МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский авиационный институт (национальный исследовательский университет)»

Институт № 6 «Аэрокосмический»

Кафедра 604 «Системный анализ и управление»

Реферат

На тему: «Описание IT решения как сложной технической системы»

По курсу: «Системный анализ»

Выполнил: Шаблий А.Д.
Аспирант группы: М8О-108А-23
Проверила: к.т.н. Дьячук А. К.

Москва 2024

Содержание

1 Введение

В современном мире важную роль в жизни общества занимают информационные технологии. В частности информационные технологии охватывают программное обеспечение цифровых устройств, которые человек применяет в повседневной жизнедеятельности: компьютеры, телефоны, планшеты, автомобили, умные холодильники и т.д. Основным поставщиком программного обеспечения для вышеупомянутых устройств являются IT-компании. Их имена известны, Google, Amazon, Yandex, Vk Group и др. Эти компании производят IT продукт, выполняющий заданный объем функционала и закрывающий обозначенные потребности потребителя.

IT продукты бывают совершенно разные. По их принадлежности:

- для гражданского применения;
- для военного применения;
- для корпоративного применения и т.д.

По сфере применения:

- офисное ПО;
- графические редакторы;
- игры и др.

По уровню критичности, например в авиационной технике:

- катастрофическое;
- аварийное;
- сложное;
- усложнение условий полета;
- без последствий.

В том числе есть класс программных решение - плагинные системы. Это способ организации приложения таким образом, что его конечный объем функционала характеризуется количеством установленных в него расширений - плагинов.

Каждый отдельный плагин включает в себя конечное множество функционала, который в свою очередь основан на требованиях.

Именно характер реализованных требований, а так же их объем потребен заказчику. Все остальное - плагины, их взаимосвязи, язык программирования, на котором они реализованы, библиотеки, которые задействованы, примененные технологии - все это скрыто от заказчика и зачастую его не интересует. Заказчику интересно, какие свои бытовые или бизнес потребности он сможет закрыть от применения IT продукта.

На стадии проектирования приложения зачастую неизвестно, какие требования будут востребованы у заказчиков и определить структуру приложения невозможно. Кроме того, перечень наиболее востребованного функционала может изменяться с течением времени. При необходимости формирования поставки, включающей требуемый объем функционала, поставщик зачастую включает и тот функционал, который не востребован и не оплачен заказчиком, но без которого не быть поставлен востребованный. Это связано с существованием зависимостей у функционала друг на друга.

Моя диссертационная работа посвящена поиску оптимальной структуры приложения, которая бы с одной стороны позволяла формировать поставки с минимальным числом невостребованного у заказчика функционала, а с другой сдерживала неконтролируемый рост сущностей и, тем самым, сдерживала стоимость сопровождения проекта.

2 Описание системы

В качесте системы для исследования я обозначил структуру приложения предназначенного для интеграции в в плагинную систему. Мною проанализирована предметная область, сформулированы ограничения на компоненты системы и предложена формулировка ее описания в виде графовой модели. Полученная модель позволяет оценивать суммарный объем требований, включенный в поставку при заданном объеме минимально необходимых для включения требований. Это применимо для вычисления коэффициента простоя подразделений компании поставщика вовлеченных в пост продажное обслуживание.

Как говорилось ранее, для описания системы был проведен анализ предметной области и сформулирован минимальный объем ограничений сущностей предметной области для описания математической модели. Так, математическая модель состоит из следующих

сущностей, к которым применены ограничения:

1. сущности классифицированы:
   (a) поставка, включающая конкретный перечень плагинов;
   (b) плагины пригодные для интеграции в плагинную систему;
   (c) файлы исходного кода распределенные по плагинам;
   (d) функциональные зависимости между файлами исходного кода и плагинами;
   (e) требования реализованные в файлах исходного кода
2. один файл может реализовывать одно или несколько требований;
3. одно требование может быть реализовано в одном или нескольких файлах;
4. один файл не может быть включен в несколько плагинов;
5. файлы имеют зависимости друг на друга, в том числе и циклические;
6. плагины так же имеют зависимости друг на друга: если в плагин включены файлы, зависящие от файлов, расположенных в другом плагине, то первый плагин зависим от второго;
7. циклические зависимости между плагинами запрещены.

