Правительство Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
"Национальный исследовательский университет
"Высшая школа экономики"

Факультет компьютерных наук

Департамент программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению 09.03.04 «Программная инженерия»

подготовки бакалавра

Программа построения дерева классов бизнес-процессов

Студента группы № 402ПИ _____ Мамонтова Мария Евгеньевна
Подпись (Ф.И.О.)

_____
(Дата)

Научный руководитель

Преподаватель _____ Климов Борис Анатольевич
(должность, ученая степень) Подпись (Ф.И.О.)

Москва – 2015

Аннотация

Данный документ является выпускной квалификационной работой, которая посвящена разработке программы построения дерева классов бизнес-процессов. Дерево классов бизнес-процессов строится при помощи средств кластерного анализа. В данной работе под бизнес-процессами понимаются функции, которые реализуются в системе.

Программа может быть использована при анализе и построении архитектуры данных и приложений в процессе анализа и разработки ИТ-архитектуры предприятия. Полученное при помощи программы распределение функций по классам позволяет выделить их изолированные группы, трактовать эти группы как системы и предлагать их древовидное отображение, что значительно упрощает структурирование информационных систем.

Актуальность работы подтверждается отсутствием аналогов, которые бы автоматизировали такой анализ данных и предлагали решения по автоматизированному построению ИТ-архитектуры предприятия в части функциональной структуры информационных систем.

Ключевые слова: архитектура предприятия, кластерный анализ, сложность системы, анализ функций, архитектура данных, ИТ-архитектура.

Дипломная работа: 46 с., 13 рис., 3 табл., 17 источников литературы.

Abstract

The current document is a graduation paper, which is dedicated to development of the program for building a tree of classes of business processes. A tree of classes of business processes is built by means of cluster analysis. In the current paper business processes are considered as functions, which should be implemented in the system.

The program could be used for analysis and construction of data and applications architecture during enterprise it-architecture analysis and development. Received by the program distribution of functions into classes helps to detect isolated groups of functions, which are considered as systems, and to show their tree representation, which considerably simplifies information systems structuring.

The topicality of the work is caused by the absence of analogues, which automate the proposed data analysis and suggest solutions for automated Enterprise IT-architecture construction in the context of information systems functional structure.

Key words: Enterprise Architecture, cluster analysis, system's complexity, functions' analysis, data architecture, IT-architecture.

Graduation work: 46 pages, 13 pictures, 3 tables, 17 sources.

Благодарность

Выражаю благодарность Смирных Евгению Алексеевичу (старший архитектор, отдел финансовых решений, управление архитектуры, ООО «РН-Информ») за постановку задачи, обсуждение методов расчета и полученных результатов.

Определения, обозначения и сокращения

EA - Enterprise Architecture

TOGAF- The Open Group Architectural Framework

FEA -Federal enterprise architecture

ADM - Architecture Development method

RUP - Rational Unified Process

SOA - Service-Oriented Architecture

SIP- Simple Iterative Partitions

БП- бизнес-процесс

ИТ-Информационные технологии

Оглавление

Введение

Задача построения дерева классов бизнес-процессов в данной работе рассматривается как часть процесса построения архитектуры предприятия. Понятий термина “архитектура предприятия” существует очень много, в самом общем виде под архитектурой предприятия (EA – Enterprise Architecture) понимается всестороннее и исчерпывающее описание (модель) всех его ключевых элементов и межэлементных отношений. [13] Бизнес-процессы в данной работе представлены функциями, которые должны быть реализованы в системе. Полученные результаты несут рекомендательный характер и могут быть использованы архитекторами для принятия решений по построению архитектуры предприятия.

Архитектура предприятия объединяет ИТ-архитектуру и бизнес-архитектуру и способствует движению предприятия к ее стратегической цели. Построение эффективной архитектуры предприятия является залогом успеха ведения бизнеса, позволяет эффективно использовать информационные технологии, повышающие приспособляемость бизнеса, организовывать сотрудничество бизнес и ИТ-подразделений, концентрировать работу организации на ее целях, сделав весь рабочий процесс более прозрачным для всех работников, а также повышать устойчивость, простоту и гибкость ИТ-системы. Таким образом, разработка архитектуры предприятия позволяет решить проблему синхронизации возможностей и потребностей бизнеса и ИТ. [10]

Чаще всего для построения архитектуры предприятия используется ряд существующих методологий, например методология TOGAF (The Open Group Architectural Framework). Методологии могут включать в себя классификацию составляющих предприятия для четкого представления его структуры, шаблонные базовые модели, на основе которых происходит построение каждой конкретной архитектуры, общий план действий по построению архитектуры или набор экспертных рекомендаций, основанный на исследованиях в данной области и личном опыте архитекторов. Каждый подход предполагает анализ имеющихся структур предприятия, данных и функций. Проблема заключается в том, что известные на сегодняшний день решения (например Alfabet компании Software AG [1] предлагают автоматизированные средства лишь по анализу уже существующей архитектуры, ее оптимизации, мониторингу и визуализированному представлению, но не по построению. Более того, большинство методологий не учитывают сложность структуры предприятия и никак не способствуют ее снижению, что значительно уменьшает эффективность архитектуры предприятия.

Задача автоматизированного анализа функций предприятия для построения архитектуры предприятия является очень актуальной, так как на данный момент не существует решений, автоматизирующих анализ функций предприятия и группирующих их в подсистемы с целью снижения сложности структуры. Задачей программы является упростить структуру, уменьшив взаимодействие между подсистемами. Минимизация взаимодействия подсистем значительно упрощает структуру предприятия, что снижает влияние отрицательных рисков и упрощает процесс построения архитектуры. Схематичное изображение структуры системы до структуризации и после отображено на рисунке 1 (*Рисунок 1*) и (*Рисунок 2*). После структуризации, как изображено на рисунке 2, функции должны быть перегруппированы в подсистемы таким образом, что взаимодействие между подсистемами будет минимальным.

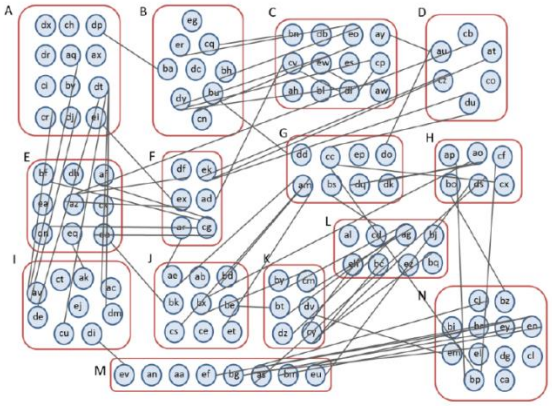


Рисунок 1. Исходная архитектура.

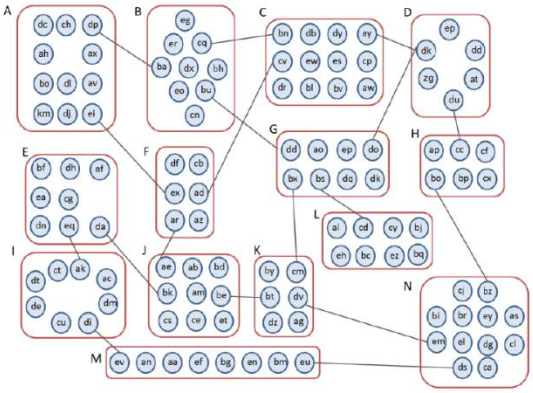


Рисунок 2. Архитектура после структуризации.

Более того, сама по себе задача построения архитектуры предприятия крайне актуальна, так как большинство организаций либо не имеют формализованной определенной архитектуры, либо эти определения неполны и недостаточно четко связаны с требованиями бизнеса, а необходимость в структурированном описании и проектировании архитектуры предприятия постоянно растет.

1. Анализ предметной области

1.1. Процесс построения архитектуры предприятия.

Архитектура предприятия как дисциплина сформировалась около 20 лет назад и на данный момент является основным средством достижения и поддержания конкурентоспособности любого предприятия или организации, особенно в сфере ИТ.

Чаще всего в понятие архитектуры предприятия включают следующие составляющие:

1. Корпоративная миссия и стратегия
2. Бизнес-архитектура
3. Системная архитектура (ИТ – архитектура)

Схематичное представление основных элементов архитектуры представлено в *Таблица 1*.

Таблица 1.Архитектура предприятия.

Корпоративная миссия и стратегия		
Бизнес-Архитектура		
Бизнес-процессы	Организационная –штатная структура	Система документа оборота
Системная архитектура		
Приложение	Данные	Оборудование

Корпоративная миссия и стратегия определяют основное направление движения компании, бизнес-архитектура включает определение бизнес-процессов и организационной структуры. На уровне системной архитектуры производится создание и описание архитектуры приложений, данных и технической архитектуры.

В данной работе рассматривается уровень системной архитектуры, включающей архитектуру приложений, данных и инфраструктуры (оборудования). Архитектура приложений, в свою очередь, включает:

- Прикладные системы
- Интерфейсы взаимодействия прикладных систем между собой и с внешними системами и источниками или потребителями данных
- Средства и методы разработки и сопровождения приложений

Архитектура данных включает:

- БД и хранилища данных
- Системы управления БД или хранилищами данных
- Правила и средства санкционирования доступа к данным

Техническая архитектура состоит из сетевой архитектуры и архитектуры платформ. [13]

В данной работе рассматривается аспект создания системной архитектуры предприятия, в частности проектирование структуры данных и приложений. Программа построения дерева классов бизнес-процессов позволяет проанализировать то, как функции совместно используют данные и на основе этой информации распределить функции по классам так, чтобы взаимодействие между классами было минимальным. При уменьшении взаимодействия между подсистемами значительно упрощается структура предприятия, и задача построения эффективной архитектуры предприятия становится гораздо более достижимой.

1.2. Методологии создания архитектуры предприятия.

Помимо общих положений в отношении построения архитектуры существует множество методологий, которое широко используются для построения архитектуры предприятия. Чаще всего используется одна из четырех методологий построения архитектуры предприятия: структура Захмана для архитектуры предприятий, TOGAF (The Open Group Architectural Framework), Архитектура федеральной организации (FEA) или Методология Gartner. Программа построения дерева классов бизнес-процессов может быть использована при построении архитектуры по любой из методологий, поэтому стоит рассмотреть методологии подробней, чтобы понять, как можно использовать программу.

