О прогнозировании развития информационных систем для бизнеса методами ТРИЗ

Сысоев С.С.
заместитель генерального директора
ООО «ПетроМС»,
к.ф.-м.н.
sysoev@petroms.ru

1. Введение

«Информационные системы для бизнеса», строго говоря, не образовывают замкнутого и четко определяемого класса информационных систем. Классификация систем по потребителю вообще затруднительна в случае с информационными технологиями, особенно если в качестве потребителя выступает такое общее понятие как «бизнес».

Тем не менее, термин этот является часто используемым, и поэтому мы позволили себе вынести его в заголовок статьи. Значение, которое вкладывается нами в понятие «ИС для бизнеса», обусловлено нашим опытом автоматизации бизнес-процессов некоторых крупных предприятий, и структурно может быть ограничено следующим списком характеристик:

1. Информационная система должна обеспечивать взаимодействие в информационном поле групп людей, объединенных с точки зрения надсистемы (бизнеса) некоторым деревом целей. Идеальной в этом смысле является ситуация, когда все задействованные в бизнесе люди – сотрудники, клиенты, контрагенты, контролирующие органы – работают в единой информационной среде.

2. Информационная система должна обеспечивать протоколирование операций. Для каждого действия, совершенного в информационной системе, можно определить, кто и когда его выполнил.

3. Информационная система должна иметь подсистемы ролей и прав доступа. Каждой роли пользователя в системе может быть сопоставлен набор возможных действий и прав.

Таким образом, предметом данного небольшого исследования являются все информационные системы, обеспечивающие взаимодействие людей с разными функциями в бизнесе, и протоколирующие это взаимодействие.

Цель данной работы – анализ применимости методов прогнозирования ТРИЗ для определения трендов развития такого рода систем. Точность умозрительных выводов относительно перспектив той или иной технологии часто оказывается недопустимо низкой, в результате чего погибают инвестиции, вложенные в перспективную, но не востребованную разработку. Инвестирование в «технологии завтрашнего дня» необходимы, так как завтрашний день, безусловно, наступит – с нами, или без нас. Подготовленная и, тем более, опережающая прогресс компания, имеет шансы на успех в области предоставления услуг в области ИТ.

Прогнозирование в этой работе ведется в соответствие с методикой АСС-прогнозирования, представленной в [1].

2. Система, надсистемы, подсистемы

Информационная система для бизнеса выполняет роль посредника между пользователем и ресурсами бизнеса. Пользователем может быть сотрудник компании, сотрудник контрагента, клиент, налоговый инспектор и т.д. Ресурсы – это сотрудники и контрагенты, материалы, товары, денежные средства, сообщения и заявления, документы, отчеты и т.д.

С точки зрения системного анализа люди и ресурсы являются подсистемами системы бизнес-процессов, которая отвечает за выполнение бизнес-задач (см. рис. 1).

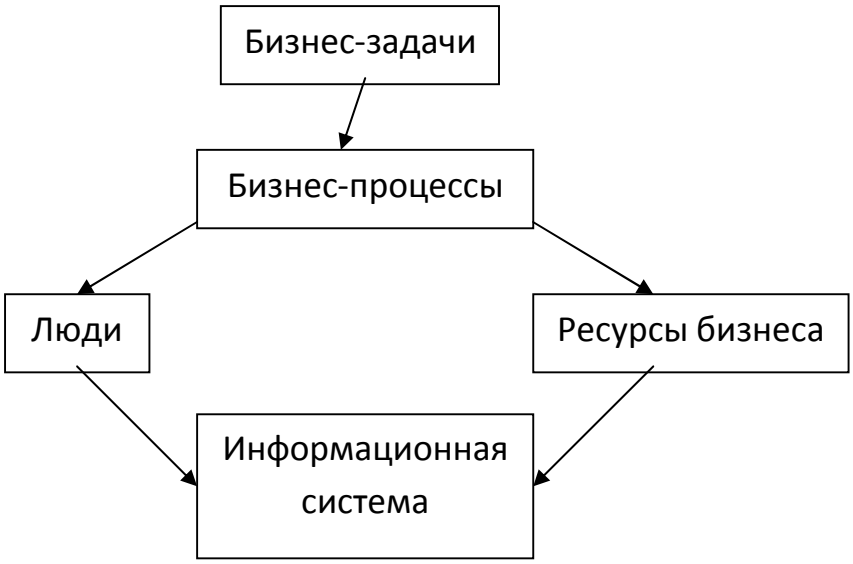

Рисунок 1. Надсистемы ИС.

