Как написать хороший диплом по программированию

Оглавление

Введение 2

Лекция 1. О научности дипломов 5

Лекция 2. О видах дипломов по программированию 19

Лекция 3. Управляющая структура текста диплома 32

Лекция 4. Название и введение 47

Лекция 5. Обзор 54

Лекция 6. Архитектура и реализация 59

Лекция 7. Практическая полезность 66

Лекция 8. Про объекты, встроенные в текст. Часть I 74

Лекция 9. Про объекты, встроенные в текст. Часть II 83

Лекция 10. О русском языке 93

Лекция 11. А теперь пишем текст 104

Заключение 110

Список литературы 111

Введение

В последние годы в России нарастает интерес к профессии программиста. Всё больше и больше студентов поступают на ИТ-специальности в университеты, и самих этих специальностей становится всё больше. Например, в самое последнее время стал популярен и востребован на практике искусственный интеллект и машинное обучение. В то же время дифференцируются давно существовавшие специализации, такие как информационные системы, языки и системы программирования, Web-программирования и др. Наконец, многие молодые люди интересуются научными исследованиями в области компьютерных наук.

Соответственно, всё больше студентов пишут и защищают дипломы. В области программирования дипломы всегда штучные – их не скачать из Интернета, не выбрать из списка заранее заготовленных тем. Как правило, студенту самому предлагается найти интересную тему, и после этого целиком и полностью самостоятельно и выполнить работу, и написать текст. И если с практической частью работы – разработкой некоторой системы или выполнением исследований – дела обстоят нормально, то вот с написанием текстов дела обстоят плачевно. Особенно это касается бакалаврских дипломов. Мне не раз приходилось слышать, что понять бакалаврский диплом, имея на руках только его текст, практически, невозможно. Причем, чем сложнее и интереснее работа – тем меньше шансов. Я и сам с этим часто сталкивался – и чтобы разобраться в том, что же человек сделал, требовался обстоятельный разговор с самим дипломником, а то и с его научным руководителем.

Заинтересовавшись этой проблемой, я выделил следующие обстоятельства.

Во-первых, программирование является молодой образовательной дисциплиной, в отличие, например, от медицины или классических инженерных специальностей. Образовательные программы, отдельные курсу, требования к дипломным работам – всё это постоянно меняется и дополняется. Поэтому каноны, правила и шаблоны просто не успевают сформироваться, хотя, точнее будет сказать, что не успевает сформироваться и образовательная ИТ-культура в целом – университеты не успевают за индустрией.

Во-вторых, в области программирования традиционно плохо с текстами и документацией. В программных проектах документации постоянно не хватает, а качество той, которая существует, оставляет желать лучшего. Так было во времена Ф.Брукса (легендарный руководитель проекта по разработке OS 360 в компании IBM), т.е. в конце 60-х годов, так обстоят дела и в настоящее время. В качестве примера можно обратиться к документации ядра открытой операционной системы Linux. Плохое состояние этой документации является известной проблемой, от которой страдают многие люди, поскольку ядро Linux является предметом многочисленных модификаций при разработке встроенных приложений.

В-третьих, весь компьютерный мир пользуется английским языком. И, соответственно, в русский язык проникает большое количество англицизмов. Ну с файлами, фреймами и фреймворками уже нужно смириться. Однако постоянно приходится слышать про некие ранеры, линеры, листнеры, воркеры и т.п. Все это, естественно, так или иначе проникает в русскоязычную документацию и студенческие тексты.

В-четвёртых, переход образовательной системы в России на трёхступенчатую модель образования, когда вместо специалистов, обучающихся пять лет, у нас появились бакалавры, обучающиеся четыре года. При этом существенно понизил качество дипломных работ и текстов дипломов по программированию первой ступени обучения, которая часто оказывается и единственной. Почему разница всего в одни год (бакалавры учатся четыре года, специалисты – пять лет,) оказала такой эффект, оставим за пределами нашего внимания, остановившись лишь на самом этом факте, с которым согласны многие университетские преподаватели, с кем я обсуждал эту тему.

Таким образом, программисты, защищающие бакалаврские дипломы, нуждаются в существенной поддержке при написании текстов дипломов. Причём эту поддержку не нужно путать с бюрократизацией процесса сдачи диплома – формальных процедур (утверждение темы, многочисленные предзащиты и пр.), большого количества различных документов и записей в информационной системе университета и т.д. Всё это нацелено на преодоление плохого качества дипломных работ студентов – обычно тексты дипломов и выступления студентов совершенно непонятны, и поэтому, например, студенты отдельно пишут технические задания к своим работам, а члены аттестационной комиссии внимательно читают отзывы научного руководителя и рецензента, в которых те, порой, значительно лучше самого студента, ёмко и кратко объясняют суть работы. Однако эти процедуры носят массовый характер, ответственным за это преподавателям почему-то постоянно не хватает времени, и в целом студенты склонны воспринимать это не как помощь, а как дополнительную нагрузку.

Впрочем, задачу подготовки хороших текстов дипломов можно не ставить, как, кстати, и делают многие вузы в России. Зачем же её ставить?

Прежде всего для того, чтобы хоть немного научить студентов писать технические тексты – отчеты, технические задания и пр. Этот навык является полезным для индустрии, где выпускникам предстоит работать. Также написание текстов развивает студентов, предлагая им задуматься над исходными посылками их дипломов, контекстом работы, ясно, кратко и содержательно сформулировать результаты и т.д. Хороший текст является естественным дополнением к хорошей работе. Также многие университеты и кафедры выкладывают тексты дипломов, и по этим текстам можно отслеживать их деятельность (а какие еще следы могут остаться в университете от студенческих дипломов?). Соответственно, самим студентам может оказаться небезразличным, что можно найти в Интернете под их фамилией, и хорошо бы, чтобы за это не нужно было краснеть. Также многие работодатели при приёме на работу молодого специалиста, могут найти и посмотреть эти тексты (я, например, так делаю, когда меня в разных компаниях приглашают участвовать в интервью). При этом, конечно, не хватает времени прочитать диплом кандидата целиком, но если с ходу в глаза бросается большое количество ошибок русского языка, а также не удаётся быстро понять, что сделал человек, то впечатление о кандидате может сложиться не очень хорошее. Наконец, мне кажется, что совершенно нормально стремиться хорошо говорить и писать по-русски, в том числе в рамках профессиональных коммуникаций. Потому что мы – русские люди, у нас есть наш русский язык, используя и развивая свой язык, мы остаёмся русскими людьми…

В течение последних пятнадцати лет я вёл специальный семинар для выпускников кафедры системного программирования Санкт-Петербургского государственного университета, посвящённый текстам дипломных работ. Семинар проходит в последнем семестре и предназначен для помощи выпускникам в написании текстов их дипломов. Поскольку выпускников на нашей кафедре оказывается довольно много – 30-50 человек ежегодно, – то я быстро пришёл к неутешительному выводу: я не могу вдумчиво прочитать каждый диплом и разобрать в его тематике, да ещё и выдать конструктивные замечания, а также добиться, чтобы все они были исправлены. При этом я обратил внимание на то, что когда текст уже написан, то активно его редактировать очень тяжело. Да и сам студент в этой ситуации очень неохотно идёт на радикальные правки: тексты писать ему объективно трудно, и если он уже потратил время и силы, написал большой труд, а вот теперь выясняется, что этот текст нужно существенно переделывать – ну уж нет, не выйдет, уважаемый профессор!

И тогда я сделал акцент на проектирование текстов. Мы вместе с *каждым* студентом создаём хорошую структуру текста его диплома (это существенно шире, чем план/оглавление), а потом студент самостоятельно переходит к написанию своего текста в соответствии с этой структурой. И чтобы минимизировать индивидуальные особенности каждого диплома и каждого студента, я создал метод проектирования текстов дипломов по

программированию. Который я и излагаю в данном учебном пособии, адресуясь, в первую очередь, к бакалаврам (но магистры также успешно пользуются этим методом). Переход к проектированию и наличие метода позволили решить вопрос индивидуальной работы с каждым выпускником кафедры — теперь я успеваю это сделать!

В рамках данного метода за годы моих экспериментов было написано несколько сотен дипломов. Я также опубликовал несколько статей про этот метод в российских журналах и зарубежных конференциях. По моему мнению, качество текстов дипломов на нашей кафедре существенно выросло за эти годы. Я пробовал провести всеобъемлющее исследование, чтобы объективизировать этот факт, но у меня не хватило на это времени. Однако подтверждением моей правоты стали отзывы самих студентов. Один из них, став уже директором собственной компании, признался мне, что работа над дипломом под моим руководством и, в частности, написание текста диплома, стало для него одним из главных событий при обучении в университете...

Финальной точкой развития метода является данное учебное пособие. Я написал его, потому что хочу, чтобы студенты, кроме прослушивания моих лекций и выполнения заданий на семинаре еще могли что-то почитать на эту тему на досуге. Тем более, что многие из них, занятые на производстве, пропускают занятия. А презентации лекций не очень хорошо подходят для ознакомления с данным предметом, так как служат, скорее, для напоминания того материала, который человек уже прослушал.

Впрочем, наличие пособия не означает, что метод больше не будет развиваться – всё живое меняется и преобразуется! Но основные идеи, а также отдельные приёмы и замечания, накопившиеся у меня за годы, нуждались в систематизации. И я надеюсь, что это пособие окажется полезным широкому кругу студентов-бакалавров, занимающихся написанием дипломов по программированию! И, возможно, не только по программированию.

Дмитрий Кознов,
Санкт-Петербург

Лекция 1. О научности дипломов

В лекции рассказывается о научном каноне создания текста дипломной работы, который включает в себя определение актуальности тематики, наличие постановки задачи, разграничение в тексте работы своего и «чужого», хорошо сформулированные достигнутые результаты, а также их индивидуальность, новизна и доказуемость (обоснованность).

Ты можешь ГОСТ поменять? ГОСТ поменять?

М. Жванецкий. Уж очень я смешной человек.

Высшие научные заведения (университеты) предназначены для духовной жизни людей, которых досуг или внутреннее стремление приводят к науке и исследованиям

К.В. Гумбольдт. О внутренней и внешней организации высших научных заведений в Берлине.

С душою прямо геттингенской,
Красавец, в полном цвете лет,
Поклонник Канта и поэт.
Он из Германии туманной
Привез учености плоды

А.С. Пушкин. Евгений Онегин.

О научном каноне в высшем образовании. Каноны в нашей жизни играют громадную роль. Писанные, написанные, в виде ГОСТов или стандартов организации ISO (International Organization for Standardization), в виде правил как должно поступать/вести себя и как недолжно. Порой этим канонам очень много лет, и часто люди не задумываются, откуда они взялись и даже что они вообще существуют. Всегда действуем таким образом – и всё тут. Но осваивая новую область, сферу деятельности, очень полезно понимать и осознанно учиться пользоваться существующими в ней канонами.

Дипломная работы выполняется в университете и создается по вполне определённому *канону*, несмотря на то, что обычно преподаватели про него явно не говорят, а напирают на некоторую естественность всех этих постановок задач, результатов и пр. А этот канон совсем не очевиден студентам, что я постоянно и наблюдаю. Неочевиден он также и специалистам из индустрии, часто занимающимися руководством дипломными работами студентов, которые проходят стажировку или уже работают к компании. Причём, эти руководители от индустрии сами заканчивали когда-то университеты...

Что же это за канон? Речь идёт о каноне научной работы. ваш диплом должен ему следовать, даже если сама работа и не является научной (о том, какие бывают дипломы по программированию, помимо научных, будет подробно рассказано в следующей лекции). Перед тем, как рассмотреть этот канон, остановимся немного на истории развития университетов.

Очерк истории университетов. Университеты в Европе появились в XI веке в Италии (Болонский университет) на основе монастырских школ, которые хранили остатки культуры со времён Римской империи. К XIII веку университеты в различных городах Европы окончательно оформились в отдельные цеховые корпорации студентов и преподавателей, эталоном здесь выступил Парижский университет. Университеты основывались, преимущественно, королями и владетельными сеньорами (кюрфюстами, пфальцграфами, герцогами и так далее), а для их открытия, вплоть до эпохи Реформации, требовалось разрешение римского папы.

Университеты имели свои особые права (привилегии), свой суд и свою экономику, были замкнутыми и независимыми сообществами, имея особые, непростые отношения с городами, в которых они находились.

Средневековый университет имел три основных (высших) факультета – богословский, юридический и медицинский (именно в таком порядке) и предназначался для выпуска сертифицированных специалистов, востребованных в обществе – прежде всего священников, а также юристов и медиков. Как правило, в университете имелся четвертый факультет – философский, – который выполнял подготовительную функцию, обучая начальным предметам; он был самым многочисленным, после его окончания можно было поступать на высшие факультеты, но большинство студентов, освоив азы грамотности, покидали стены университета. Следует отметить, что университеты, вплоть до XVII века, сочетали в себе начальное и высшее образование.

Одним из изобретений средневековых университетов, которое сохранилось до наших дней, являются лекции. В то время не было учебников (книгопечатанье было изобретено существенно позднее, во второй половине XV века), рукописные книги были большой редкостью и дорого стоили, и единственной формой передачи знаний были устные лекции, которые студенты записывали под диктовку преподавателей. Лекционная форма была пересмотрена в XVII веке, когда уже отпала необходимость записывать под диктовку, и лекции стали превращаться в специальные обучающие события, создаваемые харизмой профессора и способствующие передаче глубин познаваемого предмета – ведь ещё с древних времен известно, что настоящая мудрость передается именно при непосредственном общении. В нашем современном мире, при развитии вещательных технологий и сетевого образования, лекционный формат также претерпевает изменения.

Средневековая структура университетских факультетов была очень устойчива и, практически, в неизменном виде сохранилась вплоть до XIX века. Относительно научных изысканий следует отметить, что хоть они и велись в средневековых университетах, однако, они не были приоритетными. Программа обучения была зафиксирована и менялась крайне незначительно. Основной задачей университетов был вовсе не поиск нового знания, а сохранение и передача уже имеющегося, существующего. Есть мнение, что даже Мартин Лютер, лидер Реформации, учёный монах и книгочтей, исходно, во многом, хотел всего лишь убрать наслоившиеся за века искажения в знаний и для этого предлагал обратиться к первоисточникам и не планировал столь глобальных и глубоких последствий от своих идей. Поиск нового знания и занятия наукой стали неотъемлемой частью университетов лишь после реформы Гумбольдта.

Учились в средневековых университетах, преимущественно, представители низших и средних классов. Одним из основных преимуществ университетов был особый статус, который давался его членам в обществе (собственный суд и различные привилегии), а также перспективы выгодной работы после получения ученой степени. Всё это было не нужно дворянству, и оно обратилось к университетам лишь начиная с XVII–XVIII веков, когда престиж науки и научного образования стал высок, а в обществе укрепилась идея эпохи Просвещения о воспитании людей посредством науки. Так, например, в среде английского дворянства XIX–XX вв. было принято обучать юношей в Оксфорде и Кембридже.

В XV–XVII вв. в Европе окончательно оформилась наука в современном понимании. Коперник опроверг геоцентрическую картину мира, которой придерживались в средние века, географические открытия существенно расширили кругозор европейцев, философские изыскания Николая Кузанского, Мишеля Монтеня, Фрэнсиса Бэкона и многих других философов определили предмет и характер научного поиска. Для консолидации деятельности учёных с XV века в Италии стали основываться академии наук (они занимались, преимущественно, гуманитарными науками), а с середины XVII века академии стали появляться в Англии, Франции и других странах Европы (естественные науки). В 1724 году была основана Петербургская академия наук в России.

В это время становления и первых шагов науки университеты не были полноценно вовлечены в научный процесс, оставаясь статичными средневековыми структурами и продолжая выполнять утилитарные функции по подготовке необходимых обществу профессионалов. Хотя, безусловно, ученые в университетах были – можно вспомнить, например, Коперника (Краковский университет), а также Исаака Ньютона, который учился и долгое время работал в Тринити колледже, в Кембридже (тяготясь, впрочем, преподаванием, и студенты разбегались с его лекций). В XVIII веке, в эпоху Просвещения, университеты подверглись сильной критике научного сообщества, и, возможно, в результате также исчезли бы также, как и многие другие институты средневековья, если бы не были модернизированы. В частности, в наполеоновской Франции, а также в тех землях, которые были присоединены к ней в результате войн, старые университеты стали закрываться — осуществлялась реформа по превращению оставшихся университетов в узкоспециализированные профессиональные школы с едиными централизованными программами обучения.

Но именно *модернизация* университетов позволила, с одной стороны сохранить их как центры культуры в меняющемся мире, а с другой стороны сделать оплотом науки и образования. В Германии была создана модель классического университета, которую связывают с именем К. В. Гумбольдта.

Карл Вильгельм Гумбольдт (1767–1835 годы) был неординарным человеком. Он происходил из прусской дворянской семьи, получил блестящее образование — сначала домашнее, а потом в университетах Франкфурта-на-Одере и Гёттингена, изучая иностранные языки (он знал более десяти языков), экономику, философию, историю и государственное управление. Гумбольдт был одним из самых всесторонне образованных людей Европы того времени, дружил с И.В. Гёте, Ф. Шиллером, И. Гердером и другими выдающимися людьми своей эпохи, а его дом в Берлине был известнейшим литературно-научным салоном. При этом Гумбольдт около тридцати лет проработал на прусской государственной службе — в суде, в области образования и науки, на дипломатической службе. В частности, он успешно представлял Пруссию на Венском Конгрессе в 1813 году, где решалась судьба Европы после наполеоновских войн, был прусским посланником при папском дворе в Риме, а также в Лондоне и в Вене. В 1819 году Гумбольдт уволился с государственной службы, не сумев примерить свои либеральные воззрения с политикой линией прусского двора. К тому же он хотел более полно посвятить себя научным изысканиям. Ещё состоя на государственной службе, он совершил экспедицию в Баскские земли (современная Испания) и написал книгу про культуру и язык этого народа. Он также изучал австронезийские языки, языки северо-американских индейцев, санскрит и др. — в целом более тридцати двух языков. Гумбольдт является одним из основателей лингвистической типологии.

А в это время Пруссия, пережившая сокрушительный разгром от Наполеона и потерявшая по Тильзитскому миру (1807 год) более половины своих земель, остро нуждалась в самоутверждении и восстановлении своего международного авторитета. Прусский король Фридрих Вильгельм III заявил, что Пруссия должна «духовными силами возместить то, что она потеряла физически». И власти Пруссии организовали реформы в

области образования и науки — реформу гимназий, университетов и Берлинской академии наук.

Соответственно, появился ряд концепций обновлённого университета за авторством Ф. Шлейермахера, Х. Штеффенса, Ф.В. Шеллига, И.Г. Фихте и др. Более того, создание новой модели университета, фактически, велось уже давно. Можно упомянуть реформу Йенского университета (немецкое герцогство Саксен-Веймар-Эйзенахское), предпринятую И.В. Гёте в конце XVIII века, а также деятельность Йенского кружка (И.В. Гёте, Ф. Шиллер, И.Г. Фихте, А. Гумбольдт и др.), в котором активно участвовал и К.В. Гумбольдт, некоторое время живший в Веймаре. Также следует упомянуть основанный в 1734 году Гёттингенский университет: ряд обстоятельств способствовали его расцвету, в частности, его основателем был ганноверский курфюрст Георг Август, который одновременно являлся королем Великобритании Георгом II, что, во многом, повлияло на привлечение в университет дворянства из разных стран, включая принцев крови, что было новшеством для средневековых университетов. В Гёттингенском университете была создана особая атмосфера и «занятия наукой соединились с благородным образом жизни просвещенного человека». Таким образом, был сильно поднят престиж науки, а сам университет стал выдающимся научным центром и оставался таким более двух веков.

Однако именно К.В. Гумбольдт создал канон классического университета, которому в итоге стала следовать вся Европа в XIX веке и влияние которого простирается также и на современные университеты. Будучи крупным администратором и государственным деятелем, а также находясь в должности директора департамента образования в министерстве внутренних дел Пруссии (что, фактически, соответствовало рангу министра образования), Гумбольдт в 1808 году убедил прусского короля Фридриха Вильгельма III открыть в Берлине новый университет. При этом в документах он именовал новый университет как «высшее научное учреждение» — отчасти в силу дипломатических соображений (выше упоминалась всеобщая критика университетов в то время), отчасти же потому, что видел в новом университете именно научное учреждение, более того, «вершину, к которой сходится всё, что делается непосредственно ради нравственного усовершенствования нации». В устройстве Берлинского университета Гумбольдт сочетал идеалистические, либеральные воззрения немецких неогуманистов и прагматичный подход крупного чиновника и администратора, восполнив недостаток реалистичности, присутствовавший в других концепциях обновления университетов, появившихся в Германии в это же время, например, такую концепцию создал и описал известный философ И. Г. Фихте. Свои идеи Гумбольдт изложил в трактате «О внутренней и внешней организации высших научных заведений в Берлине», который постигла нелёгкая судьба: во-первых, он был не дописан до конца, а во-вторых, он был издан целый век спустя по цензурным соображениям.

Перечислим основные положения университетской модели К.В. Гумбольдта.

1. Утверждение единства науки и её важнейшей роли в жизни человечества.
2. Единство преподавания/обучения и научных исследований.
3. Принцип академической свободы – как преподавания, так и обучения.
4. Тесная взаимосвязь университета с государством.

Теперь прокомментируем данные положения.

Утверждение единства науки. Это принцип, который, вроде бы, напрямую не относится к организации университета. Однако это не так, как будет показано ниже. Более того, рассматривая этот принцип, мы можем уяснить на данном примере, что если инновационно-организационные предложения опираются на идеи и концепции более высокого порядка, чем непосредственно та предметная область, на преобразование которой они направлены, то это придаёт им глубину и проникновенность, и, в конечном

счёте, приводит к их успешности и долголетию. Так модель немецкого классического университета успешно воплощалась и развивалась весь XIX век и продолжает влиять на мировые университеты и в настоящее время.

Во времена Гумбольдта наука находилась в формирующейся, творческой фазе — постоянно обнаруживались новые направления исследований и основывались соответствующие области науки, осознавались доселе невидимые связи между разными областями познания. Формировалась и утверждалась место науки в обществе: как в общественном мнении (и тут уместно вспомнить многочисленные лекции о науке, которые читал брат К. В. Гумбольдта — Александр Гумбольдт, известный путешественник, географ и естествоиспытатель, искренний энтузиаст-популяризатор науки), так в государстве, и, наконец, в образовании.

Гумбольдт считал, что наука — это не свод имеющихся, зафиксированных знаний, а нечто, становящееся, образующееся, и при этом единое по духу, концепциям и методам. «Наука всегда представляет собой проблему, еще не нашедшую своего окончательного решения». При этом данная ситуация не является временной, т.е. Гумбольдт считал, что наука никогда не будет полностью завершённым сводом знаний, наука — «это нечто, еще не полностью обретенное и никогда целиком не обретаемое». Здесь интересен захватывающий дух открытых просторов, неторeных путей и в целом — бесконечности познания.

Для истинно научного познания Гумбольдт считал важным сохранять тройственное устремление духа:

- стремление выводить все из некоего первоначального принципа;
- постоянно стремиться к идеалу;
- стремиться объединить и принцип, и идеал в единую, конкретную научную идею.

