«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»

(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)

Факультет И Информационные и управляющие системы

шифр наименование

Кафедра И9 Систем управления и компьютерных технологий

шифр наименование

ОТЧЁТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
ПО ТЕМЕ

Децентрализованная разработка модульных информационных систем

Выполнил студент группы И9М33

Крылов К.А.

Фамилия И.О.

РУКОВОДИТЕЛЬ

Гущин А.Н. _____

Фамилия И.О. Подпись

Оценка _____

«_____» _____ 2018 г.

САНКТ-ПЕТЕРБУРГ
2018 г.

РЕФЕРАТ

Отчёт о НИР 18 с., 1 рис., 1 табл., 16 источников.

ДЕЦЕНТРАЛИЗОВАННАЯ РАЗРАБОТКА, ПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ, МОДУЛЬНАЯ СТРУКТУРА, СЕМЕЙСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ЭКОСИСТЕМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

Цель работы – определение задач, которые необходимо решить при организации децентрализованной разработки модульных информационных систем.

В ходе выполнения работы был произведен обзор литературы и введены понятия, связанные с понятиями децентрализованной разработки и модульных информационных систем. На основании описания предметной области был определён ряд задач, которые необходимо решить для организации процесса разработки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В индустрии разработки программного обеспечения (ПО) можно выделить тенденцию, которая направлена на снижение затрат ресурсов при создании нового ПО. Это можно объяснить тем, что индустрия стремится удовлетворить возрастающий спрос на новое ПО, которое найдёт своё применение во всех сферах жизни. Несмотря на значительный объём работы по снижению затрат ресурсов, в распространённом на данный момент подходе к разработке программного обеспечения можно выделить две существенные проблемы.

Первую можно охарактеризовать, как необходимость каждый раз заново решать незначительно отличающиеся друг от друга задачи. Несмотря на то, что в некоторых случаях можно воспользоваться готовыми решениями, зачастую такой подход отвергается ввиду следующих причин: сторонние разработчики могут недостаточно сопровождать своё решение, интеграция готового решения с существующим может вызвать большие затруднения, выбор наилучшего решения среди большого множества подходящих может быть затруднительным [1].

Даже принимая во внимание то, что существующие средства разработки позволяют быстро разрабатывать программные продукты используя лишь базовые компоненты, тем не менее, они не предоставляют уже готовых решений, которые относятся к некоторой предметной области и выполняют реальные задачи. В большинстве случаев разработчики вынуждены использовать некоторые абстрактные компоненты, определять отношения между ними, описывать бизнес-логику целевой предметной области, разрабатывать интерфейс взаимодействия с пользователем и прочие ключевые задачи разработки программного продукта.

Второй существенной проблемой является неспособность программного продукта в полной мере удовлетворять потребности большого множества пользователей с различающимися потребностями. Поскольку

конкретные цели пользователя в некоторый момент времени неизвестны заранее и определяются, для непроизводственных систем, в первую очередь личностью пользователя, то разработать программу, в которой были бы заранее учтены все возможные варианты её взаимодействия с пользователем, не представляется возможным [2].

Таким образом, решением двух озвученных проблем можно считать такой программный продукт, который обладает следующими характеристиками.

1. Процесс разработки основан на повторном использовании программного обеспечения, но недостатки повторного использования в нём неким образом компенсированы.
2. Программный продукт может предоставлять не только заданные разработчиком функции – пользователь может определить набор функций самостоятельно, в полном соответствии со своими текущими задачами.

Однако чтобы пользователь мог определить такой набор функций, его необходимо обеспечить достаточно большим множеством всевозможных функций, так как задачи, которые он будет решать, заранее неизвестны. Для предоставления такого большого множества функций одним разработчиком могут потребоваться затраты ресурсов, превышающие адекватные поставленной задаче величины. Следовательно, необходимо некоторым образом обеспечить децентрализацию и распределение разработки подобного программного продукта.

Целью данной работы является определение задач, которые необходимо решить при организации децентрализованной разработки модульных информационных систем.