Структура Захмана

Основоположником концепции архитектуры предприятия считается Джон Захман. Статья «Структура архитектуры информационных систем», опубликованная в журнале *IBM Systems Journal в* 1987 г, стала основой для создания многих других методологий описания архитектуры предприятия. Изначальная версия модели, предложенной Захманом, в последствии была доработана и использовалась многими корпорациями. Хотя изначально

модель Захмана использовалась для описания ИТ-системы, модель может быть использована для описания всей архитектуры предприятия в целом.

Модель Захмана представляет собой таблицу, в которой описательные аспекты(столбцы)- Данные(Что?), Фукнции(Как?), Местоположение (Где?), Люди(Кто?), Время(Когда?) и Мотивация(Почему?), рассматриваются со стороной разных действующих лиц (строки) -Бизнес–руководителей (Планировщик, Владелец/менеджер) и ИТ-менеджеров/ Разработчиков (Конструктор/ Архитектор, Проектировщик, Разработчик). Строки соответствуют разным уровням управления предприятием и являются представлениями аспектов со стороны определенных действующих лиц. В таблице размещаются так называемые артефакты- конкретные документы, отчеты, модели или любые другие компоненты архитектурного описания. Каждый артефакт должен располагаться в одной ячейке, если артефакт невозможно отнести ни к одной ячейке, то необходимо пересмотреть сам артефакт. Ячейка считается заполненной, если находящиеся в ней артефакты полностью определяют систему для конкретного игрока в конкретном описательном аспекте. Структура Захмана изображена на рисунке 3 (*Рисунок 3*).

		Данные ЧТО	**Функции КАК**	**Дислокация, сеть ГДЕ**	**Люди КТО**	**Время КОГДА**	**Мотивация ПОЧЕМУ**	
Бизнес-руководители	*Планировщик*	Список важных понятий и объектов	Список основных бизнес-процессов	Территориальное расположение	Ключевые организации	Важнейшие события	Бизнес-цели и стратегии	Сфера действия (контекст)
	Владелец, менеджер	Концептуальная модель данных	Модель бизнес-процессов	Схема логистики	Модель потока работ (workflow)	Мастер-план реализации	Бизнес-план	Модель предприятия
ИТ-менеджеры и разработчики	*Конструктор, архитектор*	Логические модели данных	Архитектура приложений	Модель распределенной архитектуры	Архитектура интерфейса пользователя	Структура процессов	Роли и модели бизнес-правил	Модель системы
	Проектировщик	Физическая модель данных	Системный проект	Технологич. архитектура	Архитектура презентации	Структуры управления	Описания бизнес-правил	Технологическая (физическая) модель
	Разработчик	Описание структуры данных	Программный код	Сетевая архитектура	Архитектура безопасности	Определение временных привязок	Реализация бизнес-логики	Детали реализации
		Данные	Работающие программы	Сеть	Реальные люди, организации	Бизнес-события	Работающие бизнес-стратегии	Работающее предприятие
		Данные	Функции, Процессы	Сеть, расположение систем	Люди, организации	Время, расписания	Мотивация	

Рисунок 3.Структура Захмана.

В данной модели интересующие нас функции будут рассматриваться в описательном аспекте Функции(Как?), а данные в первом столбце Данные(Что?). Модель позволяет отразить общую взаимосвязь элементов и проследить их на разных уровнях представления и использования с разной степенью детализации.

TOGAF

Следующей широко применяемой методологией является методология TOGAF, предложенная некоммерческим объединением The Open Group.The Open Group определяет архитектуру предприятия следующим образом: «Архитектура предприятия - это способ понимания различных элементов, которые в совокупности составляют предприятие, и то, как эти элементы взаимосвязаны» [10]. Если модель Захмана классифицирует артефакты, то методология TOGAF является процессом выявления артефактов. В соответствии с моделью TOGAF архитектуру предприятия можно разделить на четыре категории:

1. Архитектура бизнеса - описывает процессы, которые используются для достижения целей бизнеса.
2. Архитектура приложений - структура приложений и их взаимодействие друг с другом.
3. Архитектура данных — описывает структуру корпоративных хранилищ данных и процедуры доступа к ним.
4. Технологическая архитектура — описывает инфраструктуру оборудования и программного обеспечения, в которой запускаются и взаимодействуют приложения.

Задача анализа функций по тому, как часто они совместно используют данные, аналогично общему подходу к построению архитектуры предприятия по методологии TOGAF решается на уровне архитектуры приложений и архитектуры данных.

Также, в начале построения архитектуры предприятия в соответствии с моделью TOGAF за основу архитектуры берется базовая модель, которая далее конкретизируется. Создание конкретной архитектуры происходит путем перехода от общей архитектуры к специфицированной. TOGAF предлагает методику разработки архитектуры (Architecture Development Method -ADM) для осуществления этого перехода. Фазы методики разработки архитектуры представлены на рисунке 4 (*Рисунок 4*):

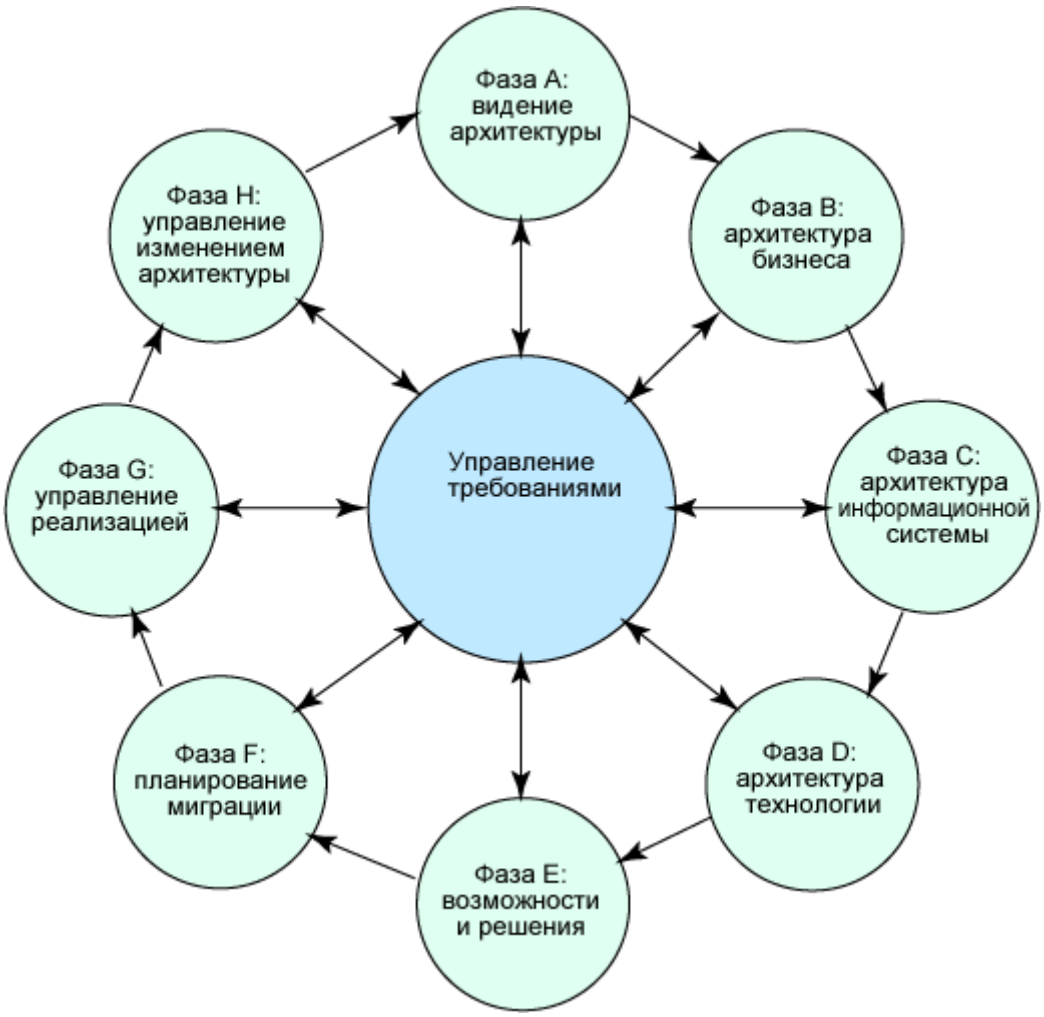


Рисунок 4.Метод разработки архитектуры (ADM) TOGAF.

Программа построения дерева классов бизнес-процессов может быть использована на Фазе C для автоматизированного анализа функций систем с целью разработки архитектуры данных.

Архитектура федеральной организации(FEA)

Следующая методология Архитектура федеральной организации(FEA) является как способом классификации артефактов, так и процессом создания архитектуры предприятия. В методологию FEA включено пять эталонных моделей: модель бизнеса, модель обслуживания, модель компонентов, технологическая модель и модель данных. Методология FEA также предоставляет инструменты для построения архитектуры предприятия.

В соответствии с методологией FEA, архитектура предприятия включает в себя базовые и служебные сегментов, которые являются некими функциональными сферами деятельности компании. Например, сегментом может считаться управление безопасностью

или управление финансами. Сегменты архитектуры являются основными аспектами бизнеса. Целью процесса FEA является создание архитектуры сегмента.

Процесс разработки архитектуры сегмента можно разделить на следующие этапы:

- Этап 1. Анализ архитектуры: формирование простого представления сегмента с привязкой к плану организации.
- Этап 2. Архитектурное определение: определение необходимого состояния сегмента, разработка архитектуры предприятия для сегмента.
- Этап 3. Стратегия инвестиций и финансирования: анализ способов финансирования проекта.
- Этап 4. План управления программой и реализация проектов: создание плана управления проектом и его реализации. [15]

Методология Gartner

Последняя методология - методология Gartner. Gartner-известная исследовательская и консалтинговая фирма. Их методология включает в себя практические рекомендации по построению архитектуры предприятия.

