими программными модулями и её загрузки в оперативную память ЭВМ [1]. Программы с модульной архитектурой можно конструировать из модулей, как из составных частей, каждая из которых может разрабатываться, программироваться, транслироваться и тестироваться независимо от других. Внутреннее строение модулей для функционирования всей программы, как правило, значения не имеет, а при модификации алгоритма работы модуля структура программы не должна меняться.

В противоположность модульной архитектуре можно привести монолитную архитектуру программного обеспечения, для которой характерно объединение всей функциональности в одной программе, которую невозможно разделить на отдельно транслируемые модули. Для такого подхода к разработке программного обеспечения характерно то, что вся логика, которая необходима для поддержки функций заложенных на этапе проектирования, уже содержится в программе.

Для программы с модульной архитектурой характерны следующие особенности:

– сокращение затрат на разработку за счёт интеграции в программу уже готовых модулей, если такие имеются;

– снижение затрат при внесении изменений в программное обеспечение на этапе сопровождения;

– увеличение затрат на этапах проектирования и разработки за счёт более сложной архитектуры программы, по сравнению с монолитной архитектурой.

Следует отметить, что использование модульной архитектуры целесообразно, если предполагается, что разрабатываемое программное обеспечение будет значительно изменяться на этапе сопровождения, в связи с изменением потребностей пользователей. Для программного обеспечения требования к которому заранее определены, и в будущем не ожидается их значительного изменения, как, например, в сфере медицины, транспорта или военного дела, целесообразно использование монолитной архитектуры, так как использование модульного подхода скорее увеличит затраты на разработку, не давая значительного прироста качества.

Таким образом, в сфере, где способность быстро изменять различные особенности программного обеспечения в соответствии с текущими потребностями целевой аудитории играет решающую роль, использование модульной архитектуры позволяет сократить расходы на разработку и ускорить выпуск новых версий, что является весомым преимуществом в пользу данного выбора. Также, появляется возможность внесения изменений в программное обеспечение за счёт разработки и интеграции новых модулей как для сторонних разработчиков, так и для конечных пользователей, однако этот аспект будет подробнее рассмотрен в разделе 4, а далее будет рассмотрена специфика изменения программного обеспечения на этапе его сопровождения.

2 ЭВОЛЮЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Можно выделить два основных подхода к изменению систем: эволюционный и революционный. Обычно, эволюционные изменения происходят постепенно, в течение продолжительного времени, изменяя лишь некоторое подмножество её свойств. Революционные изменения, напротив, являются фундаментальными, охватывают большую часть свойств системы и происходят в краткие сроки. Также, важной особенностью является то, что результат эволюционных изменений является более устойчивым и предсказуемым, нежели результат революционных изменений.

Для получения более широкого взгляда на явление эволюции программного обеспечения, можно рассмотреть естественную эволюцию, как наиболее изученный процесс эволюционного изменения, и постараться найти возможные сходства и различия между ними.

2.1 Естественная эволюция

Биологическая эволюция это происходящий в череде поколений процесс приспособления биологических систем к условиям окружающей среды. Отличительным свойством биологической эволюции является адаптивность, позволяющая биологическим системам приспосабливаться под условия существования.

Для того чтобы приступить к описанию сути эволюционного процесса, следует ввести ряд ключевых определений.

Ген – последовательность ДНК или РНК, которая несёт информацию о строении молекулы, выполняющей какую-либо функцию. Генотип – совокупность генов одной особи. Экспрессия гена – это процесс, в ходе которого наследственная информация от гена преобразуется в функциональный продукт — РНК или белок. Фенотип – совокупность характеристик, присущих особи на определённой стадии развития. Фенотип является результатом экспрессии генотипа и влияния окружающей среды, а

также совокупности этих двух факторов. Генофонд – совокупность всех генных вариаций определённой популяции. Популяция – совокупность особей одного вида, обладающая общим генофондом, способная к более-менее устойчивому самовоспроизводству [3].