Так, например, в классической библиотеке, предоставляющей читателям книги, бизнес-задачами являются:

1. Предоставление книг читателям.
2. Пополнение фондов.
3. Сохранение редких и важных книг
4. Получение прибыли от сопутствующих услуг.

Бизнес-процессы, обеспечивающие выполнение этих задач могут быть таковы:

1. Хранение книг.
2. Организация работы читального зала.
3. Процесс выдачи книг на дом.
4. Поиск книг по запросу посетителя.
5. Управление кадрами.
6. Ведение бухгалтерии.
7. …

Информационной системой в рассмотренном примере является картотека библиотеки. Пользователи информационной системы - посетители, ищущие книгу, и библиотекари, заполняющие в карточках информацию о местоположении книг. Ресурсом в рассматриваемом примере являются книги, а так же шкафы, полки, залы хранилища. В поисках книги посетитель обращается к услугам картотеки, в которой находит карточку книги. Из карточки пользователь извлекает информацию о местоположении книги.

Описанный процесс иллюстрирует очень важное понятие – модель ресурса. Связь между пользователями и ресурсами в ИС происходит через работу с моделями ресурсов. Модель книги – карточка в картотеке.

Для создания модели необходимо сначала определить ее структуру. Например, структурой карточки книги является набор заполняемых в ней полей – автор, название, количество страниц, место хранения (зал, шкаф, полка), записи о выдаче на руки, и т.д.

Данные об объектах (ресурсах) хранятся в базе данных (БД) информационной системы. Определение структур моделей объектов и связей между ними называется формированием структуры базы данных.

Сведения о филогенезе структур данных, а так же связанные с ними противоречия приведены в Приложении.

Характер взаимодействия пользователя с информационной системой определяется той частью ИС, которая отвечает за это взаимодействие – интерфейсом. Интерфейс картотеки представляет собой набор шкафов с

буквами на ящиках. Интерфейс технолога на производстве – совокупность мониторов, на которых схематично отображается текущее состояние производственного цикла. С интерфейсом связано много важных и противоречивых требований, таких как полнота и простота. Сложный, непонятный интерфейс способен полностью остановить процесс внедрения ИС, или, по крайней мере, существенно увеличить его себестоимость.

Сведения о филогенезе интерфейсов приведены в Приложении.

Часто данные об объектах нужно не только хранить, но и преобразовывать. За преобразование данных отвечают алгоритмы. Термин «алгоритм» выбран нами как наиболее общий, хотя он и не охватывает непрерывные процессы преобразования. Филогенез алгоритмов рассмотрен в Приложении.

Последней подсистемой, рассмотрению которой мы уделим внимание, будет подсистема технических средств – совокупность программных и аппаратных решений, обеспечивающих функционирование перечисленных нами ранее подсистем. Филогенез технических средств рассмотрен в Приложении.

3. Формулировка и анализ противоречий

Сформулируем противоречие, представляющееся нам ключевым не только в области информационных технологий, но и в развитии любого продукта или услуги:

Административное противоречие 1. Разработка качественной, удобной информационной системы, позволяющей автоматизировать бизнес-процессы предприятия и повысить их прозрачность, стоит дорого. Это связано с высокой наукоемкостью и сложностью используемых технологий, что влечет высокие затраты на квалифицированных инженеров и аналитиков, сложностью предметной области, в которую разработчикам и аналитикам необходимо вникать. В то же время, Заказчик не готов оплачивать такую систему полностью, просто потому, что для очень малого класса предприятий оправдан такой уровень расходов на автоматизацию. Малые предприятия при таком подходе не смогут позволить себе вообще ничего.

Анализ изобретательской ситуации позволяет нам построить элеполь (термин впервые введен в [2, 3]) следующего вида:

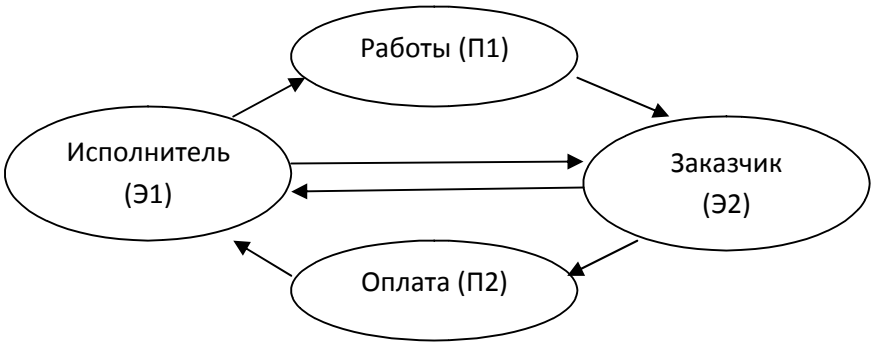

Рисунок 2

Из строения элеполя видно, что элемент «Исполнитель» через поле «Работы» изменяет элемент «Заказчик», и изменение это хорошее. В то же время элемент «Заказчик» через поле «Оплата» меняет элемент «Исполнитель», и воздействие это хорошее, но недостаточное. Элеполь представляет собой условие для применения группы стандартов (см. [4]) «Форсирование веполей».