Соответственно, наука, по мнению Гумбольдта, является вершиной интеллектуальной деятельности человека. В дальнейшем, многие учёные, например, В.И. Вернадский, считали, что наука — это вершина человеческой деятельности.

Единство преподавания/обучения и научных исследований. Вовлечение студентов в науку Гумбольдт считал главной обучающей задачей университета. При этом он утверждал, что не преподаватели служат студентам, а и те и другие служат науке. Соответственно, для студентов важно не столько усвоение конкретных знаний, сколько восприятие основных научных принципов и методов. Рассуждая о взаимодействии университетов и академий, Гумбольдт считал, что и тот и другой род учреждений в равной степени сопричастен науке, более того, «движение науки в университете, где она беспрерывно вращается в большом количестве сильных, бодрых и юных умов, безусловно, стремительней и живее». Само преподавание он не считал отвлечением от науки, ибо преподаватель, по его мнению, преподаёт то самое, что изучает, озвучивая в аудитории и получая живую обратную связь. Замечу от себя, что такое возможно, когда дистанция от переднего научного края до университетских лекций незначительна — в наше время это уже не так, и в таких науках, как математика, теоретическая физика и некоторых других требуется полжизни учиться, чтобы дойти до этой переднего края, понимать, чем же там заняты учёные. При этом специфическую роль академии наук Гумбольдт видел в аудите научных достижений, пестовании отдельных областей исследований как таковых, а также в том, что академия более свободна и может осуществлять ничем неограниченный научный поиск, нежели университеты, теснее связанные с государством (в частности, кадровый вопрос в академии решается исключительно самой академией). В целом, Гумбольдт считал важным перейти от университета как средневекового цехового объединения преподавателей и студентов (а именно так обстояло дело в средние века) к «высшему научному учреждению» нового типа.

Принцип академической свободы первоначально, в средневековых европейских университетах, означал независимость университетов от городских властей и наличие собственных, особых, привилегий: у университетов были свои собственные суды, которые могли выносить вплоть до смертных приговоров, своя полиция и своя темница и т.д. Средневековые университеты также имели полное самоуправление.

Старинный принцип академической свободы Гумбольдт наполнил новым содержанием, опираясь на свои либерально-романтические воззрения. Его академическая свобода основывалась на вере в то, что глубокие духовные устремления (место осуществления коих он видел именно в университете) могут осуществляться человеком лишь свободно, без принуждения, по доброй воле и при наличии глубокой личной мотивации. Гумбольдт конкретизирует свободу университетов следующим образом.

Преподаватель, имея обязательный предмет (курс) для чтения лекций, волен самостоятельно определять его содержание, координируясь при этом с другими преподавателями.

Студенты же могут сами выбирать курсы, свободно посещать занятия и самостоятельно определять скорость своего обучения. Единственно, в конце обучения они должны сдать государственные экзамены и удовлетворить ряду иных обязательных требований (внимательный читатель поймёт, что я намекаю на дипломную работу, хотя Гумбольдт про это и не писал прямо). Данный порядок предполагал такого студента, который «душевно, нравственно и интеллектуально может быть предоставлен свободе и самостоятельности, и тогда, избегнув принуждения, он переходит не к праздности или практической жизни, но утоляет свое стремление возвыситься до той науки, которая прежде лишь как бы издали была ему показана» (Гумбольдт). Также такой студент должен быть готов к занятиям наукой, т.е. его ум должен быть натренирован и развит, а это, в свою очередь, является задачей школы (гимназии).

Ещё в 80-х–90-х годах во многих российских университетах присутствовали отдельные элементы университетских свобод студентов — было большое количество спецкурсов по выбору, имелась возможность слушать спецкурсы на параллельных кафедрах и даже на других факультетах (бесплатно и без дополнительных административных усилий). Было заведено также свободное посещение учебных занятий и возможность досрочной сдачи семестровых экзаменов. Также и преподаватели легко могли создать новый спецкурс или спецсеминар и сразу же вставить его в расписание.

Тесная взаимосвязь университета с государством. Основная роль государства по отношению к новому университету была лаконично выражена известным немецким учёным и философом начала XIX века Х. Штеффенсом: «организовывать, помогать развитию университетов, а затем — терпеть их свободу». Гумбольдт же подчёркивал, что государство должны точно исполнять свою роль именно как хранителя и обеспечителя научно-образовательных процессов, но не как их источника. В частности, государство отвечает за подбор кадров в университетах, дабы избежать скандалов и предвзятости. При этом Гумбольдт оставляет в тени прагматичный вопрос подготовки университетами кадров для государственной службы и общественной деятельности. В этом можно видеть отход от прагматичности средневековых университетов. С другой же стороны, в этом видится убеждённость в ценности науки для общества как таковой, а также в возможности людям, получившим классическое образование в университетах, работать на различных поприщах, используя на практике развитый ум и приверженность высоким духовным идеалам. Гумбольдт верил, что государство останется в выигрыше от таких людей и таких университетов.

Можно сказать, что идеи новой модели университета «носились в воздухе» в Германии конца XVIII – начала XIX вв. Основанию Берлинского университета предшествовало несколько десятилетий развития романтизма, либерализма и неогуманизма, появление и развитие новых идей и укладов в немецких университетах (Гёттингенский и Йенский

университеты). Новые идеи развития университетов, аккумулированные Гумбольдтом в единую концепцию, которая, к тому же была им реализована, соответствовали запросу интеллектуальной элиты Германии, а также, как показали последующие события, и всей Европы, коррелируя с общим подъёмом науки. Поэтому, в частности, успешное развитие Берлинского университета могло происходить и происходило без непосредственного участия Гумбольдта — последний покинул свой пост уже в 1810 году.

Берлинский университет явил собой эталон новой модели, собрав выдающихся учёных своего времени. Можно упомянуть всемирно известного философа Г. В. Ф. Гегеля, а также основателя «исторической школы» в юриспруденции Ф. К. фон Савиньи, родоначальника системного подхода к критике источников в исторических науках Л. фон Ранке, создателя современной греческой эпиграфики А. Бёка, теолога Ф. Шлейермахера, терапевта К. В. Гуфеланда и физиолога И. Мюллера, знаменитого математика П. Г. Л. Дирихле, первооткрывателя закона изоморфизма кристаллической структуры вещества в химии Э. Мичерлиха, основателя современной географии К. Риттера.

Фактически, воплощение и развитие новой модели университета происходило стихийно по всей Германии (так, в частности, базовое сочинение Гумбольдта об устройстве университетов было опубликовано из-за цензурных причин лишь в конце XIX века и не могло влиять на этот процесс). В результате немецкие университеты в XIX веке бесспорно первенствовали в Европе. Можно сказать, пользуясь терминологии известного российского историка Л.Н. Гумилёва, что в Германии в XIX веке был интеллектуальный «пассионарный взрыв». Были и другие примеры таких «взрывов» — например, в СССР в 50-е годы XX века был небывалый подъём математики. Было разрешено сразу несколько проблем Гильберта, заложены и далеко продвинуты новые направления математики, основаны и сильно расширены многочисленные математические школы в разных городах и университетах. С тех пор математика стала самой известной областью российской науки во всём мире, визитной карточкой российской науки.

Но вернёмся в Германию начала XIX века. Я хочу упомянуть ещё об одной форме обучения, помимо лекций, которая пришла в наши современные университеты — на этот раз из классических университетов Германии XIX века. Речь идёт о *семинаре*, на котором преподаватель и студенты активно занимались практической научной работой. Первоначально семинар возник у филологов, которые совместно со студентами начали заниматься комментариями античных текстов. Семинары как практические занятия также перешли в естественные науки и стали очень продуктивной формой обучения. В настоящее время семинары и практикумы по программированию, посвящённые освоению нетривиальных идей и сложных средств разработки — например, предметно-ориентированных языков и средств Eclipse для создания соответствующего инструментария (xText, EMF и др.), — наследуют традиции этих немецких семинаров. Студенты на таких семинарах привлекаются сложностью задач, возможностью узнать новое и освоить его практически, а также сделать всё это в обозримое время, под руководством опытных практиков. При подготовке таких семинаров важно умело сочетать различные факторы и адресоваться к реальным студентам, обучающимся в данное время. Сюда же относятся и начальные практикумы по программированию, когда опытный программист-педагог учит студентов программировать на различных языках. Сюда не относятся примитивные онлайн-практикумы, когда студентам выдается набор несложных задач, при решении которых отсутствует общение с преподавателем. По моим наблюдениям, на семинарах/практикумах реальное обучение происходит именно при общении преподавателя и студентов, а также при наличии атмосферы поиска, профессионализма и увлечённости.

Отметим, что политические и общественные обстоятельства также способствовали развитию новых университетов. Геополитическое унижение Пруссии переключило фокус государственных властей на науку и образование, а последующее сокрушение империи Наполеона открыло широкую дорогу для распространения этих идей по всей Европе.

Университеты Германии сыграли не последнюю роль в её объединении (на основе Пруссии), укрепляя единство национального самопознания.

Аналогичную роль в XIX веке выполняли университеты и в России, непросто способствуя развитию науки в сугубо государственных целях, как это было в XVIII веке, а выполняя также общественную функцию. Так, например, реформатор российских университетов начала XIX века министр просвещения граф С.С. Уваров считал одной из важнейших задач университета «образовывать человека наукой». При этом в России этого времени происходило становление национальной культуры, национального самосознания. В связи с этим можно упомянуть концепцию российской национальной идеи, весьма критикуемую, но весьма ёмкую, предложенную тем же графом Уваровым и по сути, просуществовавшую целое столетие — «те начала … без коих Россия не может благоденствовать, усиливаться, жить: Православная Вера, Самодержавие, Народность». Также реформировался и, можно сказать, создавался русский литературный язык, решающий вклад в который сделали Н.М. Карамзин, А.С. Пушкин и другие писатели и поэты; при этом русский язык стал вытеснять иностранные языки из разных сфер общественной жизни (например, в университетах в это время начали читать лекции именно на русском языке). Появилась общедоступная многотомная история государства российского, написанная Н.М. Карамзиным и т.д. Дальнейший взлёт и расцвет русской культуры общеизвестен. Все эти процессы были тем фоном и скрытыми силами, влиявшими и поддерживающими развитие русских университетов, и в свою очередь, университеты влияли на эти же процессы.

В качестве немаловажного фактора, бесспорно влиявшего на развитие новых университетов, была развернувшаяся в Европе со второй половиной XVIII и длившаяся весь XIX вв. промышленная революция. Однако, её связи с университетами не являются прямолинейными — относительно небольшое количество университетских (естественнонаучных) открытий попали в индустрию и хозяйственную деятельность. Тем более, что естественнонаучные факультеты стали появляться в университетах лишь с середины XIX века. Кроме того, ещё в XVII – XVIII вв. параллельно с университетами появились узкопрофессиональные школы, в которых нельзя было получить университетских степеней, но которые больше соответствовали нуждам времени в прагматичном ключе (т.е. учили реальным профессиям, а не стремились быть оплотом культуры). И далее, в рамках этой тенденции, в начале XIX века начали активно открываться технические учебные заведения, которые стали базой для подготовки инженеров и развития технологий, необходимых для нарождающейся промышленности. Так в России в 1804 году был открыт на базе горного училища Горный кадетский корпус (ныне Санкт-Петербургский горный университет), в 1809 году был учреждён Институт корпуса Инженеров путей сообщения (ныне Петербургский государственный университет путей сообщения), в 1828 году был основан Практический технологический институт (Санкт-Петербургский государственный технологический институт, Техноложка). Сходные события происходили и в Западной Европе. И только в XX веке эти технические образовательные учреждения стали также именоваться университетами и влились в общий поток высшего образования, а до той поры существовала существенная разница между университетами и техническими образовательными учреждениями.

Модель Гумбольдта в XIX веке в Европе распространилась повсеместно. Университеты переживали небывалый подъём, они внесли огромный вклад в развитие науки, их влияние на жизнь общества трудно преувеличить.

Ситуация изменилась в начале XX века. Прежде всего, к этому времени в науке сформировались специализированные научные дисциплины и требовалось готовить непросто «свободных художников», а соответствующих специалистов — для подготовки студентов уже не хватало только усвоения ими метода научного познания, но требовалось передать/усвоить реальные знания — систематизированные базовые результаты, полученные к настоящему времени в той или иной области. При этом и сами учёные всё в

меньше степени оставались универсалами, причём даже в пределах отдельной научной области. Одним из последних универсалов в рамках своей науки был Лев Ландау (начало — середина XX века), который следил за развитием всей физики целиком и заложил принцип универсальности в основание уникальной школы теоретической физики Московского физико-технического института. Также увеличилась дистанция между университетскими программами и передним краем наук, и полноценно заниматься наукой вместе со студентами становилось всё сложнее. Таким образом, естественно-научный факультет разделился на физический, биологический, химический, математический факультеты, и в области гуманитарных наук также произошла дивергенция.

Изменились и связи университетов с государством в силу усложнения задач управления, а также из-за развития технологической индустрии, нуждавшейся в специализированных, высокопрофессиональных кадрах. Процесс технологизации науки, т.е. переориентация с задач комплексного познания Мироздания на разработку прикладных методов и методик как конечной цели, сегодня лавинообразно продолжается в связи с развитием программирования и информационных технологий.

Наконец, в западном мире произошли большие социальные, культурные и политические изменения — так, в частности, идеализм, неогуманизм и либерализм уступили место позитивизму, реализму, материализму, пессимизму и политическим технологиям.

Таким образом, модель Гумбольдта, продуктивно существовавшая и развивающаяся целый век, начала активно видоизменяться.

Однако идея Гумбольдта о единстве *высшего образования и науки* оказывается актуальной и в наши дни, с той поправкой, что к этой паре добавляется еще *индустрия/бизнес* (и получается триада!). Эта идея выступает проводником научных традиций в самые разные отрасли деятельности и знаний, попавшие к настоящему времени под крыло университетов – свободные искусства, графический дизайн, менеджмент, туризм и пр. И именно поэтому структура научной работы превращается в канон для текста университетской дипломной работы независимо от области и степени реальной научности выпускной работы. Это справедливо том числе и для дипломов по программированию.

> **Важно.** ваш диплом является научной работой по форме, даже если по содержанию он не научен. Последнее обстоятельство является допустимым и нормальным: например, вы создаёте прикладную программную систему, и нет никакой необходимости делать из этого научное исследование. Однако вы должны выполнить научное представление сделанной работе.

Этим научным представлением является текст диплома. Причём неслучайно работу над дипломом именуют «написанием диплома». То есть в качестве выпускной университетской работы предполагается именно исследование или практическая работа в форме исследования. Таковы традиции, таков канон.

Также в контексте ИТ-образования привлекательной выглядит идея Гумбольдта *об обучении принципам вместо акцента на отдельных технологиях (предметах)*. Например, эта идея нашла отражение в IEEE/ACM программе бакалавриата по программной инженерии, где многократно подчёркивается важность воспитания у студентов инженерной культуры. Дело в том, что в индустриальном программировании очень быстро сменяются технологии и языки программирования. Поэтому обучение принципам выглядит более надёжным и полезным, хотя, конечно, базовые предметные знания также необходимы — языки программирования, инструменты и интегрированные среды разработки, технологии создания приложений различных видов (баз данных, телекоммуникационных систем и систем реального времени, компиляторов и виртуальных машин и т.д.). Тут требуется искать баланс. Однако такой баланс не может быть найден

непосредственно в лоне самой предметной области, но требует идей и принципов более высоких порядков.

> **Важно.** Итак, прочитав этот краткий очерк про историю университетов, вы могли увидеть, что университеты существуют давно, имеют определённые черты разных исторических эпох, а современный их облик сложился эволюционно. Ведь всегда интересно видеть временную ретроспективу некоторого явления — тогда, возможно, удастся увидеть и его перспективу, а также яснее и полнее уяснить его текущее состояние.

О научных требованиях к дипломным работам. Теперь выделим ряд требований к научной работе, которым ваш диплом должен удовлетворять и которые, соответственно, должны быть реализованы в тексте вашей работы:

- убедительно описанная актуальность тематики диплома,
- наличие постановки задачи;
- ясно определённые результаты, достигнутые вами в дипломной работе;
- индивидуальность ваших результатов,
- разграничение своего и «чужого»,
- новизна достигнутых результатов,
- а также их доказуемость.

Необходимость удовлетворять этим требованиям вносит сумбур в умы пишущих дипломы и часто препятствует более естественному ходу изложения материала в текстах. Поэтому я детально прокомментирую данные требования с надеждой, что читатель сумеет их усвоить и в дальнейшем будет ими руководствоваться на практике при написании своего собственного диплома.

Актуальность тематики вашего диплома означает, что рассматриваемая в вашей работе проблема по каким-то причинам интересует не только вас. Проблема может быть актуальна с точки зрения человеческого познания (но это нужно обосновывать), может быть актуальной с практической точки зрения, причем последнее — и это не гипотеза автора диплома, а реальная ситуации в индустрии, и это нужно грамотно показать. Следует отметить, у вашей работы может не быть ярко выраженной новизны (про новизну смотрите ниже), но актуальность должна быть. Она может выражаться, например, в том, что многие компании решают данную проблему, что и вы в своём дипломе, поскольку она важна для пользователей, для бизнеса, и при этом трудна в реализации, т.е. для неё не существует универсального решения. Последнее обстоятельство, в свою очередь, может означать, что, в зависимости от контекста, в этой проблеме появляется много различных деталей, которые требуется принимать во внимание, а это, в свою очередь, делает каждую конкретную реализацию нетривиальной и уникальной. Можно сказать и так, что проблема актуальна, когда её решение востребовано, когда в русле этой проблемы появляются всё новые системы, сервисы, а также создаются новые технологии для реализации таких решений. Актуальность может быть также связана с модой, с тем, что «на слуху», что волнует пользователей и разработчиков (например, машинное обучение и искусственный интеллект). Таким образом, актуальность является, скорее, контекстом вашей работы, характеристикой той области, в рамках которой ваша работа выполняется.

Постановка задачи. Творческий научный поиск часто происходит спонтанно, не по плану, с захватывающим поворотами: этим наука и привлекает людей, являясь той нишей, где можно постигать неизвестное, создавать новое. Однако в научной литературе (в статьях, монографиях и дипломах и так далее) принято делать вид, что в начале работы, исходно, были поставлены определенные задачи, и именно они-то и были превращены в результаты на выходе исследования. В этом мне видится отрицание недетерминированной природы научного поиска — да, да, такая позиция свойственна науке.

Между тем хочу отметить, что ясная формальная постановка задачи в начале письменной работы структурирует изложение материала в вашем тексте, даже если сама работа и начиналась со слов научного руководителя «А давайте попробуем…», хотя, возможно, что при этом ваш текст утрачивает живость и непосредственность. На самом деле, у научного руководителя, когда он вам что-то предложил, скорее всего, были невысказанные резоны, которые полезно прояснить, как минимум, в конце работы, когда вы пишите текст. Такая деятельность оказывается полезной ретроспективой, углубляющей ваше понимание того, что вы сами же и сделали. Кроме того, плоды этой ретроспективы помогают вам сделать вашу работу понятной для более широкого круга интересантов помимо прямых специалистов в данной предметной области.

Итак, если актуальность характеризует, преимущественно, область вашего исследования (так сказать, фон вашей работы), то постановка задачи является конкретной *фигурой* на этом фоне.

> **Важно** кроме актуальной, интересной *предметной области* (так сказать, фона вашей дипломной работы) ясно и чётко определить формальную *постановку задачи* (т.е. фигуру на этом фоне) и не путать фон с фигурой.

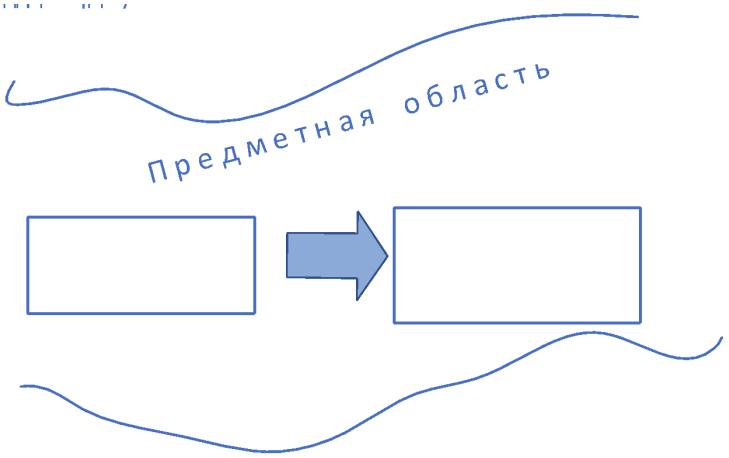


Рис. 1. От постановки задачи к результатам

Результаты научного исследования, полученные и ясно сформулированные, очень важны, ибо важно не только делать дипломную работу, но и сделать её — что-то создать, выработать некоторое новое знание. Поэтому классический вопрос рецензентов научных статей звучит следующим образом: «What are your contributions?». Результаты (contributions) в научной работе должны быть очень чётко определены и исчерпывающе описаны. Смутность, спутанность, нечёткость и мутность здесь считается огромным недостатком как самого исследования, так и соответствующего текста, поскольку научное сообщество не может понять, что же у вас получилось, и адекватно оценить это полученное (обычно рецензенты научных статей склонны весьма раздражаться в этом случае). Результаты важны еще и потому, что научный текст описывает, в основном именно их, а не тот процесс, которым исследователь пришёл к этим результатам. И это очень важно для текста вашей дипломной работы! Например, вы могли сначала создать прототип, а только потом перейти к проектированию и разработке самой системы. Так вот, в данном случае не стоит формулировать задачу/результат по разработке прототипа. Описываем финальные результаты – архитектуру и реализацию вашей системы. А вот при описании архитектуры можно упомянуть, что те или иные решения были приняты путём разработки и анализа прототипа.

Важно. Если при написании текста диплома не делались специальные усилия для акцента на результатах (они ясно не сформулированы, они не находятся легко в оглавлении работы и пр.), то текст оказывается размытым, и из него трудно понять, что же именно *сделал* автор, т.е. понять как сами результаты, так и их отдельные атрибуты (и тогда в лучшем случае всё ограничивается размытым заключением, изобилующим общими словами).

Индивидуальность результатов. Когда мы участвуем в индустриальном проекте по разработке ПО, то результатом является единый программный продукт, в рамках которого бывает непросто однозначно выделить индивидуальный вклад каждого участника. Научные результаты, напротив, традиционно носят строго индивидуальных характер. Помните многочисленные именные теоремы Эйлера, Ньютона, Вейерштрасса и т.д.? Если же научная работа выполнена коллективно, то в этом случае точно определяют вклад каждого участника. Это, в частности, выражается в порядке соавторов научной статьи (чем ближе автор располагается к началу списка, тем выше его вклад), а также, например, в детальном описании вклада каждого соавтора в отдельности при описании совместных публикаций в диссертациях. В истории науки известно много споров об авторстве тех или иных результатов — можно вспомнить, например, знаменитый спор Ньютона и Лейбница о том, кто из них создал дифференциальное исчисление, длившийся и после смерти обоих. Также следует отметить, что над решением некоторой научной проблемы могло работать много учёных, но тот, кто сделал последний шаг и завершил доказательство, считается человеком, решившим эту проблему, а все остальные участники остаются в тени. Другое отношение к индивидуальности демонстрируют строители готических соборов, а также русские иконописцы (кроме некоторых) — все они остались анонимными.