В 1960 году академик И.И. Шмальгаузен сформулировал представление об эволюции, как об авторегулируемом процессе, основанном на обратной связи. В любой авторегулируемой системе выделяются два блока: управляющий блок – регулятор и регулируемый блок – объект управления. Рассматривая процесс эволюции, в качестве регулируемого блока можно выделить популяцию, а в качестве регулятора – среду в которой обитает популяция, и к условиям которой она должна приспосабливаться. Средой, в которой происходят эволюционные преобразования популяции, является экосистема [3]. Данная система представлена на рисунке 1.

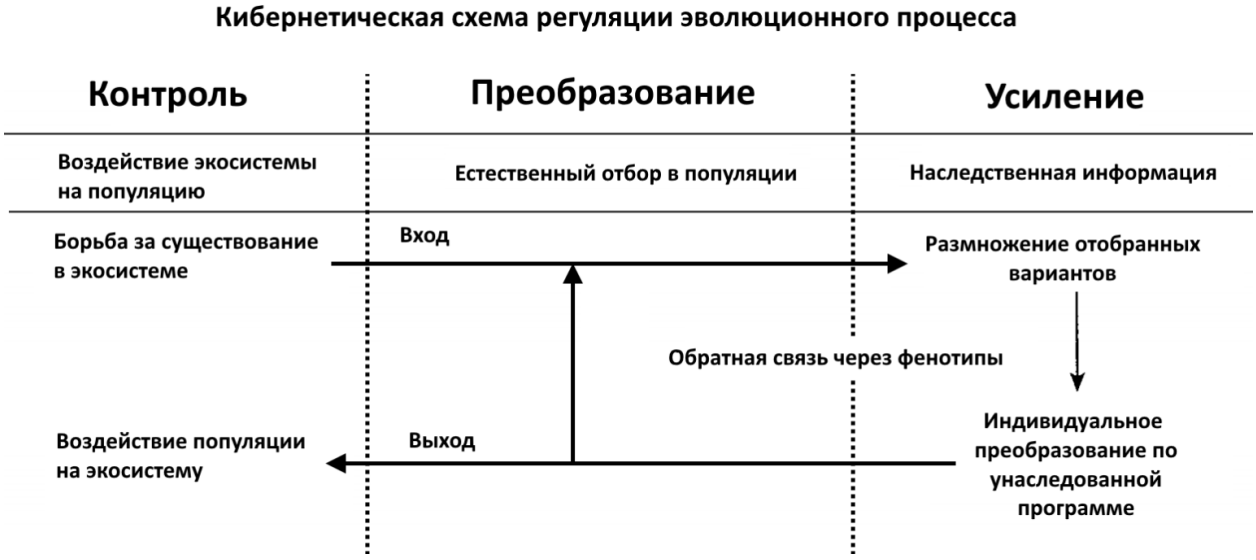

Рисунок 1 – эволюция, как авторегуляторный процесс

Популяция меняется с каждым поколением благодаря генетической изменчивости, подавляющий вклад в которую привносят случайные мутации. В данной модели генетическая изменчивость является каналом прямой связи. Канал обратной связи действует на фенотипическом уровне – на основе генетической информации экосистема «узнаёт» насколько популяция соответствует условиям регулятора.

Таким образом, в эволюционном процессе происходит два этапа перекодировки информации: первый – превращение генетической информации в фенотипическую, и второй – размножение особей, прошедших естественный отбор [3].

2.2 Эволюция программного обеспечения и её связь с естественной эволюцией

Основываясь на описании особенностей процесса естественной эволюции, представим эволюционный подход к изменениям в отношении процесса разработки программного обеспечения.

Так, например, как и в случае с изменяющейся естественной средой – изначальные требования пользователей, в соответствии с которыми создаётся программное обеспечение, в течение времени будут изменяться вслед за изменениями, происходящими в реальном мире. Регулятором являются предпочтения множества, состоящего из владельцев программного обеспечения, его разработчиков и пользователей. Также вклад могут вносить такие факторы как текущее состояние той области рынка, в которой программное обеспечение конкурирует со своими аналогами, совпадение с запросами и ожиданиями пользователей, актуальность используемой технологической платформы и инструментов и многое другое [4, 5, 6].