Модель Gartner 2002 года сформулирована в виде четырех взаимозависимых и усложняющихся уровней:

- Среда бизнес-взаимодействия (Business Relationship Grid)-уровень соответствует построению архитектуры взаимодействия предприятий. Средой взаимодействия в данном случае выступает интернет
- Бизнес-процессы и стили бизнес-процессов - описание того, как выполняются основные функции
- Шаблоны - модели и алгоритмы, такие как архитектура программных и вычислительных систем. Рекомендуется использование шаблонов Сервис-ориентированная архитектураы
- Технологические строительные блоки -технологическая архитектура- включает в себя операционные системы, серверы, базы данных, сами данные

Gartner предлагает лишь общее описание архитектуры предприятия без конкретных указаний и является примером реализации методологии высокого уровня. Для разработки архитектуры в Gartner сформулированы рекомендации в виде последовательности шагов и задач участников, однако эти рекомендации также представлены в довольно общем виде. Эксперту, который будет создавать архитектуру предприятия по подходу Gartner,

автоматизированный анализ функций систем предприятия поможет на уровне создания технологических строительных блоков и при выборе моделей и алгоритмов. [10]

Таким образом, для использования методологий построения архитектуры необходимо активизировать сложные долгосрочные проекты, которые требуют значительных ресурсных вложений. Методология архитектуры предприятия- это теоретическое описание подходов. Для упрощения задачи необходимо использовать специальные инструменты, которые на разных этапах построения помогут оптимизировать процесс.

1.3. Rational Unified Process

Необходимо понимать, что большинство инструментов, созданных для реализации, управления, оптимизации, мониторинга или иллюстрации каких-либо этапов ИТ- проекта могут быть применены в процессе построения архитектуры предприятия. В связи с этим стоит упомянуть Rational Unified Process – широко известную методологию разработки программного обеспечения.

RUP является руководством по применению эффективных подходов к разработке ПО и использованию передового опыта в таких областях, как:

- Итерационная разработке ПО
- Управление требованиями
- Использование компонентной архитектуры
- Визуальное моделировании
- Тестирование качества ПО
- Контроль за изменениями в ПО

Чаще всего RUP не связан с архитектурой предприятия, но такое взаимодействие поможет достичь положительных результатов. [3]

Существуют решения по расширению сферы деятельности RUP в направлении архитектуры предприятия. Так, был создан процесс Enterprise Unified Process (EUP), который предоставляет способы выполнения различных деятельностей предприятия.

Помимо этого новые версии RUP имеют некоторые процессы предприятия, такие как бизнес-моделирование и управление изменениями.

RUP может быть использована при построении архитектуры предприятия. Например, если рассматривать методологию TOGAF, RUP может быть применено на фазе G- управление реализацией, на которой устанавливается связь между архитектурой и разработкой. Пересечение процесса построения архитектуры и RUP происходит тогда, когда архитектура бизнеса и высокоуровневые планы реализации передаются коллективам, воплощающим реализацию. На рисунке 5 (*Рисунок 5*) представлено взаимодействие между TOGAF и RUP.

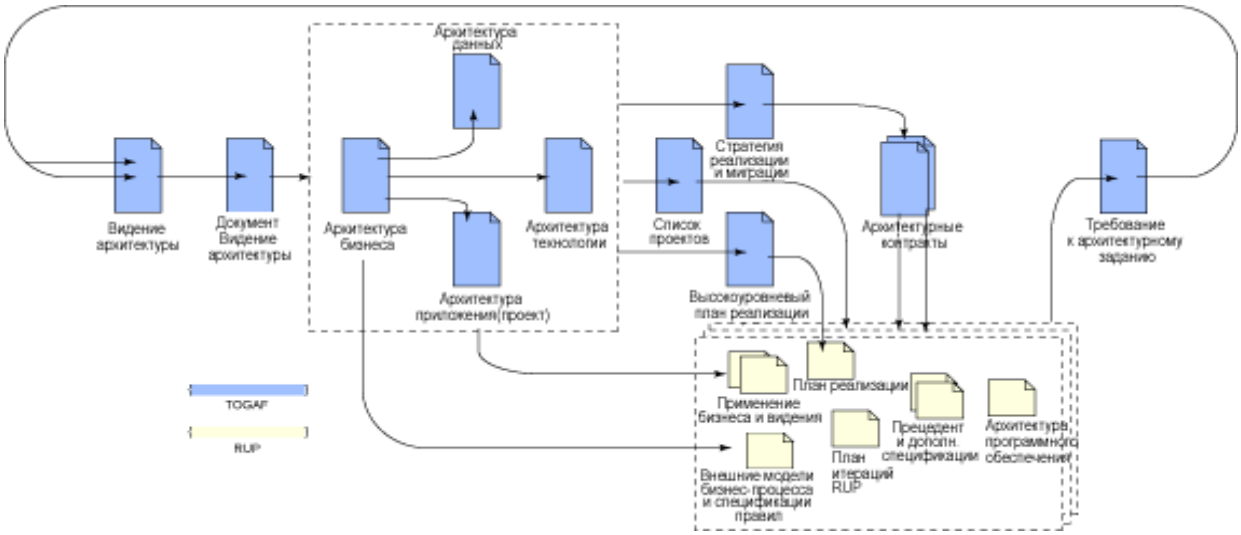


Рисунок 5.Артефакты переходят между TOGAF и RUP.

Таким образом, стоит подчеркнуть, что построение архитектуры предприятия значительно упрощается при помощи дополнительных инструментов. Программа построения дерева классов задумывается как часть автоматизированного процесса построения архитектуры предприятия, при этом она также может автономно использоваться для автоматизации части процесса.

1.4. Сервис-ориентированная архитектура

Наряду с архитектурой предприятия часто говорят о сервис-ориентированной архитектуре (Service-Oriented Architecture SOA).

Из множества определений сервис-ориентированной архитектуры сложно выделить единственное определение, поэтому приведем одно из них:

Определение IBM:

«Сервис-ориентированная архитектура - это ИТ-архитектура уровня предприятия, предназначенная для установления связи с ресурсами по требованию. Эти ресурсы представлены в виде ориентированных на задачи бизнеса сервисов, которые могут быть

включены в пул ресурсов предприятия или направления бизнеса и модифицированы для удовлетворения соответствующих бизнес-потребностей. Первичным структурным элементом приложений SOA является сервис, а не подсистема, система или компонент». [12]

На самом деле, сервис-ориентированная архитектура не является архитектурой или архитектурной дисциплиной. Сервис–ориентированная архитектура –это архитектурный стиль, который в том числе может применяться в процессе построения архитектуры предприятия. Совместная реализация архитектуры предприятия и сервис-ориентированной архитектуры может повлечь определенные сложности, ведь сферы деятельности и влияния двух подходов пересекаются. Для облегчения этой задачи команды обоих проектов должны работать в тесной связи, управление проектами и их взаимодействием должно быть четко организовано. В то же время, при возникновении конфликтующих вопросов важнее руководствоваться интересами уровня архитектуры предприятия. Хотя проблемы и есть, многие источники рекомендуют совместное использование подходов.

Итак, SOA позволяет создавать модульные и слабосвязанные сервисы, которые можно составлять и организовывать в бизнес-процессы. Для построения SOA-системы необходимо четко определить бизнес-процессы, которые составят эту систему, а также определить набор сервисов для моделирования и построения, соответствующий определенным критериям. Для уровня бизнеса SOA представляет собой набор бизнес-процессов, которые можно предложить клиентам и партнерам.

Для отбора нужных сервисов требуется изучить некоторые их характеристики:

1. Возможность настройки сервиса под потребности бизнеса
2. Возможность его компоновки с другими сервисами
3. Его потенциал для повторного использования
4. Его техническая реализуемость

Для каждой фазы жизненного цикла сервиса существуют различные инструменты проектирования и разработки.

Модель жизненного цикла сервиса, предложенная IBM Foundation, изображена на рисунке 6 (*Рисунок 6*).

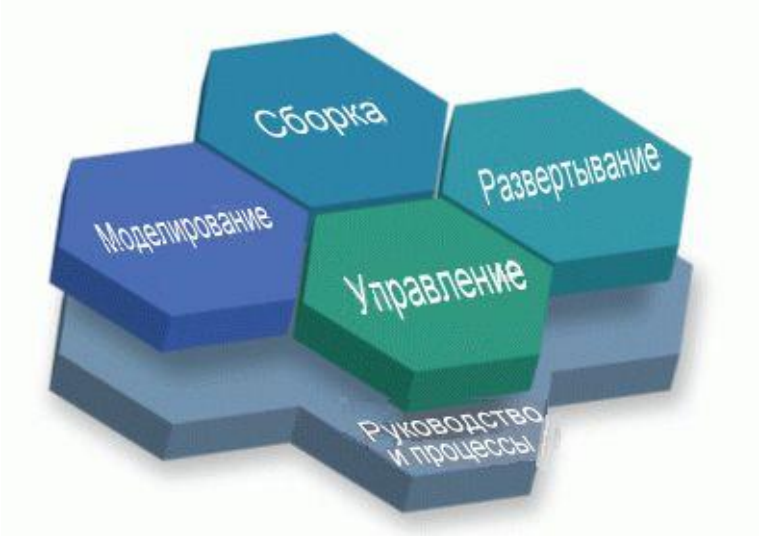


Рисунок 6.Жизненный цикл IBM SOA Foundation.

Таким образом, наличие сервис-ориентированной архитектуры не исключает реализацию архитектуры предприятия и наоборот. Совместное использование обоих подходов может принести выгоду, которая не достижима при использовании только одного из них. [4][16]

1.5. Существующие решения снижения сложности структуры предприятия

Одна из целей, которую необходимо достичь в программе построения дерева классов бизнес-процессов заключается в том, чтобы упростить структуру предприятия. Необходимость и способы понижения сложности ИТ-систем описывает в своих работах Роджер Сешнс, автор множества статей, посвященных архитектуре предприятия, сложности IT систем и создатель методологии SIP (Simple Iterative Partitions).