1 ПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В ходе решения задач, возникающих в различных предметных областях, разработчики создают новое ПО, и если в дальнейшем кто-то столкнётся с подобной задачей, то он сможет воспользоваться уже существующим решением. Для описания такого процесса используется понятие повторного использования ПО.

Повторное использование позволяет значительно сократить длительность разработки, более точно оценить сроки разработки, повысить качество конечного продукта за счёт использования уже отлаженных и проверенных компонентов. Однако у данного подхода существуют и недостатки.

1. Сопровождение – сторонние разработчики могут сопровождать ПО недостаточной документацией, а также не исправлять выявленные в процессе его работы ошибки;
2. Интеграция – внедрение выбранного ПО с другим решением, которое может также использоваться для решения поставленной задачи, может вызвать значительные затруднения;
3. Выбор – если существует большое множество доступных решений, которые подходят под условия поставленной задачи, то выбор среди них наилучшего может быть весьма затруднительным.

Подходя к классификации существующих методов повторного использования программного обеспечения, следует отметить тот факт, что чёткого разделения между ними не существует – возможно использование их различных комбинаций в рамках одного проекта. Некоторые из них непосредственно предоставляют функции уже готовых систем, некоторые лишь позволяют заложить в текущую систему возможности повторного использования в дальнейшем, однако все они, так или иначе, обеспечивают возможности повторного использования программного обеспечения.

Можно разделить методы повторного использования в соответствии с этапами разработки, во время которых они могут привнести максимальный вклад.

1. Определение требований: анализ предметной области;
2. Проектирование: принципы проектирования, использование готовых архитектурных решений (паттернов проектирования);
3. Реализация: сниппеты, принципы разработки, генераторы текста программ, использование готового фрагментов текста программ, использование готового программного кода путём подключения статических библиотек на этапе компиляции;
4. Внедрение и поддержка: использование готового программного кода или текста программ на этапе исполнения (динамические библиотеки, транслируемые языки), использование функций сервисов [1].

Использование стороннего кода на этапе исполнения, за счёт использования динамических библиотек или программного кода, написанного на транслируемом языке, делает возможным применение модульного подхода к организации структуры программы. При этом подходе программа разбивается на относительно независимые составные части – программные модули. При этом каждый модуль может разрабатываться, программироваться, транслироваться и тестироваться независимо от других [3].

2 ДЕЦЕНТРАЛИЗОВАННАЯ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Один из наиболее эффективных подходов к повторному использованию базируется 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 создается несколько дополнительных компонентов, а некоторые компоненты адаптируются согласно новым требованиям [4, 5, 6].

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 [6, 7].

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.

3 ЭКОСИСТЕМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

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 [8]. 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 [9]. 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.

В экосистемах ПО наиболее распространёнными отношениями являются мутуализм, комменсализм и антагонизм [9].

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 [8, 9].

Участники процесса могут занимать следующие роли [9, 10]:

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.

При рассмотрении экосистемы ПО как объекта исследования существуют различные подходы к выделению областей исследования, а также классификации различных её компонентов. В работе [10] производится обзор 90 работ затрагивающих вопросы экосистем ПО, и представлено описание шести возможных подходов, содержащихся среди рассмотренных работ.

1. В работах [11, 12] выделяется три области исследования экосистем ПО: внешняя организация, в которой рассматривается само понятие экосистем ПО, а также рынок вокруг них; внутренняя организация экосистем ПО, в которой главным образом рассматривается обеспечивающее её работу ПО и взаимосвязи между ним; организация экосистем ПО, в которой рассматриваются её участники и их отношения.

2. В работе [13] предлагается рассмотрение архитектурной, социальной, а также бизнес-составляющей экосистем ПО.

3. В работе [14] проводится сравнение между экосистемами ПО и биологическими экосистемами с точки зрения управления ресурсами, а также отдельно подчёркивается важность видового разнообразия, наблюдения за

показателями “здоровья” системы и поддержки социального взаимодействия в области экосистемы ПО.