Производя сравнение процессов естественной эволюции и эволюции программного обеспечения, можно выделить некоторое множество схожих элементов. Общие элементы, выявленные в процессе соотнесения процессов естественной эволюции и эволюции программного обеспечения, представлены в таблице 1.

Таблица 1 – соотнесение элементов процессов естественной эволюции и эволюции программного обеспечения

Элемент системы	Естественная эволюция	Эволюция программного обеспечения

Элементарная единица эволюции	Поколение популяции	Версия программного обеспечения
Способ изменения	Случайные изменения генома	Осознанные изменения, которые разработчики принимают, на основании внешних факторов, с учётом неполноты информации
Прямая связь (вход)	Генофонд популяции	Исходный код
Регулируемый блок	Фенотип популяции	Скомпилированный исходный код, который исполняется на компьютере пользователя
Регулятор	Естественная экосистема (живые организмы, окружение, их взаимодействие)	Предпочтения множества из владельцев, разработчиков и пользователей
Обратная связь (выход)	Успех (захват жизненно-важных ресурсов, размножение) или неуспех (вымирание) популяции	Успех (рост популярности, увеличение продаж) или неуспех (потеря аудитории, падение продаж) программного обеспечения

Основываясь на результатах соотнесения процессов естественной эволюции и эволюции программного обеспечения, можно выделить основное различие, а именно — способ внесения изменений в эволюционирующую систему. В случае естественной эволюции все происходящие изменения имеют случайную природу, следовательно результат такого процесса непредсказуем — он может равновероятно привести как к негативным, так и позитивным последствиям. В случае эволюции программного обеспечения, подавляющее большинство производимых изменений является результатом осознанных изменений производимых разработчиками. Однако следует учесть, что хоть разработчики и опираются при принятии решений на рациональные предпосылки, которые, возможно, подкреплены некоторой статистикой, невозможно полностью нивелировать фактор случайности.

Исходя из вышесказанного, можно сделать вывод, что чем ближе вносимое разработчиками изменение сдвигает набор характеристик программного обеспечения к условному «эталонному» набору характеристик,

который установлен регулятором на текущий момент, тем оно эффективнее. Иначе говоря, если усовершенствовать процесс разработки программного обеспечения таким образом, чтобы уменьшилось влияние случайности на процесс разработки, точнее оценивать текущие критерии регулятора, а также предугадывать динамику их изменения, то это в разы сократило временные и финансовые затраты на итерации разработки, а также повысило качество программного обеспечения с точки зрения пользователя.

Таким образом, для того чтобы описать текущие критерии регулятора, которые характеризуются предпочтениями множества, состоящего из владельцев программного обеспечения, его разработчиков и пользователей, необходимо использовать особую модель, которая бы учитывала всех участников процесса разработки программного обеспечения, а также позволяла принимать более эффективные решения на этапах проектирования, разработки, сопровождения и анализа состояния программного обеспечения. Одной из таких моделей является экосистема программного обеспечения.

3 ЭКОСИСТЕМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Понятие экосистемы программного обеспечения базируется на на понятии семейства программного обеспечения. Семейство программного обеспечения – это серия программных продуктов предназначенных для применения в некоторой предметной области, процесс разработки которых основан на использовании разделяемых базовых компонентов предопределённым способом. Вместе с тем все программные решения одной серии различны.

Каждый раз при создании нового решения повторно используется общее множество разделяемых базовых компонентов, которые в совокупности формируют технологическую платформу программного обеспечения. Далее в процессе разработки создаётся несколько дополнительных компонентов, а некоторые компоненты адаптируются согласно новым требованиям [7, 8, 9].

