**Содержание:**

# ВВЕДЕНИЕ

Эффективное управление предприятием в современных условиях невозможно без использования компьютерных технологий. Правильный выбор программного продукта и фирмы-разработчика - это первый и определяющий этап автоматизации бухгалтерского учета. В настоящее время проблема выбора информационной системы (ИС) из специфической задачи превращается в стандартную процедуру. В этом смысле российские предприятия сильно уступают зарубежным конкурентам. Иностранные предприятия, как правило, имеют опыт модернизации и внедрения не одного поколения ИС. В развитых западных странах происходит смена уже четвертого поколения ИС. На российских предприятиях зачастую используют системы первого или второго поколения.

Руководители многих российских предприятий имеют слабое представление о современных компьютерных интегрированных системах и предпочитают содержать большой штат собственных программистов, которые разрабатывают индивидуальные программы для решения стандартных управленческих задач.

Процедура принятия решения о выборе наиболее эффективной компьютерной системы управления нова для большинства отечественных руководителей, а ее последствия во многом будут оказывать значительное влияние на предприятие в течение нескольких лет. Т.к. применение интегрированной ИС, которая отвечала бы требованиям предприятия (масштабу, специфике бизнеса и т.д.), позволила бы руководителю минимизировать издержки и повысить оперативность управления предприятием в целом.

Целью данной курсовой работы является анализ и оценка средств реализации объектно-ориентированного подхода к проектированию экономической информационной системы

Этапы решения поставленной цели:

- Изучить понятие ИС.

- Изучить методы проектирования ИС.

- Изучить основные понятия объектно-ориентированного подхода, объектно-ориентированного подхода

- Изучить программные средства, реализующими объектно-ориентированного подход

# Глава 1. Проектирование информационных систем.

## 1.1 Основные определения

Информационная on система взаимосвязанная on совокупность on средств, методов on и персонала, on используемых для on хранения, обработки on и on выдачи информации on в интересах on достижения поставленной on цели.

Экономическая on информационная система on (ЭИС) - это on совокупности внутренних on и внешних on потоков прямой on и обратной on информационной on связи экономического on объекта, методов, on средств, специалистов, on участвующих on в процессе on обработки информации on и выработке on управленческих on решений [Смирнова on Г.Н., on Сорокин on А.А., on Тельнов on Ю.Ф. on Проектирование on экономических информационных on систем. - М: on Финансы и on статистика, 2003 on с. on 122].

Автоматизированной on информационной системой on (АИС) on называется комплекс, on включающий вычислительное on и коммуникационное on оборудование, программное on обеспечение, лингвистические on средства, on информационные ресурсы, on а также on персонал, обеспечивающий on поддержку on динамической информационной on модели предметной on области для on удовлетворения информационных on потребностей on пользователей.

В on автоматизированных on ИС часть on функций управления on и on обработки данных on выполняется компьютерами, on а часть on человеком on [О.Г. on Инюшкина Проектирование on информационных систем on (на примере on методов on структурного системного on анализа) Учебное on пособие Научный on редактор Матвеева on Татьяна Анатольевна on Екатеринбург Издательство on «Форт-Диалог on Исеть» 2014 on с. 45].

# 1.2 Проектирование информационных систем

Проектирование on экономических информационных on систем on (ЭИС) - логически on сложная, трудоемкая on и длительная on работа, требующая on высокой квалификации on участвующих в on ней специалистов. on Однако до on настоящего времени on проектирование ЭИС on нередко выполняется on на интуитивном on уровне неформализованными on методами, включающими on в себя on элементы on искусства, практический on опыт, экспертные on оценки и on дорогостоящие on экспериментальные проверки on качества функционирования on ЭИС. Кроме on того, в on процессе создания on и функционирования on ЭИС информационные on потребности on пользователей постоянно on изменяются или on уточняются, что on еще более on усложняет разработку on и сопровождение on таких on систем.

Основная on доля трудозатрат on при создании on ЭИС приходится on на прикладное on программное обеспечение on (ПО) и on базы данных on (БД). Производство on ПО сегодня on - крупнейшая отрасль on мировой экономики, on в которой on занято около on трех миллионов on специалистов on (программистов, разработчиков on ПО и on т. п.). on Еще несколько on миллионов человек on напрямую зависят on от благополучия on корпоративных информационных on подразделений либо on от производителей on ПО, таких, on как корпорации on Microsoft и on IBM on [О.Г. Инюшкина on Проектирование информационных on систем (на on примере on методов структурного on системного анализа) on Учебное пособие on Научный on редактор Матвеева on Татьяна Анатольевна on Екатеринбург Издательство on «Форт-Диалог Исеть» on 2014 с. on 67].

Рассмотрим on по порядку on эти характеристики. on На переднем on плане on первые два on пункта. Они on представляют собой on наиболее важное on отличие on от информационных on систем, функционирующих on в закрытых on сетях. Мы on не on имеем возможности on хранить и on обрабатывать on какую-либо on информацию на on стороне клиента. on Все должно on выполнятся на on сервере. При on разработке on информационной системы on с клиентским on программным обеспечением on можно было on бы on хранить часть on пользовательской информации on и обрабатывать on ее на on стороне клиента [Смирнова on Г.Н., on Сорокин on А.А., on Тельнов on Ю.Ф. Проектирование on экономических информационных on систем: Учебник on /; Под ред. on Ю.Ф. on Тельнова. - М: on Финансы on и статистика, on 2003 с. on 23]. Такая on возможность позволила on бы нам on разгрузить сервер on и трафик on сети. Например, on в случае on анализа посетителей on веб-сайтов, on мы хранили on бы основные on объемы информации