Разграничение своего и «чужого» тесно связано с индивидуальностью научных результатов и означает, что, излагая эти результаты, вам следует чётко разграничивать то, что сделано именно вами, от того, что было сделано вашим товарищами, что сделали до вас, что вы использовали в своей работе и так далее. Многим студентам, кто пишет дипломы, это непросто сделать — они склонны описывать текущее положение дел в своей области, существенно смешивая эти описания со своими собственными идеями и воззрениями, а также совмещая с описанием своих результатов. Также часто встречается ситуация, когда созданная в рамках диплома система описывается вместе с использованными технологиями. Следует отметить, что порой непросто выделить свою часть работы из коллективного проекта, поскольку оказывается, что студент выполнил много различных работ, но все они вместе не образуют замкнутого единого целого.

Новизна результатов означает, что ваша работа содержит некоторое открытие, то есть вы создали то, что до вас не существовало. Обычно речь идёт о некотором новом научном знании – доказана теорема, которую до вас не могли доказать, решена задача, волновавшая умы исследователей долгие годы и т.д. При этом новизна подразумевает наличие некоторого ясно очерченного *сообщества*, в рамках которого вы можете соотнестись, доказав эту новизну. Наличие такого сообщества крайне важно. И даже если ваш диплом не является научным, то вам всё равно нужно сделать акцент на новизне — соотнести возможности вашей системы с возможностями аналогичных, имеющихся на рынке и/или в открытом доступе, и показать какие-то конструктивные отличия. Последнее означает, что вы таки сделали что-то новое и интересное. Исключение составляют результаты, полученные в рамках индустриальных проектов — сам такой проект может обладать новизной, но его отдельные компоненты, в том числе и та, которую сделали вы, могут быть обычными, «как у всех».

Доказуемость результатов. Истинность научных результатов должна быть доказана. Тут впору вспомнить о том, что наука принципиально имеет дело с доказуемыми фактами, и если они установлены, интересны, но недоказуемы, то являются либо предметом дополнительного научного поиска (ищут способ их обоснования, объяснения, доказательств существования явления и истинности объяснений), либо их вовсе не

рассматривают в рамках науки, как, например, НЛО или паранормальные явления. Доказуемость является свидетельством истинности, объективности полученного знания, а также залогом воспроизводимости — то есть каждый интересующийся сам может убедиться в истинности полученного, нового знания, воспроизведя самостоятельно доказательство или базовые эксперименты.

Доказательство можно осуществить с использованием математического аппарата, т.е. формально, но лишь в том случае, если вы в своей работе используете математическую постановку задачи, то есть формулируйте определения и теоремы, и применяете соответствующую математическую теорию для их доказательства. Иначе, если математический аппарат не используется в работе, то истинность результата может быть доказана экспериментально.

Эксперименты следует выполнять на представительной выборке данных/тестах, хорошо покрывающих вашу задачу с обязательными последующими выводами — например, о том, что ваш алгоритм превосходит существующие аналоги. При этом, разумеется, нужно запустить и проверить на тех же данных/тестах аналогичные существующие (то есть не ваши!) алгоритмы, инструменты и тому подобное, продемонстрировав, что ваша разработка работает быстрее, успешно покрывает большее количество случаев и так далее. Бывает, что экспериментально сравниться с аналогами не представляется возможным — вы сделали нечто сильно «встроенное», и в таком случае вы можете охарактеризовать, например, точность (precision) и полноту (recall) вашего алгоритма, а также оценить время его работы, включая обоснование, что последнее (то есть время работы) оказывается приемлемым в данном контексте.

Доказуемость понимается ещё и в том смысле, что в самом тексте диплома стиль изложения материала также является доказательным в смысле обоснованности. Это означает, что нужно избегать голословных, эмоциональных заявлений по ходу работы, например, «всем известно…», «обычно это делается так …», «многие эксперты считают, что проблема заключается…» (без указания конкретных экспертов и статей, материалов в Интернете и пр. фактов), что-либо (алгоритм, язык программирования) «лучше/хуже, чем…» (например, Шерлок Холмс говорил, что «эмоции противоречат чистому и холодному рассудку»).

> **Важно.** Доказуемость в тексте вашего диплома реализуется обоснованностью ваших утверждений, т.е. проработкой аргументов, если вы их приводите, а также избеганием эмоциональных и глобалистских утверждений. Обоснованность результатов достигается, чаще всего, с помощью экспериментов. Для этого они должны быть тщательно спроектированы, выполнены, а их результаты проанализированы (подробнее про эксперименты смотри соответствующую лекцию).

В заключении этой лекции я хочу отметить, что, по моему мнению, научная форма дипломных работ уже не вполне соответствует современной реальности, и требуются новые каноны. Например, в индустриальных и групповых дипломах по программированию удобно смешивать своё и «чужое», следуя в тексте логике предметной области, ходу разработки системы — и в этом случае изложение становится более понятным и интересным. Однако на сегодняшний день новых канонов пока не создано, и даже дипломы по менеджменту туризма и дизайну интерьера оформляются в рамках изложенного выше канона. Так что можно сделать такой вывод — все описанные выше научные требования имеют прямое отношение к тексту вашего диплома.

Контрольные вопросы

1) В чём особенность научности диплома в университете?
2) Должен ли ненаучный (например, производственный) диплом иметь научные черты? Пожалуйста, ответ обоснуйте.
3) Когда возник лекционный формат обучения в университетах, как он видоизменялся и в каком виде он существует сейчас.
4) Что вы можете рассказать про семинар как форму обучения в университете?
5) Перечислите научные требования к диплому.
6) Расскажите о необходимости актуальности вашего диплома.
7) Что такое постановка задачи диплома?
8) Расскажите о результатах дипломной работы.
9) Почему ясно сформулированные, ёмкие результаты требуют специальной работы – и часто уже после того, как диплом сделан и осталось всего лишь написать текст?
10) Как вы понимаете индивидуальность результатов дипломной работы?
11) Расскажите о разграничение своего и «чужого» в тексте диплома.
12) Что означает новизна результатов дипломной работы?
13) Расскажите об обоснованности суждений в тексте диплома.
14) Что означает и зачем нужна доказуемость результатов диплома по программированию?
15) Какие вы знаете способы доказательства истинности, корректности, значимости результатов дипломов по программированию?

Список литературы

1) К.В. Гумбольдт. О внутренней и внешней организации высших научных заведений в Берлине / Пер. с немец. в журн. «Неприкосновенный запас», N 2, 2002.
2) А.Ю. Андреев Российские университеты XVIII – первой половины XIX века в контексте университетской истории Европы. Издательство «Знак», 2009.
3) Н.В. Ростиславлева. Теория и практика раннего либерализма в Германии (первая половина XIX в.). Диссертация на соиск. уч. степ. докт. истор. н-к. М., 2011.
4) Рекомендации по преподаванию информатики в университетах. Пер. с англ. Изд-во СПбГУ, 2002.
5) А.М. Цвелик. Жизнь в невозможном мире. Издательство Ивана Лимбаха. 2012.
6) Л.Н. Гумилёв. Этногенез и биосфера земли. Гидрометеоиздат. 1990.

Лекция 2. О видах дипломов по программированию

В лекции рассказывается о разных категориях дипломных работ по программированию — о научных, исследовательских и производс-твенных дипломах. Также рассматривается вопрос выбора темы диплома, рабочие продукты, создаваемые в ходе работы над дипломом.

Чего автор хотел сказать этим своим ... произведением?

Землетрясение. М. Зощенко

В этой лекции мы рассмотрим различные категории дипломов по программированию. Эта тема является важной, поскольку тут существует изрядная путаница — прежде всего, в самих университетах при позиционировании программирования и определении требований к дипломам. Так, все ещё встречается пренебрежительное отношение к индустриальным тематикам дипломов и благоговенье перед так называемыми научными дипломами. Также иногда встречаются завышенные требования к индустриальным дипломам – например, требование акта о внедрении результатов работы в производство (тут следует понимать, что далеко не все результаты индустриальных проектов, в особенности, выполняемых студентами, попадают в финальные продукты, продающиеся на рынке и/или имеющие реальных пользователей). Наконец, в компаниях и университетах выполняется большое количество технологических исследований, которые при подготовке дипломов порой выдаются за научные, что не соответствует действительности и сильно вредит самим работам, затуманивая изложение результатов ненужными абстракциями. Так, например, студент реализовал интересную и актуальную для пользователей функциональность в контексте некоторой среды разработки (другими словами, он дополнил эту среду), а заявляет, что создал новый метод — и в результате и слушателям, и читателям непросто разобраться, что же он сделал на самом деле.

Вся эта путаница приводит к тому, что студенты не могут уяснить точные требования к своим дипломам, затрудняются с выбором тем, привнося в этот процесс как мечты о том, что диплом должен быть чем-то «большом и светлым», так и ненужные беспокойства о необходимой «солидности» своей выпускной работы. Кроме того, претензии студентов на занятия настоящей наукой, порой, не учитывают реальную сложность научных исследований и высокий порог вхождения.

Мне кажется, что естественное течение дел в этом вопросе, соответствующее реальным склонностям и фактическим возможностям студента, а также конструктивные традиции де факто российского ИТ-образования способны обеспечить интересный процесс работы над дипломом и хороший результат. Причём я искренне считаю, что работа над дипломом и сама защита должны носить *праздничный, торжественный характер* — получение диплома о высшем образовании является значимой вехой в жизни человека.

> **Важно** при создании диплома точно определиться с его категорией: от этого будет зависеть позиционирование ваших результатов, акценты, а также то, что вы будете оставлять в тени. От категории диплома также зависит способ, которым вам следует обосновывать новизну своих результатов и проводить

сопоставление с существующими разработками и результатами в вашей области.

Выделим следующие категории дипломов по программированию:

- научные дипломы,
- исследовательские дипломы,
- производственные дипломы.

Эти категории важны не для абстрактной классификации рассматриваемого предмета — дипломов по программированию, — но сугубо в практических целях.

Давайте рассмотрим их детально.

Научный диплом — это полноценное научное исследование (не только по форме, но и по содержанию). С научностью бывает много путаницы, поскольку часто студенты считают, что если их работы посвящена созданию чего-то нового, то это и есть наука. Ну, а если при этом еще используются отдельные научные методы, то это точно наука. Да и вообще, заниматься наукой — это почётно!.. Однако тут всё сложнее.

Научная работа выполняется в рамках определенного *научного* сообщества и соответствующего круга проблем и задач, рассматриваемых данным сообществом. Например, если вы создаете компилятор или даже новый язык программирования – то это ненаучная работа, хотя очень интересная и нужная (при этом вряд ли вы создаёте полноценный компилятор или язык программирования — скорее всего, это прототип или фрагмент). Ненаучная она потому, что не решает какую-то определённую проблему научного сообщества Programming Languages (это сообщество группируется вокруг конференций POPL и PLDI (так называемые конференции ранга A*). И можно посмотреть в трудах этих конференций, каким проблемам посвящены научные исследования этой области).

> **Научный диплом** посвящён решению некоторой проблемы определённого научного сообщества. Он обладает несомненной научной новизной, т.е. является некоторым (большим или не очень) научным открытием.

Исследовательский диплом. Скорее всего, диплом про новый компилятор или новый язык программирования будет принадлежать к этой категории. Что это за категория?

В индустриальном программировании существует такой термин R&D (Research and Development), который переводится как исследование и разработка. Этим термином называют отделы прикладных исследований в промышленных компаниях. Современные крупные компании, такие как IBM, Microsoft, Huawei и др. имеют специальные исследовательские подразделения, которые, впрочем, часто называются иначе — IBM Research, Microsoft Research и т.п. Более старое название подобных подразделений — R&D. Чем характеризуются исследования в таких компаниях? Прагматичным, прикладным характером, связанным с приоритетами компании, с определенными нуждами. При этом они могут иметь научную новизну, а их результаты часто публикуются в научных конференциях. Но это не является основным, да и не все подобные результаты публикуются в научной прессе. В частности, результаты могут не публиковаться и потому, что компания не хочет раскрывать свои новые подходы.

В чём индустриальная актуальность отличается от научной новизны кроме того, что последняя возникает, обычно лишь тогда, если задача поставлена в рамках научного сообщества? Например, создаются сервисы навигации по исходным кодам C-программ для специализированной среды разработки, и эти сервисы ориентированы на сверхбольшие проекты и принципиально не подразумевают использование результатов компиляции. Но сколько уже было создано средств разработки для C-приложений? Сколько было создано

сервисов навигации по исходному коду? Наконец, существуют и специальные средства для «легковесного» анализа С-приложений. Таким образом, заслуга автора данной работы заключается в том, что он сумел всё это собрать вместе и приложить к решению конкретной практической проблемы в своей компании. И научной новизны в этой работе нет. Но польза для компании большая, и исследовательская деятельность налицо — вряд ли существует ещё одно в точности такое же решение. В то же время производственной данную дипломную работы тоже не стоит называть по многим причинам. Например, она может остановиться на уровне прототипа и не иметь полноценной реализации и конкретных пользователей.

> **Важно.** Основное отличие исследовательских дипломов от индустриальных заключается в том, что последние всегда являются частью индустриального проекта для конкретного заказчика или частью проекта по созданию рыночного продукта/сервиса.

Кроме того, исследовательские дипломы часто инициируются внутри самого университета преподавателями, которым интересно попробовать какую-нибудь технологию, какой-нибудь метод и т.д. При этом работа могла бы быть научной, если бы исходная задача ставилась из научного сообщества.

В заключении дадим определение исследовательского диплома.

> **Исследовательский диплом** посвящён решению некоторой актуальной нетривиальной индустриальной задачи или интересной технологической проблемы. Такой диплом подразумевает исследования, которые, однако, проводятся вне научного контекста. Результаты работы не обязательно имеют внедрение, а могут оставаться на уровне прототипа. Такой диплом может быть выполнен в индустриальной компании в качестве R&D-разработки, а может — в университете.

Производственный диплом. Речь идёт о самом распространённом виде дипломов по программированию. Сегодня большинство студентов-выпускников, как правило, уже работают в промышленных компаниях и очень часто берут темы дипломов именно у себя на работе. К этому в университетах относятся по-разному, в том числе не всегда положительно. Отчасти это происходит из-за университетского снобизма: мы де тут, в университете занимаемся высокой наукой, а тут какая-то презренная прагматичная индустрия... Но от части подозрительное отношение к производственным дипломам в университетах формируется из-за сложностей в коммуникациях с соответствующими компаниями: специалисты из индустрии не являются профессиональными педагогами и, как правило, не могут обеспечить приведённые выше научные требования к дипломной работе (невзирая на то, что они сами заканчивали университеты и сталкивались с этим!). Бывает также и так, что студенты не могут толком объяснить, в чём же именно заключалась их работа, что они сделали, в чём состоит «благородная сложность» их работы. Поэтому важно, чтобы тут была оказана помощь из университета. Тем более, что в бурно развивающейся сфере индустриального программирования для университетских преподавателей важно знакомиться с новыми задачами, технологиями, тенденциями, предметными областями, и одним из способов это осуществлять является помощь студентам защищать производственные дипломы.

> **Важно.** Если вы хотите писать диплом на работе в компании, то нужно озаботиться необходимой поддержкой процесса со стороны университета.

Мне кажется, что производственные дипломы, при необходимой поддержке и организации процесса со стороны университета, очень ценны, поскольку знакомят университетских преподавателей с новыми тематиками, направлениями, тенденциями,

предметными областями и технологиями, способствуют созданию и укреплению двухсторонних связей университета и индустрии. Ну и для самих студентов это удобно, поскольку позволяет им, не отвлекаясь на параллельные и зачастую искусственные активности, осмыслить и оформить то, чем они занимались в течение довольно долгого времени в компании и продемонстрировать, что они являются сформировавшимися специалистами, готовыми к работе в индустрии. Тут важно напомнить университетским преподавателям и администраторам, что именно подготовка специалистов для работы в индустрии, а не создание «правильных» дипломов, является основной задачей образования в сфере программирования.

> **Производственный диплом** является частью реального индустриального проекта, выполненного студентом в некоторой промышленной компании.

Производственные дипломы тем более интересны, что относительно них можно сформулировать большое количество нетривиальных и содержательных инженерных вопросов. Меня, например, на предзащитах и собственно, защите всегда интересует степень готовности, завершённости программной системы, созданной выпускником в рамках индустриального диплома. Например, возможны следующие варианты вопросов про это.

- Сколько система имеет конкретных пользователей и каковы они? Ведь имеется существенная разница, например, между внутренними пользователями (т.е. они сидят в соседнем от автора системы помещении) и независимыми клиентами, которым система или сервис продан.
- Как система тестировалась, как можно охарактеризовать её качество и степень готовности для применения?
- Если система используется реальными пользователями, то каковы их впечатления? При этом, если вы создали только часть системы, то каковы впечатления пользователей именно от вашей части, то есть от реализованной вами функциональности – потому что ведь именно ваша функциональность может оказаться неиспользуемой! В связи с этим уместно напомнить правило 80 на 20, которое гласит о том, что 80 процентов пользователей используют лишь 20 процентов функционала системы…
- Если система является прототипом, то кому и что показал этот прототип показал, что прояснил, как эта информация будет использоваться в дальнейшем в компании?

Также интересным является обоснованность выбора тех или иных технологий разработки. Причём, эта обоснованность может базироваться на совсем не академических аргументах. Так, например, менеджер одного большого индустриального проекта обосновывал выбор технологий разработки от компании Microsoft тем, что данная компания очень устойчива, и это понижает риски долгосрочного использования её инструментов.

Также, как правило, можно содержательно пообщаться с автором диплома по поводу отдельных проектных решений, и тут оказывается важной эрудированность автора дипломной работы, его осведомлённость касаемо имеющихся вариантов решения тех же задач и проблем. Но помимо этих рассуждений о том, как правильно или неправильно реализовывать те или иные задачи (часто университетские преподаватели западают именно на такого рода вопросы) важны также всевозможные конкретные соглашения и обстоятельства реального проекта, в контексте которого была выполнена дипломная работа.

Программирование и наука. Данная тема важна для нашего предмета — написание дипломов по программированию, и вот по какой причине. Помимо обозначенной выше априорной научности любого диплома – по форме, в силу научных университетских

традиций, ведущих начало от классической университетской модели К.В. Гумбольдта, — имеются ожидания, что дипломы в нашей сфере будут иметь реальную научную составляющую. Часто этого неявно (а иногда и явно) ожидают от своих студентов выпускающие кафедры, существенно выше порой оценивая математизированные дипломы по сравнению с индустриальными. И сами студенты, порой, желают иметь дипломы на фундаментальные, теоретические темы. И они, и многие люди из индустрии с уважение относятся к тем, кто такие дипломы сумел подготовить. Поэтому ответы на вопросы о том, что такое наука, откуда и когда она появилась, какой путь прошла и в каком сейчас находится состоянии, а также то, как с ней связано программирование, мне кажутся актуальными.

Наука в современном понимании сформировалась к XVII веку. Помимо нарастающих открытий в различных областях сформировалось научное мировоззрение, включающее в себя:

- восприятие природы как объекта изучения и освоения, причём на основе субъектно-объектных, рациональных отношений, где человек оказывается то независимым исследователем, то колониальным покорителем, то песчинкой в урагане стихий (последнее, скорее, является обратной, иррациональной стороной главной, рациональной составляющей), и значительно реже он является восхищённым зрителем красоты, непостижимой сложности и цельности творений природы;
- эволюционную идею — в биологии, космологи, истории науки и пр. — включая видение человечества как целого на фоне поступательного прогресса: исторического, общественного, научного и технического.

Сегодня научное мировоззрение нам кажется естественным и единственно верным, но так было не всегда. В более ранние времена — в Древнем Вавилоне и Египте, в Древней Греции и Риме, в Средние Века — науки не существовало, хотя отдельные научные результаты и были достигнуты (точнее, имеется нечто, что можно трактовать как научные результаты). Такое положение дел не является свидетельством неразвитости этих эпох — последние имели иное мировоззрение, приоритеты, ценности и устремления. Кажется, что искать в древних эпохах «зачатки» астрономии, математики, механики, физики и других наук означает смотреть на эти эпохи не как на таковые, какими они были, но сквозь призму современного мировоззрения. Это, безусловно, проще, но уводит нас далеко от непосредственного восприятия познаваемого объекта, фактически, низводя его уникальность до наших исходных, априорных представлений. Невольно возникает вопрос — а ищем ли мы тогда действительно новое знание?

Появление науки в Западной Европе было обусловлено конкретными культурными, идейными и историческими контекстами. К таким контекстам следует отнести географические открытия, расширившие картину мира европейцев и показавшие им другие, совершенно непохожие человеческие и природные миры, и в целом планету Земля. Сюда же следует отнести Реформацию, которая поколебала всеохватный авторитет Церкви в миропонимании людей и жизни общества. Важную роль в становлении науки сыграло Итальянское Возрождение, возбудившее многогранный интерес к прекрасному, к человеку, к окружающими миру, истории и литературе, а также к естественным наукам. В дальнейшем глобальное распространение науки на Земле коррелирует с доминированием мировоззрения и ценностей Западного мира. Таким образом, наука является продуктом развития Западного Мира.

Существенной характеристикой науки явилось научное сообщество учёных, специально организованное и имеющее определённые формы существования. Необходимость однозначного признания тех или иных научных идей в этом сообществе, понятие коллективной общепринятости результатов и, соответственно, общая смена взглядов и в конечном счёте построение единой, общей картины мира. Полемичность науки характеризуется процессом выработки/смены взглядов (и здесь явно наследует

традицию средневековых схоластических споров и диспутов), но не является составляющей её существования. Основной формой появления и передачи новых знаний являются научные публикации – ранее преимущественно книги (трактаты, монографии), позднее к ним добавились периодические научные журналы, а в конце XIX века появились конференции, симпозиумы и конгрессы.

Однако в дополнение к этой традиционной форме сегодня существует ещё одна — новые рыночные продукты, которые, например, в области информационных технологий и программирования представляют многие новейшие идеи и разработки, не представленные в виде научных результатов и часто оказывающиеся предметом вторичного научного изучения. Таковы, в частности, телекоммуникационные технологии, большие данные (в частности, в сфере software engineering), и, собственно, сама программная инженерия (software engineering).

Институциональными формами науки являются:

- академии наук, специально создаваемые для организации учёных и научной деятельности (в Италии с XV века, в остальной Европе — с XVII века), а также активно основывающиеся и продуктивно работающие в настоящее время отдельные исследовательские научные институты;
- университеты, превратившиеся, после реформы Гумбольдта из замкнутых средневековых учреждений, сохраняющих культуру и обеспечивающих поддержание элементарной грамотности, а также выпускающих востребованных обществом профессионалов — богословов, медиков и юристов — в активные научно-образовательные центры и динамичные общественные институты;
- различные лаборатории прикладных исследований при индустриальных компаниях: в современном мире создание новых наукоёмких технологий, находящих массовое применение посредством созданий новых рыночных продуктов, происходит, преимущественно, в таких лабораториях.