Стандарт 2.2.2 – Дробление инструмента. Поскольку стоимость копирования продуктов в области ИТ является практически нулевой, то работы по созданию системы можно сделать один раз, а оплату получить от большого количества разных Заказчиков.

Чтобы это простое решение не показалось нам панацеей, вспомним этапы создания ИС – ее жизненный цикл:

1. Проведение обследования Заказчика, предметной области, постановка задачи (Этап разработки технического задания).
2. Техническое проектирование – выбор технологий, проектирование архитектуры системы, формирование плана разработки, …
3. Рабочий проект – воплощение системы с использованием выбранных технологий, по сформированному плану. Тестирование.
4. Ввод в действие – опытная эксплуатация системы, приемка, ввод в промышленную эксплуатацию.

5. Сопровождение – обновление, разрешение нештатных ситуаций.
6. Утилизация.

Увеличение количества Заказчиков не повлияет на стоимость работ первых трех этапов, причем только для «одинаковых» Заказчиков. Этапы 4 и 5 полностью продублируются для каждого нового Заказчика. Пожалуй, только с этапом 6 не будет проблем. Возникает противоречие:

Техническое противоречие 1. ЕСЛИ систему сделать более универсальной, ТО для нее применимо копирование для большего количества Заказчиков, НО сложность разработки, внедрения и использования сильно возрастают, что снижает ее привлекательность для каждого Заказчика.

Тем не менее, копирование в нашем случае – очень соблазнительный (бесплатный) прием. Попытки защитить стандарт 2.2.2 и разрешить ТП1 хотя бы для первых трех этапов нашли отражение в следующих тенденциях развития ИС:

1. Возникновение узкоспециализированных ИС, всесторонне автоматизирующих некоторый простой процесс. (Пример – обилие систем автоматизации сопровождения и помощи пользователям ServiceDesk). Использован ***принцип местного качества***.
2. Создание вертикальных решений – ИС, автоматизирующих «типового» заказчика в какой-либо предметной области. (Пример – системы управления отелями и отельными сетями). Использован ***принцип вынесения***.
3. Создание горизонтальных решений – автоматизация группы процессов, присутствующих на любом предприятии. (Пример – платформа 1С:Предприятие и ее конфигурации). Использован ***принцип вынесения***.
4. Создание универсальных платформ, позволяющих адаптацию под Заказчика перенести на этап ввода в действие (Пример – 1С, Lotus Notes, DocsVision, BizTalk, SAP). Использован ***принцип универсальности.***

Как ни странно, все перечисленные схемы успешно существуют на сегодняшний день, ни одна из них не вытеснила другие.

Ввод в действие – достаточно дорогой этап (особенно, если внедряется универсальная платформа), а стандарт 2.2.2 не дает намека на форсирование веполя на этом этапе. Двигаемся дальше:

Стандарт 2.2.4 – Динамизация веполя.

Динамизация веполя может включать:

1. Динамизацию Заказчика.
2. Динамизацию Исполнителя.
3. Перемешивание Заказчика и Исполнителя.

Возможны и другие варианты, но мы остановимся на этих, как на наиболее перспективных.

Динамизацию Заказчика можно понять как изменение Заказчика под внедряемую систему. Тогда процесс внедрения заключается не в настройке системы, а в «настройке» Заказчика – изменение орг.структуры, замена сотрудников, изменение бизнес-процессов.

Динамизация Исполнителя хорошо прослеживается у системных интеграторов, выполняющих заказы силами субподрядчиков.

Перемешивание Заказчика и Исполнителя – очень удачный прием, в результате которого часть работ (в идеале все) по внедрению выполняют сотрудники Заказчика, находясь под контролем у Исполнителя.

Стандарт 2.2.5 – Структуризация веполя. Упорядочивание поля работ, как правило, приводит к созданию методологий и технологий работы, что иногда приводит к существенному повышению качества. Часто перспективы развития ИТ видят именно в этом направлении.