Принципиальная схема приведена на рисунке 1

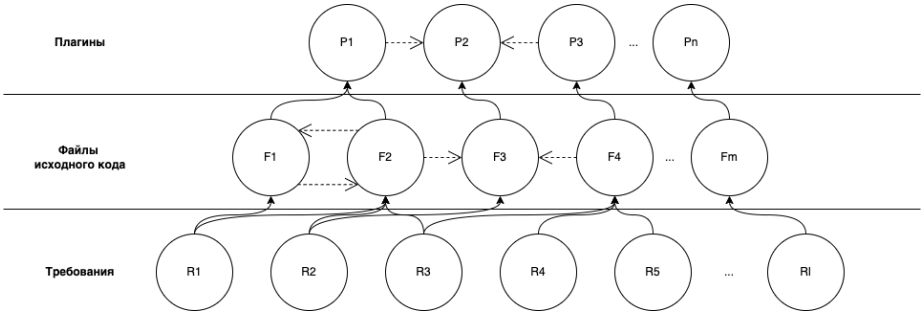

Рис. 1: Система с приведенными подсистемами

Рассматриваемая система является частью жизненного цикла ПО. Начиная от формирования изначального потенциально возможного

перечня требований оформленных в виде ТЗ и заканчивая формированием поставки под нужды конкретного потребителя.

Так, имеется IT продукт с реализованным набором требований заданных в ТЗ. Требования реализованы в файлах исходного кода, которые в свою очередь распределены по плагинам, из которых можно выполнить поставку. Предположим, что заказчику потребны не все реализованные требования, а часть из них. Тогда необходимо сформировать поставку, в которую потребные требования войдут точно, но еще и войдут требования, которые ему не потребны. Отношение непотребных требований к общему числу требований я называю качеством поставляемого функционала, от качества зависит эффективность использования IT продукта заказчиком.

Схема с отображением проблемы приведена на рисунке 2

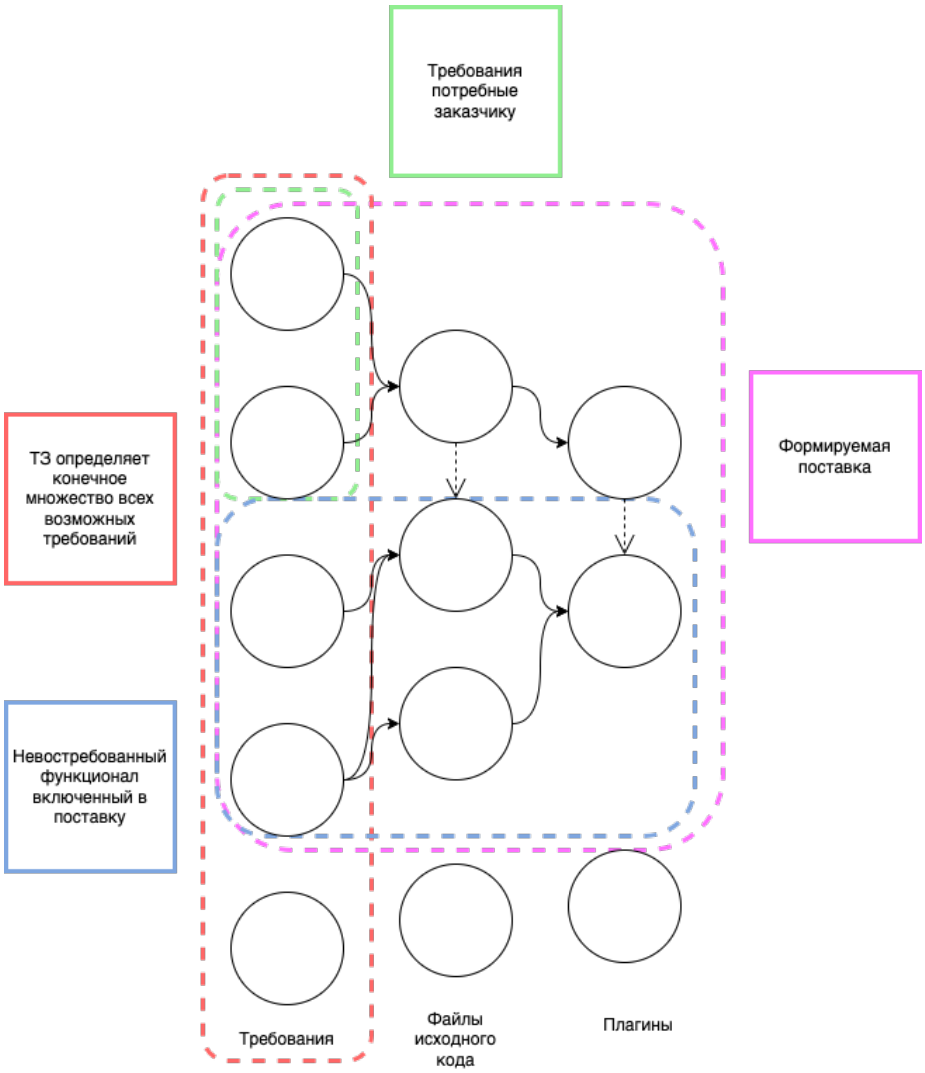

Рис. 2: Формирование поставки по потребным заказчику требованиям

3 Аспекты внешней среды

Среди аспектов внешней среды системы я выделил изначально сформированный объем требований к системе и требования заказчика, которые могут изменяться от поставки к поставке. От потребных заказчику в рамках поставки объема функционала зависит цена послепродажного обслуживания ПО.