Каждый ИТ-проект обладает определенным набором рисков:

- Превышение бюджета
- Сроки
- Недостающий функционал
- Низкая безопасность
- Отсутствие гибкости

Одним из ключевых факторов появления этих рисков является сложность системы. Чем сложнее система, тем больше вероятность возникновения этих рисков. Само понятие “сложность системы” относится не только к сложности функционала, но и к сложности структуры системы. Важными факторами является то, как система структурирована в

подсистемы, и как эти подсистемы взаимодействуют. Таким образом, чем меньше сложность системы, тем выше вероятность успеха предприятия. [5]

Проблема, как утверждает Сешнс в своей статье “Контроль сложности архитектуры предприятия” («Controlling Complexity in Enterprise Architectures») [7], заключается в том, что существующие методологии создания архитектуры предприятия не обладают достаточным количеством средств для учета сложности системы и ее понижения, что значительно снижает эффективность архитектуры.

В соответствии с методологией SIP (simple iterative partitions) предлагается производить деление («partitioning») бизнес-процессов на уровне бизнес-модели. Для такого деления используется математический аппарат, введенный в методологии SIP, который определяет те бизнес-процессы, которые являются взаимодействующими («synergetic») и группирует их. Как описывается в статье, функции являются взаимодействующими, если для каждой функции необходимо, чтобы другая давала наиболее эффективный результат. [7]

Роджер Сешнс предлагает совместно использовать разработанную им методологию проектирования архитектуры предприятия Snowman Architecture (или Vertically Aligned Synergistically Partitioned) и методологию TOGAF.

Основные четыре компоненты архитектуры Snowman представлены на рисунке 7 (*Рисунок 7*).

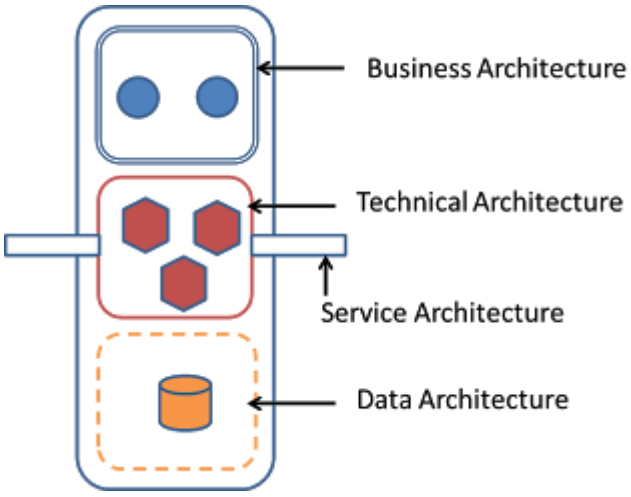


Рисунок 7.Базовая архитектура Snowman.

Архитектура Snowman Architecture похожа на сервис-ориентированную Архитектуру (SOA), но имеет существенные улучшения. Архитектура Snowman использует методологию SIP для выделения взаимодействующих модулей при помощи группирования

бизнес-процессов. Для каждого такого модуля и строится бизнес-архитектура, технологическая архитектура и архитектура данных. Графическое отображение архитектуры SOA представлено на рисунке 8 (ниже*Рисунок 8*).

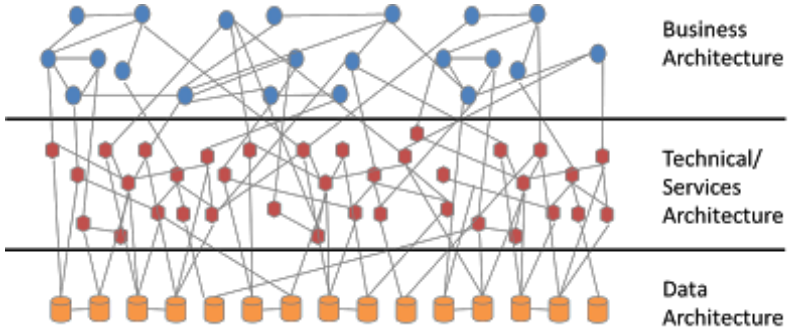


Рисунок 8.Типичная Сервис-Ориентированная Архитектура.

Несмотря на всю выгоду инновационного подхода, предложенного Роджером Сешнсом, не стоит отказываться от традиционных концепций построения архитектуры, так как они годами разрабатывались ведущими экспертами в области архитектуры предприятия и имеют свои преимущества. Предлагается использовать комбинированный подход, заимствуя у традиционных методологий подходы, которые проработаны более качественно.

В отличие от методологии SIP, совершающей разделение на модули на уровне бизнес-архитектуры, в данной работе мы будем совершать разделение системы на модули на уровне архитектуры данных. Выделение групп бизнес-процессов сама по себе задача непростая, ведь для этого необходимо проанализировать все бизнес-процессы, проанализировать связи между бизнес-процессами, определить отношения между ними и уже тогда разбивать на группы. Определение отношения между бизнес-процессами также является сложной трудоемкой задачей, особенно учитывая сложность системы и количество бизнес-процессов.

В данной же работе деление на модули производиться не при помощи аналитического анализа функционала бизнес-процессов, а на основе информации об использовании функциями предприятия данных.

Таким образом, рекомендацией по выбору способа построения архитектуры предприятия является совместное использование одновременно различных технологий. На данный момент, построение архитектуры предприятия строится в соответствии с мировыми методологиями, которые дают рекомендации по построению архитектуры, предлагают применять эталонные модели или советуют последовательно разобрать предприятие на

составляющие, классифицировать эти составляющие и скомпоновать в соответствии с методологией. Распространенной практикой является нанимать отдельного специалиста по построению архитектуры (или иметь его в штате, так как многие авторы отмечают, что архитектура предприятия - это процесс, а не единовременно проведенная работа), однако проблема заключается в том, что нет уникального способа построения архитектуры. Два разных специалиста, применив одну и ту же методологию, могут получить разные результаты, а какой результат будет лучше работать, можно понять только на практике. Более того, большинство методологий не учитывают сложность системы и не пытаются снизить ее сложность.

Автоматизация анализа функций является лишь частью создания автоматизированного процесса построения архитектуры предприятия. Однако эта часть поможет однозначно определить связь между функциями и помочь архитектуру скомпоновать данные и процессы в соответствующие слабосвязанные блоки, что значительно упрощает структуру предприятия. [10][15]

2. Кластерный анализ функций

Таким образом, мы обращается к конкретной проблеме понижения сложности структуры предприятия путем анализа функций систем и разбиения множества данных функций на слабовзаимодействующие системы на основе информации о совместном использовании этими функциями данных.

Необходимо сгруппировать функции так, чтобы взаимодействие между группами было минимальным. Для данной задачи может быть использован инструмент кластерного анализа. Рассмотрим алгоритмы кластеризации, наиболее подходящие для решения поставленной задачи.

2.1. Описание алгоритмов

Термин *кластерный анализ*, введенный Tryon в 1939 году, включает в себя набор различных алгоритмов классификации. Кластеризация решает вопрос о том, как *организовать* наблюдаемые данные в наглядные структуры. Иначе говоря, как распределить наблюдаемые объекты так, чтобы объекты из одной группы были более похожи друга на друга, чем на объекты из других групп. [8]

Чаще всего методы кластерного анализа делят на две группы:

1. Иерархические
2. Неиерархические.

Для того, чтобы определить, какой подход лучше использовать для решения задачи группирования функций, подробно рассмотрим каждый из них.

Иерархические

Рассмотрим сначала иерархические методы кластеризации. В иерархических алгоритмах меньшие кластеры объединяются в большие или наоборот, кластеры с большим количеством элементов разделяются на кластеры с меньшим. Иерархические методы предоставляют наглядную картину разбиения данных. Помимо этого, нет необходимости знать предполагаемое число кластеров до начала анализа, что подходит для решаемой задачи, так как заранее неизвестно, на какое число групп нужно поделить исходное множество функций.

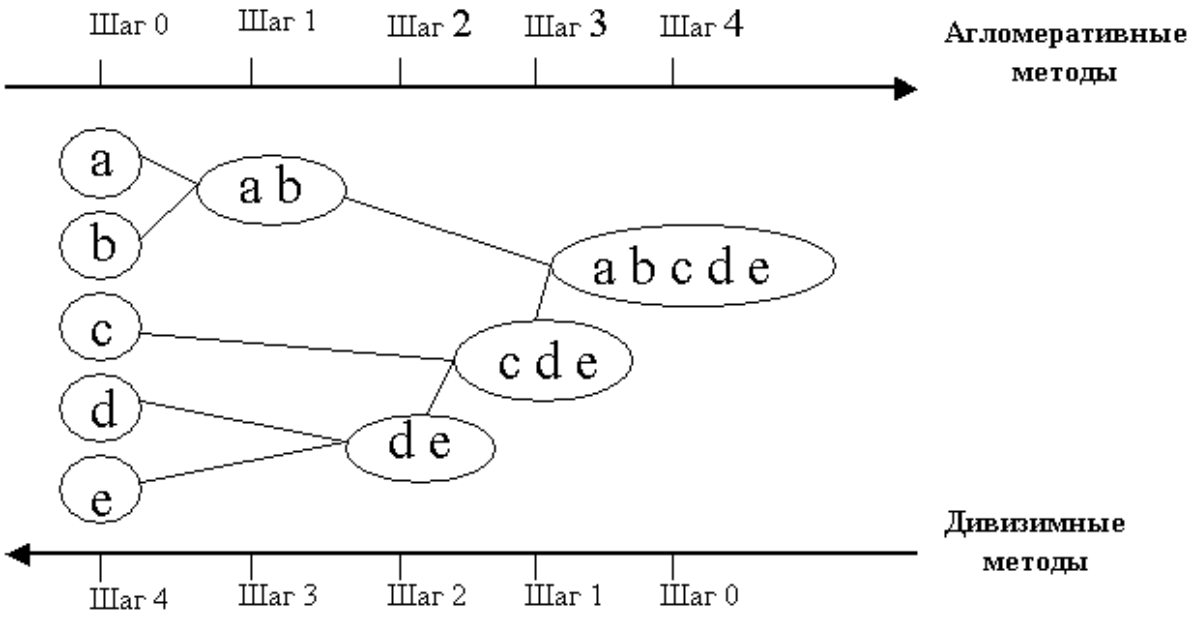


Рисунок 9. Агломеративные и дивизимные методы.