Стандарт 2.2.6 – Структуризация вещества. Компания, занимающаяся вводом в действие ИС, может отбирать себе Заказчиков по некоторому признаку, позволяющему в ее случае удешевить процесс. Пример – поиск Заказчиков в одном регионе, или в одной предметной области.

Стандарт 2.3.1 – Согласование ритмики поля и изделия. Терминология и настройки ИС должны быть настолько понятным для специалистов предметной области, что ИС внедряется сама силами Заказчика.

4. Выводы

В результате проведенных исследований можно сделать следующие выводы:

1. Для анализа понятия «ИС для бизнеса» успешно использованы методы системного анализа и системный оператор. В результате получены предварительные результаты относительно трендов развития подсистем (см. Приложение 1).
2. Для анализа административных противоречий были использованы принятые в ТРИЗ формулировки противоречий, вепольный анализ, система стандартов и элементы функционального анализа. Общий результат – информационные системы для бизнеса развиваются в направлении упрощения их продаж, то есть копирования для разных Заказчиков. Подтверждение этому можно найти в результатах анализа развития архитектур ИС (Приложение 1).

Общий вывод: методы прогнозирования ТРИЗ применимы для анализа сложной и запутанной ситуации с развитием информационных систем для бизнеса. Небольшая попытка анализа административного противоречия по системе стандартов привела к практически полной классификации подходов к разработке и внедрению ИС, а также позволила сделать предположения относительно тенденций их развития.

5. Список литературы

1. Рубин М.С. АСС-Прогнозирование. 2010. (http://www.triz-summit.ru/file.php/id/f4764/name/acc-прогнозирование-5.pdf).

2. Рубин М.С. Филогенез социокультурных систем. Секреты развития цивилизаций. 2010. (http://www.temm.ru/ru/section.php?docId=4472)

3. Рубин М.С., Одинцов И.О. Опыт применения методов ТРИЗ для повышения эффективности разработки ПО. 2009. (http://www.temm.ru/ru/section.php?docId=4419)

4. Альтшуллер Г.С. В сб. "Нить в лабиринте". - Петрозаводск: Карелия, 1988. - С. 165-230. Маленькие необъятные миры: стандарты на решение изобретательских задач стандартные решения изобретательских задач (76 стандартов).

5. http://ru.wikibooks.org/wiki/История_развития_ЭВМ).

Приложение. Филогенез информационных систем.

Исходя из результатов системного анализа (Глава 2), мы будем рассматривать информационные системы для бизнеса с точки зрения их подсистем, в соответствие со следующей структурой:

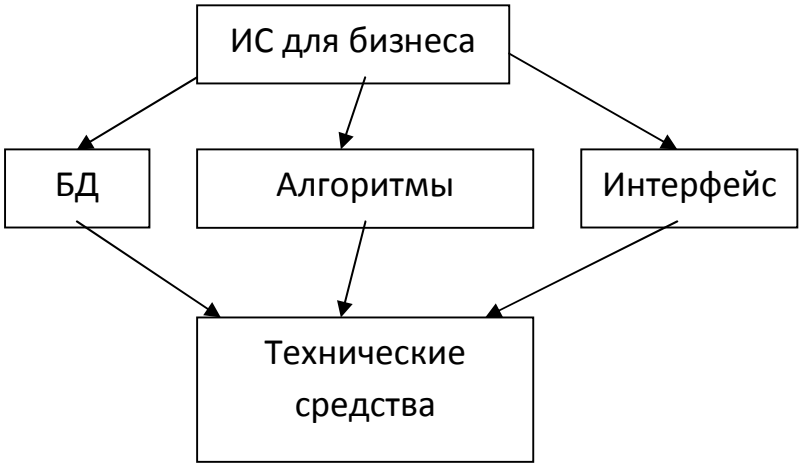

Рисунок 3. Подсистемы ИС.

П1. Структура данных

Структура данных может оцениваться по следующим характеристикам:

1. Полнота (точность модели).
2. Скорость доступа к данным на изменение (добавление) данных.
3. Скорость доступа к данным на чтение (формирование отчетов).

Эволюция структур данных в информационных системах кратко может быть представлена в виде следующей таблицы:

До 1968-го года	Последовательный доступ к данным
1968 - 1980	Сетевые и иерархические структуры данных
1980 – по настоящее время	Реляционная модель данных, SQL
1995 – по настоящее время	Объектно-ориентированная модель данных
1990-е, 2000-е	Развитие технологий OLAP, противопоставление их OLTP
1990-е	Развитие персональных БД
2000-е годы	Гибриды реляционной и объектно-ориентированной моделей (Oracle)
2000-е годы	Смещение акцента с персональных БД к облачным технологиям

Примечательно, что наш пример с картотекой библиотеки, с точки зрения эволюции структур данных более совершенен, чем компьютерные информационные системы до 1968 года. Технические средства того времени не позволяли осуществить мгновенный доступ к данным, как это позволяет сделать картотека.