on у клиентов, on а на on сервере - on лишь общедоступные on статистические отчеты, on выжимки и on сравнительные on показатели с on другими клиентами on [В.В. on Мухортов, on В.Ю. on Рылов on ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ on ПРОГРАММИРОВАНИЕ, АНАЛИЗ on И on ДИЗАЙН Методическое on пособие Новосибирск on 2002 с. on 113]. Но on мы не on имеем on такой возможности, on поэтому надо on тратить большие on деньги на on накопители on жестких дисков on и вычислительные on мощности серверов. on Многопользовательский on доступ и on разграничение доступа on являются общими on требованиями для on всех информационных on систем. Важным on критерием является on ограничение по on объему передаваемой on информации. На on сервере может on быть канал on с on большой пропускной on способность, но on по этому on каналу идет on информация от on множества клиентов. on В свою on очередь, у on пользователя информация on идет только on для on него, но on очень часто on пользователи сидят on на плохих on каналах, например, on на on модемном соединении, on или же on просто, в on силу удаленности on и большого on количества шлюзов on между клиентом on и сервером, on скорость передачи on информации on очень медленная. on В связи on с тем, on что в on сети Интернет on находится огромное on количество людей, on среди которых on есть и on злоумышленники, то on необходимо on предъявлять повышенные on требования к on безопасности. И on наконец, переносимость. on Конечно, эта on особенность не on столь важна, on но допустим, on вам потребовалось on открыть зеркало on сайта на on другом континенте. on Принципиально надо on решить on две проблемы. on Во-первых, on настройка серверной on платформы и on вашего программного on обеспечения для on функционирования вашей on информационной системы. on Во-вторых, on перевод системы on на другой on язык. На on другом on континенте может on просто не on оказаться ни on требуемой вашей on информационной on системой платформы, on ни специалистов, on которые бы on могли все on это установить, on настроить и on поддерживать. Например, on будет другая on разновидность Unix. on Все on эти характерные on особенности, в on основном, и on определяют стадию on проектирования.

## 1.3 Основные принципы построения ЭИС

Методологические on принципы:

1. Системный on подход. Каждое on явление рассматривается on во взаимосвязи on с другими. on Система сосредотачивает on внимание на on объекте как on на

едином on целом, а on не на on отдельных его on частях.

Этапы on формирования системы: on определение целей on системы, on определение требований on к системе; on определение функциональных on подсистем on ИС, структуры on и задач on в общей on системе управления; on выявление и on анализ on связи между on подсистемами; Установление on порядка функционирования on всей on системы в on целом и on ее динамики; on синтез интегрированной on системы [Смирнова on Г.Н., Сорокин on А.А., on Тельнов on Ю.Ф. on Проектирование on экономических информационных on систем: Учебник on /; Под ред. on Ю.Ф. on Тельнова. - М: on Финансы on и статистика, on 2003 с. on 155].

2. Принцип on решения новых on задач - не on просто использовать on ЭВМ on для традиционных on методов, но on и перестраивать on эти методы on в соответствии on с on теми возможностями on которыми располагает on ЭВМ. Выявить on и решать on задачи, которые on не решаются on в виду on их on сложности.

3. Принцип on первого руководителя on разработка и on внедрение on ЭИС производятся on под непосредственным on руководством первого on руководителя, иначе on система ориентируется on на рутинные on проблемы on [О.Г. on Инюшкина Проектирование on информационных систем on (на примере on методов on структурного системного on анализа) Учебное on пособие Научный on редактор Матвеева on Татьяна Анатольевна on Екатеринбург Издательство on «Форт-Диалог on Исеть» on 2014 с. on 144].

Разработчик on обязан стремиться on к тому, on чтобы on предлагаемые on им проектные on решения подходили on к возможно on более широкому on кругу on заказчиков.

5. Принцип on развития исходя on из перспектив on развития on объектов автоматизации on ЭИС должна on создаваться с on учетом возможности on пополнения on и обновления on функций и on состава ЭИС on без нарушения on ее on функционирования.

6. Принцип on совместимости при on создании система on должны on быть реализованы on информационные интерфейсы, on благодаря которым on она on может взаимодействовать on с другими on системами в on соответствии с on установленными правилами.

7. Принцип on модульности в on построения программного on и информационного on обеспечения ЭИС, on то есть on ЭИС, строится on из on набора функционально on

независимых блоков on [В.В. on Мухортов, on В.Ю. on Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ on ПРОГРАММИРОВАНИЕ, АНАЛИЗ on И ДИЗАЙН on Методическое пособие on Новосибирск 2002 on с. on 44].

8. Принцип on разработки сверху on - вниз проектируемая on система рассматривается on как древовидная on структура, составленная on из on отдельных модулей.

9. Принцип on стандартизации при on создании ЭИС on должны on быть рационально on применены типовые on унифицированные и on стандартизованные элементы, on проектные решения, on ППП.

10. Принцип on эффективности заключается on в достижении on рационального соотношения on между затратами on и целевыми on эффектами, включая on конечные результаты on автоматизации on [О.Г. Инюшкина on Проектирование on информационных систем on (на примере on методов структурного on системного анализа) on Учебное пособие on Научный редактор on Матвеева Татьяна on Анатольевна on Екатеринбург Издательство on «Форт-Диалог on Исеть» 2014 on с. 122].

11. Принцип on единой информационной on базы что – on бы on исходная информация on один раз on воспринятая и on введенная в on ЭВМ могла on быть использована on многократно.

## 1.4 Подходы к проектированию экономических систем

Существует on следующие подходы on к проектированию on экономических информационных on систем

1. Традиционный on подход, основанный on на системном on анализе предметной on области и on последовательного проектирования on системы. on Такие подходы on реализуются в on «водопадной модели». on Необходимость on типизации проектных on решений обуславливается on следующим on [О.Г. on Инюшкина Проектирование on информационных систем on (на примере on методов on структурного системного on анализа) Учебное on пособие Научный on редактор Матвеева on Татьяна Анатольевна on Екатеринбург Издательство on «Форт-Диалог on Исеть» 2014 on с. on 89]:

1) при on внедрении типовой on системы существенно on снижаются on затраты на on проектирование;

2) при on индивидуальном проектировании on трудно обеспечить on должный научно-технический on уровень on разработки.