Иерархические методы кластеризации в свою очередь подразделяются на агломеративные и дивизимные. Схематическое изображение агломеративного и дивизимного подхода изображено на рисунке 9 (*Рисунок 9*).

В соответствии с агломеративным алгоритмом, в начале процесса кластеризации все элементы исходного множества представляют собой различные кластеры. Затем кластеры последовательно объединяются в зависимости от того, какой используется способ расчета расстояния. Расстояние показывает на сколько объекты различны, а мера близости – насколько объекты похожи.

Существуют различные способы расчета расстояния, выбор способа должен исходить из требований каждой конкретной задачи.

Приведем наиболее используемые способы,

Где

$$A_i - \text{значение } i - \text{того свойства объекта } A$$

$$B_i - \text{значение } i - \text{того свойства объекта } B$$

Метрики для количественных шкал:

1) *Евклидово расстояние*

   Наиболее распространенный способ вычисления расстояния. Данный способ удобен тем, что на расстояние между двумя величинами не влияют другие объекты.

$$\text{Расстояние (A, B)} = \sum_i \sqrt{(A_i - B_i)^2}$$

2) *Квадрат евклидова расстояния*

   Может использоваться, если необходимо сильнее подчеркнуть различие далеко находящихся друг от друга объектов.

$$\text{Расстояние (A, B)} = \sum_i (A_i - B_i)^2$$

3) *Расстояние городских кварталов (манхэттенское расстояние)*

   Это расстояние рассчитывается как среднее разностей по координатам. В большинстве случаев эта мера расстояния приводит к результатам, подобным расчетам расстояния евклида. Однако, для этой меры влияние отдельных выбросов меньше, чем при использовании евклидова расстояния, поскольку здесь координаты не возводятся в квадрат.

$$\text{Расстояние (A, B)} = \sum_i |A_i| + |B_i|$$

4) *Расстояние Чебышева*

   Принимает значение наибольшего модуля разности между значениями соответствующих свойств (признаков) объектов. Это расстояние может оказаться полезным, когда необходимо определить два объекта как "различные", если они различаются каким-либо одним измерением. Расстояние Чебышева вычисляется по формуле:

$$\text{Расстояние (A ,B)} = max(|A_i| + |B_i|)$$

[17]

Метрики для качественных шкал (номинальных и порядковых):

1) Коэффициент Рао: отношение числа совпадающих единичных свойств к числу всех значений m.

$$R(i,j)=n_{i,j}/m$$

Где

$n_{i,j}$ – число совпадающих единичных свойст, m – число всех значений

2) Коэффициент Роджерса-Танимото:

$$R(i,j)=n_{i,j}/(n_i+n_j-n_{i,j})$$

Где

$n_{i,j}$ – число совпадающих единичных свойст, n_i – число единичных значений свойст

Коэффициент Рао и коэффициент Роджерса-Танимото учитывают только совпадающие единичные значения, а совпадающие нулевые игнорируются.

3) Расстояние Хемминга – отношение количества совпадающих значений к числу всех значений N.

[14]

Таким образом, при помощи расстояния или меры близости можно сравнить объекты. После первого этапа группировки, когда объекты сгруппированы в кластеры, содержащие больше одного объекта, необходимо использовать специальный критерий, введенный для определения расстояния между кластерами.

Существуют различные способы расчета расстояния между кластерами, из наиболее известных можно выделить:

1) *Метод ближнего соседа или одиночная связь.*

   За расстояние между кластерами берется расстояние между двумя ближайшими объектами. Этот метод позволяет выделять кластеры сколь угодно сложной формы при условии, что различные части таких кластеров соединены цепочками близких друг к другу элементов.

2) *Метод наиболее удаленных соседей или полная связь.*

   За расстояние между кластерами берется расстояние между двумя наиболее отдаленными объектами. Подходит, если объекты сильно отличаются друг от друга.

3) *Метод невзвешенного попарного среднего.*

   За расстояние между кластерами берется среднее расстояние между всеми парами объектов в них.

4) *Метод взвешенного попарного среднего (метод взвешенного попарного арифметического среднего.*

   Так же, как и метод невзвешенного попарного среднего используется расстояние между всеми парами объектов, однако еще учитывается весовой коэффициент. В качестве весового коэффициента выступает размер кластера. Метод стоит использовать в тех случаях, когда делается предположение о том, что размер кластеров будет различный.

5) *Невзвешенный центроидный метод*

   В качестве расстояния между кластерами берется расстояние между их центрами тяжести.

6) *Взвешенный центроидный метод*

   Аналогичен невзвешенному методу, однако для учета различия в размерах кластера предлагается использовать веса.[17]

В дивизимных алгоритмах в начале процесса группирования все объекты исходного множества включаются в один исходный кластер. Затем этот кластер делится на два других

кластера. Существуют различные способы распределения объектов по новым двум кластерам.

Один из возможных вариантов-

- 1 шаг: Выбрать два наиболее далеких элемента, которые и будут первыми исходными элементами новых кластеров
- 2 шаг: Распределять объекты делимого кластера в тот кластер, для которого расстояние между рассматриваемым объектом и исходным объектом кластера будет максимальным.

Неиерархические методы кластеризации

Методы неиерархической кластеризации направлены на то, чтобы разделить исходный набор данных на непересекающиеся группы, при этом количество групп (кластеров) заранее известно. Данная группа методов подходит для кластеризации большого количества данных, в отличие от иерархических алгоритмов.

Самым известным неиерархическим методом кластеризации является метод k-средних. Метод k-средних представляет собой следующую последовательность шагов:

Шаг 1: Выбрать k элементов, каждый из которых на первой итерации будет первым элементом одного из k кластеров. Каждый элемент будет являться центром кластера.

Шаг 2: Распределить оставшиеся элементы исходного множества по кластерам. Для распределения по кластерам необходимо определить критерий сравнения объекта с центром. Например, можно относить объект в кластеру, расстояние к центру которого у текущего объекта минимальное.

Шаг 3: После того, как все элементы распределены, рассчитать новый центр кластера. Это может быть среднее всех элементов.

Шаг 4.Повторять распределение исходного набора объектов по кластерам. Остановить распределение, когда полученный центр равен центру на предыдущей итерации. [17]

2.2. Выбор алгоритмов

1) Иерархический или неиерархический метод

Для решения задачи построения дерева классов бизнес-процессов больше подходят иерархические алгоритмы кластеризации, так как они, как и требуется по условию задачи, позволяют построить дерево, а также не требуется знать заранее количество формируемых кластеров.

2) Агломеративный или дивизимный

Для построения дерева классов можно использовать как агломеративный, так и дивизимный подход. Для сравнения результатов и получения двух решений стоит реализовать оба подхода.

3) Определение меры близости функций

Будем искать не расстояние, а меру близости функций. Для определения меры близости функций необходимо выбрать метрику для качественной шкалы, так как в исходной матрице программы используется качественная шкала измерений. Из рассмотренных методов наиболее подходящим является коэффициент Роа, который определяет отношение числа совпадающих единичных свойств к числу всех свойств. Для задачи важно учесть, что функции изменяют одни и те же наборы данных, то есть определить, сколько у функций совпадает свойств. После анализа исходных данных может понадобиться модифицировать подход.

4) Определение меры близости сформированных кластеров

Методы ближнего и дальнего соседа не подходят для данной задачи, так как необходимо учитывать совместное использование данных всеми функциями кластера, а не только одной. Поэтому объединение кластеров по двум функциям будет нерепрезентативным.

Центроидные методы не имеют для данной задачи смысла, так как близость значений суммы связностей функций в кластерах не означает, что эти кластеры стоит объединить.

Остается метод попарного среднего. Им и будем пользоваться для решения задачи, однако мы не будем искать среднее.

Рассчитывать близость кластеров будем следующим образом:

$$R=\sum_i^n \sum_j^n F(i,j)$$

Где

- i-Элементы из первого кластера
- j-Элементы из второго кластера
- F-таблица близости функций.

Тогда, мы будем объединять два кластера, если эти два кластера чаще используют одни и те же наборы данных чем другие кластеры.

5) Разделение кластера

   При реализации разделения кластера будем следовать общему подходу, но при принятии решения о распределении объекта в тот или иной кластер может понадобиться учитывать близость ко всем объектам, уже распределенным в кластеры.

Таким образом, имеются общие подходы к проведению кластерному анализу, которые может понадобиться адаптировать и модернизировать после анализа данных и получения результатов первых этапов разработки программы.

3. Реализация программы

План работ:

1) Реализация считывания исходной матрицы типа Функции-Данные и определение взаимосвязей каждой пары бизнес-процессов
2) Реализация расчета меры близости объектов
3) Подбор и реализация оптимального метода расчета меры близости кластеров
4) Визуализация результата
5) Интерпретация результата

3.1. Реализация считывания исходных данных и формирование матрицы близости функций

Исходные данные программы –это реальные анонимные данные из области финансов большого предприятия. Исходные данных программы представлены в таблице 1 (*Таблица 1*).

Таблица 1.Таблица исходных данных.

Функции\Данные	**ЭД1**	**ЭД2**	**ЭД3**	**ЭД4**	**ЭД5**	**ЭД6**	**ЭД7**	**ЭД8**
БП001	0	0	0	0	0	0	0	0
БП002	0	0	0	1	2	2	0	0
БП003	1	0	0	0	0	0	0	0
БП004	1	0	2	0	0	0	0	0
БП005	0	0	0	0	0	0	0	0
БП006	0	0	2	0	1	0	0	0
БП007	0	0	0	0	0	0	0	0
БП008	0	0	0	0	0	0	0	0
БП009	2	0	0	0	1	0	0	0
БП010	0	0	1	0	0	0	2	0
БП011	0	0	0	0	0	0	0	0

Таблица 1 строится на основании анализа бизнес-процессов предприятия. В таблице 1 (Таблица 1) используется порядковая шкала измерений (качественная), то есть числа в таблице означают относительную позицию объектов таблицы.

В таблице 0 означает, что функция (БП) не использует данные,1- функция читает данные, 2-функция изменяет данные. Таким образом, количественные методы расчета расстояния не подходят. Следовательно, необходимо обратиться к качественным шкалам.