4. В работе [15] был произведён анализ работ, представленных на конференции IWSECO (International World Software ECOsystems) в период с 2009 по 2010 годы. В ходе анализа выделяются три основных области, к которым можно отнести результаты рассмотренных работ: архитектура, стратегии и решения, социальные сети. Далее, в рамках данной работы область архитектуры экосистемы ПО рассматривается отдельно, в частности, рассматриваются этапы организации технологической платформы, дальнейшей разработки экосистемы программного обеспечения и способы оценки эффективности её работы.

5. В работе [16] было проведено систематическое исследование 44 источников, связанных с рассмотрением области экосистем ПО. В результате исследования были выделены наиболее выделяющиеся характеристики экосистем, преимущества и ограничения связанные с их использованием. Также, на основании содержательной части работ было выделено 8 наиболее часто встречающихся областей исследований, которые связаны с экосистемами программного обеспечения. Каждая из рассмотренных работ была отнесена к той или иной области исследования, а распределение работ по областям было отображено на радиальной диаграмме (Рис. 1).

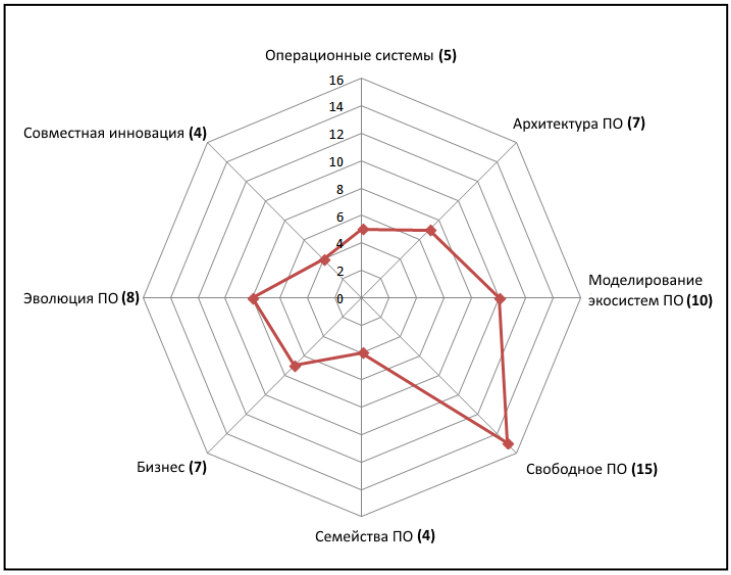

Рисунок 1 – Распределение рассмотренных работ по областям исследований

6. В работе [9] приводится классификация экосистем ПО которая выделяет два независимых параметра: уровень абстракции (операционная система, приложение, инструмент разработки для конечного пользователя) и используемая платформа (ПК, веб-сервис, мобильное устройство). Таким образом, можно сформировать двумерную матрицу, элементами которой являются существующие реализации экосистем ПО (Табл. 1). Также в ней детально рассмотрены особенности процесса перехода от семейства программного обеспечения к экосистеме программного обеспечения.

Таблица 1 - Классификация существующих экосистем ПО

Уровень абстракции	Технологическая платформа		
	ПК	Веб-сервис	Мобильное устройство
Операционная система	MS Windows, Linux, Apple OS X	Google AppEngine, Yahoo developer, Coghead, Bungee Labs	Nokia S60, Palm, Android, iPhone
Приложение	MS Office	SalesForce, eBay, Amazon, Ning	Не существует на текущий момент
Инструмент разработки для конечного пользователя	MS Exel, Mathematica, VHDL	Yahoo! Pipes, Microsoft PopFly, Google’s mashup editor	Не существует на текущий момент

Также в работе [10], выделяются три области, к которым можно отнести основные результаты рассмотренных исследований:

1. Программная инженерия: проблемы определения требований к экосистеме ПО, особенности проектирования архитектуры, организация децентрализованной разработки.