Для on разработки и on внедрения традиционного on проектирования on ЭИС существует on целый ряд on объективных on предпосылок:

1) управление on предприятием осуществляется on на основе on единых положений;

2) структура on системы управления on на всех on предприятиях одинакова on и зависит on только от on размера on предприятия;

3) технические on средства ЭИС on стандартизированы;

В on основе типового on проектирования лежит первоначальная классификация экономических объектов по их on важнейшим параметрам on [Смирнова on Г.Н., Сорокин on А.А., on Тельнов on Ю.Ф. on Проектирование on экономических информационных on систем: Учебник on /; Под ред. on Ю.Ф. on Тельнова. - on М: Финансы on и статистика, on 2003 с. on 75]. Затем on создание типовых on схем и on решений, on внедрение которых on в дальнейшем on на конкретном on предприятии сводится on к привязке on их on в условиях on данного предприятия. on Декомпозиция функциональных on компонентов ЭИС on является основой on технологии типового on проектирования. on Типовое проектирование on предполагает разбиение on ЭИС на on отдельные составляющие on и создание on для каждого on из законченного on проектного решения, on которое затем on с некоторыми on модификациями будет on использоваться при on проектировании on ЭИС [Мацяшек on Л.А. Анализ on требований и on проектирование систем. on Разработка информационных on систем с on использованием UMLM./ on Л.А on Мацяшек: Издательский on дом «Вильямс», on 2002. - с. on 166]. В on соответствии on с этим принципом on система хорошо on структурирована, удовлетворяет on следующим требованиям:

1) каждый on уровень иерархии on обозрим и on понятен без on детального знания on нижних on уровней:

2) минимизированы on связи между on элементами на on одном on уровне иерархии;

3) не on должно быть on связей между on элементами через on 1 on уровень;

4) элемент on более высшего on уровня должен on вызывать on элемент следующего on уровня как on единое целое, on передавая ему on входную on информацию;

5) элемент on следующего уровня on после окончания on своей on работы возвращает on вызывающему его on элементу управление on и результаты on работы.

В on соответствии с on перечисленными требованиями on для on компонентов функционирования on структуры можно on установить следующую on структуру on по уровням on [В. В. on Мухортов, В. Ю. Рылов ОБЪЕКТНО- ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 99]:

1) элементы on автоматизированных on подсистем;

2) элементы on автоматизированных on функций;

Комплексы on последних уровней on - это элементы on машинных процедур on и элементы on процедур, реализуемых on персоналом управления. on В on основе разработки on типовых проектов on лежат такие on принципы как on унификация on и стандартизация. on Под унификацией on понимается реализация on при on разработке программ on принципа единообразия on в методах, on средствах и on содержании, on и формах on представления информации. on [В. В. Мухортов, В. Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 55].

Структурный on подход в on полном объеме, on основанный на on функциональном моделировании on систем (модели on SADT диаграммы on типа IDEF0) on и on на моделировании on данных и on их отношений on (модели ERD). on Такой on подход позволяет on автоматизировать разработку on информационной системы on до on более глубокого on уровня с on описанием структур on данных и on их отношений on (уровень датологического on проектирования информационной on системы).

Объектно-ориентированный on подход, позволяющий on описывать on все функции, on связи, события, on входные и on выходные данные on и процессы on их преобразований on с позиций on объектно-ориентированного on подхода on на специальном on языке UML [Буч Г., on Рамбо Д., on Якобсон И. on Язык on UML.

Руководство on пользователя Издательский on дом «Вильямс», on 2010. с. 23]. При объектном on методе проектирования on в качестве on типизируемого on элемента выступает on система управления on объектом в on целом т.е. on создается типовой on проект ЭИС, on обобщенного объекта on из некоторого on класса объектов on управления.

**Вывод:** on В результате on работы над on первой главой on ознакомились on с определением on ЭИС, методами on проектирования ИС, on достоинствами on и недостатками on различных методов on проектирования.

# Глава 2. Объектно-ориентированный подход

## 2.1 Сущность объектно-ориентированного подхода

Структурный анализ (Structured Analysis, SA) и структурное проектирование (Structured Design, SD) – результат появившегося в 1970-х структурного программирования, развивался из классического системного анализа. Сравнительно позже появились и стали невероятно популярны объектно-ориентированные языки. По мере нарастания их популярности была разработана методология помощи программисту в разработке приложений с использованием объектно-ориентированных языков. Эта методология стала известна как объектно-ориентированный анализ и проектирование (object-oriented analysis and design, OOAD) [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 245]. OOAD – это подход к инженерии ПО, моделирующий систему как группу взаимодействующих объектов. Объектно-ориентированный анализ (Objectoriented analysis, OOA) использует методы объектного моделирования для анализа функциональных требований к системе. Объектно-ориентированное проектирование, ООП (Object-oriented design, OOD) разрабатывает аналитические модели для создания спецификаций реализации (например, ТЗ). Концептуальной основой ООП является объектная модель, которая строится с учетом принципов абстрагирования, инкапсуляции, модульности, иерархии, типизации, параллелизма, устойчивости. Основными понятиями объектно-ориентированного подхода являются объект и класс. Объект – представляет собой определенную сущность, соответствующую значимому предмету или явлению предметной области, характеризуется классом, состоянием (state (data elements)) и поведением

[Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 134]. Для этих взаимодействующих (collaborating) между собой объектов можно создать различные модели, характеризующие статическую структуру, динамическое поведение и развертывание в действии (run-time deployment). Класс – это множество объектов, связанных общностью структуры и поведения. Следующую группу важных понятий объектного подхода составляют полиморфизм (способность класса принадлежать более чем одному типу) и наследование (построение новых классов, на основе существующих с возможностью добавления или переопределения данных и методов) [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 178]. На сегодняшний день существует более тридцати объектно-ориентированных методов проектирования (например, IDEF4 – Object-Oriented Design – методология ООП, позволяющая отображать структуру объектов и принципы их взаимодействия) с множеством различных нотаций представления объектных моделей.