Функции считаются связанными, если они используют одни и те же наборы данных. Соответственно важен факт использования функциями данных. Поэтому, как и было определено при выборе методов, воспользуемся подходом, близким к коэффициенту Роа. Однако будем также учитывать, читает функция данные или изменяет и не будем делить на общее количество измерений.

Для определения близости двух функций введем следующую формулу:

$$\textit{Связность (БП1, БП2)} = \sum_i^n (T(\text{БП1}, i) * T(\text{БП2}, i)) \qquad (1)$$

Данная формула является нашим предположением, а также элементом исследования в данной работе.

Где

Т-исходная *Таблица 1*.

БП1-порядковый номер первой рассматриваемой функции

БП2-порядковый номер второй рассматриваемой функции

Другими словами, если две функции совместно читают данные, то их связь увеличивается на единицу, если одна из функций еще и изменяет данные, то на 2, если обе изменяют, то на 4. Обуславливается различие связности функции при чтении и при изменении данных тем, что если функция данные изменяет, необходимо обеспечивать поддержку более сложных операций и поэтому такие функции более важно разместить в один класс, чтобы минимизировать сложность доступа к данным.

После проведения такого анализа в программе формируется матрица связности функций предприятия, анализ которой и должен привести к формированию классов функций.

Таблица представлена следующим образом (*Таблица 2*):

Таблица 2. Таблица связности функций.

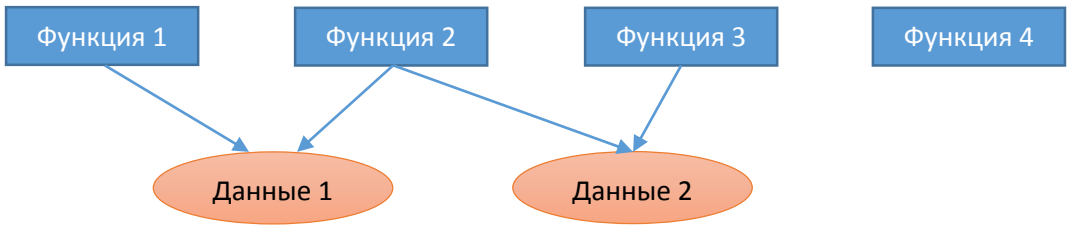
	Функция 1	Функция 2	Функция 3
Функция 1	0	2	3
Функция 2	2	0	4
Функция 3	3	4	0

3.2. Расчет меры близости функций и кластеров

Чем больше число в ячейке таблицы *Таблица связности функций.* на пересечении разных функций, тем больше одинаковых данных функции обрабатывают. Данные числа будем использовать как меру близости для сравнения функций, так как если функции обрабатывают один и тот же набор данных, целесообразно сгруппировать эти функции в одну группу для упрощения для обеих функций доступа к данным и минимизации перенаправления данных в разные блоки.

Однако по таблице связности можно получить информацию о расстоянии только между теми функциями, которые читают или изменяют одни и те же данные. Если же этого не происходит, в таблице связности *Таблица 2* стоит 0. Предлагается определить меру близости для функций, не использующих совместно данные.

Приведем пример:



- Функция 1 обрабатывает Данные 1.
- Функция 2 обрабатывает Данные 1 и Данные 2.
- Функция 3 обрабатывает Данные 2.
- Функция 4 не обрабатывает Данные 1 и Данные 2

Очевидно, что Функция 1 и Функция 2 связаны, необходимо задать способ определения их близости.

Для этого найдем транзитивное замыкание каждой функции.

Определение: Прямое транзитивное замыкание некоторой вершины xi T+ (xi) - это множество вершин, достижимых из вершины xi, т.е. T (xi) = { xj | ∃ путь из xi в xj }.

[11]

Таким образом найдем длину пути между всеми функциями и выделим независимые блоки функций. В приведенном примере длина пути между Функцией 1 и Функцией 3 равна одному, так как они связаны через одну функцию.

Так, на первом шаге выделяется определенное количество изолированных блоков функций. Внутри изолированного блока можно определить меру близости каждой пары функций. Следующий этап –определение способа разделения на кластеры для дивизимного алгоритма и способа объединения для агломеративного.

3.3. Дивизимный иерархический алгоритм.

Необходимо определить, каким образом делить кластеры. В соответствии с дивизимным алгоритмом, на первом шаге нужно выбрать первый элемент для каждого из новых двух кластеров.

Возьмем две функции из таблицы близости *Таблица 2*, между которыми самый длинный путь.

Далее оставшиеся элементы необходимо распределить по кластерам. Для решаемой в данной работе задачи важно минимизировать связь между кластерами, то есть распределить функции так, чтобы функции из разных кластеров как можно меньше использовали данные. Поэтому стандартный способ, при котором сравнивается расстояние между распределяемым объектом и исходным объектом кластера не подходит.

Для разделения кластера с учетом всех взаимосвязей между функциями используется следующий алгоритм:

1) Разделение начинается с исходного кластера, включающего все функции исходной изолированной системы
2) Далее кластер разделяется на две части и рекурсивно вызывается алгоритм разделения каждой из частей
3) При разделении кластера рассчитывается взаимосвязь рассматриваемого объекта и функций, которые уже были распределены в кластеры.

```
for(int j=0;j<claster1.size();j++) {

   centroid1 += initialBP. get(cluster1.get(j)).get(bp3);
}
for(int j=0;j<cluster2.size();j++) {

   centroid2 += initialBP.get(cluster2.get(j)).get(bp3);
}


if (centroid1 > centroid2) {

   cluster1.add(bp3);

} else if(centroid1 < centroid2) {

   cluster2.add(bp3);

}
```

Где

- cluster1- первый формируемый кластер
- cluster1- первый формируемый кластер
- centroid1- сумма взаимосвязей (количества совместно используемых данных) между объектом, который необходимо распределить в кластер, и объектами, которые уже находятся в кластере.

- initialBP- *Таблица 2. Таблица связности функций.*
- bp3-номер распределяемого объекта в исходном множестве функций

Другими словами, для решения задачи минимизации взаимодействия, необходимо учесть близость распределяемого объекта ко всем объектам кластеров. Таким образом, объект распределяется в тот кластер, с объектами которого данный объект чаще совместно использует данные. Так, будет минимизировано взаимодействие между кластерами.

3.4. Агломеративный иерархический алгоритм

В выборе методов был определен способ определения близости кластеров:

$$R=\sum_i^n \sum_j^n F(i,j)$$

Где :

- i-Элементы из первого кластера
- j-Элементы из второго кластера
- F-таблица близости функций.

Алгоритм объединения кластера на языке программирования java будет выглядеть следующим образом:

```
while (Количество кластеров >1) {

for (int i = 0; i < Количество элементов в кластере;i++) {
   for (int j = i + 1; j < Количество элементов в кластере;j++) {

      double value = Мера близости кластеров i и j по формуле
      if (value > max) {
         max = value;

         a = Кластер i;
         b = Кластер j;
      }
   }
```

```
Cluster c = splitCluster(a, b); //Объединяются два кластера в кластер c, кластер c включает
//две ветви- a и b
currentBP.remove(a);//убирается первый кластер
currentBP.remove(b); //убирается второй кластер
currentBP.add(c); //Добавляется новый сформированный кластер


}
}
```

Cluster c = Объединяются кластеры a и b;

Где

max – максимальное значение расстояния двух кластеров (в данном случае, чем больше расстояние, тем больше данных совместно используют функции из разных кластеров).

3.5. Визуализация и интерпретация

После того, как сформированные изолированные системы были распределены по классам и представлены в виде иерархического дерева, визуализируется сформированная структура классов с отображением входящих в класс функций.

Для интерпретации результатов добавлена функция расчета меры близости сформированных кластеров. Число само по себе не имеет значения, которое носило бы информационный характер для аналитика, но можно сравнивать близость кластеров в агломеративном и дивизимном разбиении. Чем больше число, тем больше данных совместно используют кластеры.

3.6. Структура программы

Программа построения дерева классов бизнес-процессов реализована на языке программирования Java.

Были использованы библиотеки:

1) Apache poi 3.11 –для работы с Exel-файлами
2) Jgraphx 3.3.0.0 – для рисования интерфейсов данных

Приведем описание важных классов программы:

Класс InMatrixAnalysis :

1) Классу передается исходная матрица функций и данных (*Таблица 1.Таблица исходных данных*.).
2) В классе осуществляется выявление функций, которые не читают ни один набор данных, и данные, которые не читают функции. Такие функции и данные не используются далее при проведении кластерного анализа и игнорируются при расчете матрицы связности (*Таблица 2. Таблица связности функций*.).

3) Фиксируются данные, совместно используемые функциями, таким образом, что по номерам двух функций можно получить все наборы данных, используемые двумя функциями.
4) Производиться расчет таблицы *Таблица связности функций*. .

Класс BusinessProcessesAnalysis :

Класс, производящий подготовительный анализ матрицы связности (*Таблица связности функций*.)

1) Определение функций, использующих данные, которые не используют другие функции.
2) Выявление изолированных систем.
3) Расчет близости каждой пары функций, входящей в изолированную систему.

Класс AgglClustering

Класс, производящий аггломеративный иерархический кластерный анализ.

Класс DivClustering

Класс, производящий дивизимный иерархический кластерный анализ. Как уже описывалось, для того, чтобы разделить кластер так, чтобы взаимодействие функций из сформированных кластерах было минимальным, необходимо рассчитывать близость распределяемой функции ко всем функциям кластера.

Класс Cluster

Класс для формирования объектов типа Cluster, которые используются в процессе кластеризации.

1) При создании объекта, конструктору класса передается список функций, входящих в кластер.
2) Реализовано добавления в кластер нового объекта, который является наследником кластера. Через наследников реализована связь в иерархии кластеров.
3) Отображение иерархии для каждого кластера.

Класс Tree

1) Класс отображения иерархии сформированных кластеров.