Переход к сетевым и иерархическим структурам данных привел к улучшению характеристик доступа к данным. Тем не менее, вопросы полноты страдали, так как указанные структуры накладывали существенные ограничения на процессы моделирования.

Ограничения на структуры данных были сняты в модели сотрудника IBM доктора Кодда, предложенной им в 1972 году. Модель была названа реляционной. Для доступа к данным был предложен язык SQL, являющийся и на настоящее время стандартом организации доступа к данным.

Реляционная модель и язык доступа к данным SQL – наиболее существенные изобретения в теории баз данных и эволюции структур данных. Они сняли противоречия, возникавшие в предшествующих структурах, и до недавнего времени новых противоречий не возникало.

OLTP – Online Transaction Processing – реляционная структура данных, обеспечивающая высокую скорость транзакций по добавлению записей в БД. Упор в такой структуре делается на высокую «размазанность» данных по таблицам, большое количество связей.

Системы, данные которых организованы по принципу OLTP, хорошо справляются с быстрым добавлением записей (например, получение данных о покупках с касс в сети гипермаркетов в режиме реального времени). Однако такая структура данных препятствует быстрому формированию аналитических отчетов, так как большое количество переходов по ссылкам серьезно усложняет отчет.

В 1993 году, тем же доктором Коддом, который изобрел реляционную модель и SQL, была предложена структура данных OLAP – Online Analytical Processing. Данные в модели организованы в виде гиперкубов. Аналитические отчеты строятся путем формирования срезов этих кубов по выбранным плоскостям.

Структура OLAP совершенно не подходит для быстрого обновления данных. Формирование кубов обычно запускают по ночам на базе данных из OLTP структуры. Это компромиссное решение противоречия:

Структура данных должна быть хорошо нормализована (содержать минимум избыточной информации), чтобы обеспечить высокую скорость выполнения операций добавления и изменения данных, и должна содержать большое количество избыточной информации (как, например, гиперкуб), чтобы обеспечивать быстрое формирование сложных отчетов.

ИКР: данные отчета САМИ меняются динамически на глазах у пользователя по мере добавления новых данных в базу.

Следует отметить, что указанное противоречие возникает при довольно больших объемах операций с данными, что пока не грозит малому и среднему бизнесу.

П2. Интерфейс

Интерфейс пользователя к ИС может оцениваться по следующим характеристикам:

1. Полнота (возможность осуществления всех требуемых операций в ИС, и при этом невозможность осуществления запрещенных операций).
2. Защищенность «от дурака». Предотвращение выполнения формально допустимых операций, влекущих нежелательные последствия для пользователя.
3. Наглядность и простота. Большой набор управляющих элементов и выводимых данных требует организации дорогостоящих процессов обучения пользователей, повышает вероятность ошибок, вызывает утомление пользователя и негативное отношение к работе в системе.
4. Эстетика. Красота интерфейса может быть рассмотрена отдельно от наглядности.
5. Доступность операций. Операции, выполняемые пользователем при решении его задач в системе, должны требовать от него минимального количества действий. Обычно это требование формулируют следующим образом: наиболее часто выполняемые операции должны

осуществляться наиболее просто. Часто употребляемая мера доступности операций – «количество кликов» мышью.

Нетрудно заметить, что требование полноты интерфейса конфликтует со всеми другими требованиями, кроме эстетики.

Полнота, тем не менее, является ключевым требованием к интерфейсу. Интерфейс, не дающий доступа ко всем необходимым операциям, не работоспособен. Эволюция интерфейсов с точки зрения полноты заключается в росте количества проверок, предотвращающих выполнение недопустимых и нежелательных операций (в частности, усложнение механизмов защиты «от дурака»).

Наглядность и простота не являются главными критериями работоспособности системы, так как могут быть компенсированы процессом подготовки пользователя. Подготовка пользователя – процесс дорогостоящий и не надежный. Идеальная система вообще не должна требовать от пользователя каких-либо других знаний, кроме знаний о предметной области. Поэтому интерфейсы достаточно заметно эволюционируют в сторону идеальности.

Для работы с первыми поколениями ЭВМ требовалась специальная подготовка в высших учебных заведениях. Сейчас в технические задания на разработку ИС почти всегда включают пункт об отсутствии специальных требований к пользователю – должно быть достаточно основных навыков работы на компьютере. И число этих основных навыков последовательно сокращается.