Остановимся на очень важном аспекте науки — целостности научного знания. Наука не может останавливаться на наблюдениях и фиксации бесчисленных явлений природы. Н. Кузанский писал, что наука призвана занять серединное положение между безбрежностью многообразных актов чувственного восприятия (то есть восприятий органов чувств), с одной стороны, и непостижимым и всеохватным божественным абсолютом, с другой стороны. фактически, сводя множественное к постижимо-единому. В.И. Вернадский определял науку как установление научных фактов и построение на их основе эмпирических обобщений — законов. Он, как и многие другие учёные, негативно относился к логическим построениям многих философов, претендующих на познание Мироздания, но создававших свои построения в отрыве от непосредственной реальности. Склонность к необоснованным обобщениями и разного вида заблуждениям Ф. Бэкон классифицировал в виде призраков, «которые осаждают умы людей» (он выделял призраки рода, призраки пещеры, призраки рынка и призраки театра). Бэкон призывал людей через науку подняться над человеческим (обыденным) взглядом на Мироздание и принять нелогичную, невообразимую для человека, но реально истинную природу вещей. Он считал, что основная проблема познания заключается в том, чтобы разуму человека выйти за пределы собственной природы и следовать природе самой по себе.

Программирование возникло на стыке вычислительной математики и инженерных задач в 30–40-х годах XX века в виду роста числа вычислительных задач (преимущественно, в военной области), а также развития электронной элементной базы — двоичных цифровых схем и доступных технологий для их реализации (электро-механические реле, вакуумные лампы, полупроводники, интегральные схемы). Механические счётные машины конструировались с давних времён — например, Б. Паскалем в XVII в., — но именно в указанный период появилась критическая масса обстоятельств, которая привела к тому, что одновременно в США, Германии,

Великобритании, России и ряде других стран стали появляться первые вычислительные машины. Принципы организации архитектуры вычислительных машин собрал и опубликовал в 1945 году Джон фон Нейман в знаменитом труде First Draft of a Report on the EDVAC. Эти принципы составили так называемую архитектуру фон Неймана, и с тех пор она является основой аппаратных вычислительных средств.

Программы для первых ЭВМ создавались в машинных кодах, исполнение которых иногда требовало участие человека (оператора). После появления полностью автоматизированных вычислительных устройств, в 50-х годах, стали создаваться первые высокоуровневые языки программирования — Fortran, COBOL и другие, — потом появились операционные системы, реализующие для прикладных программ аппаратно-независимый интерфейс работы с различными устройствами компьютера (60-е годы), далее, в 80-е годы, стали активно развиваться и использоваться первые интегрированные среды разработки для создания больших приложений на мэнфреймах (так называемые CASE-системы) и так далее.

Прогресс средств разработки программ ошеломляет, но одновременно оставляет нерешёнными большое количество вопросов:

- создание надлежащих абстракций проектирования/программирования, позволяющих разрабатывать системы всё большей сложности и размера;
- решение различных интеграционных задач разработки;
- комплексные технологии управления большими проектами и предсказуемость проектов (сроки и бюджет);
- решение вопросов качества и безопасности приложений и другие.

Для систематического решения подобных задач ещё в 1968 году была образована новая область знания и наука — программная инженерия (software engineering). В то время пришло понимание того, что крупные программно-аппаратные системы требуют, помимо создания аппаратной части (то есть определённого ансамбля вычислительных устройств), математических алгоритмов, ещё нечто весьма существенное: стало очевидно, что программы для ЭВМ значительно сложнее, чем ввод (кодирование) математических алгоритмов. В это время стали появляться самые разные задачи, помимо сложных расчётов, например, управлению сложным оборудованием, включая сами ЭВМ, реализация различных цифровых сервисов и т.д. Возникли модели процесса разработки (водопадная, спиральная, итеративно-инкрементальная и прочие), стандартны процесса (например, широко известный CMM/CMMI), методологии разработки (MSF, XP, Scrum, RUP), модельно-ориентированные средства разработки (структурный анализ, объектно-ориентированный анализ и проектирование, UML), предметно-ориентированные языки программирования, концепция и средства для разработки семейств (линеек) продуктов, методы и инструменты для поддержки различных видов деятельности и фаз процесса разработки, исследования в области архитектуры программных систем, а также огромное количество различных частных задач по автоматизации разработки различных специализированных программных систем. При этом следует отметить, что программная инженерия, фактически, не использует математический аппарат.

Наряду с бурным и многоплановым развитием программирования возникли также попытки концептуального осмысления этого нового научно-инженерного направления и создание классического раздела науки. Самой известной такой попыткой явилось изобретение американским математиком Норбертом Винером и его коллегами *кибернетики* — науки об управлении в живых и механических системах. Соответствующую монографию он опубликовал в 1948 году, и она имела большой резонанс в мировом научном сообществе.

В некотором роде кибернетика была большим, чем стремлением концептуализации программирования: она была одной из попыток интеграции науки в целом вокруг задач управления и математических способов их решения. Следует отметить, что с середины XIX века наука начинает утрачивать целостность и единство в виду большого количества

разнообразных специальных областей, где освоение каждой из которых оказывается делом целой жизни. Соответственно, из науки уходят энциклопедисты, занятие наукой становится всё труднее сочетать с другими общественными функциями, например, с государственной службой или работой в индустрии. Разбиваясь на отдельные узкие дисциплины, наука всё сильнее замыкается в себе, и её способность это делать постепенно становится её характеристической чертой. К доказательствам этого факта можно отнести авторитет и снобизм узких специалистов, а также ту «естественность» и скорость, с которой вокруг областей, являющихся в высшей степени практическими, прикладными, образуются замкнутые академические сообщества со своими темами исследований, тенденциями, неписанными правила и авторитетами. При этом данные области отбрасывают из рассмотрения явления и феномены, открытые вопросы и отдельные аспекты (например, практическая программная реализуемость тех или иных теоретических построений), насущные и актуальные для практики, но несоответствующие букве «научности».

Междисциплинарные исследования, объединяющие различных узких специалистов и основанные на математике, были начальной стартовой задачей группы Норберта Винера. Этой группой было инициировано значительное количество проектов в самых разных сферах — в физиологии, биологии, медицине и др. Всё это также было теснейшим образом связано с развитием вычислительных средств и программированием главным образом, вследствие военных запросов и Второй мировой войны, мобилизовавшей силы человечества на создание новейшего вооружения. И везде Винера неизменно интересовали вопросы управления, предсказания, самообучения и так далее, везде он стремился применить аппарат гармонического анализа, нелинейных динамических систем и пр.

Идеи Норберта Винера вызвали как энтузиазм исследователей, так и резкую критику. Видимо, одним из самых резких прецендентов такой критики имел место в России, где философы объявили кибернетику лженаукой, сумев добиться её официального запрета, который, впрочем, недолго продержался и не смог нанести заметного урона. Однако сама кибернетика в итоге не смогла объединить различные науки и выковать общие основы для решения задач управления в разных областях на основе математики. Впрочем, на трудности использования математических моделей при описании процессов в живой природе, при их исходном сходстве и некоторых очевидных резонансах, например, в области физиологии, указывал ещё российский академик А.Ухтомский в 193…. в своей интереснейшей статье …. о …. В итоге кибернетика стала одной из математических наук и в настоящее время оставила попытки объединить остальные науки в лоне решения задач управления. А программирование как наука так и осталась между зарождением (эдакий вариант вечной молодости) и успешным прикладными результатами.

При этом нельзя сказать, что в программировании отсутствует научность. К настоящему времени образовалось достаточное количество математизированных областей, например, языки программирования, теория баз данных, формальные методы, искусственный интеллект, логическое программирование и др., которые в настоящее время объединяются под названием Computer Science (а иногда к этому названию добавляют слово Theoretical). Но они не образуют единства, более того, их связь с практикой является постоянно обсуждаемой и весьма болезненной темой. При этом программная инженерия как наука стоит особняком, в стороне от различных разделов Computer Science. И в рамках программной инженерии также наблюдается бурный рост исследовательской деятельности: согласно S. Alam, S. Zardari, M. Bano за 2007–2019 годы в этой области было выпущено около 150 000 научных статей! Однако следует отметить, что такое количество статей крайне трудно даже просмотреть… Вместе с тем в программной инженерии отсутствуют единые основы, общепринятые концепции, базовая теория, что, в свою очередь, является необходимым атрибутом устоявшейся научной дисциплины (смотрите, например, исследование J.Pontus, M. Ekstedt, I. Jacobson). Таким образом, программирование оказалось семейством научно-практических активностей,

бурно и стремительно видоизменяющих современную реальность, включая самого человека, но не имеющего единого научного базиса.

Про науку в целом можно в настоящее время можно констатировать, что она теряет целостность, связность, а та самая интеграция эмпирических фактов в единую, истинную философию, о чём писал Ф. Бэкон на заре составления научного проекта Нового Времени, оказывается сегодня, фактически, недостижимой задачей. В то же время наука, способствуя увеличению могущества человечества, оказывается причиной крайне разрушительных деструкций в виду создания новейшего смертоносного оружия, а также разрушения экологии нашей планеты. Вместе с тем современное научное мировоззрение продолжает оставаться в пределах координат, обозначенных в XVI–XVII веках, основываясь на идее всеобщего исторического прогресса человечества путём научно-технического освоения природы с целью расширения его возможностей. Пожалуй, единственной добавкой к этим идеям, возникшей за последнее 50–70 лет, является бизнес и глобальный рынок, активно использующие научные результаты и активно влияющие на развитие науки, что отсутствовало в XVI–XVII веках. Однако сегодня явно востребованы принципиально новые научные концепции, и в целом — новое общечеловеческое мировоззрение, включающее интеграцию, гуманизм, бескорыстие и ответственность, а также самого человека, как ценность. Новые концепции науки, новое мировоззрение — и это то, что ещё только предстоит создать!

О выборе темы диплома. Перед началом работы над дипломом нужно, конечно же, сначала выбрать тему. Давайте обсудим этот вопрос. При этом я обозначу ошибки и трудные места, поскольку промахи с выбором темы приводят к большим сложностям — вплоть до переноса защиты на следующий год, смены темы и руководителя или даже вообще к уходу из университета. В работе над дипломом, как и при разработке ПО, справедливо одно и тоже правило — чем на более ранних стадиях была сделана ошибка, тем дороже стоит её исправление. Именно поэтому ошибки и неточности при выборе темы являются самыми «дорогими».

Особенность дипломов по программированию заключаются в том, что они всегда подразумевают практическую (реальное программирование на некоторых языках программирования, в некоторых средах разработки) и/или научную часть. И данная часть является уникальной, поскольку часто связана с некоторой компанией, где студент работает, с какой-то уникальной исследовательской задачей и т.д. В целом имеется огромное количество уникальных тем для дипломов, область индустриального программирования и исследований является живой и в неё вовлечено большое количество людей, денег, идей. И эти темы быстро развиваются, то есть то, что было актуально 3–7 лет назад может быть сегодня уже неактуально.

Итак, в дипломе по программированию имеется уникальная содержательная часть. Поэтому и текст, который её описывает, тоже оказывается уникальным. Таким образом, в нашей области, фактически, невозможно, как в некоторых других областях, списать работу из Интернета.

Итак, темы дипломов по программированию уникальны и *рождаются*. Я считаю, что в нашей области нельзя действовать так, как в других областях — заранее готовить набор некоторых тем и предлагать их готовыми для студентов. Я склонен предлагать определённые тематики, подобласти, предполагая, что тема диплома «высекается» самим студентом — его активностью и желанием, вдохновением и трудолюбием. Поэтому можно выбрать (создать) себе тему по сердцу, что невероятно привлекательно — как для студентов, так и для преподавателей. И написать работу с удовольствием, искристо и вдохновенно!

Выбирайте тему диплома именно в таком настроении.

Про недостаточно сложные темы и рабочие продукты диплома. Не надо стремиться взять для диплома слишком трудную задачу — как в практическом плане, так и в теоретическом смысле. Решение трудной задачи целесообразно представлять в рамках диплома, если вы занимались этой задачей два и более лет, и к моменту выбора темы у вас уже есть результаты, которые получились, так сказать, естественным путём. Но если это не так, если вас влечёт новая задача, и вы готовы рискнуть… Но даже в этом случае нужно соблюдать определённую разумность.

Нужно понимать, что дипломная работа по программированию включает в себя не только решение некоторой задачи, как в математике. Многие студенты считают, что самое главное для диплома — это … программный код. Ну и, соответственно, пишут его едва ли не до дня защиты, стремясь всё написанное включить в диплом. Но следует понимать, что ваш код, кроме научного руководителя, никто не смотрит (за исключением некоторых особенных преподавателей в ИТМО). А хорошо выполненная работа содержит много больше, чем только сложный код. Как правило, требуется ещё также апробация и/или эксперименты, которые доказывают практическую полезность полученных результатов. Кроме того, нужен хороший обзор сходных разработок или научных работ. Ну и в конце концов, требуется полноценный текст диплома, а также презентация (ppt или pdf) и нормальная дипломная речь. Для того, чтобы создать все эти рабочие продукты, требуется время и силы.

Специально отмечу, что в дипломах по программированию (научных, исследовательских или производственных) требуется наличие программного кода, программной реализации. Поскольку мы говорим именно о программировании. Иначе целесообразно защищать работу по другим учебным специальностям, например, по математике. Впрочем, учебные программы по программированию в российских университетах составляются и оформляются во многом ad hoc, и математизированные результаты пользуются традиционным уважением у преподавателей, и часто — у студентов (высокая наука!). Поэтому об окончательных требованиях к диплому справьтесь у своих кураторов в вашем университете. Тем не менее будем исходить из того, что программная реализация в дипломе нужна — ведь вам присваивают квалификацию программиста, с которой вы может работать в индустрии.

Рассматривая дипломную работу как специальный проект, можно выделить так называемые *рабочие продукты* этого проекта — различные части финальной поставки, являющейся результатом проекта[1]. Напомню, что в финальную поставку программного обеспечения входит не только код, но и документация, обучение пользователей/команды сопровождения, средства инсталляции продукта или его развёртка на серверах заказчика, поддержка продукта (ответы на вопросы, разбор проблем) и т.д. Как в случае с программным обеспечением, точная идентификация рабочих продуктов и учёт времени ресурсов на их создания являются важной составляющей успешного дипломного проекта.

Рабочими продуктами диплома по программированию являются:

- формальная постановка задачи;
- практическая и/или научная часть, включающая программный код;
- эксперименты и/или апробация;
- обзор сходных продуктов/решений/технологий;
- текст диплома;
- презентация (ppt/pdf), и часто несколько версий (для предзащит);
- формальные документы (всевозможные акты, заявления и пр.);
- подготовленная речь для выступления на защите.

[1] Строго говоря, в проектах бывают ещё промежуточные рабочие продукты, но в рамках данного пособия они нас не интересуют.

Специально отмечу, что результаты дипломной работы и рабочие продукты дипломного проекта — это разное. Когда мы говорим о результатах, то имеем в виду параметры и характеристики практической и/или научной составляющей вашего диплома. То есть результаты являются тем, что выносится на защиту, чем вы обосновываете возможность выдачи вам университетского диплома по программированию. А рабочие продукты — это результаты (да, тоже результаты) разных видов деятельности, выполненных вами в ходе дипломного проекта. Как мы обсуждали выше, эти виды деятельности имеются, поскольку ваши результаты должны быть надлежащим образом обоснованы и оформлены.

Кроме необходимости текста, презентации и подготовленного выступления (дипломной речи) важно и то, что сами результаты практической и/или научной составляющей вашего диплома должны быть разнообразными: непросто реализация сложной функциональности или доказательство некоторой важной теоремы. В дипломной работе интересен кругозор и различные связи, ответы на многочисленные вопросы, обоснование интуитивных гипотез и так далее. Конечно, все это целесообразно делать, если выполнена основная работа, но объем затрат на получение этих дополнительных результатов может быть сопоставим с работами для получения основных результатов. Этими дополнительными результатами как раз и являются выполненные эксперименты и/или апробация, обзор сходных продуктов/ решений/технологий и так далее.

Теперь я представлю пример того, как можно организовать хорошую работу на относительно несложную тему. Недавно в компании Microsoft написали статью о том, как следует реализовывать универсальное дерево синтаксического разбора программы (Abstract Syntax Tree, AST) для мультиязыковой среды разработки (IDE, Integrated Development Environment). Реализовать такое AST по имеющейся статье является относительно простой задачей. Обозначим эту задачу/результат как основной. А теперь выделим дополнительные задачи/результаты.

1) Интеграция созданного AST с *несколькими* (например, двумя) языками программирования для того, чтобы показать, что заявленная универсальность действительно имеет место и успешно реализуема на практике.
2) Реализация двух разных IDE-сервисов на основе созданного AST с тем, чтобы показать, что созданное AST универсально не только в смысле поддержки разных языков программирования, но и в смысле разработки разных сервисов. В качестве таких сервисов можно предложить инкрементальный синтаксический анализ (parsing) и простую операцию рефакторинга.
3) Анализ других реализаций и подходов к решению этой же задачи в рамках существующих многоязыковых сред разработки и написание интересного обзора.

Таким образом, исходная несложная тема оказывается, для которой грамотно сформулированы задачи, оказывается вовсе не такой простой, очень интересной и, будучи хорошо реализованной, заслуживает высших балов при защите.

> **Важно** не то, столь сложна тема вашего диплома, а то, на сколько тщательно, всесторонне и продумано вы развернули её в задачи, а после — сколь качественно выполнили все эти задачи.

О выборе производственной темы. Мне приходилось сталкиваться с ситуациями, когда уже работающие на производстве студенты выбирают какую-то тему, не связанную с их работой. И так происходит не потому, что хотят позаниматься в рамках диплома чем-то новым и интересным (это как раз понятно и нормально). А потому, что не понимают, как оформить результаты их работы на производстве в виде диплома. И получается противоестественная ситуация — уже сформировавшийся специалист не может предъявить результаты своей профессиональной работы как свидетельство того, что он получил требуемую профессию. В случае подобных затруднений студенту лучше всего найти себе помощника из преподавателей кафедры, на которой он защищается и который

бы помог ему создать управляющую структуру диплома — постановку задачи и результаты — и оформить по сделанной практической работе хороший диплом.

А иначе часто получается, что студенту нужно будет делать для диплома что-то специальное, отдельное, не связанное с его/её работой в компании. Это, в свою очередь, может потребовать много усилий (а еще нужно и работать, и в университете учиться), и в итоге может получиться искусственный дипломный проект. А в программной инженерии важно, чтобы проекты были максимально реалистичными, близкими к жизни — и для этого есть все возможности!

Про групповые студенческие проекты. Невзирая на требование индивидуальности результатов и необходимость разделения своего и «чужого», дипломные работы в некотором смысле могут быть коллективными. Часто бывает, что несколько студентов-выпускников (например, друзей) объединяются в один проект – или на работе в компании, или студенты берут вместе один объемный исследовательский проект. Обычно такие группы состоят из двух–трёх человек (больше я бы не рекомендовал в виду сложности в организации процесса). В рамках такого проекта результаты получаются более весомыми, а работа над дипломом — более интересной. Педантично разбивать такие группы на совершенно отдельные, независимые задачи-проекты не всегда целесообразно. Но вот чётко выделить минимально пересекающиеся задачи для разных дипломов – необходимо.

Также отмечу, что в групповом дипломном проекте важно наладить адекватную проектную среду — разбиение по задачам, координацию, также хорошо, когда кто-нибудь из студентов берет на себя лидерские и/или организационные функции. Однако в этом случае нужно, чтобы у каждого участника такого проекта была своя хорошо выделенная подзадача. И вокруг этой подзадачи, кроме, собственно, её надлежащей реализации, студент выполняет микроисследование: делает обзор соответствующих существующих тулов, думает над выбором технологий реализации и возможных аналогов, проектирует и выполняет эксперименты и прочее. Объём этих дополнительных работ может быть значительным, и именно в свете этого обычный групповой программистский проект превращается в набор хороших дипломных работ. Следует также отметить, что в таких групповых проектах существуют дополнительные риски — например, групповой проект потерпел неудачу: главная идея оказалась неэффективной, неудачной и прочее, то есть не получилось сделать что-то впечатляющее. Или участники проекта не смогли найти общий язык и рабочая группа распалась. Или кто-то один всех подвёл. И так далее. Такие риски нужно учитывать и понимать, что в таком проекте все в одной ложке. Ведь аттестационная комиссия не будет разбираться в причинах провала — ей нужны положительные результаты!

> **Важно.** Групповые дипломные проекты могут быть, от них не надо сторониться. Но их следует тщательно организовывать — как эффективный проект по разработке программного обеспечения, и как работу над отдельными дипломами. Следует обратить внимание на следующие моменты.
>
> 1) Нужно чётко и ясно выделить *дипломную задачу* для каждого участника, но *реальных задач* в проекте у каждого участника может быть значительно больше, и не все их нужно включать в диплом, поскольку требуется обеспечить концептуальную целостность каждого диплома.
> 2) Вокруг выделенной дипломной задачи, кроме, собственно, её надлежащей реализации, студент выполняет микроисследование — делает обзор соответствующих существующих программных средств, обосновывает выбор технологий реализации, проектирует и выполняет эксперименты и прочее.
> 3) Однако, прежде всего, требуется хорошо выполнить (реализовать) саму выделенную дипломную задачу, а также выполнить весь проект в целом – без этого трудно говорить о хороших дипломах участников такого проекта.

4) Следовательно, в групповых дипломных проектах существуют дополнительные риски — назовём их групповыми.

Контрольные вопросы

1) Перечислите категории дипломов по программированию.
2) Дайте определение научному диплому.
3) Дайте определение исследовательского диплома.
4) Дайте определение производственному диплому.
5) В чем отличие исследовательского диплома от научного диплома?
6) В чем сходство исследовательского диплома и производственного диплома?
7) Расскажите, какие содержательные вопросы касаемо производственного диплома можно поставить, сформулировать?
8) Если у вас производственный диплом, то имеются ли у вас собственные ответы на эти вопросы?
9) Перечислите рабочие продукты проекта по выполнению дипломной работы по программированию.
10) Расскажите о выгодах групповых дипломных проектов.
11) Расскажите о выделении дипломной задачи для каждого участника группового дипломного проекта.
12) Как соотносятся дипломная задачи и реальные задачи участника дипломного проекта?
13) Расскажите о дополнительных (по отношению к самому программистскому проекту) работах каждого участника группового дипломного проекта.
14) Как вы понимаете риски группового дипломного проекта? Что, на ваш взгляд, следует предпринять, чтобы их избежать?

Список литературы

1) А.В. Ахутин. Понятие «природа» в античности и в Новое время. М. Наука. 1988.
2) Ван Дер Варден Б.Л. Пробуждающаяся наука I. Математика древнего Египта, Вавилона и Греции / Перевод с англ. М.: ГИФМЛ, 1959.; Пробуждающаяся наука II. Рождение астрономии / Перевод с англ. М.: ГИФМЛ, 1991.
3) Ф. Бэкон. Новый органон. Рипол Классик, 2018.
4) И. Ньютон. Математические начала натуральной философии (Philosophiae Naturalis Principia) / Перевод с латыни А. Н. Крылова. Собр. трудов. М. – Д., т. 7, 1936.
5) В.И. Вернадский. Избранные труды по истории науки. М.: Наука, 1988.
6) Очерки истории информатики в России. Н.: Научно-исследовательский центр ОИГГМ СО РАН, 1998.
7) Н. Винер. Кибернетика, или управление и связь в животном и машине / Перевод с англ. М.: Советское радио, 1968.
8) J. Pontus, M. Ekstedt, and I. Jacobson. Where's the theory for software engineering? IEEE Software, 2012, 29 (5): 96-96.
9) S. Alam, S. Zardari, M. Bano. Software engineering and 12 prominent sub‑areas: Comprehensive bibliometric assessment on 13 years (2007–2019). IET Software, 2022, 16 (2): 125-145.