Обычно, разработка семейства программного обеспечения ведётся внутри компании, с привлечением только собственных сотрудников. Однако компания может сделать выбор в пользу раскрытия своей технологической платформы и предоставления открытого доступа к ней. Произойти это может от осознания факта, что компания больше не в силах самостоятельно разработать все компоненты программного продукта, и в то же время удовлетворить все пользовательские потребности. Как только компания делает такой выбор, она совершает переход от семейства программного обеспечения к экосистеме программного обеспечения [9, 10].

В результате такого перехода доступ к технологической платформе получают независимые разработчики, которые могут вносить непосредственный вклад в её развитие, добавляя в её состав новые компоненты. Таким образом, разработчики, при условии достаточной развитости экосистемы программного обеспечения, получают широкий спектр доступных возможностей, предоставляемых технологической платформой и компонентами, которые были созданы другими разработчиками.

Понятие экосистемы заимствовано из области экологии, где экосистемой называют биологическую систему, состоящую из сообщества живых организмов, среды их обитания, системы связей, осуществляющей обмен веществом и энергией между ними [11]. Это обусловлено тем, что процессы, протекающие в биологических экосистемах, находят своё отражение в отношениях между участниками децентрализованной разработки программного обеспечения.

1. Сообществом живых организмов можно считать множество независимых разработчиков, каждый из которых преследует свои цели.
2. Средой обитания можно считать совокупность технологической платформы и дополнительных компонентов, которые используются при разработке нового программного обеспечения.
3. Обмен веществом или энергией заменяется на выгоду, которая может быть как коммерческой (продажи, отчисления), так и некоммерческой (известность, опыт, идеология).

Связи между участниками также могут быть представлены как симбиотические отношения, характерные для биологических экосистем. Симбиотические отношения – это категория биотических отношений, в которых сожительство повышает адаптивные возможности организма за счёт использования особенностей партнёра [11]. Выделяют следующие виды симбиотических отношения:

1. Мутуализм – два участника получают взаимную выгоду от взаимодействия друг с другом.
2. Комменсализм – один участник получает выгоду от взаимодействия, второй не получает ничего.
3. Антагонизм – два участника конкурируют за общие ресурсы и выгода одного, исключает выгоду другого.
4. Паразитизм – один участник получает выгоду от взаимодействия, второй получает ущерб.

5. Аменсализм – один участник получает ущерб от взаимодействия, второй не получает ничего.

6. Нейтрализм – два участника не получают ничего от взаимодействия.

Для лучшего представления такого способа описания взаимодействия участников экосистемы программного обеспечения на рисунке 2 изображено некоторое множество проектов и типы связей между ними для ядра операционной системы Linux [12].

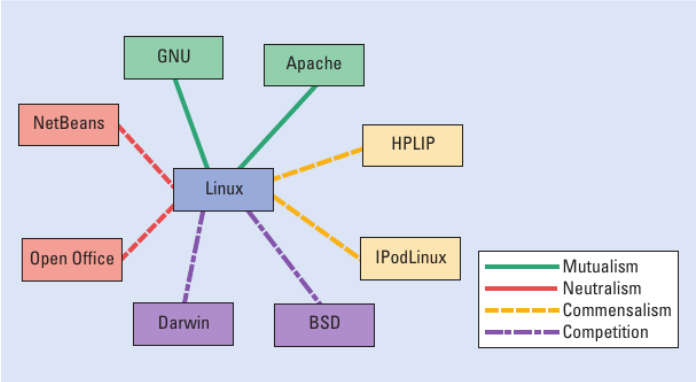

Рисунок 2 – проекты и типы связей между ними для ядра операционной системы Linux

Также в работе [12] отображена степень успешности двух проектов, выраженная в процентном отношении захвата рынка, для операционной системы Mac OS и веб-браузера Safari, которые связаны друг с другом отношениями мутуализма – данные отношения представлены на рисунке 3.