Преимущества объектно-ориентированных методологий:

- упрощение и ускорение программной реализации системы по сравнению со структурными методологиями;

- повторное использование кода в других проектах, благодаря независимости объектов и инкапсуляции, что сокращает стоимость проектирования, программирования и проверки; повторное использование кода может способствовать улучшению качества последующих проектов [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 134]; отсутствие разделения между фазами анализа и разработки обеспечивает взаимодействие с пользователями до самого конца проекта; аналитики и программисты не связаны ограничениями внедрения системы, поэтому могут формулировать проекты, которые будут соответствовать различным средам исполнения; программное обеспечение устойчиво к изменениям, что обеспечивает более высокий уровень уверенности в его корректности, способствуя снижению рисков при разработке сложных систем; те преимущества, которые представляет объектно-ориентированное программирование по сравнению со структурным: при разработке объектов со сложным взаимодействием, аналитик думает на ином уровне детализации, чем это возможно в структурном коде, т.е. об атрибутах объекта; стандартизация объектов увеличивает степень понимания проекта [Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем: Учебник/; Под ред. Ю.Ф. Тельнова. - М: Финансы и

статистика, 2003 с. 345]

Недостатки объектно-ориентированных методологий:

- изначальная модель слишком упрощена для того, чтобы быть адекватной;

- чрезмерная фокусировка на коде;

- не так много внимания уделяется командной работе, как в структурных методологиях;

- определение всех необходимых для системы классов и объектов – это не такая, на самом деле, простая задача;

- попытка сочетания объектного программирования с анализом различных функций системы; однако, эти функциональные методы совершенно не соответствуют OOAD [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 179];

- преувеличение значимости и универсальности объектной методологии, когда, фактически, другой подход мог бы подойти лучше для анализа и разработки системы в зависимости от конкретных обстоятельств;

- требует новый вид управления проектами, который включает различные типы анализа, отличные от традиционного функционального подхода декомпозиции.

- функциональное описание системы в UML основано на сценариях использования, которые подходят для документирования требований, не основанных на взаимодействии с системой (таких как алгоритм или математические требования) или нефункциональных требований (такие как платформа, производительность, синхронизация, безопасность); следование шаблонам не гарантирует качества сценариев, качество зависит только от навыков создателя сценария;

- объектный подход к моделированию данных при том, что большинство ИС используют реляционные модели; [О.Г. Инюшкина Проектирование информационных систем (на примере методов структурного системного анализа) Учебное пособие Научный редактор Матвеева Татьяна Анатольевна Екатеринбург Издательство «Форт-Диалог Исеть» 2014 с. 256]. ИС чаще разрабатываются через комбинацию объектно-ориентированных языков программирования и реляционных баз данных.

В процессе объектно-ориентированного анализа основное внимание уделяется определению и описанию объектов в терминах предметной области. Основная идея объектно-ориентированного анализа и проектирования состоит в рассмотрении предметной области и логического решения задачи с точки зрения объектов [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002].

В процессе проектирования определяются логические программные объекты, которые будут реализованы средствами объектно-ориентированного языка программирования.

В процессе конструирования обеспечивается реализация основных компонентов средствами объектно-ориентированных языков программирования.

Объектно-ориентированный подход использует объектную декомпозицию. При этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира.

## 2.2 Понятия класс и объект, основные свойства объектной модели

Понятие объект впервые было использовано около 30 лет назад в технических средствах при попытках отойти от традиционной архитектуры фон Неймана и преодолеть барьер между высоким уровнем программных абстракций и низким уровнем абстрагирования на уровне компьютеров. С объектно-ориентированной архитектурой также тесно связаны объектно-ориентированные операционные системы. Однако наиболее значительный вклад в объектный подход был внесен объектными и объектно-ориентированными языками программирования: Simula, Smalltalk, C++, Object Pascal [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002]. На объектный подход оказали влияние также развивавшиеся достаточно независимо методы моделирования баз данных, в особенности подход сущность – связь [Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем: Учебник/; Под ред. Ю.Ф. Тельнова. - М: Финансы и статистика, 2003 с. 166].

Концептуальной основой объектно-ориентированного подхода является объектная модель. Основными ее элементами являются:

- абстрагирование;

- инкапсуляция;

- модульность;

- иерархия.

Кроме основных, имеются еще три дополнительных элемента, не являющихся в отличие от основных строго обязательными:

- типизация;

- параллелизм;

- устойчивость.

Абстрагирование – это выделение существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют его концептуальные границы относительно дальнейшего рассмотрения и анализа [Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UMLM./ Л.А Мацяшек: Издательский дом «Вильямс», 2002. с. 349]. Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности его поведения от деталей их реализации. Выбор правильного набора абстракций для заданной предметной области представляет собой главную задачу объектно-ориентированного проектирования.

Принцип абстрагирования реализуется в ряде методов при решении задач с использованием объектной модели. В литературе можно встретить разные определения и расшифровки того, что понимается под термином абстрагирование. Хорошей является такая абстракция, которая подчеркивает детали, существенные для рассмотрения и использования, и опускает те, которые на данный момент несущественны». Если объединить эти точки зрения, получим определение абстракции [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 168]: Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки

зрения наблюдателя. Абстрагирование концентрирует внимание на внешних характеристиках объекта и позволяет отделить наиболее существенные особенности его поведения от менее существенных. Граница между существенными и несущественными с точки зрения разрабатываемой программной системы особенностями поведения объекта называется барьером абстракции. Последний определяется исходя из принципа минимизации связей, согласно которому интерфейс объекта должен описывать только существенные аспекты поведения [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 155]. Так же следует соблюдать принцип наименьшего удивления. Следуя ему абстракция должна охватывать только поведение описываемого ей объекта, и, соответственно, не привносит сюрпризов и побочных эффектов, лежащих вне сферы ее применимости. Выделение полного и достаточного набора абстракций при решении задачи с применением объектного подхода представляет собой главную задачу объектно-ориентированного проектирования. Во время разработки программной системы могут появляться абстракции разных категорий, начиная с объектов, которые почти точно соответствуют реалиям предметной области, и кончая объектами, целесообразность использования которых сомнительна [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 233].

Инкапсуляция – это процесс отделения друг от друга отдельных элементов объекта, определяющих его устройство и поведение. Инкапсуляция служит для того, чтобы изолировать интерфейс объекта, отражающий его внешнее поведение, от внутренней реализации объекта. Объектный подход предполагает, что собственные ресурсы, которыми могут манипулировать только методы самого класса, скрыты от внешней среды. Абстрагирование и инкапсуляция являются взаимодополняющими операциями: абстрагирование фокусирует внимание на внешних особенностях объекта, а инкапсуляция (или, иначе, ограничение доступа) не позволяет объектам пользователям различать внутреннее устройство объекта [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 235].