Лекция 3. Проектирование текста диплома

В лекции излагается метод проектирования текста диплома. Сначала даётся краткое введение в проектирование в целом, а также определяется синтаксис текста диплома как высокоуровневый предмет проектирования текста. Потом вводится понятие управляющей структуры, создание которой должно помочь вам реализовать синтаксис в вашем тексте. Управляющая структура определяется как следующая триада: (i) постановка задачи, (ii) результаты работы, (iii) оглавление. Подробно рассказывается о постановке задачи (цель и раскрывающие её задачи), о результатах работы и важности создавать текст, непосредственно нацеленный именно на описание этих своих результатах, об оглавлении, включая шаблоны для производственных и исследовательских дипломов. Рассказывается также о балансировке управляющей структуры текста дипломной работе, а также обсуждаются типовые ошибки при её создании.

О проектировании:

Вдохновенный прыжок от фактов настоящего к возможностям будущего

Осуществление очень сложного акта интуиции

Целенаправленная деятельность по решению задачи

Отыскание существенных компонентов какой-либо физической структуры

Из книги Дж. К. Джонса «Инженерное и художественное конструирование»

О проектировании искусственных систем. Я хочу дать широкий контекст для предлагаемого мною метода проектирования текстов дипломов по программированию, вписав этот метод в тематику проектирования искусственных систем.

> **Проектирование** искусственных систем заключается в выполнении предварительной работы по созданию многопланового, ясного образа объекта или явления перед началом его создания. Фактически, создание объекта/явления оказывается его *воплощение* по результатам проектирования.

Ошибки проектирования, например, при строительстве больших домов, мостов, самолетов, кораблей и других искусственных систем имеют всем известные печальные последствия. В конце концов Творец, создавая этот Мир, явно действовал по некоторому Плану, поскольку я не верю, что случайным образом, стихийно мог образоваться и ясный тихий осенний день в Петербурге, и удивительные прибрежные бухты залива Петра Великого в Приморье, и сложнейший организм человека, и структура атома и химических элементов и так далее и так далее. Наблюдая всё это, можно прийти к мысли о высшей целесообразности и гармонии всего Мироздания и его отдельных частей. И хочется, чтобы человеческая активность также была бы была источником прекрасных и совершенных творений. И в частности, такими бы был ваш диплом и его текст.

Итак, *проектирование* является базовой идеей и одновременно конкретной практической деятельностью в различных сферах человеческой активности, широко применяясь при создании различных искусственных систем. Проектирование тесно связано с использованием *чертежей*, хотя и не исчерпывается этим. Чертеж — это упрощенное изображение инженерного объекта, выполненное с определенной точки зрения и опускающие многочисленные несущественные детали. Чертежи появились в Эпоху Возрождения и Новое Время. Чертежи готических соборов, средневековых замков, древнегреческих храмов, древних дворцов и египетских пирамид не сохранились. Хотя геометрия и законы перспективы, в частности, золотое сечение, активно использовались в этих постройках. Однако те фрагментарные изображений, которые сохранились с древних времен и которые представляются поклонниками теории прогресса и эволюционного (разве что, за исключением географических карт) как предвестники чертежей, не вселяют уверенности в этом. Быть может, чертежей тогда вообще не было, и искусственные системы создавали как-то иначе…

Чертежи появились во время зарождения современного научного мировоззрения – сначала как рисунки в Эпоху Возрождения, вместе с развитием живописи и расширением области интересов человека. В частности, Леонардо да Винчи оставил большое число таких рисунков и чертежей, и в Букингемском Дворце, в Лондоне имеется прекрасная коллекция, которая произвела на меня сильное впечатление, передавая широту и глубину интересов этого человека, а также демонстрируя богатство языка рисунков и чертежей…

Широкое распространение чертежи получили с конца XVIII века, когда известный французский геометр Гаспар Монж обобщил принципы изображения трёхмерных геометрических фигур в своём знаменитом труде Géométrie descriptive, который был первым систематическим изложением начертательной геометрии. Этот труд лёг в основу разработки инженерных чертежей, которые вошли в обиход инженеров (последние также появились примерно в это же время).

В Эпоху Промышленной Революции выделились инженерные профессии в машиностроении, строительстве, электротехнике и т.д. И везде активно применялись чертежи. Наверное, каждый из нас видел чертежи электропроводки, планы и схемы дома, квартиры перед постройкой или капитальным ремонтом, а также чертежи различных механизмом или их отдельных узлов.

Итак, инженерное проектирование часто связывают, и не без основания, с процессом создания чертежей, или, обобщая, — с визуализацией информации об инженерных объектах. Чертежи оказывают значительную практическую пользу потому, что ведущим каналом восприятия современного человека является зрение — до 90% информации об окружающем мире он получает посредством зрения. Поговорка о том, что лучше один раз увидеть, чем сто раз услышать, свидетельствует об этом же.

Однако к 60-м годам XX века, с ростом масштаба и гетерогенности искусственных систем, в индустриальных и исследовательских кругах возникло понимание, что проектирование — это нечто большее, чем создание чертежей. Сами системы, усложняясь, стали включать в себя разнородные компоненты, из которых далеко не каждую представлялось возможным начертить (в частности, программное обеспечение). Кроме того, предметом разработки становились не сами инженерные объекты, а сложные процессы, например, динамическая, развивающаяся система транспорта в большом городе, социальные и геополитические процессы, сложные, развивающиеся во времени проекты и так далее. Подобные искусственные системы нуждались (и продолжают нуждаться) в тщательном предварительном проектировании, и далеко не всегда чертежи могли помочь в этой деятельности, а когда могли — то лишь как средство вторичной фиксации уже найденных, созданных решений.

Важно. Создание чертежей не тождественно проектированию, поскольку предметом проектирования всё чаще оказываются не объекты, а динамические процессы.

В результате появились интересные идеи о проектировании сложных систем как процессов, а не объектов, о специальном участии человека в проектировании, а также о групповом проектировании. Там же были собрано много интересных методов, ознакомиться с которыми можно в книге Дж. Джонса «Методы проектирования».

И далее эти методы, во многом, перешли в область проектирования программного обеспечения, поскольку этот вид искусственных систем активно появлялся и развивался в это время. Возникли структурный анализ (structured analysis), объектно-ориентированный анализ и проектирование (object-oriented analysis and design), UML и т.д. Эти методы должны быть вам известны из университетских курсов, а я здесь отмечу про них лишь следующие.

Увлечение сообщества программистов стандартизацией и графическими нотациями, а также использование формальных средств для описания графических языков (метамоделирования, логических языков ограничений и пр.), привело к деградации области проектирования программного обеспечения. Были созданные сложные для понимания формальные описания различных визуальных языков и потеряна идея целостности проектирования — прежде всего, связей визуальных спецификаций с другими артефактами программных проектов (в частности, с программным кодом). В итоге создание визуальных спецификаций программного обеспечения превратилось в специальную деятельность, которую обеспечивали специальные программные инструменты, при этом и эта деятельность, и инструменты оказались плохо интегрированы в программные проекты. Кроме того, не был учтён тот факт, что проектирование ПО не может опереться на *очевидные* визуальные образы, как прочие виды инженерии, занятые созданием материальных, видимых объектов — ведь программное обеспечение нематериально, невидимо. А коль так, то версий визуализации ПО может быть много (каждый может «видеть» программное обеспечение как-то по-своему), поэтому визуальные спецификации здесь не привносят магической пользы, подобно чертежам, например, в строительстве. В свете этого не удалось сформировать культуру предварительного проектирования программного обеспечения, и эта деятельность осталась неформализованной, растворённой и тесно переплетённой непосредственно с разработкой ПО. Однако она, отделённая от излишнего применения визуального моделирования, по-прежнему является важной и востребованной, о чём говорят позиции архитекторов, имеющиеся в различных компаниях по производству ПО (IBM, Microsoft, Huawei и прочих).

В этом же процессе поиска путеводных нитей в море информации возникала специальная наука и область деятельности под название инженерия знаний (knowledge management). Последняя ориентируется на разработку и использование системных подходов к работе с различной информацией, уделяя значительное внимание бизнесу и производству. Наконец, следует упомянуть про интеллект-карты (mind maps), широко применяемые для визуализации, проектирования и работы в повседневной жизни: Тони Бьюзон, автор этого подхода, считает, что радиальная структура информации соответствует структуре нейронов в коре головного мозга человека.

В целом следует отметить, что дальнейшее развитие идей и методов проектирования после 60-х годов (исключая проектирование программного обеспечения, но об этом было рассказано выше) *застопорилось*. Аналогичное происходило и с другими областями, например, семиотикой (универсальной науки о знаковых системах), кибернетикой (унифицированной наукой об управлении на основе математики) и др. По всей видимости, во всех этих случаях исследователи выходили на новые рубежи, где требовались совершенно иные, нетрадиционные подходы. В частности, в области

проектирования дальнейший прогресс лежит не в области средств визуализации (в частности, прояснении того, как же всё таки изображать программное обеспечение), ни в групповых методах проектирования, как считал Дж. Джонс. Сложные и успешные, и в то же время гармоничные рукотворные творения, с которыми я сталкивался на практике, создавались людьми, могущими вместить в свое сознание большой объем сложной информации, обладающих знаниями и опытом в соответствующей области, а также имеющих интуицию, воображение и … талант. И если всё таки говорить о развитии методов проектирования, то перспективным кажется движение в направлении развития человека и его внутренних инструментов с тем, чтобы он был способен *мочь*, *иметь возможность* проектировать действительно сложные системы, гармонично привнося в Мир свою уникальность на радость Мирозданию и людям.

О синтаксисе и проектировании текста диплома. Теперь перейдём к задаче проектирования текстов дипломов. Речь идёт о проектировании сложного объекта, с разработкой которого вы столкнулись впервые. Таким образом, налицо действительно непростая ситуация, к которой следует подходить нетривиально, поскольку простые методы не подходят для сложных, непростых ситуаций. Тщательное предварительное проектирование текста перед тем, как приступать к его написанию как раз и является ответом на этот вызов.

Сделаю еще одну предварительную ремарку. Многие студенты считают, что главная задача текста диплома — объяснить всевозможные детали и особенности выполненной работы. Но это совсем не так. Перед тем, как читатель погрузится (если погрузится) в эти детали, он хотел бы понять, о чём работа в принципе и понять, что же было сделано. И даже если он начал читать всё целиком, ему также нужна помощь в том, чтобы сориентироваться в вашем тексте, изучая детали и при этом владея общей картиной. Складывать общую картину из большого числа деталей — мучительно для читателя.

То, что сможет решить обозначенные выше задачи, можно назвать специальным *синтаксисом* текста. Давайте посмотрим на составные части синтаксиса.

- Краткое описание предметной области вашей работы — то есть ответ на вопрос «Про что это всё» (это «Введение»).
- Мотивация вашей работы — то, зачем она выполнена, кому она нужна (это тоже «Введение»).
- Постановка задачи работы — это то, что планировалось сделать (разделы «Введение» и «Постановка задачи»).
- Что было сделано — конкретно, по пунктам, без тумана и общих слов, кратко и в то же время содержательно (как правило это даётся в «Заключении»).
- Понятное оглавление, которое коррелирует с постановкой задачи и результатами.
- Краткое описание основного содержания работы (это, как правило, описывается в главе под названием «Архитектура»).

Очень важно иметь в тексте этот синтаксис хорошо проработанным. А для этого у вас должна иметься исходная ясность — в том смысле, что она предваряет написание текста. То есть у вас всё это разложено по полочкам перед тем, как вы начинаете писать текст. Тогда эта ясность будет пронизывать ваш текст и позволит обеспечить, как минимум, два способа прочтения вашего диплома: первый способ — ознакомительный, для уяснения так сказать, общего сюжета, совершенно без деталей; второй способ — для специалистов, которым, как раз интересны детали.

Наивно полагать, что специалисты быстро во всём разберутся, каким бы ни был ваш текст. Я сам с большим интересом, но быстро наступающим разочарованием много раз читал тексты бакалаврских дипломов с многообещающими названиями, свидетельствовавшими о том, что данные дипломы принадлежат сфере моих научных и практических интересов – но разобраться в текстах было невозможно.

Хочу обратить внимание на то, что одну и ту же информацию, одну и ту же сделанную работу можно описать в тексте очень по-разному. Пусть, например, студент расширил функциональность некоторой среды разработки/фреймворка, реализовав для него некоторую дополнительную функциональность. В этом случае следует объяснить, зачем в принципе нужна такая новая среда/фреймворк, и далее — зачем в ней нужна данная функциональность. А потом объяснить самое интересное: что же конкретно сделал студент? Причём реализованная функциональность — это далеко не всё, что требуется: а эксперименты, а интеграционные вопросы, а обоснование выбора технологий, алгоритмов, методов, а общий дизайн приложения (то есть архитектура)? Например, студент может объявить, что он создал некий новый метод и выполнил его реализацию — так часто и формулируют подобные результаты. Однако такая формулировка запутывает суть в угоду наукообразию, в результате которого синтаксис работы оказывается запутанным и неясным.

Для того, чтобы воплотить этот синтаксис в тексте, сделать его легко воспринимаемым, я предлагаю вам специально поработать перед началом писательской активности — спроектировать текст, осмыслив (возможно даже переосмыслив) сделанную вами практическую работу, которая к моменту написания текста либо закончена, либо может быть не завершена до конца, но уже получены основные результаты. (Приступать к проектированию и написанию текста диплома в ситуации, когда вы находитесь лишь в самом начале своего дипломного проекта бессмысленно).

Выше определённый синтаксис текста является лишь позиционной, верхнеуровневой идеей. Для её практической реализации введём понятие *управляющей структуры текста диплома*. Предварительно определим её как связную, сбалансированную триаду, состоящую из следующих элементов:

- постановку задачи,
- результаты и
- оглавление.

Сразу хочу подчеркнуть, что управляющая структура существенно шире оглавления или «плана сочинения». Когда я учился в школе, то у нас на уроках литературы требовали создавать предварительные планы для сочинений. Считалось самоочевидным, что это помогает написать хороший текст: то есть здесь исходно содержалась та же идея — предварительное проектирование текста. Но эта идея осталась не раскрытой и, фактически, план требовался лишь для того, чтобы он был — без плана снижали оценку за сочинение. В результате я так и не понял, зачем он нужен, не убедился в том, как именно он мне помогает писать сочинение, не научился грамотного его составлять.

Рассмотрим детально каждый элемент этой триады, а потом дадим более полное определение управляющей структуры и рассмотрим, как выглядит процесс её разработки.

Постановка задачи дипломной работы включает одну главную, основную *цель* работы. Следуя словарю С.И. Ожёгова, *цель — это стремление, другими словами то, что надо, желательно осуществить*.

> **Целью дипломной работы** является некоторая короткой формула того, что нужно было сделать; эта формула состоит из одного предложения.

В качестве примера можно привести следующую цель производственного диплома — разработка подсистемы динамического символьного исполнения на основе символьной виртуальной машины V#, ориентированной на платформу .NET.

Очень важно, что в качестве цели вы приводите именно ёмкую формулу, и это в самом деле одно предложение. Аналогичная ситуация имеется при подготовке патентов на изобретения — точно также там требуется в одном предложении представить формулу

предлагаемого (патентуемого) изобретения, и порой это предложение оказывается очень длинным. В нашем же случае важно, чтобы в рамках определения цели своей работы вы не пустились бы в пояснения и объяснения, не начали давать определения каким-либо понятиям и так далее — все это следует делать во введении, поскольку оно как раз и предназначено для описания нужного контекста постановки задачи.

> **Важно.** Цель вашей дипломной работы не должна копировать заголовок. Иначе то получается досадная тавтология.

А студенты часто делают именно так, экономя время и силы. Я очень много раз делал такого рода замечания, и чтобы их успешно реализовывать, вам потребуется как следует включиться. Потому как название диплома все таки должно быть кратким — я не сторонник длинных, в несколько строк, названий, из которых как будто бы должно быть полностью понятно всё про работу. А вот цель работы должна содержать максимально полную информацию о работе и может быть довольно объёмной, но при этом лучше вложиться в одно предложение (формула!).

Недавно, со мной произошла следующая ситуация. Я предложил одному студенту расширить цель его работы с тем, чтобы она перестала быть полной калькой с названия его работы. В ответ на это моё предложение он точечно перефразировал в цели своего диплома некоторые понятия, даже не увеличив общего числа слов, а также оставил неизменной структуры предложения.

Далее, для осуществления цели выделяют набор задач. Согласно тому же словарю С.И. Ожёгова, *задача — это то, что требует исполнения*.

> **Задачами дипломной работы** является описание конкретных работ, которые планируются в начале и должны быть выполнены в процессе работы над дипломом.

Особо подчеркну, что задач работы, в отличие от цели, должно быть несколько, т.е. больше, чем одна. Оптимально, когда имеется три–пять задач. Если их меньше трёх, то постановка задачи оказывается недостаточно проработанной, если больше пяти — то постановка задачи превращается в детальное техническое задание к программной системе (а у нас речь идёт о тексте дипломной работы, если кто забыл!). Важно, что задачи оформляются непросто текстом–описанием, а списком–перечислением. Так делается для удобства обсуждения и дальнейшей трассировки задач и результатов в том числе и членами аттестационной комиссии на вашей защите. Каждая задача является небольшим текстом размером в одну–три строки.

> **Важно**. И цель, и задачи должны относится именно к вашей работе, но не ко всему проекту, если вы работали в команде. Важно выделить именно вашу часть, возможно, при этом включив в диплом не всё, что вы сделали в рамках своего дипломного проекта.

> **Важно**. Не бойтесь исключить часть сделанной вами работы из диплома. Не набивайте его под завязку, в ущерб концептуальной целостности. Последняя для диплома очень важна!

В качестве примера приведём набросок задач, раскрывающих приведённую выше цель.

- Выполнить обзор существующих систем …
- Разработать требования к системе ….
- Спроектировать архитектуру системы ….
- Реализовать систему ….
- Выполнить тестирование и апробацию системы ….

При этом я не рекомендую очень подробно описывать задачи, составляя своего рода техническое задание к системе. А то трудно будет писать содержательные результаты.

Результаты дипломной работы. Задачам диплома взаимно–однозначно соответствуют её *результаты*: вы формулируйте результаты в виде отчёта по каждой задаче, кратко объясняя, как вы каждую из них реализовали. Результаты диплома очень важны, поскольку они являются тем, что выносится на защиту. Таким образом, защищая диплом, вы предъявляете именно результаты, то есть то, что вы в итоге сделали, получили, чего вы достигли.

> **Важно**, что результаты вашего диплома — это именно *ваши результаты*, а не результаты всего вашего индустриального проекта, или группового дипломного проекта, который вы выполняли совместно с вашими товарищами. При этом не обязательно включать в результаты диплома всё, что вы сделали в проекте. Следует понимать, что диплом не является отчётом по проекту для представления в компании, диплом — это выпускная работа, выполняемая и защищаемая в университете.

Нужно, чтобы результаты, также как и задачи, образовывали список, где каждый пункт представлен небольшим объемом текста — примерно в две–пять строк. В этом списке можно делать подпункты, но не злоупотребляйте этим: я рекомендую иметь ни более одной порции подпунктов для какого-то одного объёмного результата (две порции — это совсем редкий случай).

> **Важно.** Оформленные списком результаты целесообразно поместить в заключение вашего текста, а также на отдельный слайд презентации, которую вы будете докладывать на защите.

При этом список результатов (также как и список задач) должен целиком помещаться на один слайд в вашей презентации — то есть данный список не должен быть очень длинным, а его пункты — чрезмерно объёмными. Это именно краткая выжимка, но при этом конкретная и содержательная. Последнее также существенно, так как многие студенты затрудняются при создании такой содержательной выжимки и часто в виде результатов представляют задачи, слегка поменяв фразы.

> **Важно.** Результаты не должны быть калькой с задач, хотя такой вариант и является вариантом взаимно-однозначного соответствия между ними.

Поговорим отдельно о содержательности результатов, точнее, о содержательности их краткого представления. Итак, ещё раз подчеркну, что в результатах очень важна конкретность, специфичность, содержательность. Например, если вы писали в одной из задач, что собираетесь сделать обзор алгоритмов, решающих некоторую проблему, то в соответствующем результате вам надлежит перечислить рассмотренные вами алгоритмы, да ещё кратко сформулировать выводы из этого обзора. В общем-то, именно эти выводы и являются мотивацией вашей работы — созданный вами алгоритм должен полностью или частично закрывать брешь в существующих результатах. Или вы описывайте результат, посвящённый реализации вашей системы — укажите использованные языки и среды программирования, кратко укажите решённые проблемы. Или если у вас имеется экспериментальные результаты, то кратко опишите их основные параметры — использованные наборы данных, метрики, существующие алгоритмы/системы, с которыми вы сравнивались и краткие выводы по экспериментам.

> **Результат дипломной работы** — это краткое и вместе с тем содержательное описание реализованных вами задач, содержащее специфичные детали и конкретные характеристики.

Хочу ещё раз подчеркнуть, что результаты являются непросто формальным артефактом, а составляют краткую выжимку, суть проделанной вами работы. Тут важно и то, что это именно суть, и то, что она выражена кратко. Ведь суть — это не общие слова, которые могут подойти не только к вашей работе, но и ко многим другим. Суть означает краткое описание полученных вами результатов, используя конкретные *характеристики*. Наличие таких характеристик является важным, ибо если написать, например, что тестирование успешно выполнено — то это как раз и есть общие слова. А, например, указание видов тестов, процента тестового покрытия, времени исполнения тестов (и при этом, чтобы все эти характеристики не были бы стопроцентно положительными) — это как раз и есть краткое и содержательное описание результатов тестирования. При этом важно кратко выразить суть, так как результаты должны быть легко доступными и обозримыми, поскольку аттестационной комиссии должно быть «понятно в сборе» то, что вы сделали в своём дипломе.

> **Важно.** При описании результатов не следует использовать неопределенные слова — некоторые свойства, несколько проектов, ряд методов и т.д. Наоборот, следует указывать конкретное число, перечислять поименно, и давать краткие характеристики, а также делать ясные и содержательные выводы.

Рассмотрим, как могут выглядеть результаты для уже рассмотренного шаблона оглавления. Поскольку текст дипломной работы описывает результаты, причем максимально однозначно и легко трассируемо, то в этом случае имеем список результатов, представленных ниже.

- Выполнен обзор следующих существующих систем…
- Разработаны требования к системе…
- Спроектирована архитектура системы…
- В соответствии с созданной архитектурой выполнена реализация системы…
- Выполнено тестирование и апробация системы…

Теперь я опишу те характеристики, которые важны при описании различного вида результатов. Причём я считаю, что описывать эти характеристики имеет смысл в том самом кратком описании результатов, о котором я толкую.