Класс InterfacesFrame

1) Отображение интерфейсов данных -данных, которые читают функции из обоих кластеров.
2) Сравнение численной интенсивности потока данных между кластерами ,полученной при агломеративном и дивизимной разбиении

3.7. Интерпретация результатов

Для начала анализа функций необходимо открыть исходный файл с данными. Исходный файл, при помощи которого производилось исследование,-это Exel-таблица, в которой представлены реальные анонимные данные из области финансов большого предприятия. После того, как был считан исходный файл и произведен анализ, номера систем добавляются в список изолированных систем. Номера функций отображаются в специальном окне, как изображено на рисунке 10 (*Рисунок 10*).

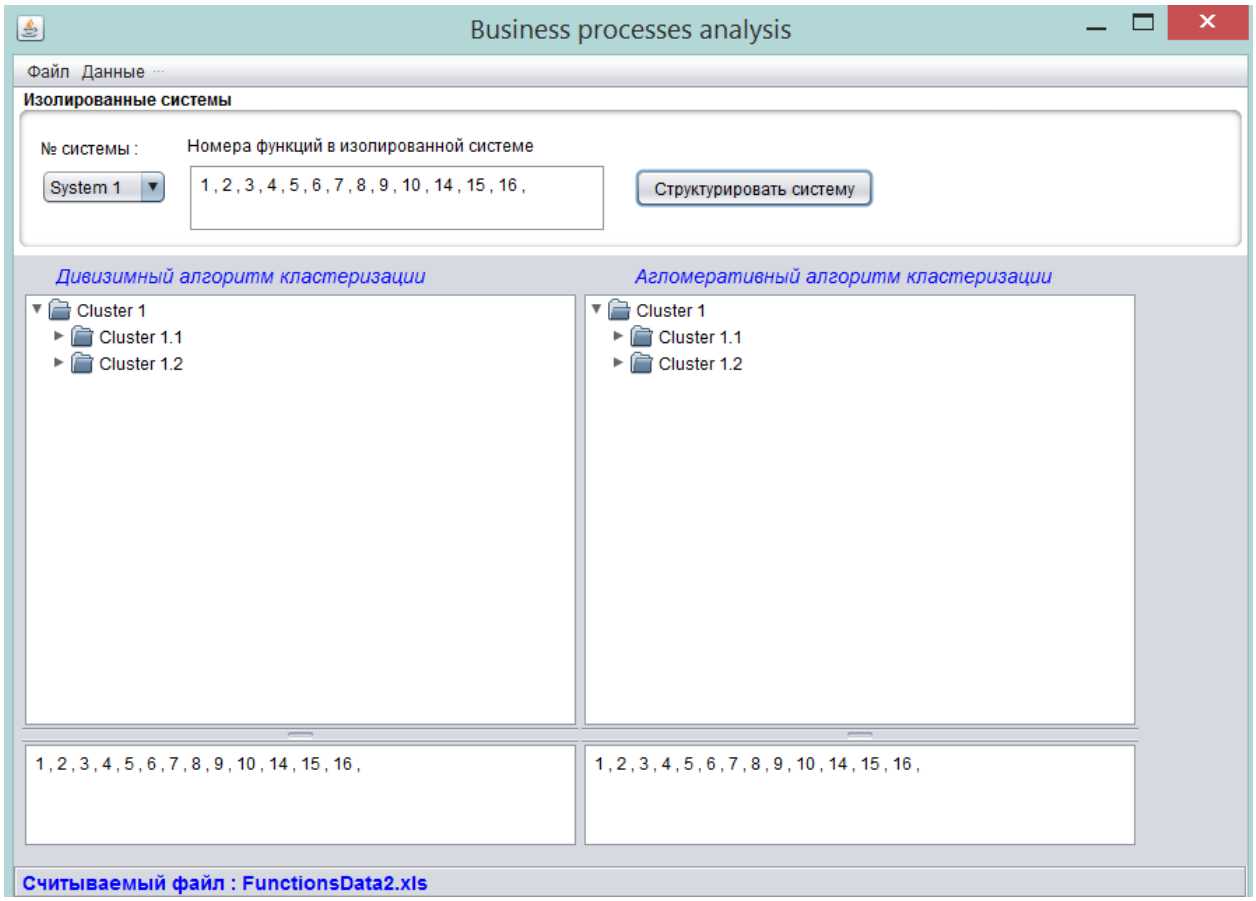


Рисунок 10.Интерфейс программы.

Изолированные системы - первый важный результат программы. Так как функции в разных системах не взаимодействуют между собой, аналитик может воспользоваться данной информацией и сформировать соответствующие предложения по системной архитектуре.

Помимо этого производиться анализ консистентности исходных данных. Во первых выявляются элементы данных, которые не используются ни одной из функций. Во вторых выделяются функции, которые не используют ни один набор данных или совместно не используют данные с другими функциями. Результат отображается в специальных окнах программы, как показано на рисунке 11 (*Рисунок 11*).

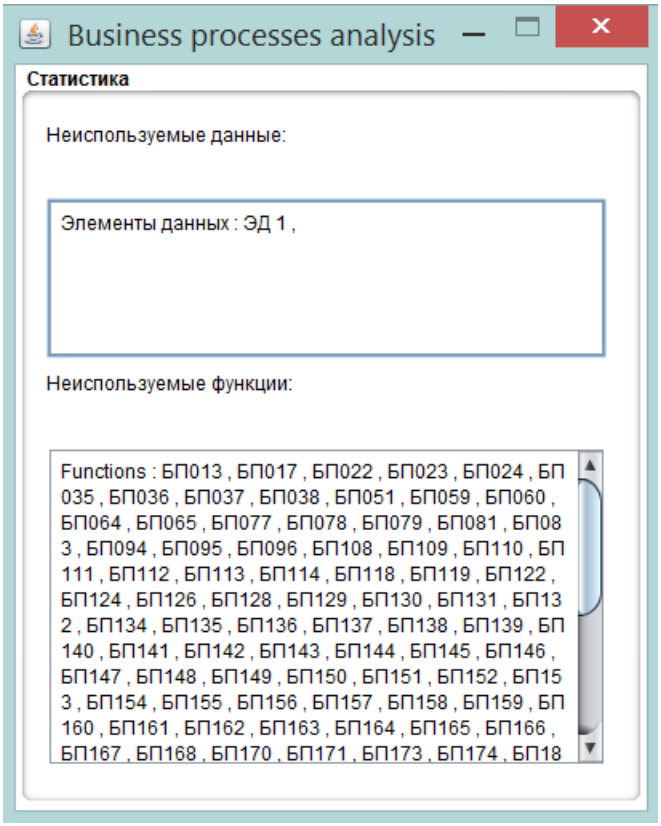


Рисунок 11. Интерфейс программы, неиспользуемые данные и функции.

Данная информация также полезна для аналитика и означает необходимость проверить корректность исходных данных и проанализировать возможные причины таких результатов.

Далее можно структурировать изолированную систему. В двух панелях с дивизимном и агломеративным алгоритмом будут отображены результаты разбиения, как показано на рисунке 10 (*Рисунок 10*).

Аналитик может использовать данное разбиение следующим образом:

1) Определить необходимое количество систем, на которые следует разделить изолированную систему.
2) Спускаясь вниз по иерархии (раскрывая узлы дерева), раскрыть нужнее количество систем.
3) Просмотреть номера функций для каждой системы.
4) Сравнить разбиение, предложенные двумя алгоритмами и выбрать то, которое лучше отвечает потребностям конкретной задачи.

Результаты разбиения агломеративного и дивизимного алгоритма могут быть разными, и для интерпретации результатов и принятия решения об эффективности аналитику необходимо воспользоваться дополнительными средствами по интерпретации.

Так, в программе есть способ расчета интенсивности потоков данных между кластерами и отображения интерфейсов данных.

Чем больше численный результат вычисления интенсивности потоков, тем больше данных совместно используют функции из разных кластеров. Соответственно, если производится деление двух одинаковых наборов функций, то более эффективным с точки зрения совместного использования данных будет то разбиение, для которого число в окне “Результат” меньше (*Рисунок 12.Расчет близости кластеров.*).

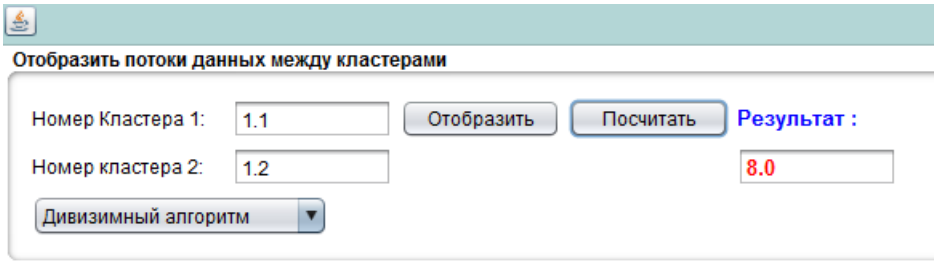


Рисунок 12.Расчет близости кластеров.

Далее, можно посмотреть интерфейсы между сформированными кластерами. Интерфейс-совокупность возможностей, способов и методов взаимодействия двух систем. [9]

В данной работе интерфейсы систем – это наборы данных, которые необходимо перенаправлять между системами. В качестве демонстрации возможностей разработанной программы на рисунке 13 (*Рисунок 13*) показаны рассчитанные интерфейсы.