2. Бизнес и менеджмент: показатели “здоровья экосистемы ПО”, децентрализованное управление, особенности взаимодействия с участниками, модели получения выгод и их особенности, вопросы лицензирования.

3. Взаимодействие участников: определение ключевых ролей, рассмотрение моделей добавления новых компонентов в экосистему и показателей “открытости” и “закрытости” экосистемы для участия в её развитии.

Рассмотрение области программной инженерии в работе [10] содержит описание наиболее часто встречающихся тезисов, касающихся организации процесса разработки и особенностей архитектуры экосистем ПО. Описанные тезисы также являются условиями, выполнение которых является необходимым для обеспечения эффективной децентрализованной разработки модульной информационной системы.

1. Архитектура экосистемы ПО должна учитывать её ограничения и особенности, а также поддерживать интеграцию разнообразной функциональности в доступной и безопасной форме.

2. Использование интерфейсов предоставляет сторонним разработчикам возможности для использования возможностей платформы экосистемы ПО. Стабильность и простота интерфейсов платформы являются важными факторами успешной интеграции компонентов и их дальнейшего взаимодействия.

3. Внесение изменений в существующий интерфейс какого-либо компонента может, в свою очередь, привести к несоответствию со всеми зависимыми от него компонентами.

4. Централизация управления не эффективна при широкомасштабной разработке ПО, вместо этого следует использовать децентрализованный подход, в котором основным управляющим механизмом является сама архитектура экосистемы ПО.

5. Непрерывная разработка ПО требует внесения изменений в организацию процесса разработки ПО: направленность на интеграцию компонентов, независимое внедрение новых версий компонентов, замедление темпа выпуска новых версий чтобы сторонние разработчики успевали подстраиваться под вносимые изменения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был произведен обзор литературы и введены понятия, связанные с понятиями децентрализованной разработки и модульных информационных систем. На основании описания предметной области был определён ряд задач, которые необходимо решить для организации процесса разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Крылов К.А., Повторное использование программного обеспечения в области разработки персональных информационных менеджеров – Спб.: БГТУ, III общероссийская молодежная научно-техническая конференция «Старт-2017», 2017.

2. Гущин А.Н., Личностно-ориентированные информационные системы – Спб.: БГТУ, 2012. – 122 с.

3. Дорот В. Л. Толковый словарь современной компьютерной лексики, 3 изд. – БХВ-Петербург, 2004.

4. Иан С. Инженерия программного обеспечения 6-е издание – М: Издательский дом «Вильямс. – 2002.

5. Clements P., Northrop L. Software product lines. – Addison-Wesley, 2002.

6. 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.

7. 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.

8. Николайкин, Н. И. , Николайкина, Н. Е., Мелехова, О. П. Экология. — 5-е. — М.: Дрофа, 2006. — 640 с.

9. Yu L., Ramaswamy S., Bush J. Symbiosis and software evolvability //IT Professional. – 2008. – Vol. 10, №. 4.

10. Manikas K., Hansen K. M. Software ecosystems–A systematic literature review //Journal of Systems and Software. – 2013. – Vol. 86, №. 5. – P. 1294-1306.

11. Jansen S., Brinkkemper S., Finkelstein A. Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems //IWSECO@ ICSR. – 2009.

12. Boucharas V., Jansen S., Brinkkemper S. Formalizing software ecosystem modeling //Proceedings of the 1st international workshop on Open component ecosystems. – ACM, 2009. – P. 41-50.

13. Campbell P. R. J., Ahmed F. A three-dimensional view of software ecosystems //Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. – ACM, 2010. – P. 81-84.

14. Dhungana D. et al. Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature //Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. – ACM, 2010. – P. 96-102.

15. dos Santos R. P., Werner C. M. L. A Proposal for Software Ecosystems Engineering //IWSECO@ ICSOB. – 2011. – P. 40-51.

16. Barbosa O., Alves C. A systematic mapping study on software ecosystems. – 2011.