- Когда вы формулируете результат выполнения обзорной задачи, вам следует перечислить все те системы, которые вы рассмотрели. Это та конкретика, которая отсутствовала в соответствующе задаче и которая важна для понимания того, что же вы сделали в рамках этого результата. Также важно сформулировать какие-то выводы относительно рассмотренных систем, поскольку они, эти выводы, служат мостиком от обзора к вашей работе — имеются проблемы у существующих систем, и именно эти проблемы закрывает ваша разработка! Аналогично, вместо систем могут выступать существующие алгоритмы, методы и так далее.
- Если у вас есть результат про требования, то он должен включать как функциональные, так и нефункциональные требования, и последние должны определять требования по безопасности, быстродействию и удобству дальнейшего сопровождения системы и так далее.
- Важно, чтобы результат по архитектуре содержал перечень и краткие характеристики целостных свойств вашей системы (смотрите лекцию про архитектуру и реализацию). Вы можете, например, упомянуть, что вы создали

клиент-серверную или микросервисную архитектуру, что серверная часть реализована с использование многопоточности для увеличения производительности; вы можете указать, какие сторонние библиотеки и фреймворки вы использовали, упомянуть про главные компоненты вашей системы и их интерфейсы. Важно, что всё это надо написать кратко и одновременно содержательно.

- При описании реализации можно перечислить список трудностей, которые вы преодолели (смотрите лекцию про архитектуру и реализацию), можно также дать перечень реализованной функциональности. Наконец, упомяните про использованные языки и среды реализации.
- Описывая тестирование и апробацию вашей системы, пожалуйста, объясните, сколько тестов вы создали и как (например, это были синтетические тесты или какие-то пользовательские истории), откуда вы брали тестовые данные, каковы оказались результаты тестирования, были ли отзывы пользователей и удалось ли вам реализовать их замечания.

Очевидно ли вам теперь, чем отличаются результаты от задач?!

Оглавление. Теперь перейдём к оглавлению текста вашей работы. Оглавление состоит из глав и разделов. Я не рекомендую делать в оглавлении больше, чем два уровня, иначе в каждом разделе объём текста может оказаться совсем небольшим, вплоть до одного абзаца. Тут следует иметь в виду, что абсолютная структура иссушает текст — разделы должны содержать достаточное количество информации, иначе вместо текста получается нетекст.

Теперь я приведу несколько шаблонов оглавления дипломов по программированию.

Шаблон № 1 (для производственных дипломов).

- Введение.
- Постановка задачи.
- Глава 1. Обзор.
- Глава 2. Требования.
- Глава 3. Архитектура.
- Глава 4. Особенности реализации.
- Глава 5. Тестирование и внедрение.
- Заключение.
- Список литературы.

Сразу постараюсь ответить на главный вопрос, который мне часто задают студенты — почему этот шаблон не учитывает особенности созданной системы? То есть почему в этом шаблоне оглавление не формируется по функциональности системы, почему нет глав про обзор предметной области, которые соответственно называются, например «Системы сборки», «Задачи и проблемы трансляции видео в Интернете» и т.д. Мне кажется, что задача шаблона – предложить некий язык для использования в разных ситуациях. В данном случае единый язык важен в виду огромного разнообразия предметных областей в программировании – легко может произойти ситуация, что на кафедре нет специалистов, которые хотя бы даже слышали о предметной области, которой принадлежит некоторый диплом, и в этом нет ничего зазорного! С другой стороны, кафедра должна оценить каждый конкретный диплом – прослушать презентацию и что-то уяснить, пролистать текст работы (именно пролистать) и понять в некотором приближении, что же было сделано выпускником. Далее, если в оглавлении присутствует обилие терминов из вашей предметной области, то это зачастую запутывает читателя (даже искушенного) — оказывается неясным, про что данная глава, является ли она обзорной или описывает некоторый результат работы, и если верно последнее, то какой именно результат она представляет? Напомню, что целесообразно максимально расширить круг потенциальных

читателей дипломных текстов, поскольку в настоящее время тексты дипломов активно выкладываются в Интернет и отдельные работы могут быть найдены через Интернет-браузер, путём поиска по ключевым словам; хороший шаблон, который при этом осознанно используется (а вот с этим бывает много проблем…) делает текст, исходя из моего собственного опыта, существенно, существенно понятнее для беглого просмотра.

Теперь перейдём непосредственно к обсуждению самого шаблона.

Отмечу, что главы про требования может не быть, поскольку требования вам были сообщены готовыми. Если же это не так, и вы потратили значительные усилия на их выявление, то тогда пишите такую главу. Иначе эту главу следует опустить.

Глава про архитектуру созданной вами системы нужна для того, чтобы представить ваше решение целиком: и его мотивацию и назначение, и его функциональность, включая основной сценарий, и разбиение на компоненты, и основные архитектурные решения. Иначе читателю не представляется возможным понять, какова ваша система в сборе.

Глава про особенности реализации вашей системы важна для описания реализации отдельных компонент и/или отдельных локальных трудностей (особенностей), с которыми вы столкнулись. Эта глава показывает нетривиальный характер проделанной вами программисткой работы.

Глава про тестирование и внедрение даёт представление читателю о том, на сколько зрел ваш продукт, каково его состояние: ведь под словами «разработана система» может скрываться широкий спектр вариантов — от небольшой программки, написанной автором для себя до продукта, который активно продается и имеет множество пользователей. Описание процесса тестирования, тестов, процесса внедрения, обратной связи от пользователей и её учёта — всё это и показывает зрелось вашей системы.

Отмечу, что именно такое структурирование диплома отвечает его задачам — показать, что студент разбирается в программной инженерии и способен выполнять различные виды деятельности в индустрии, помимо кодирования. Напротив, структурирование диплома сообразно функциональности созданной системы оставляет вне рассмотрения множество вопросов, а также создаёт трудности для восприятия текста.

Шаблон № 2 (для исследовательских дипломов).

- Введение.
- Постановка задачи.
- Глава 1. Обзор.
- Глава 2. Метод/алгоритм/подход.
- Глава 3. Реализация
- Глава 4. Эксперименты.
- Заключение.
- Список литературы.

Исследовательские дипломы очень близки к производственным, поэтому и шаблоны оказываются похожими. Только вместо требований идет исследовательская составляющая — созданный алгоритм/метод/подход.

При этом архитектура с реализацией могут схлопнуться в одну главу, поскольку программирования в таком дипломе может оказаться меньше, чем в производственном, поскольку автор потратился ещё и на исследования. А, следовательно, код устроен не слишком сложно, и можно обойтись без архитектуры.

Глава про тестирование и внедрение превратилась в главу про эксперименты, и это неслучайно. Предложенные результаты некоторого исследования, для которых автор выполнил программную реализацию, нужно сравнить с аналогами и/или измерить их качество. Первое не всегда удаётся сделать, поскольку исходный аналогов может быть недоступен, а сами они могут быть встроенными в большие коммерческие продукты, из которых крайне непросто вычленить нужные функции. Тем не менее, качество результата, как правило, можно измерить, применяя, например, такие понятия как точность (precision)

полнота (recall), а также аппарат несложной математической статистики. Возможны и другие подходы к обоснованию того, что у автора получилось что-то интересное и стоящее, но важно, чтобы они опирались на что-либо измеряемое, реальное и объективное.

Для научных дипломов мы не буем приводить шаблона, так как такие дипломы бывают очень разными: существует значительное количество разных научные областей, связанных с программированием — Computer Science, искусственный интеллект, базы данных, языки программирования, телекоммуникационные системы, информационные системы, программная инженерия и так далее, и каждая из этих областей имеет значительное число подобластей. При этом и методы исследования могут существенно разниться, например:

- могут доказываться некоторые теоремы (в том числе с использование популярного сегодня языка автоматических доказательств Coq);
- могут разрабатываться новые алгоритмы, представляемы более или менее формально, а также для низ могут формально доказываться свойства — безопасности, корректности, достижимости, вычислять сложность и так далее;
- вся дипломная работа может быть посвящена экспериментальному исследованию некоторых существующих алгоритмов, включая более или менее сложную статистику;
- могут быть реализованы экспериментальные расширения для некоторых языков программирования;
- может быть предложена новая идея или новый метод для разработки некоторого специфичного программного обеспечения с выполнением пилотной реализации и так далее.

Мне кажется, что часто встречающееся в российской академической школе мнение о том, что наука — это то, где есть теоремы, является несостоятельным. В программной инженерии и языках программирования часто отсутствуют теоремы, и тем не менее исследования и их результаты оказываются вполне научными. Ещё раз подчеркну, что научный диплом определяется постановкой задачи, вытекающей из соответствующего научного сообщества. Именно это, а не абстрактная научность или использование математики решают здесь всё, поскольку, как я рассказывал выше, общественный, коллективный аспект науки чрезвычайно важен. Хотя иногда здесь встречаются исключения, например, предлагается радикально новая идея, не вписывающаяся в существующие контексты научных сообществ. Тогда требуется большая тщательность по разработке и представлению этой идеи в рамках диплома, поскольку иначе никто ничего не поймёт и у вас будут проблемы.

> **Важно.** Хорошо, если удается заранее выполнить как саму работу, так и написать текст. И дать почитать этот текст другим людям, собрать их обратную связь и существенно улучшить текст. Целесообразно дать тексту диплома «выстояться», а не сдавать сырой текст, написанный второпях, накануне.

Балансировка управляющей структуры. Итак, теперь всё готово для того, чтобы сформулировать точное определение управляющей структуре текста дипломной работы и детально его обсудить.

> **Управляющая структура** дипломной работы является сбалансированной триадой, включающей в себя *постановку задачи* (цель и задачи), *результаты* и *оглавление*. При этом каждый элемент, являясь множеством элементов, взаимно-однозначно отображается на другой, и в итоге имеется трассировка любой задачи на некоторый результат и далее на определенную главу, где эта задача/результат описана.

Идея такого тройственного взаимно-однозначного отображения представлена на рисунке 2, а соответствующий пример приводится на рисунке 3[2]. Таким образом каждой задаче соответствует один и только один результат, и поэтому можно легко увидеть, как именно была реализована данная задача, что важно для аттестационной комиссии, оценивающей ваш диплом. Далее, каждый результат оказывается описанным в подходящей главе вашего текста. При этом в одной главе описан один результат, и отсутствует задача/результат, которой не соответствует никакой главы. Такая фокусировка текста на описание результатов важна, ибо текст диплома должен описывать результаты и только результаты: большой ошибкой является написать диплом не о результатах, а о чём-то другом, например, о предметной области или о каких-то второстепенных деталях реализации. Кроме того, члены аттестационной комиссии смогут быстро трассировать описание результатов в тексте вашей работы, а не выискивать их описание где-то в гуще текста, что может оказаться невозможным без полного прочтения всего текста диплома. Бывает часто, что если управляющая структура плохо спроектирована, то некоторые результаты остаются неописанными и наоборот, нечто выносится в результаты, являясь лишь атрибутами или отдельными характеристиками…

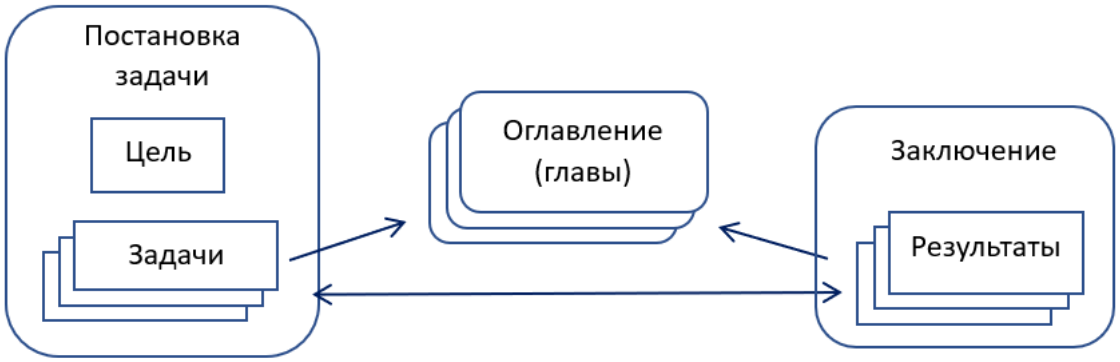


Рис .2. Схема управляющей структуры текста диплома

Сделаю небольшое философское отступление. Управляющая структура является триадой (треугольником). Общеизвестно, что триада является устойчивой фигурой в отличие от диад (пар), поскольку последние жестко поляризуют сложные явления. Как писал известный математик и философ В. В. Налимов, европейская культура существенно опирается на диады, например, добро/зло, истина/ложь, демократия/тирания, материализм/идеализм и другие. Однако, опора на диады, по свидетельству многих мыслителей, ведет к противоречивости и неразрешимостям (например, В.В. Налимов отмечал, что семантический континуум не разбивается аккуратно на дихотомические пары). В связи с этим можно отметить, что триады более устойчивы, поскольку третий элемент не даёт двум другим полярно противопоставиться, обособиться и контрпродуктивно поляризоваться. Неслучайно, например, у Платона одним из основных строительных элементов Мироздания являлись треугольники (смотрите диалог Платона «Тимей»).

Управляющая структура создается непросто с помощью copy/paste задач в результаты и далее подобным же образом — в оглавление, всего лишь с небольшой попутной стилистической доработкой (а именно таким образом большинство студентов начинают работать над управляющей структурой своего диплома). Процесс создания хорошей управляющей структуры оказывается сложнее, и для его успеха требуется сделать несколько итераций. Всё начинается с задач, но далее приходится их исправлять, например, потому, что может оказаться, что результаты лучше структурировать иначе, но

[2] Данный пример взят из диплома Дмитрия Копина «Поиск клонов в XML-документации» (дипломная работа специалиста, СПбГУ, кафедра информатики, 2013).

при этом необходимо сохранить взаимно-однозначное соответствие задач и результатов. Далее, результаты могут измениться в виду того, что при соответствующей результатам структуре оглавления очень неудобно писать текст, например, одна глава получается очень объёмной, а на другую не хватает материала. Но, изменив результаты, вам следует изменить, соответственно, и задачи. При этом, как я уже писал выше, постановка задачи не является чем-то неизменным и преопределённым с самого начала. Данный итеративный процесс будем называть балансировкой.

> **Балансировка управляющей структуры** — это учитывание нескольких (более одного) параметров при разработке управляющей структуры с поддержанием взаимно-однозначного отображения каждой вершина триады на все другие; этими параметрами являются (i) ясность постановки задачи, (ii) полнота и лаконичность результатов, (iii) естественность разбиения текста на главы и некоторые другие.

Зачастую студентам оказывается сложным учитывать более одного параметра при создании чего бы то ни было, в том числе и управляющей структуры: то задачи оказываются запутанными и очень контекстно зависимыми, то результаты — формальными и «невесомыми», то оглавление — экзотическим, трудно воспринимаемым или разбивающим текст на неравновесные части.

Отмечу, что иногда результаты понимают как некоторые важные атрибуты сделанной работы — например, достигнута высокая производительность алгоритма, решающего данную задачу, и именно этот факт выносится в отдельный результат. Мне кажется, что в этом случае результатом является сам алгоритм, а не его хорошая производительность, даже если алгоритм и разрабатывался ради неё; его высокую производительность можно отметить в описании данного результата.

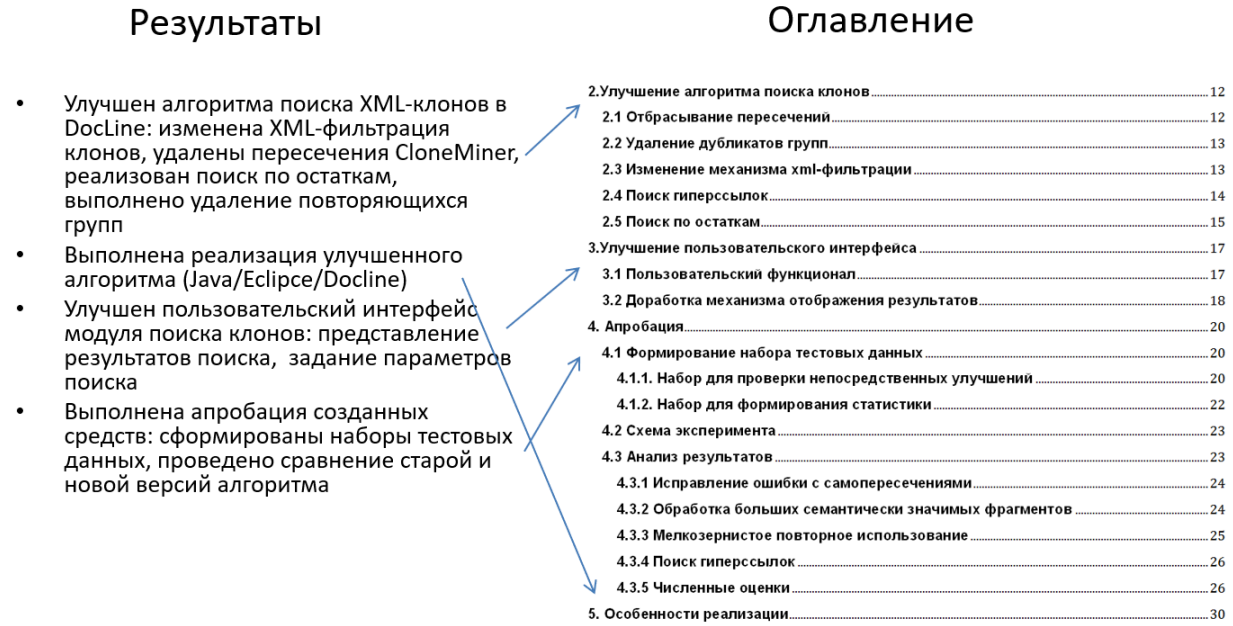


Рис. 3. Пример взаимно-однозначного отображения результатов на оглавление

Наконец, отметим, что взаимно-однозначное соответствие в триаде может иногда не соблюдаться. Например, в качестве результата вы отдельно отметили выполненную публикацию или полученный грант — зачем такой результат описывать в отдельной главе?

Также бывает удобно иметь для двух результатов разные разделы в одной и той же главе, поскольку эти разделы не могут составить полноценны отдельных глав. Встречаются и другие варианты. Но тут следует помнить, что любое правило всегда можно нарушить, поскольку жизнь часто течёт мимо правил. Но эти отдельные факты правил не отменяют! Известна поговорка, что исключение лишь подтверждает правило.

Типовые ошибки в управляющей структуре. Я уже обозначил выше некоторые ошибки при составлении управляющей структуры. Продолжу это делать и отмечу другие ошибки, с которыми я часто сталкивался в надежде, что вы сможете избежать хотя бы некоторых из них.

1) Задачи оказываются планом работ или детальным техническим заданием. В этом месте студентам оказывается трудно поиграть, вникнув в логику «научной» игры (смотрите мои размышления о постановке задачи в предыдущей лекции).
2) Результаты являются полной копией задач. Тут мне приходится давать многочисленные замечания, фактически, в каждому студенту, которому я помогаю спроектировать управляющую структуру. Студентам оказывается непросто лаконично, но ёмко сформулировать свои результаты, выделив при этом значимые атрибуты результатов и избегнув голословных констатаций об успешности, эффективности, оптимальности и т.д.
3) В оглавлении часто оказывается глава, которая содержит всего один подраздел. В таком случаем, по моему мнению, подраздел не нужен — получается, что в главе нет другой заявленной информации.
4) После получения первого варианта многие застревают и не могут внести необходимые изменения. Мне кажется, что тут срабатывает «однопроходный» шаблон мышления — если я что-то сделал, затратив какое-то количество усилий, то именно так, безо всяких изменений, будет теперь всегда; и мотиваций на затрату дополнительных усилий у меня нет. У многих студентов я часто наблюдаю именно такой шаблон не только при создании управляющей структуры текста диплома…
5) Несогласованность терминологии — как внутри одного элемента триады, так и между разными элементами. Потом эта несогласованность, перейдя в основной текст, даст обильные плоды, но не те, какие нужно.
6) Отсутствие синтаксически-текстуального подобия между соответствующими элементами в разных триадах (например, как результат рьяной борьбы с copy/paste). Однако нужно, чтобы читатель мог бы осуществить трассировку по разным элементам триады визуально, то есть узнал бы главу, соответствующие определенному результату, непосредственно по её названию.
7) Наличие различных ошибок в постановке задачи/результатах/оглавлении —пунктуационных, грамматических, а также орфографических. Хочу заметить, что если вы даже управляющую структуру не можете сделать без таких ошибок, то мало шансов создать хороший текст диплома…

Общие замечания и промежуточные итоги. Заканчивая обсуждение управляющей структуры, хочу отметить следующее: хорошая структура очень важна для того, чтобы текст вашего диплома состоялся.

Например, в авионике (разработка систем управления и обслуживания самолетов), согласно стандарту DO-178B/C, существует следующая иерархия компонент системы по степени безопасности: (i) катастрофический уровень (catastrophic), (ii) опасный уровень (hazardous), (iii) важный уровень (major), (iv) незначительный уровень (minor), (v) не важный (No Safety Effect). Безопасность означает угрозу жизни людей при проявлении ошибок в компонентах. Соответственно, тщательность разработки и затраты для разработки компонент каждого уровня оказываются различными.

Так вот, я считаю, что для текстов дипломов управляющая структура, в соответствии со стандартом DO-178B/C, принадлежит катастрофическому уровню — ошибки, недоработки и несуразицы, допущенные тут, радикально влияют на весь текст. Невнятная управляющая структура приводит к тому, что ваш текст оказывается набором слабо связанных фактов, изложенных смутно и путанно. Поэтому предлагаю вам, не скупясь, как следует вложиться в проектирование текста вашего диплома, в создание управляющей структуры, подойдя сложно (в благородном понимании) к этой деятельности, в частности, выполнив несколько итераций в работе над управляющей структурой (помните про балансировку!).

Предлагаю вам подойти к разработке управляющей структуре как к проектированию — то есть вы создаёте в*и*дение текста, его сжатый образ текста, задаёте основные измерения, параметры вашего текста. И далее, в процессе написания, вы реализуете это в*и*дение, значительно упрощая процесс разработки и существенно повышая его качество. Таким образом в результате должно стать легче писать текста, а удовлетворение от того, что получается должно нарастать. Очень, очень важно не остаться снаружи своего текста, вяло создавая какую-то управляющую структуру…

Контрольные вопросы

1) Дайте различные определения проектирования. В чем их сходство и различия?
2) Приведите различные примеры проектирования в различных видах деятельности человека.
3) Как вы поняли роль чертежей и, шире, визуализации, при проектировании искусственных систем?
4) Почему проектирование не исчерпывается создание чертежей?
5) Расскажите о причинах неудачи широкого использования методов проектирования программного обеспечения на основе чертежей (визуальных моделей).
6) Как вы поняли, что такое синтаксис дипломной работы?
7) Можете ли вы объяснить, зачем синтаксис нужен?
8) Дайте определение проектирования текста диплома.
9) Что такое управляющая структура текста диплома?
10) Чем она отличается от плана текста?
11) Как синтаксис диплома соотносится с управляющей структурой?
12) Какие составные части включает в себя постановка задачи?
13) Какова роль введения относительно постановки задачи?
14) Дайте определение цели дипломной работы.
15) Какова часто встречающаяся ошибка при составлении цели работы?
16) Дайте определение задач дипломной работы.
17) Почему цель диплома одна, а задач много?
18) Что такое результаты дипломной работы?
19) Чем результаты диплома отличаются от соответствующих задач?
20) Как оглавление текста диплома связано с его результатами?
21) Что означает взаимно-однозначное отображение друг на друга элементов триады управляющей структуры?
22) В чём смысл взаимно-однозначного отображения друг на друга элементов триады управляющей структуры?
23) Что такое балансировка управляющей структуры диплома?
24) Опишите шаблон производственного диплома.
25) Опишите шаблон исследовательского диплома.
26) На сколько вы прониклись значением управляющей структуры для успеха вашего текста?
27) Расскажите о балансировке управляющей структуры диплома.
28) Расскажите о типовых ошибках при создании управляющей структуры
29) Как вы поняли, создание управляющей структуры исчерпывает проектирование текста диплома?
30) В чем, на ваш взгляд, выражается критичность управляющей структуры при создании текста диплома (смотрите аналогию с критичными компонентами в авионике)?