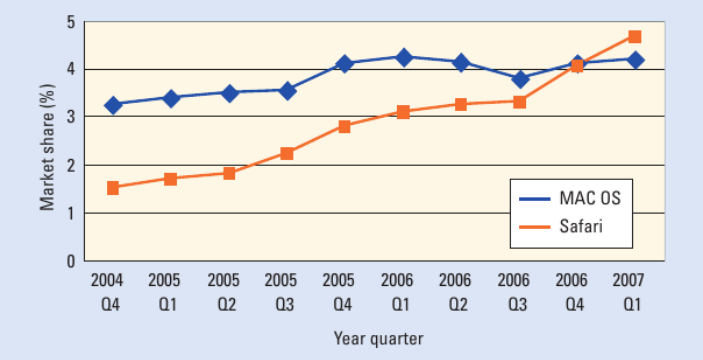

Рисунок 3 – степень успешности проектов Mac OS и Safari

Таким образом, основываясь на приведённых рассуждениях, можно сформулировать само понятие экосистемы ПО. Экосистема программного обеспечения – это организация процесса разработки программного

обеспечения, при которой его независимые друг от друга участники, совместно используя технологическую платформу, вступают друг с другом в симбиотические отношения, характерные для биологических экосистем [11, 12].

Участники процесса могут занимать следующие роли [12, 13]:

1. Владелец платформы – организация, которая несёт ответственность за развитие экосистемы ПО, оценивает её состояние и принимает организационные решения.

2. Внутренний разработчик – коллектив является частью организации, которая предоставляет технологическую платформу. Имеет доступ к технологической платформе и занимается её непосредственным развитием.

3. Стратегический партнёр – сторонние организации, которые связаны долговременными отношениями с владельцем платформы. Разрабатывают дополнительные компоненты для своих нужд или нужд владельца платформы. Владелец платформы может предоставить возможность вносить изменения в технологическую платформу. Владелец платформы может напрямую повлиять на деятельность стратегического партнёра и предсказать его поведение.

4. Сторонний разработчик – сторонние разработчики или организации, никак не связанные с владельцем платформы. Разрабатывают дополнительные компоненты для собственных нужд, с целью получения личной выгоды. Владелец платформы не имеет прямого влияния на них, а также не может достоверно предсказать их поведение. Способны обеспечить весьма существенный толчок в развитии экосистемы.

5. Пользователь – лицо или организация, пользующаяся программным обеспечением, которое было получено в результате взаимодействия прочих участников. Является участником, который, хоть и косвенно, но оказывает самое существенное влияние на развитие экосистемы программного обеспечения.

Для того, чтобы представить сложность отношений между участниками экосистемы программного обеспечения можно привести иллюстрацию упрощённой модели взаимодействия участников, которая описана в работе [14]. Данная иллюстрация представлена на рисунке 4.