Например, заказчик заказал 10 требований, а поставлен был функционал, реализующий 15. 5 требований он не оплатил, но осуществлять их поддержку все равно необходимо. В данном случае коэффициент простоя равен $5/15 = 1/3$. Почти 30% работы персонала по сопровождению заказчиком не оплачена.

С другой стороны существуют затраты на разработку и поддержку IT продукта. Чем больше в нем сущностей, тем дороже управление его конфигурацией, сложение отношения между сущностями и усложняется поиск и отладка неисправностей. Не говоря уже о когнитивном усложнении восприятии проекта и, как следствие, повышение требований к квалификации разработчиков, что в свою очередь тоже может быть переведено в денежный эквивалент.

Подытоживая, от этих денежных показателей зависит рентабельность IT продукта, а значит и конкурентоспособность бизнеса.

4 Выявление альтернатив системы

Так какие же варианты? Первое, что следует рассмотреть, как предоставлять заказчику только потребный функционал. Самый очевидный способ реализовать схему 1 требование - 1 плагин. Но этот вариант при достаточно большом объеме требований приведет к неконтролируемому росту сущностей в проекте ПО, следовательно, неконтролируемый рост затрат на его разработку и поддержку. Чтобы их компенсировать производитель будет вынужден повышать цену на свой IT продукт, что снижает конкурентоспособность бизнеса.

Другой альтернативой является схема все требования - 1 плагин. В этом случае коэффициент простоя будет слишком велик, если заказчик заинтересован лишь в части возможностей всего продукта. И службы пост продажного обслуживания будут перегружены, что так же приводит к стоимости поставляемого решения и снижения конкурентоспособности бизнеса.

В своей диссертации я решаю задачу оптимальной декомпозиции функционала по плагинам с целью снижения коэффициента простоя и контроля стоимости разработки и поддержки проекта.

Здесь тоже есть варианты. Первый - декомпозиция по частоте вызова функций. Если функция вызывается редко, она может быть обособлена, чтобы не тянуть свои зависимости в поставку. Часто вызываемые функции напротив, образуют ядро приложения.

Альтернативный - по функциональному признаку. Обосабливать функции по классу решаемых задач. Тогда, если заказчику потребены функции из определенного класса задач, то ему количество функций для решения задач других классов в поставке будет меньше.

5 Описание модели

Попытки описания механизмов, позволяющих реализовать разделение функционала уже существуют в наше время. Пример - спецификация OSGI, которая лежит в основе IDE Eclipse и называется Equenox.

Спецификация OSGI предполагает построение (синтез) системы из обособленных компонентов. Для разделения зависимостей на сторонние библиотеки и сервисы. Степень разделения и ее характер зависит от реализации. В IDE Eclipse она реализована для решения целей и задач стоящих перед самой IDE, однако предполагает ее расширение по правилам конкретного бизнеса.

Так, в своей работе мне необходимо выработать, сформировать распределение заданного множества требований реализованных в файлах исходного кода по плагинам для достижения конкурентоспособности бизнеса по ранее описанным критериям.

Для этого я разработал графовую модель, в которую вошли выявленные сущности предметной области с характерными им ограничениями. Как один из этапов, она решает задачу разрешения циклических зависимостей между файлами исходного кода для построения графа трассируемости требований к ПО на файлы исходного кода. Благодаря этому достигается сокращение сущностей на слое файлов исходного кода за счет слияние циклически зависимых вершин графа в одну вершину и, как следствие, упрощается задача распределения файлов по плагинам.

На языке программирования Java написан решение обозначенной задачи, получены результаты экспериментов для оценки времени работы приложения от типа используемых коллекций, в которых хранится информация о графе:

1. вершины - файлы исходного кода и требования к ПО;
2. ребра - зависимости между файлами исходного кода и трассируемость требований к ПО.

Работа алгоритма заключается в итеративном поиске циклов в графе с последующим слиянием входящем в цикл вершин до тех пор, пока в графе не будут отсутствовать циклы.

Схема работы разработанного решения приведена на рисунке 3.