Список литературы

1) С.И. Ожёгов. Словарь русского языка. М. 1989.
2) Дж. К. Джонс. Инженерное и художественное конструирование /Пер. с англ. М.: Мир, 1976.
3) Дж.К. Джонс. Методы проектирования /Пер. с англ. М.: Мир, 1983.
4) Т. Бьюзен. Супермышление /Пер. с англ. Мн.: ООО «Попурри», 2003.

5) Т.А. Гаврилова, Д. В. Кудрявцев, Д. И. Муромцев. Инженерия знаний. Модели и методы. Санкт-Петербург, 2016.
6) Д.А. Марка, К. МакГоуэн. Методология структурного анализа и проектирования SADT /Пер. с англ. М.: Мета Технология, 1993.
7) Software Considerations in Airborne Systems and Equipment Certification. DO-178C. Federal Aviation Administration. 2012.
8) В.В. Налимов. Спонтанность сознания. Водолей Publisher, Томск-Москва. 2007.
9) К.И. Вигерс. Разработка требований к программному обеспечению / Пер. с англ. Русская редакция, 2004.

Лекция 4. Название и введение

В лекции рассказывается об особенностях формулировки названия диплома, о важности первого предложения введения, а также о конусовидной структуре введения, ведущего к постановке задачи. Также рассказывается о стиле написания введения методом пересекающихся контекстов.

Как корабль назовёте, так он и поплывёт.
А.Некрасов. Приключения капитана Врунгеля

Без принужденья в разговоре
Коснуться до всего слегка…
А.С. Пушкин. Евгений Онегин

Название диплома – это визитная карточка вашей работы. Важно, чтобы название было лёгким, элегантным, интригующим. Не старайтесь всё объяснить в названии, хотя, в некоторых университетах требуют именно так. Пример всё объясняющего названия представлен ниже.

> Создание предметно-ориентированного отладчика в среде разработки DevM для пошаговой симуляции верхнеуровневых сценариев работы сетевого роутера.

Кажется, что действительно всё объяснено. Однако без контекста такое название всё равно порождает много вопросов. Во-первых, не ясно, что это за среда разработки DevM: на предзащите данного диплома один из членов комиссии и вправду стал искать эту среду в Интернете! А в Интернете эту среду не найти, поскольку она является внутренним продуктом некоторой крупной компании, и название это означает как среду разработки, так и одновременно предметно-ориентированный язык, созданный в этой же компании. Во-вторых, не ясно, что такое верхнеуровневые сценарии работы сетевого роутера. Автор диплома имел в виду сценарии обработки событий уровня сетевого управления, но это совершенно неясно людям, незнакомым с телекоммуникациями. В-третьих, может быть непонятным то, как именно отладка (и, собственно, отладчик) связаны с пошаговой симуляцией, и чем является симуляция, а также почему сценарии целесообразно эмулировать… Одним словом, кажущаяся понятность «скрывает» множество вопросов — и их ничуть не меньше, чем если бы название диплома было бы таким: «Отладчик предметно-ориентированного языка для разработки семейства сетевых роутеров». Однако этот последний вариант действительно является названием, в то время как предыдущий тяготеет к аннотации.

Рассмотрим ещё один пример очень длинного названия.

> Глава I, в которой рассказывается о том, как Муми-тролль, Снусмумрик и Снифф нашли шляпу Волшебника, как неизвестно откуда появились пять маленьких тучек, а Хемуль обзавелся новым хобби.
>
> *Название первой главы в сказке Туве Янсон «Шляпа волшебника»*

Отмечу, что в данном случае это длинное название является заголовком главы, а не всего текста, и таким образом имеет контекст; однако я не советую подобным образом именовать главы в тексте вашего диплома! А название работы, вообще говоря, контекста не имеет и встречается во множестве документов, сопровождаемое лишь вашей фамилией. В связи с этим оно, как мне кажется, должно быть достаточно лаконичным, литературно грамотным, в меру понятным и интригующе-приглашающем читателя ознакомиться с вашей работой.

Заголовок является, как правило, назывным предложением. Напомним, что назывным называется односоставное предложение, в котором подлежащее выражено существительным в именительном падеже. Односоставное предложение — это такое предложение, в котором имеется всего один главный член предложения, в данном случае, это только подлежащее, следовательно, сказуемое в назывном предложении отсутствует.

> **Название (заголовок)** дипломной работы является назывным предложением и обозначает некоторый объект или явление, которому посвящена работа, определяя ряд его характеристик, свойств, а также контекстов и ограничений.

Ниже приведены удачные, на мой взгляд, примеры названий дипломных работ, взятые с сайта кафедры системного программирования СПбГУ.

- Реализация дистиллятора для простого функционального языка на Haskell.
- Быстрая локализация изображений, получаемых некалиброванной камерой.
- Управление ограничениями в символьной виртуальной машине V#.
- Разработка веб-приложения «Электронный дневник» для частной школы.
- Автоматизация разметки видимого горизонта дороги по видеопотоку.
- Разработка веб-сервиса для поиска студенческих IT-стажировок.

Вот ряд практических советов разного рода, которые можно использовать при составлении названий дипломов и заголовков (глав и разделов).

- Не следует увлекаться в заголовках знаками препинания — «,» «;» «(» «)». Например, не стоит писать подобным образом: «Элементы для Flash-приложений на языке haXe. Библиотека Sparkle». Хотя, все эти знаки препинания можно использовать, например: «Быстрая локализация изображений, получаемых некалиброванной…».
- В заголовке не нужно повествовательности и описательности — такой (неправильный в данном случае) стиль часто сопровождается словами «потому что», «которые», «следовательно» и т.п.
- Не нужно повторов в заголовке, в частности, использования одних и тех же предлогов или других слов. Например, «Измерения чего-то *в* коде **в** процессе решения каких-то задач, *в* различных средах разработки» — здесь три раза встречается предлог «в» и в целом наличествует явный перебор информации; если хочется всё же сохранить в заголовке всю эту информацию, то нужно искать альтернативу двум из трёх предлогов «в».
- Не стоит ставить точку в конце заголовка. Заголовок является элементом плакатного стиля, поэтому структура предложения здесь несколько нарушается. Но иногда точки в заголовках все таки ставятся: так делают в рамках специальных шаблонов по оформлению научных статей некоторых журналах, но только в подзаголовках. Однако в дипломных текстах так делать не следует.
- Можно использовать знаки препинания «:» или «–» для акцентирования и предания заголовку выразительности, например: «Инструменты для управления вариативностью — готовность к промышленному применению», «Разработка документации: поиск клонов при организации повторного использования текста».

- В заголовке целесообразно использовать ключевые слова и словосочетания вашей предметной области (особенно в англоязычных заголовках): тогда в Интернете ваша работа будет лучше находиться. Вот примеры, в которых ключевые словосочетания выделены курсивом: «*Clone detection* technique in *documentation reuse*», «*Domain-specific modeling* in *enterprise architecture management*: case study», «Teaching to write *software engineering documents* with focus on document design by means of *mind maps*».

Заканчивая разговор о названии, я предлагаю вам подойти к задаче создания подходящего названия творчески: поиграйте с названием вашей работы, придумайте несколько вариантов, много вариантов — больше десяти. Посмотрите и убедитесь, что каждый вариант освещает вашу работу как-то иначе, обратите внимание на небольшие нюансы и выберите наиболее подходящий вариант. При этом обращайте внимание не только на точность и тематическую корректность названия, но также на его благозвучность.

Введение является одновременно одной из самых ответственных частей текста вашего диплома и одновременно одной из самых сложных. Почему — ответственной? Дело в том, что большинство читателей дальше введения не продвинуться в чтении вашей дипломной работы: они прочитают название, поинтересуются, о чём работы, прочитав введение — и тем самым вполне удовлетворят свой интерес к вашей работе! И лишь немногие интересанты будут читать дальше. И поэтому во введении следует максимально доходчиво описать … а что же, собственно, описать?

Общераспространённым мнением является, что во введении следует описать краткое содержание всей работы. Но это, фактически, невозможно сделать на двух-трёх страницах — а именно таков должен быть размер вашего введения. Дело в том, что программирование является очень обширной областью деятельности, и для того, чтобы вы могли описать свою работу, даже кратко, нужно сначала описать контекст. Дипломная записка отличается от научной статьи — последняя, как правило, сильно специализирована и предназначена для узких специалистов. А ваш диплом предназначен, как минимум, для аттестационной комиссии, которая должна понять и надлежащим образом оценить вашу работу. Коме того, текст вашего диплома предназначен также для широкого круга читателей — ваших товарищей, которые будут защищать дипломы после вас, и любого, заинтересовавшегося вашей тематикой и нашедшего ваш текст в Интернете. Так вот, всех этих потенциальных читателей нужно ввести в курс дела, т.е. объяснить предметную область и проблематику вашей работы. Именно это и является основной задачей введения. А уже потом, будучи надлежащим образом введёнными, ваши читатели захотят или не захотят читать дальше.

Однако вышесказанное не является единственной задачей введения. Важно, чтобы оно выполняло ещё одну функцию, а именно, подводило читателя к постановке задачи вашей работы. Что такое постановка задачи к диплому, мы обсуждали в лекции, посвящённой управляющей структуре. Здесь же отметим, что постановка задачи может быть непонятной читателям, а также аттестационной комиссии, в силу тех же причин, по которым может оказаться неясной и тематика вашей работы в целом. Часто встречающейся ошибкой студентов является довольное общее введение, а потом — резкий скачок сложности информации при определении цели и задач диплома. Так вот, это неправильно! Введение должно подготовить читателя к постановке задачи с тем, чтобы не было никаких скачков. Эта идея отражена на рисунке 4.

Что означает воронка, представленная на этом рисунке? Что специфичность информации во введении нарастает. Таким образом вы начинаете совсем с общеизвестных вещей и, не задерживаясь на них, устремляетесь всё глубже и глубже. И в конце концов приходите к постановке задачи.

При этом важно первое предложение введения, которое, вместе с тем, является и первым предложением всей вашей работы. Оно является самым общим утверждением в вашей работе. Если ваш диплом посвящён разработке некоторой информационной системе, то можете начать введение с того, что сообщите читателю о том, что информационные системы глубоко вошли в обиход людей и современного бизнеса. Если же ваш диплом посвящен решению какой-то задаче из области компьютерной криминалистики, то вы можете начать с сообщения про рост в мире компьютерных преступлений и необходимость создания специальных инструментов для их расследования. Если же ваш диплом посвящён верификации и тестированию (вы, например, улучшаете какой-то известный символьный движок для более полной генерации тестов программных приложений), то в начале введения вы можете порассуждать о значимости средств обеспечения качества для современного программного обеспечения. При этом важно не впасть в крайность, т.е. не начать сообщать читателю совсем уж тривиально-известные вещи, например, про рост значимости ПО в современном мире, про повсеместное распространение компьютеров и пр. Важно, чтобы вы с первых строк уже прицеливались в собственную предметную область, но не бросались, тем не менее, с места в карьер.

> Важно, что **первое предложение** вашего введения (оно же – и первое предложение всей вашей работы) следует тщательно выверить. Это предложение не должно быть «с места в карьер»: тут не следует писать про вашу задачу, и даже про вашу область. Первое предложение должно *плавно вводить читателя в курс дела*, повествуя о более-менее общеизвестных вещах.



Рис. 4. Введение: от общего к частному, в конечном счёте – к постановке задачи дипломной работы

Таким образом, в первом предложении начинаем с общих слов, но имеющих отношение к вашей теме. например, не надо первой фразой сообщать о том, что компьютеры сегодня распространены повсеместно. Но, если ваш диплом посвящён работе с большими данными, то первое предложение введения можно посвятить констатации важности этих данных в современном бизнесе, индустрии и социальной сфере.

Далее за первым предложением следует первый абзац вашего введения. В этом абзаце можно плавно продолжить сообщать общеизвестную информацию, начиная, тем не менее, подводку к теме вашего диплома.

Обращу внимание на ещё один важный момент. Дело в том, что большинство дипломных работ, будучи реальными, интересными, живыми, оказываются на

пересечении различных контекстов (другими словами — предметных областей). Например, человек пишет диплом про создание системы сборки Python-приложений. Его контекстами оказывается, во-первых, язык Python и его специфические конструкции, во-вторых, средства разработки (IDEs) вообще и конкретно нацеленные на Python, в-третьих, собственно, сами системы сборки – их функциональность, самые известные существующие системы, вопросы их интеграции с IDEs и так далее. Все эти контексты необходимо надлежащим образом осветить во введении. Идея нескольких контекстов, на пересечении которых находится постановка задачи диплома, отражена на рисунке 5.

Отмечу, что введение к тексту диплома существенно отличается от введения к научной статье. Объем (размер) научной статьи, как правило, относительно невелик по сравнению с текстом диплома. Также надо отметить, что статья предназначена, как правило, для узких специалистов, а, кроме того, ориентирована на рецензентов, которые принимают решение о её принятии на научную конференцию или в журнал. В итоге в научной статье тщательно обосновывается тема исследования, прямо во введении дотошно формулируются достигнутые результаты исследования, обильно цитируются различные источники и так далее. В нашем случае столь тщательно обосновывать все утверждения нет необходимости, формулировка результатов дипломной работы помещается в заключение, а важной оказывается постановка задачи, в которую и перетекает введение.

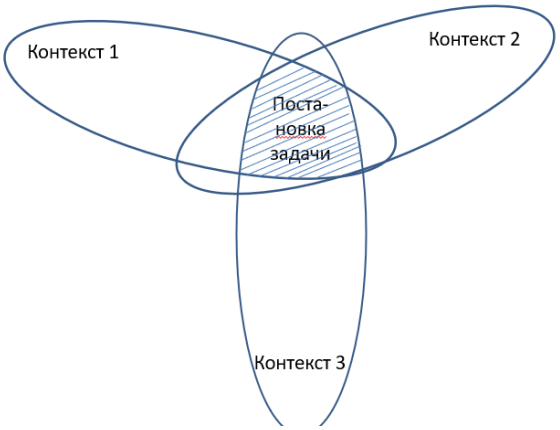


Рис. 5. Многоконтекстность дипломной работы и соответствующая структура введения

Подытожу вышесказанное определением основной задачи введения дипломной работы.

> **Главной целью введения** является ознакомление читателей с предметной областью и проблематикой вашей дипломной работы. При этом оказывается важным максимально подвести читателя к постановке задачи вашей работы, объяснив все нужные контексты, на пересечении которых находится работа.

В заключении отмечу, что текст введения должен лёгким, очень открытым и расположенным к читателю, дружественным. С одной стороны, вы как автор действительно заинтересованы в том, чтобы самые разные читатели поняли предметную область вашей работы, а также ту проблему, которую вы решали. С другой стороны, объем введения ограничен двумя-тремя страницами, поэтому вы вводите читателя в курс дела без излишних деталей, скорее, называя, чем объясняя или давая строгие определения. При этом вы обильно цитируете различные источники как для обоснования высказываемых вами взглядов и утверждений, так и направляя читателя к источникам, где он сможет, при желании, восполнить свои знания по вопросам, которые вы затрагиваете в своей работе.

Важно. Разные части и разделы текста вашего диплома следует писать по-разному, то есть в разном стиле.

Что имеется в виду? Рассмотрим примеры.

- Детальные объяснения, неуместные во введении, весьма уместны в главе «Обзор», где вам действительно нужно объяснять особенности используемых подходов и где для этого предусмотрен подходящий объем текста.
- Ещё более «объяснительными» должны быть главы основной части работы, где вы описываете свои результаты.
    - Напротив, в начале каждой главы нужно уделить один или несколько абзацев для обзора предстоящего материала, а также для его связи с материалом предыдущих глав – и тут следует называть, указывать, и часто обосновывать, мотивировать, но не объяснять детали.

Контрольные вопросы

1) Что такое заголовок дипломной работы?
2) Что такое назывное предложение?
3) Что вы можете сказать о знаках препинания в заголовках?
4) Почему встречаются длинные заголовки?
5) Что можно сказать в защиту компактных заголовков?
6) Какова основная задача введения к диплому?
7) Охарактеризуйте принцип «воронки» или от общего к частному, которым целесообразно руководствоваться при написании введения.
8) Расскажите о принципе пересекающихся контекстов, который также может быть полезен при написании введения.
9) Расскажите о важности первого предложения введения.
10) Расскажите о характере (стиле) текста введения.

Список литературы

1) T.M. Annesley. The Title Says It All. Clinical Chemistry, March 2010, 56 (3): 357–360.
2) T.M. Annesley. “It was a cold and rainy night”: Set the Scene with a Good Introduction. Clinical Chemistry, May 2010, 56 (5): 708–713.

Лекция 5. Обзор

В лекции рассказывается о требовании к тексту диплома разделять своё и «чужое», об отдельной главе под названием «Обзор», куда можно поместить всё чужое, о видах этого «чужого»: описание предметной области, описание «поляны» (похожих методов, систем и пр.), описание используемых средств (методов/технологий/ алгоритмов), описание объемлющего проекта (при условии, что он есть).

.

Вспомним одну из важных характеристик дипломной работы – «разделять своё и чужое». Зачем это делается? А делается это для того, чтобы ясно выделить описание ваши результатов в тексте (главы основной части), отделив это от описания предметной области, используемых технологий, контекстного проекта, в рамках которого вы работали со создали результаты своего диплома и т.д. Глава под названием «Обзор» как раз и служит для описания всего «чужого», т.е. того, что сделали не вы, но о чём по тем или иным причинам вам следует написать. С тем, чтобы остальные главы вашего текста были именно про ваши результаты и вместе с тем могли бы опираться на необходимую «чужую» информацию, описанную в обзоре.

При этом для обзора я рекомендую иметь одну неименованную главу под названием «Обзор» с тем, чтобы не иметь несколько обзорных глав. Тут следует понимать, что текст вашего диплом не является ни учебником, ни описанием предметной области или обзором технологий, а призван описать сделанную вами работу. Я рекомендую одну обзорную главу также и потому, чтобы сразу, в оглавлении вашего диплома, читателю было бы видно, где находится описания результатов вашей работы, а где — нужная, но вспомогательная информация, то есть «чужое».

Хочу отметить одну важную деталь. Очень часто обзор начинает доминировать — и в вашей презентации, и в тексте вашего диплома: во время презентации вы тратите основное время на описание контекста, а основной объем текста диплома оказывается обзором (половина вашего текста и даже более того). При этом часто оказывается, что написание текста вы стартовали именно с обзора, увлеклись — и не осталось ни сил, ни времени на написание основной части — то есть текста именно про вашу работу.

> **Важно.** Объем главы «Обзор» не должен превышать 30% от всего текста вашего диплома. Иначе ваш диплом будет не про результаты вашей работы, а про нечто иное.

Теперь перечислим различные виды «чужого», т.е. той информации, которую целесообразно поместить в обзор:

- описание предметной области,
- описание «поляны» — похожих методов, систем и прочего,
- описание используемых средств — методов/технологий/алгоритмов,
- описание объемлющего проекта, если он есть.

Давайте разберёмся с этими пунктами детально.

Описание предметной области. Речь идёт о той области бизнеса, науки, технологий, к которой принадлежит ваша работы. Выше уже указывалось, что промышленное программирование и сопутствующие ему науки весьма многочисленны. И порой требуется весьма искусное описание предметной области вашего диплома для того, чтобы читатели и прежде всего аттестационная комиссия хоть что-то поняла.

Например, ваш диплом посвящён решению «вечной» проблеме — сборке мусора в C++ приложениях. Проблема эта является неразрешимой в виду невозможности контролировать динамически выделяемую память из-за адресной арифметики, а также из-за свободы для приведений указателей. Адресная арифметика позволяет весьма эффективно распоряжаться динамически выделенной памятью, например, распаковывать сетевые сообщения, зная, сколько байт занимает каждое поле и имея информацию о последовательности этих полей. Однако она же и препятствует созданию точных и полных сборщиков мусора — утилит, которые контролирую корректное освобождение динамически выделенной памяти. Поэтому появляются всё новые и новые сборщики мусора, которые в чём-то превосходя свои аналоги, но в чём-то неизменно им проигрывают…

вы заметили, что я начал писать обзор предметной области для диплома, посвященного сборке мусора в C++? А сборка мусора является относительно общеизвестной задачей. Имеются другие, гораздо менее известные предметные области, например, поиск похожих комментариев в Java-коде, различные задачи в области управления данными (Data Management), динамическое символьное исполнения, задача онбординга для платёжных систем, проблемы разработки подсистемы управления устройствами (device management) для телекоммуникационных систем, томография, компьютерное зрение, сейсморазведка, компьютерная криминалистика, логическое программирование, слабые модели памяти и так далее. Для аттестационной комиссии, а также для других возможных читателей вашего диплома (например, вашего потенциального работодателя) важно хоть как-то разобраться в предметной области вашего диплома.

Разумеется, значительную часть предметной области вы описываете во введении. Но этого может оказаться недостаточно, поскольку на сегодняшней день в информатике и индустриальном программировании существует огромное количество различных тем и областей, и одному человеку, практически, невозможно за всем уследить. Если ваша предметная область является специальной и известной лишь в узких кругах, то в этом случае вам и следует отдельно описать её в обзорной главе. Но здесь важно не переусердствовать, соотнося объем обзора с общим объемом текста всего диплома.

Если же это не так и ваша предметная область достаточно известна, то эту часть обзора можно пропустить, ограничившись введением.

Но описание предметной области — это далеко не всё, что должно быть и может содержаться в главе «Обзор».

Описание «поляны». В этой части обзора речь идёт о том, что уже сделано по вашей тематике до вас: какие имеются индустриальные системы, аналогичные вашей, какие есть сходные алгоритмы, подходы, методы и так далее. В общем, как обозначенную вами проблему решают другие и по-другому. Может оказаться, что вы придумали нечто совсем новое, и пользователи ещё не знают, что им это нужно. Придумал же Стив Джобс айфон с дизайном, которому не было аналогов. Или компания Microsoft впервые предложила когда-то метафору письменно стола как основу интерфейса операционной системы с пользователем (интерфейс OS Windows). Но даже в этих случаях существуют аналоги, пусть и сильно уступающие. И у этих аналогов имеются проблемы, задачи, с которыми они не справляются. Вот это всё и следует обсуждать в данной части обзора.

Итак, требуется определить список аналогов. Это список не должен быть слишком большим, иначе он будет плохо сфокусирован. Например, если вы предлагаете открытую (open source) систему по управлению мастер-данными, то не нужно обозревать все имеющиеся системы управления мастер-данным – существует значительное количество таких систем, более того, каждая из них оказывается многофункциональной и весьма объёмной. В этом случае можно ограничится лишь обзором имеющихся открытых систем в этой области, даже если их и немного. Однако, если вы занимаетесь сборкой мусора для C++, то вам придётся рассмотреть все имеющиеся сборщики, которых за последние

десятилетия создано немало. Но в этом случае наверняка имеются уже готовые обзоры, которыми вы можете воспользоваться.