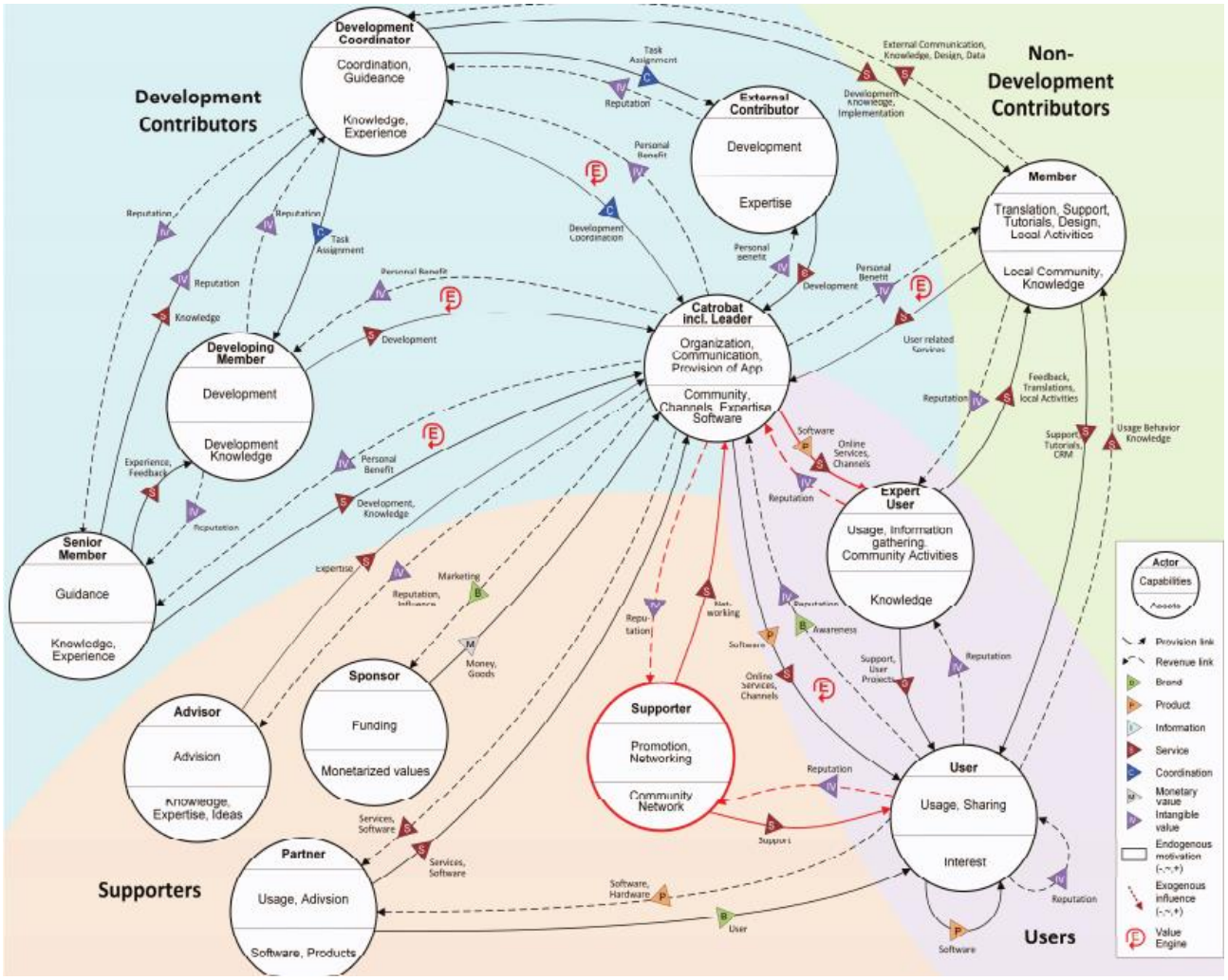

Рисунок 4 – взаимодействие участников экосистемы программного обеспечения

Возвращаясь к понятию эволюции программного обеспечения и принимая во внимание описанные в данном разделе принципы взаимодействия участников экосистемы программного обеспечения, можно отметить, что успех или неудача программного обеспечения зависит от куда большего количества важных факторов, чем если рассматривать процесс разработки в отрыве от экосистемы программного обеспечения для которой предназначено программное обеспечение. Правильный подход к построению взаимодействия с участниками экосистемы программного обеспечения

является настолько же важным, если не важнее, самой задачи разработки программного обеспечения.

4 ВЛИЯНИЕ МОДУЛЬНОГО ПОДХОДА НА ЭВОЛЮЦИЮ ЭКОСИСТЕМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Используя весь материал рассмотренный в рамках предыдущих глав, будет определено влияние модульного подхода на построение программного обеспечения. В рамках данной работы влияние рассматривается только на этапе сопровождения, для которого характерно внесение изменений в функциональность программного обеспечения, однако масштаб изменений может варьироваться в зависимости от сферы задач, для которых разрабатывается данное программное обеспечение. Как уже было упомянуто – в данной работе акцент делается на программное обеспечение, для которого способность быстро изменяться является ключевой.

Использование модульного подхода позволяет строить архитектуру программы таким образом, чтобы модули, отвечающие за определённые функции программы, могли изменяться независимо от других модулей, входящих в состав программы. Развивая эту идею дальше, можно предоставить возможность использования не только тех модулей, которые были разработаны внутренними разработчиками или стратегическими партнёрами, но и модулей, которые разрабатываются сторонними разработчиками.