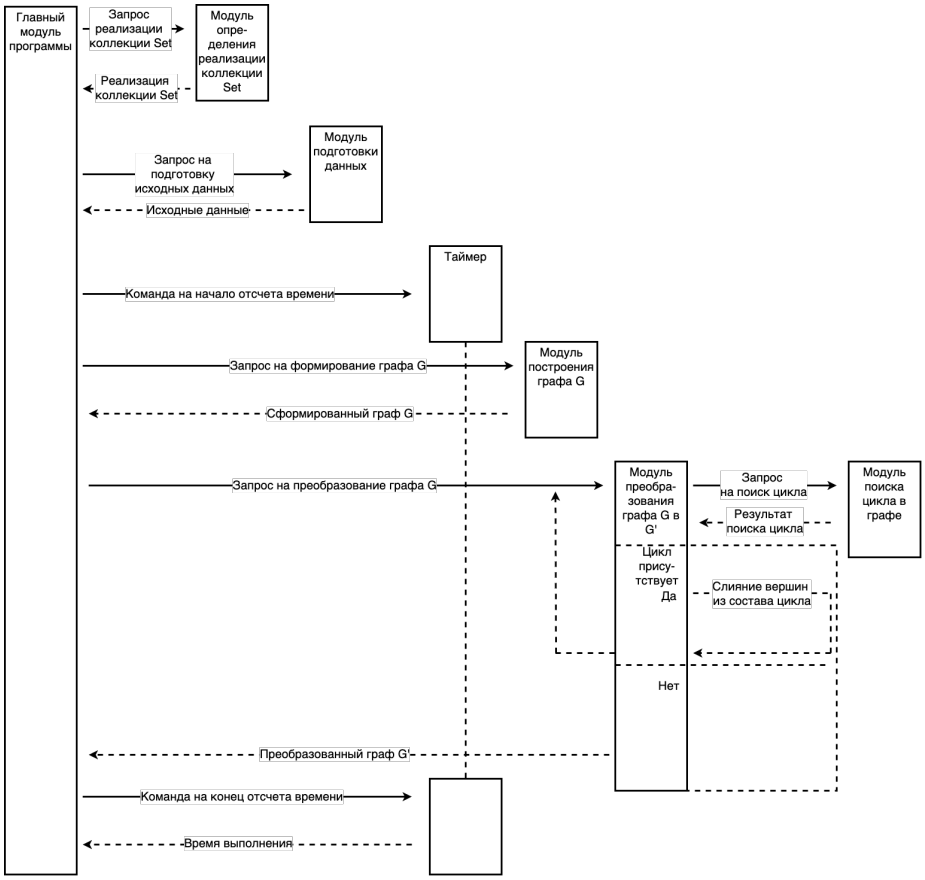

Рис. 3: Схема разработанного решения

В дальнейшем предполагается доработка модели для поиска оптимального распределения файлов исходного кода по плагинам при заданных ограничениях на систему, например обособление по частоте вызовов функций при числе плагинов не более пяти единиц.

Решение этой задачи предполагается выполнять с применением таких алгоритмов как:

1. генетический алгоритм;
2. PageRank;

3. решающий лес;

После решения этой задачи различными подходами будет произведена оценка полученных результатов и сделан вывод о том, при каких начальных условиях или их диапазонах лучше применять тот или иной алгоритм для решения задачи оптимальной декомпозиции.

Поиск оптимального решения предполагает введения ценовых коэффициентов или задание целевых функций вычисления стоимости сформированного алгоритмом оптимизации решения. Таким образом будет достигнута возможность проведения объективной оценки результата работы того или иного алгоритма.

6 Заключение

В заключении хочу сказать, что полученные мною результаты применяются для различных приложений и решений выполненных как набор плагинов для интеграции в соответствующие им плагинные системы.

Так, на конференции в МГТУ им. Баумана я в своем докладе «Исследование правил построения конфигуратора ARINC 653 спецификации в IDE Eclipse» показал результаты применения сформулированной модели для решения задачи по критерию частот вызова функций. В качестве результата продемонстрировал снижение невостребованного функционала до 10% при различных запросах состава требований.

На конференции в Воронеже в ВУНЦ ВВС «ВВА имени профессора Н.Е. Жуковского и Ю.А. Гагарина» в своем докладе «Реализация инструментального средства конфигурирования компонентов БРЭО» показал актуальность применения плагинных систем для решения задач в авиационной сфере.

Подготовил выступление на конференции, организуемой Союзом Машиностроителей России, в качестве материала выступления мною предоставлено средство интеллектуального конфигурирования системы определения состояния воздушного судна. В этой работе я сформировал среду конфигурирования, которая независимо подключается к базе данных и заполняет ее конфигурационными параметрами и их значениями с целью дельнейшей их обработки непосредственно в самой системе.

Подготовлена публикация ВАК, в которой описан алгоритм разрешения циклических зависимостей графовой модели трассируемо-

сти требований к ПО на файлы исходного кода. Работа передана в редакцию ИПМ им. М.В.Келдыша РАН.