Модульность – это свойство системы, связанное с возможностью ее декомпозиции на ряд внутренне связных, но слабо связанных между собой модулей. Инкапсуляция и модульность создают барьеры между абстракциями.

Процесс выделения ключевых абстракций при решении поставленной задачи в большей степени относится к объектно-ориентированному проектированию.

Инкапсуляция же, напротив, является главенствующим принципом объектно-ориентированного программирования [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 211]. Интерфейс отражает внешнее поведение абстракции, специфицируя поведение всех объектов данного класса. Внутренняя реализация описывает представление этой абстракции и механизмы достижения желаемого поведения объекта. Принцип разделения интерфейса и реализации соответствует сути вещей: в интерфейсной части собрано все, что касается взаимодействия данного объекта с другими объектами, а реализация скрывает от других объектов все детали, не имеющие отношения к процессу взаимодействия объектов [Крэг Ларман Применение UML 2.0 и шаблонов проектирования. 3-е издание. Издательство: Вильямс с. 188]. Инкапсуляция — это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации. Современные объектно-ориентированные языки, такие как C++ и Java, имеют развитые средства поддержки принципа инкапсуляции. Эти средства выражаются в наличии механизмов управления доступом к методам и данным объекта

Иерархия – это ранжированная или упорядоченная система абстракций, расположение их по уровням. Основными видами иерархических структур применительно к сложным системам являются структура классов (иерархия по номенклатуре) и структура объектов (иерархия по составу). Примерами иерархии классов являются простое и множественное наследование (один класс использует структурную или функциональную часть соответственно одного или нескольких других классов), а иерархии объектов – агрегация [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО- ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 137].

Инкапсуляция и модульность позволяют в значительной степени упростить описание и процесс разработки системы абстракций, но зачастую, еще лучше позволяет бороться со сложностью принцип иерархии. Инкапсуляция убирает из поля зрения внутренние содержание абстракций, модульность объединяет логически связанные абстракции в группы, а иерархия позволяет разделить абстракций на уровни, т.е. образует из абстракций иерархическую структуру.

Иерархия — это упорядочение абстракций путем расположения их по уровням. В объектно-ориентированных системах используются два вида иерархических структур: структуры классов (иерархические отношения «is a») и структуры объектов (отношения вида «part of») [ Буч Г., Рамбо Д., Якобсон И. Язык UML.

Руководство пользователя Издательский дом «Вильямс», 2010. с. 233.]. Отношения вида «is a» реализуются в объектно-ориентированных языках с помощью наследования или генерализации. Наследование означает такое отношение между классами (отношение родитель/потомок), когда один класс заимствует, а также расширяет и/или специализирует (уточняет) структуру и функциональный контракт одного или нескольких родительских классов. Иными словами, наследование создает такую иерархию абстракций, в которой подклассы наследуют строение и функциональность от одного или нескольких суперклассов. В наследственной иерархии общая часть структуры и поведения сосредоточена в наиболее общем суперклассе.

Суперклассы, при этом, отражают наиболее общие, а подклассы -более специализированные абстракции, в которых члены суперкласса могут быть дополнены, модифицированы и даже скрыты. При одиночном наследовании класс может иметь только одного родителя, но реализовывать несколько интерфейсов. При этом интерфейсы могут наследовать от нескольких родительских интерфейсов. При множественном наследовании у класса может быть несколько суперклассов [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 177.].

Типизация Понятие типа пришло в ООП из теории абстрактных типов данных [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 355]. Несмотря на то, что в некоторых языках существуют отличия между типом и классом, в современных объектно-ориентированных языках (в нашем случае C++ и Java) эти понятия неразделимы. Мы будем подразумевать под типами как примитивные типы (char, int, float), так и языковые средства абстракций, определяемых пользователем (классы, структуры и интерфейсы) [Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем: Учебник/; Под ред. Ю.Ф. Тельнова. - М: Финансы и статистика, 2003 с. 333]. Типизация, как и инкапсуляция, больше относится к области объектно-ориентированного программирования, нежели к области объектно-ориентированного проектирования.

Типизация – это ограничение, накладываемое на класс объектов и препятствующее взаимозаменяемости различных классов (или сильно сужающее ее возможность). Типизация позволяет защититься от использования объектов одного класса вместо другого или по крайней мере управлять таким использованием.

Центральное место в типизации занимают механизмы согласования типов. Конкретный язык программирования может иметь сильный или слабый механизм типизации, или даже не иметь его вовсе. Сильно типизированные языки непреклонно следуют правилам использования типов [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 244]. Так, в языках C++ и Java нельзя вызвать метод у объекта, если он не зарегистрирован в его классе, суперклассе или интерфейсе. Причем нарушение такого рода будет обнаружено уже на стадии компиляции. В Smalltalk, напротив, во время исполнения любое сообщение может быть послано любому объекту, при этом возникнет ошибочная ситуация, если объект не в состоянии обработать это сообщение (т.е. в его классе или суперклассе нет соответствующего метода).. Сильная типизация заставляет разработчика соблюдать правила использования абстракций, поэтому она приносит неоценимую помощь в больших проектах. Однако у нее есть теневая сторона, заключающаяся в необходимости перекомпиляции всех подклассов, а также классов, использующих данный класс при внесении изменений в его интерфейс [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 211].

Параллелизм – свойство объектов находиться в активном или пассивном состоянии и различать активные и пассивные объекты между собой.

Виртуальный полиморфизм — одно из самых мощных и эффективных средств объектно-ориентированного программирования, отличающее его от программирования на основе абстрактных типов данных. Таким образом, C++ и Java являются сильно типизированными языками с динамическим связыванием [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 231]. Параллелизм в объектно-ориентированном программировании, как и другие принципы, возник не на пустом месте, а явился результатом привнесения объектной идеи в теорию параллельных вычислений. Необходимость в разработке теории параллельных вычислений воз- никла давно. Задачи автоматизации очень часто требуют одновременной обработки нескольких событий. При решении задач, связанных с большой вычислительной трудоемкостью, очень часто бывает недостаточно мощности одного процессора и приходится искать решение основанное на распараллеливании вычислений на многопроцессорных системах. Существует очень много классов задач, где возможность использования параллелизма может сильно улучшить характеристики разрабатываемой информационной системы. Фундаментальным понятием и единицей действия в теории параллельных

вычислений является поток управления. Традиционно, многопоточность бывает двух видов — тяжелая (основанная на процессах в операционной системе) и легкая (основанная на потоках в рамках одного процесса).