Предоставление такой возможности создаёт новые задачи, связанные с контролем качества модулей и безопасности использования программного обеспечения пользователями, а также с управлением сообществом независимых разработчиков, которым невозможно дать конкретные указания. Однако, это также открывает и огромный потенциал для роста целевой аудитории, создания и укрепления новых связей с другими участниками внутри экосистемы программного обеспечения.

Потенциальные последствия подобного процесса, с некоторыми допущениями, можно сравнить с развитием операционных систем – представить операционную систему как программу с модульной архитектурой, в которой системное и прикладное программное обеспечение

является модулями. Операционные системы предоставляют пользователю возможность сформировать функциональность из доступных модулей таким образом, чтобы получившаяся программа как можно более полно отвечала его запросам.

Такой подход позволяет, подняться на уровень выше и взглянуть на проблему разработки программного обеспечения, которое бы успешно эволюционировало под другим углом – самыми успешным в эволюционном плане программным обеспечением на текущий момент являются программы, которые позволяют пользователю изменять состав функций таким образом, чтобы они максимально удовлетворяли его потребности. Возвращаясь к сравнению эволюции программного обеспечения с естественной эволюцией – это как если бы живые организмы могли изменять своё ДНК не за счёт случайных мутаций, а как если бы сама экосистема выбирала те гены, которые бы максимально точно подходили под текущие условия окружающей среды и задачи конкретного организма.

Если принять такую идею, то вместо задачи разработки программного обеспечения, которое бы удовлетворило потребности некоторого подмножества пользователей, необходимо решать задачу предоставления пользователю возможности добавлять функции в программное обеспечение, не имеющее заранее заданных ограничений на множество возможных функций. Основными проблемами, в таком случае, становятся формирование множества модулей пользователем, а также разработка и интеграция этих модулей в единую систему. В рамках данной работы рассмотрим только задачу формирования множества модулей пользователем, так как рассмотрение задачи разработки и интеграции модулей потребует введения большого количества дополнительного материала, связанного с разработкой программного обеспечения, что целесообразно сделать в отдельной работе, посвящённой рассмотрению этого вопроса.

Рассматривая операционную систему, как пример программного обеспечения с неограниченным множеством возможных функций, приведём

существующий процесс формирования множества программного обеспечения пользователем. Можно выделить следующие этапы этого процесса:

- формирование потребности;
- поиск;
- выбор;
- установка;
- использование;
- удаление.

Этапы формирования потребности, использования и удаления не будут рассмотрены в данной работе, так как выходят за установленные рамки рассмотрения. Поиск, выбор и установка программного обеспечения отличается в различных операционных системах.

В операционных системах Android и iOS самую важную роль в поиске, выборе и установке программного обеспечения играют магазины приложений. Кроме распределения программного обеспечения по категориям, для удобного поиска, они предоставляют пользователям возможность оставлять оценки и отзывы на программное обеспечение, что формирует важнейший для эволюции программного обеспечения эффект «обратной связи» – разработчики могут учесть допущенные ошибки при проектировании и исправить их, тем самым повышая способность своего программного обеспечения успешно эволюционировать [15, 16].

В операционных системах Windows, MacOS и Linux магазины приложений существуют, но не являются такими популярными – большинство пользователей ищут программное обеспечение на специализированных сайтах, где производится сравнение программного обеспечения предназначенного для выполнения определённых функций. В Windows и MacOS приложения обычно скачиваются с сайтов разработчиков, а в Linux самым распространённым способом является использование пакетных менеджеров.