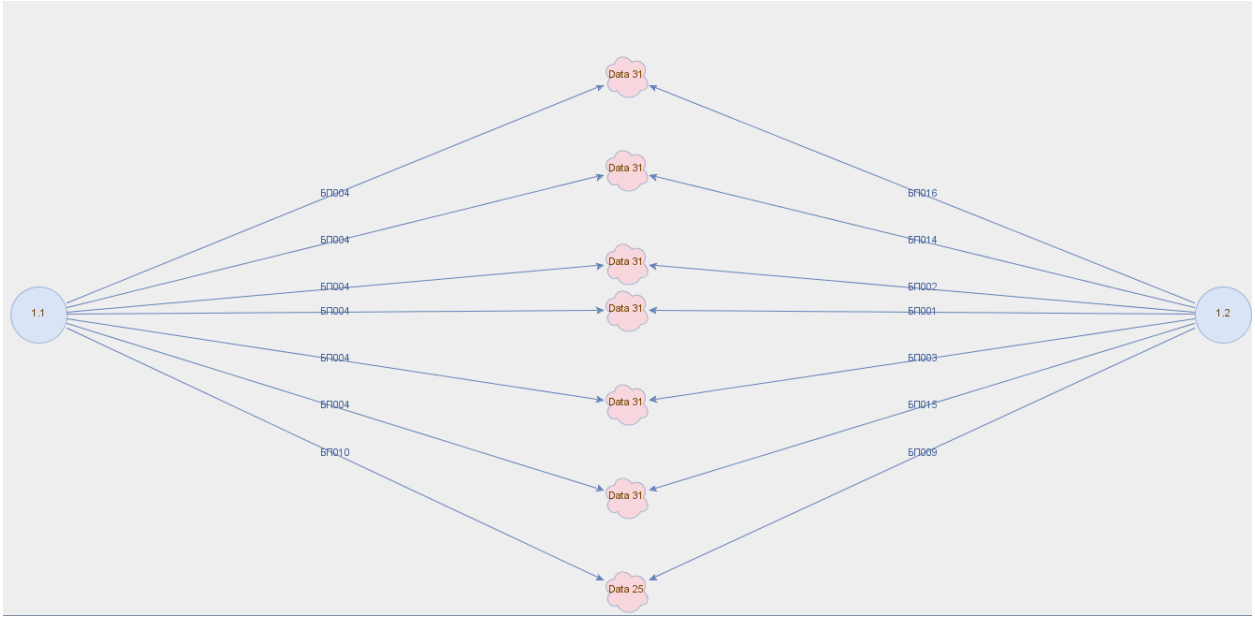


Рисунок 13. Интерфейсы систем.

На рисунке отображается, какие данные используются какими функциями из обеих систем.

Так, на рисунке отображено, что функция с название БП004 из первого кластера и функция БП016 из второго используют Данные 31.

Аналитик может сравнить интерфейсы, получающиеся при двух алгоритмах, и выбрать то разбиение, которое соответствует целям и задачам структуры предприятия.

3.8. Оценка результатов

Полученные результаты являются ценными, так как подтверждается возможность структурирования изолированных наборов функций и предлагаются возможные варианты такого структурирования.

Для подтверждения результатов была произведена теоретическая проверка распределения функций по кластерам, при этом сопоставлялось предполагаемое разбиение, построенное на основе исходной информации о совместно-используемых данных, и то, как функции распределяются по подсистемам при помощи программы. Проверка подтвердила справедливость распределения, действительно функции, которые читают одни и те же наборы данных, оказываются в одном кластере.

Приведем результат проверки, в которой анализировалась небольшая исходная матрица (*Таблица 3*):

Таблица 3.Пример таблицы исходных данных.

Наименование	ЭД1	ЭД2	ЭД3	ЭД4	ЭД5	ЭД6	ЭД7	ЭД8
БП001	1	2	2	1	0	0	0	0
БП002	1	2	2	0	0	0	0	0
БП003	2	2	1	0	0	0	0	0
БП004	2	2	1	0	0	0	0	0
БП005	0	0	0	0	0	0	0	0
БП006	0	0	0	0	2	0	0	0
БП007	0	0	0	1	1	0	0	0
БП008	0	0	0	2	2	0	0	1
БП009	0	0	0	1	2	0	0	0
БП010	0	0	0	2	2	0	0	1
БП011	0	0	0	2	2	0	0	0
БП012	0	0	0	0	0	2	1	1
БП013	0	0	0	0	0	2	2	2
БП014	0	0	0	0	0	2	2	2

Без помощи программы можно выделить следующие группы:

- 1,2,3,4
- 6,7,8,9,10,11
- 12,13,14
- 5-неиспользуемая функция

Проанализируем данные при помощи программы. При разделении на две группы два алгоритма одинаково разбили функции.

- Группа 1: 1 , 2 , 3 , 4
- Группа 2: 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14
- Неиспользуемая функция 5.

При делении на 4 группы результат выглядит следующим образом:

Дивизимный алгоритм:

- Группа 1.1: Функции 1 , 2 , 4
- Группа 1.2: Функции 3
- Группа 2.1: Функции 6 , 7 , 8 , 9 , 10 , 11
- Группа 2.2: Функция 12 , 13 , 14
- Неиспользуемые функции: 5

Агломеративный алгоритм:

- Группа 1.1: Функции 1 , 2 , 3
- Группа 1.2: Функции 4
- Группа 2.1: Функции 6 ,7, 8 , 9 , 10 , 11
- Группа 2.2: Функция 12 , 13 , 14
- Неиспользуемые функции: 5

Различие в Группе 1 несущественно, различия в отделении функций 3 и 4 нет.
Полученные результаты соответствуют ожидаемым результатам. Результат разбиения минимизирует количество совместно используемых данных между подсистемами.

Ожидаемый результат легко представить при относительно простой структуре данных и небольшом количестве функций и данных, но если матрица состоит из 1000 элементов данных и 1000 функций, анализ получается очень непростым. Программа построения дерева классов бизнес-процессов быстро (матрица из 200 элементов данных и 700 функций анализируется за 30 секунд) анализирует данные, выявляет неиспользуемые данные и функции, определяет изолированные системы функций и предлагает структуризацию изолированных систем.

Так как направление исследования является новым и строится на предположениях, необходимо проведение дополнительных исследований в соответствующей области.

Одним из вопросов, которые необходимо рассмотреть в дополнительных исследованиях является способ расчета матрицы связности функций (*Таблица 2*). Могут быть предложены альтернативные алгоритмы кластеризации изолированных систем. Помимо этого, необходимо получить комментарии аналитика после тестирования программы.

4. Заключение

В настоящее время большинство крупных компаний понимают необходимость построения архитектуры предприятия и активно развиваются в направлении структурированного описания и проектирования архитектуры. Это в основном осуществляется при помощи специалистов в области построения архитектуры предприятия, которые используют многочисленные методологии и рекомендации, которые помогают в построении архитектуры. Однако до сих пор не существовало систем, которые бы автоматизировано предлагали решения по конструированию и оптимизации структуры предприятия, есть только системы, которые осуществляют мониторинг и отображение уже имеющейся структуры. Программа построения дерева классов бизнес-процессов являются частью, которая может войти в комплекс систем по автоматизированному построению архитектуры предприятия. Программа осуществляет анализ функций систем и группирует функции в подсистемы на основе информации об использовании этими функциями данных. Группирование функций в слабовзаимодействующие подсистемы поможет уменьшить сложность системы, что увеличит ее прозрачность, снизит множество рисков и увеличит вероятность построения эффективной архитектуры предприятия.

Список использованных источников

[1] **AG Software** Software AG's Alfabet Enterprise Architecture Management [В Интернете] // Software AG. - 20 Апрель 2015 г.. - http://www.softwareag.com/corporate/products/aris_alfabet/ea/overview/default.asp.

[2] **Danili A. Slusarenko A.** http://www.intuit.ru/studies/ courses/995/152/info [В Интернете]. - 2005 г.. - 2015 г.. - http://www.intuit.ru/studies/ courses/995/152/info.

[3] **InterfaceLtd** Rational Unified Process [В Интернете]. - май 2015 г.. - http://www.interface.ru/rational/rup01_t.htm.

[4] **Jensen Claus T.** If you have SOA, do you still need EA? [В Интернете]. - IBM Software Group. - май 2015 г.. - http://www-01.ibm.com/software/solutions/soa/newsletter/aug10/enterprise_architecture.html.

[5] **Sessions Roger** A Fundamental Metric for Predicting IT Success [В Интернете]. - 23 April 2012 г.. - май 2015 г.. - https://drive.google.com/file/d/0B_PdYY3dTisIcFQyRjlwSlF0Uzg/view.

[6] **Sessions Roger** Comparison of the Top Four Enterprise Architecture Methodologies [В Интернете]. - 2007 г.. - https://drive.google.com/file/d/0B_PdYY3dTisIMTREUDNuVkhsbXM/view.

[7] **Sessions Roger** Controlling Complexity in Enterprise Architectures [В Интернете]. - 2007 г.. - май 2015 г.. - https://drive.google.com/file/d/0B_PdYY3dTisIN3dDV0pGSW14SEU/view.

[8] **Statsoft** http://www.statsoft.ru/home/textbook/modules/stcluan.html [В Интернете]. - 22 апрель 2015 г.. - http://www.statsoft.ru/home/textbook/modules/stcluan.html.

[9] **Wikipedia** Интерфейс [В Интернете] // Wikipedia. - 11 май 2015 г.. - https://ru.wikipedia.org/wiki/Интерфейс.

[10] **А. Данилин А. Слюсаренко** Архитектура и стратегия. «Инь» и «Янь» информационных технологий предприятия [Книга]. - Москва : [б.н.], 2005.

[11] **Волченская Т. В. Князьков В. С.** Компьютерная математика:Часть 2. Теория графов [Книга]. - Пенза : Изд-во Пенз. ун-та, 2002.

[12] **Ибрахим Мамду** Сервис-ориентированная архитектура и архитектура предприятия: Часть 1. Взаимодействие SOA и EA [В Интернете]. - 2008 г.. - 11 май 2015 г.. - http://www.ibm.com/developerworks/ru/library/ws-soa-enterprise1/.

[13] **Калянов Г. Н.** Архитектура предприятия и инструменты ее моделирования [Статья] // Автоматизация в промышленности. - июль 2014 г.. - стр. 9-12.

[14] **Мандель И. Д.** Кластерный анализ [Книга]. - Москва : Издательство "Финансы и статистика", 1988.

[15] **Р.Сешнс** Сравнение четырех ведущих методологий построения архитектуры предприятия,http://msdn.microsoft.com/ [В Интернете] // Сравнение четырех ведущих методологий построения архитектуры предприятия. - 05 2007 г.. - 5 2015 г.. - http://msdn.microsoft.com/ru-ru/library/ee914379.aspx.

[16]**Тилак М.** Архитектура на практике: Часть 1.Реализация сервис-ориентированной [В Интернете]. - IBM, 9 апрель 2010 г.. - май 2015 г.. - http://www.ibm.com/developerworks/ru/library/ar-arprac1/.

[17]**Чубукова И. А.** Курс лекций по Data Mining [Книга]. - 2006.