В сложных ИС неизбежно возникает противоречие:

ЕСЛИ удовлетворить требование полноты интерфейса, ТО количество управляющих элементов и выводимых данных должно быть велико (в соответствие с требованиями предметной области), НО тогда простота и наглядность интерфейса полностью теряются.

Последовательное разрешение этого противоречия достигается внедрением изобретений в области технических средств:

1. Сигнальные лампы, перфокарты и распечатки в первых ЭВМ.
2. Возникновение консолей – мониторов и клавиатур, интерфейс командной строки.

3. Возникновение графических режимов, позволяющих использовать все пространство экрана для размещения управляющих элементов и данных (интерфейс Norton Commander и др.).

4. Эволюция графических режимов, внедрение новых видов манипуляторов – мыши, джойстики.

5. Возникновение touch screen – позволяет использовать в качестве манипуляторов собственные руки.

6. Революционные на сегодняшний момент разработки – в качестве устройств ввода используется собственное тело. Управляющие элементы проецируются, например, на ладони. Выполнять операции пользователь может с закрытыми глазами.

Развитие простоты и наглядности интерфейсов особенно быстрыми темпами идет в отраслях ПО, охватывающих наибольшее количество неклассифицируемых пользователей – **индустрия компьютерных игр и индустрия мобильных приложений**. К сожалению, использование опыта этих ключевых с точки зрения развития интерфейсов областей, происходит в области ИС для бизнеса с очень большим запозданием.

ИКР. Выполнение операций в системе происходит САМО без усилий со стороны пользователя по его желанию.

ИКР1. Пользователь выполняет операции в системе самым естественным для себя образом, с помощью навыков, обретенных им в детстве безотносительно к работе в ИС. Например – движением рук, ног, глаз и т.д.

П3. Технические средства ИС

Материалов по эволюции технических средств в свободном доступе гораздо больше, чем материалов по эволюции других частей ИС. Поколения ЭВМ проходят в любом курсе информатики, знания о них доступны большинству школьников.

Вероятно, такая популярность именно этой ветви развития связана с тем огромным эффектом, который имеют технические изобретения на процессы развития ИС.

Рассмотрим краткую историю развития технических средств, с точки зрения типов базовых элементов (подробнее см. [2]).

Доисторические времена	Пальцы, однотипные предметы
V век до нашей эры	Абак, счеты
Середина 17-го века по начало 20-го века	Развитие механических арифмометров
Весь 19-й век – по начало 20-го века	Развитие механических вычислительных машин, действующих по программе. Машина Бэббиджа Счетно-перфорационные машины Холлерита – основателя компании, в будущем ставшей IBM
30-40-е годы 20-го века	Развитие вычислительных машин на базе электромагнитных реле. Одна из самых известных моделей – ENIAC. Скорость вычислений – до 100 кГц. Недостатки – низкая скорость переключений реле, высокое энергопотребление, огромные размеры
40-50-е годы	Эра ламповых ЭВМ. Первое поколение ЭВМ. Лампы позволили существенно увеличить скорость переключений, снизить энергопотребление и сложность конструкции.
50-60-е годы	Насыщение роста ламповых ЭВМ. Лампа заменяется транзистором. Переход на микроуровень – серьезный скачок в развитии, увеличение скоростей, объемов данных и т.д. Второе поколение ЭВМ.
60-е гг. – по наше время	Эпоха интегральных схем, отказ от проводов в большинстве узлов. Третье поколение ЭВМ. На микроуровень перешли связи между элементами. Возникновение сетей передачи данных.
70-е гг. – по наше время	Развитие персональных компьютеров и сетей.
90 – е гг. по наше время	Развитие Интернет
2000-е годы	Возникновение облачных технологий, датацентров и т.д.
Начало 2000-х годов	Насыщение развития интегральных схем. Транзисторы приближаются по величине к молекулам, пропадает p-n переход.

Ведущие производители технических средств для ЭВМ осознают необходимость перехода к следующему типу вычислительных элементов, к новому поколению ЭВМ.

Компания IBM, стоявшая у истоков практически всех поколений ЭВМ, в середине 2000-х годов объявила о серии экспериментов по созданию квантового компьютера. Большинство инженеров понимают, что бесконечно уменьшающийся транзистор станет молекулой, в которой уже нельзя пренебрегать «вредными» квантовыми эффектами. Создатели квантового компьютера решили обратить вред в пользу и использовать квантовые эффекты для скачкообразного повышения производительности вычислительных машин.