На основании анализа процесса формирования множества программного обеспечения в различных операционных системах, можно сделать вывод, что наличие магазина приложений положительно сказывается на отношениях в экосистеме между пользователем и разработчиком, он позволяет участникам экосистемы обмениваться информацией о качестве разрабатываемого программного обеспечения оставляя оценки и отзывы, формируя «регулятор», который был описан в разделе эволюции программного обеспечения. Таким образом, если использовать аналог магазина приложений для модулей программного обеспечения, то это позволит обеспечить возможность быстрого и достаточно точного реагирования таких участников экосистемы как владельцы программного обеспечения, а также его внутренние и сторонние разработчики, на текущие запросы пользователей.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был произведён обзор литературы и введены понятия, связанные с модульной архитектурой, эволюцией программного обеспечения и экосистем программного обеспечения. На основании приведённых понятий была проведена оценка влияния модульного подхода на эволюцию экосистемы программного обеспечения, а также возможные способы улучшения использования подобного подхода.

Подводя итог, можно отметить, что успех или неудача программного обеспечения зависит от куда большего количества важных факторов, чем если рассматривать процесс разработки в отрыве от экосистемы программного обеспечения для которой предназначено программное обеспечение. Правильный подход к построению взаимодействия с участниками экосистемы программного обеспечения является настолько же важным, если не важнее, самой задачи разработки программного обеспечения.

Рассмотренный в данной работе модульный подход к разработке даёт возможность создавать программное обеспечение, которое способно более эффективно эволюционировать и точнее подстраиваться под изменяющиеся потребности участников экосистемы программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дорот В. Л. Толковый словарь современной\nкомпьютерной лексики, 3 изд. – БХВ-Петербург, 2004.
2. Three-tier Application Model [Электронный ресурс]. URL: https://docs.microsoft.com/en-us/previous-versions/office/developer/server-technologies/aa480455(v=msdn.10), дата обращения (25.12.2018).
3. Северцов А. С. Теория эволюции: учебник для студентов вузов, обучающихся по направлению 510600 «Биология»/Алексей Сергеевич Северцов //М.: Гуманитар. изд. центр ВЛАДОС. – 2005. – 380 с.
4. Kutschera U., Niklas K. J. The modern theory of biological evolution: an expanded synthesis //Naturwissenschaften. – 2004. – V. 91. – №. 6. – P. 255-276.
5. Hall B. K., Hallgrímsson B. Strickberger's evolution. The integration of genes, organisms, and populations 4th edition. – 2008. – P. 760.
6. Godfrey M. W., German D. M. The past, present, and future of software evolution //Frontiers of Software Maintenance, 2008. FoSM 2008. – IEEE, 2008. – P. 129-138.
7. Иан С. Инженерия программного обеспечения 6-е издание – М: Издательский дом «Вильямс. – 2002.
8. Clements P., Northrop L. Software product lines. – Addison-Wesley, 2002.
9. Bosch J. From software product lines to software ecosystems //Proceedings of the 13th international software product line conference. – Carnegie Mellon University, 2009. – P. 111-119.
10. Van Den Berk I., Jansen S., Luinenburg L. Software ecosystems: a software ecosystem strategy assessment model //Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. – ACM, 2010. – P. 127-134.
11. Николайкин, Н. И. , Николайкина, Н. Е., Мелехова, О. П. Экология. — 5-е. — М.: Дрофа, 2006. — 640 с.
12. Yu L., Ramaswamy S., Bush J. Symbiosis and software evolvability //IT Professional. – 2008. – Vol. 10, №. 4.
13. Manikas K., Hansen K. M. Software ecosystems–A systematic literature review //Journal of Systems and Software. – 2013. – Vol. 86, №. 5. – P. 1294-1306.
14. Vorraber W. et al. Analyzing and Managing Complex Software Ecosystems-A Framework for Creating a Common Understanding and Aligning Shared Goals for Developers and Business Managers, Applied to a Free Open Source Software Project //IEEE software. – 2018.

15. Ickin S., Petersen K., Gonzalez-Huerta J. Why Do Users Install and Delete Apps? A Survey Study //International Conference of Software Business. – Springer, Cham, 2017. – P. 186-191.

16. Panichella S. et al. How can i improve my app? classifying user reviews for software maintenance and evolution //Software maintenance and evolution (ICSME), 2015 IEEE international conference on. – IEEE, 2015. – С. 281-290.