Потоки управления при тяжелой многопоточности существуют каждый в своем отдельном адресном пространстве в рамках операционной системы (с каждым процессом ассоциируется ровно один по- ток управления) [О.Г. Инюшкина Проектирование информационных систем (на примере методов структурного системного анализа) Учебное пособие Научный редактор Матвеева Татьяна Анатольевна Екатеринбург Издательство «Форт-Диалог Исеть» 2014 с. 23]. Переключение контекстов исполняемых потоков при операциях межпроцессного взаимодействия в таких системах, как правило, сопряжено с большими накладными расходами. Параллелизм дает возможность объектам действовать одновременно. Параллелизм — это свойство, отличающее активные объекты от пассивных [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 175]. Как только в систему введен параллелизм, сразу возникает вопрос о синхронизации активных объектов друг с другом и последовательными пассивными объектами. Параллелизм может обеспечиваться как средствами языка (Java, Ada, Smalltalk), так и специально написанными библиотеками, которые используются при написании параллельной системы с использованием языков, не имеющих встроенной поддержки этого принципа (C++). Следует, однако, отметить, что даже если язык имеет встроенную поддержку параллелизма, все равно, необходимо учитывать устройство и особенности организации многопоточности в конкретных операционных системах, под управлением которых планируется работа разрабатываемой программы[В. В. Мухортов, В. Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 87].

Устойчивость – свойство объекта существовать во времени (вне зависимости от процесса, породившего данный объект) и/или в пространстве (при перемещении объекта из адресного пространства, в котором он был создан).

Любой объект или данные в программной системе существуют во времени и пространстве (памяти ЭВМ) [О.Г. Инюшкина Проектирование информационных систем (на примере методов структурного системного анализа) Учебное пособие Научный редактор Матвеева Татьяна Анатольевна Екатеринбург Издательство «Форт-Диалог Исеть» 2014 с. 34]. Одни объекты существуют только как промежуточные результаты вычисления выражения, другие могут вообще пережить программу, оставаясь сохраненными в базе данных. Этот спектр

подразделяется на следующие уровни [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 76]:

- Промежуточные результаты вычислений выражений.

- Локальные переменные и объекты в блоках, а также, при вызове процедур и функций (как правило эти данные существуют на стеке).

- Статические переменные классов, а также, глобальные переменные и объекты в динамической памяти.

- Данные, сохраняемые между сеансами выполнения программы.

- Данные, сохраняемые при переходе на другую версию программы.

- Данные, переживающие программу.

**Вывод:** В результате работы над второй главой ознакомились с сущностью объектно-ориентированного подхода, основными понятиями, такими, как класс и объект, изучили основные свойства объектной модели.

# Глава 3. Программными средства, реализующие объектно-ориентированного подход

## 3.1 Языки ООП

Многие современные языки специально созданы для облегчения объектно-ориентированного программирования. Однако следует отметить, что можно применять техники ООП и для необъектно-ориентированного языка и наоборот, применение объектно-ориентированного языка вовсе не означает, что код автоматически становится объектно-ориентированным.

Современный объектно-ориентированный язык предлагает, как правило, следующий обязательный набор синтаксических средств:

Объявление классов с полями (данными –членами класса) и методами (функциями – членами класса).

Механизм расширения класса (наследования) - порождение нового класса от существующего с автоматическим включением всех особенностей реализации класса-предка в состав класса-потомка. Большинство ООП-языков поддерживают только единичное наследование [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 123].

Средства защиты внутренней структуры классов от несанкционированного использования извне. Обычно это модификаторы доступа к полям и методам, типа public, private, обычно также protected, иногда некоторые другие.

Полиморфные переменные и параметры функций (методов), позволяющие присваивать одной и той же переменной экземпляры различных классов.

Полиморфное поведение экземпляров классов за счёт использования виртуальных методов. В некоторых ООП-языках все методы классов являются виртуальными.

Видимо, минимальным традиционным объектно-ориентированным языком можно считать язык Оберон, который не содержит никаких других объектных средств, кроме выше перечисленных. Но большинство языков добавляют к указанному минимальному набору те или иные дополнительные средства. В их числе:

- Конструкторы, деструкторы, финализаторы.

- Свойства(аксессоры).

- Индексаторы.

- Интерфейсы – как альтернатива множественному наследованию.

- Переопределение операторов для классов.

Часть языков (иногда называемых «чисто объектными») целиком построена вокруг объектных средств - в них любые данные (возможно, за небольшим числом исключений в виде встроенных скалярных типов данных) являются объектами, любой код – методом какого-либо класса и невозможно написать программу, в которой не использовались бы объекты. Примеры подобных языков – Java или Ruby. Другие языки (иногда используется термин «гибридные») включают ООП-подсистему в исходно процедурный язык. В них существует возможность программировать, не обращаясь к объектным средствам. Классические примеры - C++ и Delphi [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ

ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 211].

Большинство существующих методов объектно-ориентированного анализа и проектирования (ОО-АП) включают как язык моделирования, так и описание процесса моделирования. Язык моделирования – это нотация (в основном графическая), которая используется методом для описания проектов [Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов/ Д. Розенберг., К. Скотт Издательский дом «Вильямс», 2012. с. 326]. Нотация представляет собой совокупность графических объектов, которые используются в моделях; она является синтаксисом языка моделирования. Например, нотация диаграммы классов определяет, каким образом представляются такие элементы и понятия, как класс, ассоциация и множественность. Процесс – это описание шагов, которые необходимо выполнить при разработке проекта.