Квантовый компьютер должен был использовать совершенно новую парадигму вычислений. Большой популярностью пользуется работа Шора, в которой предложен алгоритм разложения любого числа на простые множители за полиномиальное от разрядности числа время. В классических вычислениях такая задача считается NP-полной.

К сожалению, создание квантового компьютера для практически полезных задач до сих пор остается под большим вопросом. Компьютер, созданный IBM, стоил 18 миллиардов долларов, и позволял разложить на множители число 15 по алгоритму Шора. Более современные исследования позволяют смотреть на перспективы квантовых вычислений чуть более оптимистично, но быстрой замены транзистору в этом направлении скорее всего искать не стоит.

Исследования по созданию компьютеров на биологических принципах (ДНК-компьютеры) менее известны, и пока так же не дали обнадеживающих результатов.

Организация транзисторов в виде нейронной сети показала практическую полезность в решении ряда задач, но она не решает основные противоречия, возникающие в современном поколении ЭВМ:

ЕСЛИ транзистор уменьшать, ТО скорость операций и тактовая частота увеличиваются, НО транзистор уже нельзя уменьшать, так как исчезает p-n переход.

Помимо скорости, одна из важнейших характеристик вычислений – их энергоемкость. Энергоемкие вычисления создают большое количество

проблем – потребление энергии, необходимость отвода тепла. Развитие элементной базы последовательно снимало эту проблему – каждое новое поколение позволяло достичь существенно меньшего потребление энергии при вычислениях, но компьютеры каждого поколения, пользуясь предоставленным преимуществом, наращивали элементную базу и тем самым снова увеличивали энергопотребление. Современные суперкомпьютеры более требовательны к энергетическим ресурсам, чем ENIAC, работавший на электромагнитных реле.

Ключевое противоречие: ЕСЛИ изобретение уменьшает требование ЭВМ к потребляемой энергии, ТО ЭВМ начинают расти и потреблять прежнее и даже большее количество энергии.

П4. Алгоритмы информационный систем

Под анализом развития алгоритмов для ИС мы будем понимать следующие направления анализа:

1. Развитие программной архитектуры.
2. Развитие технологий программирования.
3. Развитие парадигмы вычислений с точки зрения теории алгоритмов и теории сложности.

Развитие программной архитектуры

1. В памяти машины – данные, программу выполняет оператор.
2. В памяти машины и программа и данные.
3. Данные выносятся в базу данных.
4. Управление данными отделяется от взаимодействия с пользователем, двухзвенная архитектура.
5. Между управлением данными (СУБД) и работой с пользователем (интерфейс) возникает модуль логики вычислений (сервер приложений) – трехзвенная архитектура.
6. Разделение логически независимых компонентов в двухзвенной и трехзвенной архитектуре позволяет физически разнести их в рамках одной сети или Интернет, добавлять однотипные компоненты независимо друг от друга.

7. Дробление приложений по функциональному признаку, концепция сервисно-ориентированной архитектуры.
8. Дробление логически неделимых компонентов – образование кластеров, развитие облачных вычислений.

Общая логика развития – элементы системы дробятся, динамизируются и усложняются, усложняются связи между ними.

Развитие технологий программирования

1. Программа записана в виде алгоритма действий оператора.
2. Программа вводится в машину в виде потактовых команд – управляющих сигналов для всех вычислителей.
3. Отдельные часто используемые наборы управляющих сигналов запоминаются в ПЗУ машины в виде микропрограмм, реализующих команды процессора. Возникает понятие программной архитектуры процессора, определяемой системой команд.
   ЕСЛИ система команд простая (команд мало – RISC процессор), ТО для нее просто реализовать оптимизации в компиляторе с языка высокого уровня, НО для выполнения определенных действий потребуется больше команд.
   ЕСЛИ система команд сложная (CISC процессор), ТО некоторые операции выражаются короткими наборами команд, НО существенно усложняется процесс оптимизации кода в компиляторе. Споры о преимуществах подходов к организации архитектуры идут давно (например, противостояние Sun и Intel).
4. Для более удобного процесса написания последовательности команд программы, процессов управления памятью и т.д. появляются программы – ассемблеры, линкеры, загрузчики, редакторы текста. Так, программы начинают помогать писать программы.
5. Появляются операционные системы – программы, управляющие поведением программ, предоставляющие им доступ к ресурсам машины.
6. Появляются языки высокого уровня (ЯВУ) и компиляторы с них. ЯВУ позволяют ускорить процесс разработки программ, сделав его более естественным и автоматизировав многие операции.