## 3.2 Унифицированный язык моделирования UML

Унифицированный язык моделирования UML (Unified Modeling Language) – это преемник того поколения методов ООАП, которые появились в конце 1980-х и начале 1990-х гг. Создание UML фактически началось в конце 1994 г., когда Гради Буч и Джеймс Рамбо начали работу по объединению методов Booch и OMT (Object Modeling Technique) под эгидой компании Rational Software. К концу 1995 г. они создали первую спецификацию объединенного метода, названного ими Unified Method, версия 0.8. Тогда же, в 1995 г., к ним присоединился создатель метола OOSE (Object-Oriented Software) Ивар Якобсон [Крэг Ларман Применение UML 2.0 и шаблонов проектирования. 3-е издание. Издательство: Вильямс]. Таким образом, UML является прямым объединением и унификацией методов Буча, Рамбо и Якобсона, однако дополняет их новыми возможностями. Главными в разработке UML были следующие цели:

- предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий разрабатывать осмысленные модели и обмениваться ими;

- предусмотреть механизмы расширяемости и специализации для расширения базовых концепций;

- обеспечить независимость от конкретных языков программирования и процессов разработки [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 125].

- обеспечить формальную основу для понимания этого языка моделирования (язык должен быть одновременно точным и доступным для понимания, без лишнего формализма);

- стимулировать рост рынка объектно-ориентированных инструментальных средств;

- интегрировать лучший практический опыт.

Язык UML находится в процессе стандартизации, проводимом OMG (Object Management Group) – организацией по стандартизации в области объектно-ориентированных методов и технологий, и в настоящее время принят в качестве стандартного языка моделирования и получил широкую поддержку в индустрии ПО. Язык UML принят на вооружение практически всеми крупнейшими компаниями – производителями ПО [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 122.]. Кроме того, практически все мировые производители CASE-средств, помимо Rational Software (Rational Rose) поддерживают UML в своих продуктах (Paradigm Plus, System Architec, Microsoft Visual Modeler, Delphi, PowerBuilder и др.).

Создатели UML представляют его как язык для определения, представления, проектирования и документирования программных систем, организационно-экономических, технических и др. UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов. Стандарт UML версии 1.1, принятый OMG в 1997 г., предлагает следующий набор диаграмм для моделирования [Крэг Ларман Применение UML 2.0 и шаблонов проектирования. 3-е издание]:

- диаграммы вариантов использования – для моделирования бизнес-процессов организации (требований к системе);

диаграммы классов – для моделирования статической структуры классов системы и связей между ними;

- диаграммы поведения системы;

- диаграммы взаимодействия – для моделирования процесса обмена сообщениями между объектами. Существуют два вида диаграмм взаимодействия: диаграммы

последовательности; кооперативные диаграммы.

- диаграммы состояний – для моделирования поведения объектов системы при переходе из одного состояния в другое;

- диаграммы деятельностей – для моделирования поведения системы в рамках различных вариантов использования или моделирования деятельностей;

- диаграммы реализации: диаграммы компонентов – для моделирования иерархии компонентов (подсистем) системы; диаграммы размещения – для моделирования физической архитектуры, системы.

У большинства людей понятие проектирование ассоциируется со структурным проектированием по методу сверху вниз на основе функциональной декомпозиции, согласно которой вся система в целом представляется как одна большая функция и разбивается на подфункции, которые, в свою очередь, разбиваются на подфункции и т.д. Эти функции подобны вариантам использования в объектно-ориентированной системе, которые соответствуют действиям, выполняемым системой в целом [Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов/ Д. Розенберг., К. Скотт Издательский дом «Вильямс», 2012. – с 333].

В объектно-ориентированном подходе основная категория объектной модели – класс – объединяет в себе на элементарном уровне как данные, так и операции, которые над ними выполняются (методы). Именно с этой точки зрения изменения, связанные с переходом от структурного к объектно-ориентированному подходу, являются наиболее заметными. Разделение процессов и данных преодолено, однако остается проблема преодоления сложности системы, которая решается путем использования механизма компонентов [Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UMLM./ Л.А Мацяшек: Издательский дом «Вильямс», 2002. - с 128].

## 3.3 Объектно-ориентированное проектирование и его связь с другими методами проектирования

Данные по сравнению с процессами являются более стабильной и относительно редко изменяющейся частью системы. Отсюда следует главное достоинство объектно-ориентированного подхода, которое Гради Буч сформулировал

следующим образом: объектно-ориентированные системы более открыты и легче поддаются внесению изменений, поскольку их конструкция базируется на устойчивых формах. Это дает возможность системе развиваться постепенно и не приводит к полной ее переработке даже в случае существенных изменений исходных требований [Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. с. 147]. Буч отмечает также ряд следующих преимуществ объектно-ориентированного подхода:

Объектная декомпозиция дает возможность создавать программные системы меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования не только программ, но и проектов, что в конце концов ведет к созданию среды разработки и переходу к сборочному созданию ПО. Системы зачастую получаются более компактными, чем их структурные эквиваленты, что означает не только уменьшение объема программного кода, но и удешевление проекта за счет использования предыдущих разработок.

Объектная декомпозиция уменьшает риск создания сложных систем ПО, так как она предполагает эволюционный путь развития системы на базе относительно небольших подсистем. Процесс интеграции системы растягивается на все время разработки, а не превращается в единовременное событие [Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов/ Д. Розенберг., К. Скотт Издательский дом «Вильямс», 2012.].

Объектная модель вполне естественна, поскольку в первую очередь ориентирована на человеческое восприятие мира, а не на компьютерную реализацию.

Объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования.

К недостаткам объектно-ориентированного подхода относятся некоторое снижение производительности функционирования ПО и высокие начальные затраты [Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UMLM./ Л.А Мацяшек: Издательский дом «Вильямс», 2002. с. 169]. Объектная декомпозиция существенно отличается от функциональной, поэтому переход на новую технологию связан как с преодолением психологических трудностей, так и дополнительными финансовыми

затратами. Безусловно, объектно-ориентированная модель наиболее адекватно отражает реальный мир, представляющий собой совокупность взаимодействующих (посредством обмена сообщениями) объектов. Но на практике в настоящий момент продолжается формирование стандарта языка объектно-ориентированного моделирования UML, и количество CASE-средств, поддерживающих объектно-ориентированный подход, невелико по сравнению с поддерживающими структурный подход.