7. Возникает парадигма объектно-ориентированного программирования. Программа овеществляется – в ней создаются объекты, для которых описывается поведение. Объекты взаимодействуют друг с другом, с пользователями, с данными. В результате этого взаимодействия решаются задачи ИС.

8. Инструментарий программиста пополняется – развиваются компиляторы, дополняются новыми возможностями операционные системы, появляются отладчики программ, библиотеки типовых функций и т.д.

9. Возникают среды разработки (IDE), объединяющие в себе все необходимые ресурсы для программирования – удобный текстовый редактор с подсветкой операторов, отладка программы с отслеживанием ее работы прямо по тексту, компиляция прямо из IDE и т.д.

10. Возникают среды проектирования программ, которые на основании унифицированного описания задачи порождают код программы.

11. Программа обретает вид продукта – помимо рабочего кода к ней добавляются документация, красивая коробка, процесс продаж, процесс внедрения на предприятие.

12. Программа сама становится платформой разработки – часть разработки (настройки) передается пользователям, с целью адаптации программы для решения конкретных бизнес-задач

Таким образом, процесс перехода от стационарной системы к динамичной уже произошел – пользователям предлагается самостоятельно настраивать программу, изменяя ее свойства (параметры).

Согласно законам развития технических систем, следующий этап – самоорганизация, система сама настраивается под задачу.

Развитие парадигмы вычислений

Развитие парадигм вычислений в процессах развития ИС фактически отсутствует. Практически все современные компьютеры с точки зрения вычислений являются эквивалентами машины Тьюринга.

Известные парадигмы вычислений:

1. Итеративная (классический компьютер)
2. Декларативная (генетическое программирование)
3. Функциональная (квантовый компьютер)

В итеративной модели вычислений существенно понятие сложности алгоритма, так как результат в ней достигается путем последовательного выполнения шагов алгоритма. Развитие итеративной модели шло двумя путями – оптимизация итеративных алгоритмов и ускорение процесса выполнения шагов (повышения тактовой частоты). Повышение тактовой частоты достигло своего предела, а теория сложности уперлась в сложные теоретические задачи (P != NP?, NP-полнота некоторых задач и т.д.). Ряд практически важных задач оказался принадлежащим к классу NP-полных задач, что вызывает пессимистические прогнозы относительно разрешимости этих задач при росте объемов входных данных. **Итеративная модель вычислений находится на верхнем пределе своего существования**.

В отличие от итеративной, декларативная модель не описывает последовательность шагов, приводящую к результату. В декларативной модели описываются условия на результат, который должен быть найден в процессе эволюции некоторой системы. Так, появление человека можно считать промежуточным результатом работы некой огромной декларативной машины с принципом действия, основанном на генетическом отборе.

Серьезных исследований относительно перспектив этой парадигмы автору работы найти не удалось. Некоторый успех имел язык программирования PROLOG, организованный по декларативному принципу. Одной из проблем этого языка стал тот факт, что для его выполнения использовалась итеративная модель – вычислительная машина, и следовательно возможные вычислительные преимущества терялись на процессе моделирования.

Декларативные языки прочно связаны в сознании общественности с искусственным интеллектом, отношение к которому в научных кругах колеблется от прохладного до агрессивного («искусственным интеллектом занимаются те, кому естественного не хватает»).

Есть ряд работ, в которых в качестве декларативного компьютера используется набор пробирок с образцами генетического материала, который в питательной среде начинает развиваться. Тема ДНК-компьютеров так же пока не получила широкого распространения и известности.

Судьба парадигмы функционального программирования несколько более насыщена событиями, чем случай декларативной парадигмы. Функциональное программирование основано на Лямбда-исчислении, и предполагает, что результат вычислений является результатом вычисления суперпозиции набора функций. Вместо того, чтобы последовательно менять результат (как это делается в итеративных программах), можно раскрыть суперпозицию (построить результирующую функцию), и применить эту функцию к входным данным (получить результат). Существует большое количество языков программирования, организованных по функциональному принципу, однако все они, как и декларативные языки, выполняются на итеративных машинах, и не дают выигрыша в производительности. Основное их преимущество – высокая наглядность и простота программ, реализующих выполнение некоторого класса операций (как правило, связанных с вложенными рекурсиями и обходом деревьев).

Операции с данными в квантовом компьютере осуществляются путем применения линейных операторов на многомерном линейном пространстве над полем комплексных чисел. Операторы можно объединять в суперпозиции (если в их последовательности нет измерений), что близко парадигме функционального программирования.

Подходы к программированию имеют, по всей видимости, следующие корни:

Итеративный – опыт человека, животных.

Функциональный – математика, функциональный анализ.

Декларативный – биология.