Кроме того, диаграммы, отражающие специфику объектного подхода (диаграммы классов и т.п.), гораздо менее наглядны и плохо понимаемы непрофессионалами. Поэтому одна из главных целей внедрения CASE-технологии, а именно снабжение всех участников проекта (в том числе и заказчика) общим языком для передачи понимания, обеспечивается на сегодняшний день только структурными методами [Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов/ Д. Розенберг., К. Скотт Издательский дом «Вильямс», 2012. с. 53].

При переходе от структурного подхода к объектному, как при всякой смене технологии, необходимо вкладывать деньги в приобретение новых инструментальных средств. Здесь следует учесть и расходы на обучение (овладение методом, инструментальными средствами и языком программирования). Для некоторых организаций эти обстоятельства могут стать серьезными препятствиями.

Объектно-ориентированный подход не дает немедленной отдачи. Эффект от его применения начинает сказываться после разработки двух-трех проектов и накопления повторно используемых компонентов, отражающих типовые проектные решения в данной области [Крэг Ларман Применение UML 2.0 и шаблонов проектирования. 3-е издание]. Переход организации на объектно-ориентированную технологию – это смена мировоззрения, а не просто изучение новых CASE-средств и языков программирования.

Основой взаимосвязи между структурным и объектно-ориентированным подходами является общность ряда категорий и понятий обоих подходов (процесс и вариант использования, сущность и класс и др.). Эта взаимосвязь может проявляться в различных формах. Так, одним из возможных подходов является использование структурного анализа как основы для объектно-ориентированного проектирования [В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002 с. 211]. Такой подход

целесообразен ввиду широкого распространения CASE-средств, поддерживающих структурный анализ. Его можно считать слишком прагматическим, но в некоторых ситуациях иной подход невозможен. При этом структурный анализ следует прекращать, как только диаграммы потоков данных начнут отражать не только деятельность организации (предметную область), а и систему ПО.

После выполнения структурного анализа и построения диаграмм потоков данных вместе со структурами данных и другими продуктами анализа можно различными способами приступить к определению классов и объектов. Так, если взять какую-либо отдельную диаграмму, то кандидатами в объекты могут быть внешние сущности и накопители данных, а кандидатами в классы – потоки данных [Крэг Ларман Применение UML 2.0 и шаблонов проектирования. 3-е издание. с. 231].

Другой формой проявления взаимосвязи можно считать интеграцию объектной и реляционной технологий. Реляционные СУБД являются на сегодняшний день основным средством реализации крупномасштабных баз данных и хранилищ данных. Причины этого очевидны: реляционная технология используется достаточно долго, освоена огромным количеством пользователей и разработчиков, стала промышленным стандартом, в нее вложены значительные средства и создано множество корпоративных БД в самых различных отраслях, реляционная модель проста и имеет строгое математическое основание; существует большое разнообразие промышленных средств проектирования, реализации и эксплуатации реляционных БД. Вследствие этого реляционные БД в основном используются для хранения и поиска объектов в так называемых объектно-реляционных системах.

Объектно-ориентированное проектирование имеет точки соприкосновения с реляционным проектированием. Например, как было отмечено выше, классы в объектной модели могут некоторым образом соответствовать сущностям [Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UMLM./ Л.А Мацяшек: Издательский дом «Вильямс», 2002. с 344]. Одним из примеров практической реализации взаимосвязи между структурным и объектно-ориентированным подходом является программный интерфейс (мост) между структурным CASE- средством Silverrun и объектно-ориентированным CASE-средством Rational Rose, разработанный российской компанией Аргуссофт. Этот мост создает диаграммы классов Rational Rose на основе RDM-модели (Relation Data Model – реляционная модель данных) Silverrun и наоборот. Аналогичные интерфейсы существуют также между CASE-средствами ERwin.[ Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем: Учебник/; Под ред. Ю.Ф. Тельнова. - М: Финансы и

статистика, 2003]

**Вывод:** В результате работы над третьей главой ознакомились с программными средствами, реализующими объектно-ориентированного подход и проанализировали их достоинства и недостатки.

# ЗАКЛЮЧЕНИЕ

Эффективное управление предприятием в современных условиях невозможно без использования компьютерных технологий. Правильный выбор программного продукта и фирмы-разработчика - это первый и определяющий этап автоматизации бухгалтерского учета. В настоящее время проблема выбора информационной системы (ИС) из специфической задачи превращается в стандартную процедуру. В этом смысле российские предприятия сильно уступают зарубежным конкурентам. Иностранные предприятия, как правило, имеют опыт модернизации и внедрения не одного поколения ИС. В развитых западных странах происходит смена уже четвертого поколения ИС. На российских предприятиях зачастую используют системы первого или второго поколения.

Руководители многих российских предприятий имеют слабое представление о современных компьютерных интегрированных системах и предпочитают содержать большой штат собственных программистов, которые разрабатывают индивидуальные программы для решения стандартных управленческих задач.

Процедура принятия решения о выборе наиболее эффективной компьютерной системы управления нова для большинства отечественных руководителей, а ее последствия во многом будут оказывать значительное влияние на предприятие в течение нескольких лет. Т.к. применение интегрированной ИС, которая отвечала бы требованиям предприятия (масштабу, специфике бизнеса и т.д.), позволила бы руководителю минимизировать издержки и повысить оперативность управления предприятием в целом.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UMLM./ Л.А Мацяшек: Издательский дом «Вильямс», 2002. - 432 с

2. Крэг Ларман Применение UML 2.0 и шаблонов проектирования. 3-е издание / Издательство «Форт-Диалог Исеть» 2015.

3. Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов/ Д. Розенберг., К. Скотт Издательский дом «Вильямс», 2012. - 532 с.

4. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя Издательский дом «Вильямс», 2010. -235 с.

5. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем: Учебник/; Под ред. Ю.Ф. Тельнова. - М: Финансы и статистика, 2003.

6. О.Г. Инюшкина Проектирование информационных систем (на примере методов структурного системного анализа) Учебное пособие Научный редактор Матвеева Татьяна Анатольевна Екатеринбург Издательство «Форт-Диалог Исеть» 2014.

7. В.В. Мухортов, В.Ю. Рылов ОБЪЕКТНО- ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, АНАЛИЗ И ДИЗАЙН Методическое пособие Новосибирск 2002.