После того, как вы разобрались со списком аналогов, вам следует определиться с критериями их обзора. Не стоит их просто описывать – получится очень объемно и всё равно неполно. Например, пусть ваш диплом ориентируется на реализацию каких-то специфических возможностей продуктов этого класса – например, на реализацию средств навигации по коду для C/C++ сред разработки для встроенных приложений. В этом случае вам следует рассматривать в этих средах разработки именно средства навигации, сформулировав дополнительные характеристики для их описания.

Далее, целесообразно (но не строго обязательно) свести описание выбранных аналогов в соответствии с критериями, которые вы определил, в одну таблицу с графами-характеристиками и значениями некоторых метрик, опять же созданных вами. Или сделать эти характеристики основой текстового писания рассматриваемых аналогов с тем, чтобы у вас получились единообразные описания каждого продукта/метода/алгоритма, а не вольный рассказ-сказка о ваши впечатлениях или собранной как получилось информацией.

Ещё одним важным моментом является способ исследования выбранных вами аналогов: очень часто студенты не исследовали их заранее и приходится делать это прямо в момент написания текста диплома (известно, что даже у опытных исследователей эксперименты часто запаздывают…). Самый простой способ заключается в поиске подходящей открытой информации (white papers). Можно также искать аналитические обзоры: например, для средств управления мастер-данными выходят ежегодные обзоры агентства Gartner. В качестве ещё одного способа можно порекомендовать установить все найденные аналогичные системы на свой компьютер и изучить их в действии. Возможны и другие варианты.

И последнее. вам нужны выводы по этой части обзора. Эти выводы должны показывать имеющиеся у аналогов бреши, часть из которых вы и закрываете своей работой.

Описание используемых средств. В своей работе вы наверняка используете что-то готовое – от языков программирования и сред разработки до методов, подход и идей. Кое-что из этого также следует описать в обзоре с тем, чтобы читатель понял, что же именно вы использовали, а что реализовали самостоятельно. Конечно, нет нужды писывать широк известные языки программирования, такие как Java, C++ и пр. Но вот если вы использовали малоизвестные языки, например miniKanren или OCL, то их стоит описать. Но даже касательно общеизвестных языков могут оказаться специфические и малоизвестные черты, которые принципиальны для вашей работы. Например, вы могли использовать последнюю версию C++ (C++17) для того, чтобы использовать имеющиеся в ней возможности метапрограммирования, а также преследуя задачу повышения производительности кода. Эти черты C++ надо описать в обзорной главе, а также ясно объяснить, почему C++17 в вашем случае обеспечит более высокую производительность.

Нужно отметить, что описание используемых средств можно приводить не только в обзорной главе. Иногда такие описания приводят прямо по тексту в тех местах, где эти средства используются. Это имеет смысл, если такие описания не очень большие – до 0.5 страницы, – и тех случаях, когда соответствующие средства не столь значимы, чтобы их отдельно описывать в отдельной главе. Но если такая вставка превышает 0.5 страниц, то она начинают «рвать» основной текст и затрудняют понимание, особенно при поверхностном чтении. На первых порах можно следовать абсолютному правило: «все чужое – в обзор».

Описание объемлющего проекта. Любой из видов дипломов может выполняться в рамках некоторого проекта, в котором, помимо вас, участвуют и другие. Наиболее очевидная ситуация с производственными дипломами – студент работает в команде над

каким-то проектом и выбирает для защиты подсистему, которую он разработал (тут отдельный вопрос – что выбирать в таком случае: многие студенты стримятся забрать в диплом всё, что они сделали в проекте, но это «всё» может не образовывать отдельного объекта; и тогда нужно «резать», убирать лишнее – но об этом я подробно рассказывал в лекции про управляющую структуру). Однако весь проект в целом тоже нужно описать (но кратко), поскольку иначе описание отдельной подсистемы может оказаться неясным. И это лучше всего сделать именно в главе «Обзор».

Исследовательские проекты также могут выполняться группой разработчиков. Тоже, но реже бывает с научно-исследовательскими проектами. Таким образом, во всех этих случая нужно также в главе с обзором описать контекстный проект.

Комментарий. Некоторую часть «чужого» всё таки бывает целесообразно описывать прямо по ходу работы, в главах, посвящённых результатам, и не выносить эту информацию в «Обзор». Это бывает целесообразно, когда данная информация является небольшой и не заслуживает отдельного раздела в «Обзоре». Или же данная информация нужна и используется лишь в одном месте текста, и она органично вписывается именно в это место текста вашего диплома. При этом я не рекомендую злоупотреблять этим подходом, описывая основную «чужую» информацию в главе «Обзор».

Контрольные вопросы

1) Какова основная задача обзорной главы?
2) Почему нежелательно иметь несколько таких глав?
3) Какую информацию в данной лекции называют «чужой»?
4) Каковы рекомендации по соотношению объёма «Обзора» и текста всего диплома?
5) Расскажите про правила «разделять своё и чужое» и «всё чужое – в обзор».
6) Перечислите основные элементы обзорной главы.
7) Расскажите про описание предметной области.
8) Как вам кажется, когда описание предметной области в дипломе не требуется?
9) А когда описание предметной области в дипломе нужно обязательно?
10) Зачем нужно описание существующих подходов (описание поляны)?
11) На сколько подробным должно быть описание используемых технологий/продуктов и как должно быть сфокусировано это описание?
12) С какой целью описывают используемые в дипломе средства в обзоре?
13) В какой ситуации бывает целесообразно описывать используемые в вашей работе «чужие» результаты — программные средства и технологии, алгоритмы и пр. — прямо по ходу описания результатов, т.е. в главах основной части?
14) Расскажите про необходимость описывать объемлющий вашу дипломную работу проект (разумеется, если он имеется)?
15) Когда целесообразно описывать чужое в главах основных частей текста диплома?

Лекция 6. Архитектура и реализация

В лекции даётся определение архитектуры программной системы с точки зрения удобства её описания в тексте диплома — компактное описание системы в терминах целостных атрибутов. Приводятся примеры целостных атрибутов. Даётся также определение компоненты системы — основной структурной единицы, исполь-зуемой при описании структуры системы в рамках определения её архитектуры. Приводится ряд примеров из реальных дипломов студентов.

.

Целое всегда больше суммы своих частей

Как правило, диплом по программированию – это описание некоторой программной системы, которую вы создали. Бывают, конечно, и другие дипломы (виды дипломов мы обсуждали в первой лекции), но наличие сложного программного приложения, которое вы самостоятельно разработали, является неоспоримым доказательством того, что вы успешно выучились и готовы работать в индустрии. (Те же студенты, кто пишет научные дипломы, могут пропустить эту лекцию – у них всё обстоит иначе.)

Итак, в производственных дипломах имеется две задачи: разработать архитектуру системы и реализовать её. Соответствующие результаты — созданная архитектура и выполненная реализация — являются общепринятыми результатами производственных дипломов. Однако столь же общепринятой является и путаница касательно того, в какой главе писать о чём писать, и чем текст про архитектуру системы должен отличаться от текста про её реализацию. Интуитивно понятно, что архитектура должна описывать главные решения, принятые при проектировании системы. Ну а если студент, в общем-то, не проектировал, а лишь немного подумал и сразу начал писать код?

Подойдём к вопросу с другой стороны. От чего мы стремимся уйти в текстах производственных дипломов? От единственной объемной главы под названием «Реализация», где описана вся проделанная вами работа. Или от набора глав, которые соответствуют разным функциональным модулям разработанной вами системы.

Почему первый вариант плох – понятно: вместо нормального текста есть она большая куча (глава), в которую помещено всё самое содержательное. А чем второй вариант плох? Тем, что в тексте вашего диплома нет места, где бы про вашу систему было рассказано целиком – про основные решения, которые были использованы при разработке, про её функциональность, про структуру компонент и т.д.

Архитектура системы. Итак, читатель производственного диплома хотел бы прочитать про вашу систему без лишних деталей и получить о ней общее, целостное представление. И может быть, этим и ограничиться. Это описание не содержится во ведении, поскольку последнее описывает мотивации разработки и существующую ситуацию на рынке, а также предметную область и прочее. А читателю нужно важно где-то в одном месте прочитать вкратце все основное про предлагаемую систему. Например, в нормативных шаблонах документации одной корпорации значился особый вид документа — паспорт системы. Этот шаблон предназначался для описания основных технических параметров системы: разработческих (средства разработки и платформы исполнения), функциональных (количество поддерживаемых соединений, одновременно работающих пользователей, временные ограничения и прочее), структурных (различные компоненты и подсистемы). И в нашем случае (то есть в ситуации разработки и оформления выпускного университетского диплома по программированию) нам также нужен некоторый паспорт системы, который мы будем называть её *архитектурой*.

> **Архитектурой программной системы** является описание системы как единого целого, без лишних деталей и подробностей, с использованием *целостных атрибутов*, то есть таких её характеристик, которые затрагивают всю систему целиком, не локализуясь на отдельных её частях. Примерами целостных атрибутов системы являются: (i) выбранные технологии разработки, (ii) структура компонент системы, (iii) описание её функциональности (в частности, главный сценарий её работы).

Не имея компактного описания такой архитектуры, читателю придётся, для получения связной картины читать текст диплома целиком, да ещё изрядно подумать, соединяя разные детали в общую картину…

Вы можете заметить, что данное выше определение архитектуры не соответствует другим имеющимся определениям этого понятия, которые можно найти в Интернете и/или в литературе. На это можно ответить следующим образом. Один мой товарищ провел исследование и нашёл в литературе более ста разных определений архитектуры ПО. О чём это говорит?

Во-первых, о том, что данный предмет является живым и интересует большое количество как теоретиков, так и практиков. Мне кажется, что если в Интернете вы можете найти всего одно определение какого-то понятия или явления, то вы, скорее всего, имеете дело с чем-то неинтересным, неживым, с тем, что мало кому интересно. Напротив, много разных точек зрения, всевозможных материалов, плохо стыкующихся или вовсе нестыкующихся друг с другом определений — о, значит, вы интересуетесь чем-то действительно живым и содержательным!

Во-вторых, наличие большого количества разных определений для одного и того же понятия или явления наводит на мысль, что рассматриваемое понятие/явление многогранно, а разные определения происходят из разных контекстов, служат разным целям. А какова ваша цель? Написать понятный диплом про созданную вами программную систему! Это и есть тот контекст, который породил данное выше определение.

Давайте посмотрим, что означают перечисленные в определении архитектуры целостные атрибуты системы – выбранные технологии разработки, структура компонент системы, описание её функциональности (в частности, главный сценарий её работы).

Выбранные технологии разработки. Важно в одном месте перечислить их все – и язык(и) разработки, и используемое IDE, и библиотеки, включая специализированные (например, для тестирования), а также специфическое аппаратное обеспечение – например, какую-нибудь плату или графический процессор (GPU).

Структура компонент системы. Тут предполагается использование диаграмм компонент UML – пример представлен на рисунке 6. В этом примере показано, что в рамках дипломной работы создана система, которая является специализированным клиентским приложением для портала государственных открытых данных РФ и предназначена для оценки качества этих данных по разным критериям[3]. Приложение имеет главную компоненту «Assessor», которая позволяет пользователю выбрать параметры проверки через специальный пользовательский интерфейс, запустить и осуществить проверку. При этом проверяемые данные скачиваются с портала в специальную базу данных данного приложения (компонента «База данных MySQL»). Отчёт по выполненной проверке формируется компонентой «Reporter». При этом можно добавить существенные детали про реализацию всех этих компонент – прежде всего про использованные технологии.

[3] Данный пример взят из диплома Ольги Андреевой «Оценка открытых государственных данных РФ» (диплом специалиста, кафедра информатики СПбГУ, 2015 год).

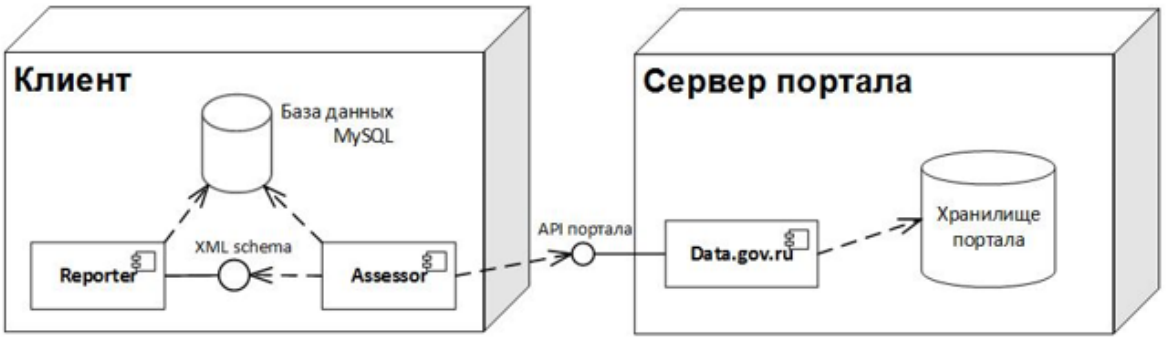


Рис. 6. Пример диаграммы компонент

Главный сценарий работы системы. Диаграмма компонент является статическим описанием системы. Некоторая динамика на ней (скорее, вокруг неё, при её описании в тексте) всё же присутствует – вы могли заметить, что я, фактически, описал сценарий работы системы, описывая компоненты с рисунка 7. Однако на самой диаграмме компонент никакой динамики нет – и это принципиально важно. UML предлагает использовать концепцию separation of concerns, то есть показывать на разных диаграммах статику и динамику. В этом есть резон, поскольку иначе, в погоне за наглядностью, автор документа рисует совсем нетипичную диаграмму, которая в большинстве случаев понятна и очевидна лишь ему самому. Но главный сценарий работы системы, явно представленный и подробно описанный, может существенно прояснить то, как система работает, точнее, какую основную задачу, полезную для пользователей, она решает. На рисунке 7 представлен такой сценарий для системы с рисунка 6. Мы видим, что пользователь заполняет параметры для своего запроса к системе — соответствующий фрагмент пользовательского интерфейса представлен на рисунке 8. В итоге формируется отчёт, пример которого представлен на рисунке 9. Этот фрагмент пользовательского интерфейса также важен для понимания основной функциональности системы. Таким образом мы видим, что при описании главного сценария работы системы мы также кратко описали её пользовательский интерфейс.

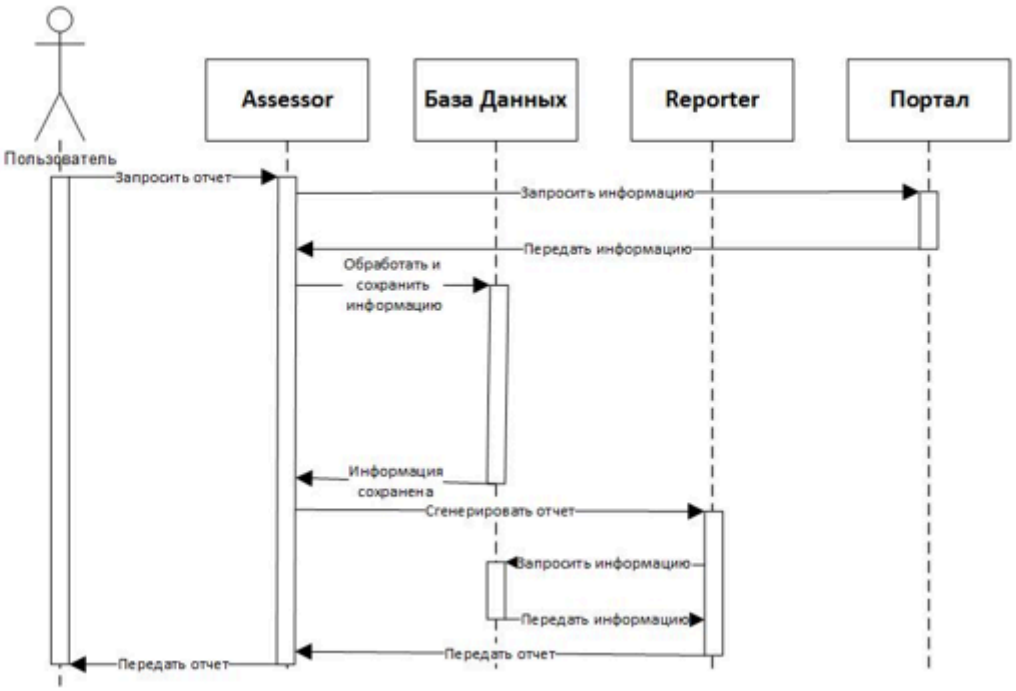


Рис. 7. Пример основного сценария системы

Важно отметить, что в данной главе мы описываем интерфейс системы кратко, без лишних деталей – иначе глава с архитектурой может сильно увеличится в размерах. В частности, здесь не стоит подробно описывать все поля ввода, а также значения всех элементов отчёта. Я не думаю, что это вообще целесообразно описывать в тексте диплома – иначе получится не диплом, а специальный рабочий документ с описанием интерфейса система. И этот текст будет весьма объёмным.

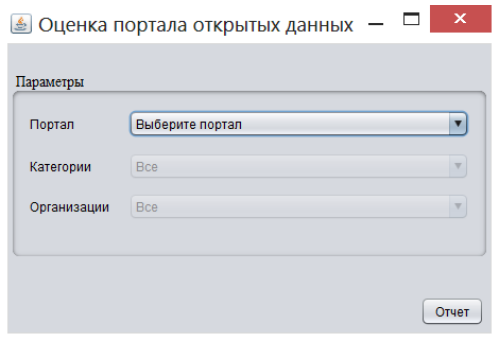


Рис. 8. Фрагмент пользовательского интерфейса системы

О компонентах системы. Теперь обсудим вопрос о том, чем могут быть компоненты на диаграмме компонент в главе, посвященной архитектуре вашей системы. Ведь легко предвидеть вопрос о том, что как таковых, никаких компонент в коде нет, а есть, к примеру, только классы.

Ответом на этот вопрос является следующий совет: выделять в компоненту те элементы кода (например, классы, процедуры, структуры данных и пр.), которые тесно связаны друг с другом и/или реализуют определённую, хорошо очерченную функциональность системы, и/или код, размещённый на одном устройстве, и/или код, разрабатываемый одним разработчиком/подкомандой и так далее. Вы являетесь полным хозяин при

создании компонентного описания вашего кода, и это описание должно служить вашим целям – наглядно представить структуру созданной вами системы.

Так, известный стандарт в области авионики AADL предлагает следующие варианты программных компонент, характерные для программного обеспечения в этой сфере:

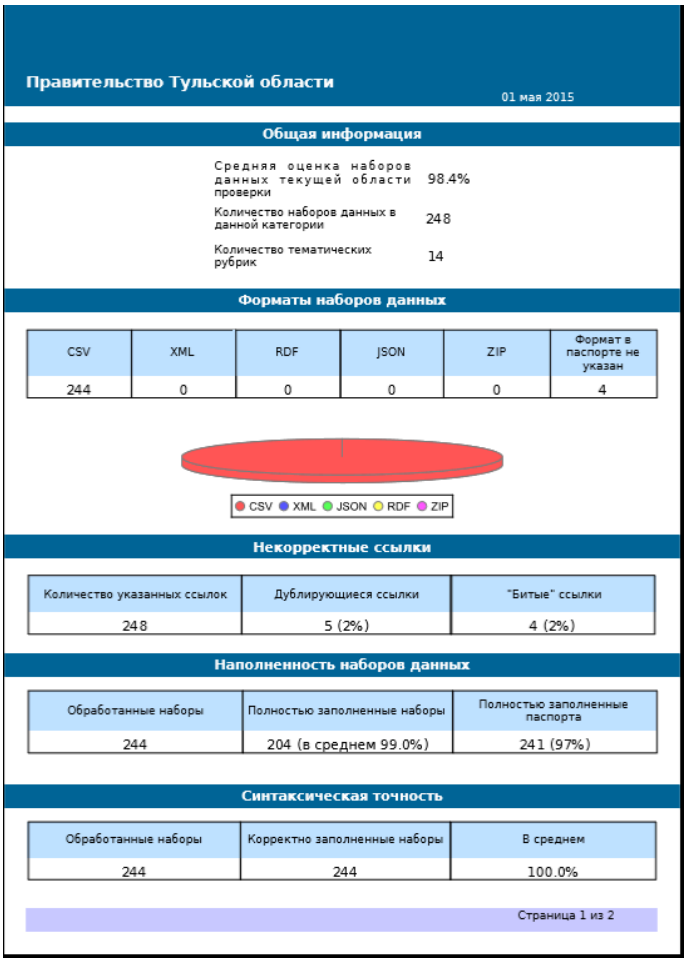
- набор данных,
- подпрограмма или группа подпрограмм,
- нить (thread) или группа нитей,
- процесс.

Именно эти кирпичики в данной предметной области считаются удобными для верхнеуровневого представления кода, при этом они могут иметь вложенную структуру, т.е. одни компоненты могут включать в себя другие.

Таким образом, диаграмма компонент обеспечивает высокоуровневое представление структуры вашей системы, контрастируя, например, с диаграммой классов, поскольку классов может быть весьма много и все они, скорее всего, не поместятся на одну компактную диаграмму. Кроме того, классы содержат слишком много деталей, а также многие из них являются вспомогательными, служебными, и весьма непросто сформулировать критерий о том, какие классы являются значимыми, основными – и именно их показать на диаграмме, – а какие являются вспомогательными, несущественными с точки зрения представления структуры системы «в большом».

Компонента, созданная при описании вашей системы, является некоторым фрагментом кода, выделенным вами для удобства представления системы «в большом», в силу одного или нескольких соображений, представленных ниже.

- Фрагмент кода, независимый с точки зрения физической организации, например, поставляемый в виде dll-файла.
- Код, взаимодействующий с окружением по строго определённым правилам при условии, что его реализация скрыта от окружения.
- Фрагмент кода, который исполняется на отдельном компьютере/ процессе или в рамках одной нити (thread).
- Код, который разрабатывался одним разработчиком или одной небольшой командой; в этом случае количество компонент в системе может определяться количеством разработчиков/команд.
- Код, который реализует замкнутую, отдельную, независимую функциональность системы, в том числе код, который соответствует микросервису.



Правительство Тульской области

01 мая 2015

Общая информация

Средняя оценка наборов данных текущей области проверки	98.4%
Количество наборов данных в данной категории	248
Количество тематических рубрик	14

Форматы наборов данных

CSV	XML	RDF	JSON	ZIP	Формат в паспорте не указан
244	0	0	0	0	4

Некорректные ссылки

Количество указанных ссылок	Дублирующиеся ссылки	"Битые" ссылки
248	5 (2%)	4 (2%)

Наполненность наборов данных

Обработанные наборы	Полностью заполненные наборы	Полностью заполненные паспорта
244	204 (в среднем 99.0%)	241 (97%)

Синтаксическая точность

Обработанные наборы	Корректно заполненные наборы	В среднем
244	244	100.0%

Страница 1 из 2

Рис. 9. Ещё один фрагмент пользовательского интерфейса системы: пример автоматически генерируемого системой отчёта

Таким образом вы можете выделить в компоненту, практически, какой угодно код, а точнее то, что вам удобно выделить, чтобы наглядно представить структуру вашей системы.

Реализация системы. Теперь обсудим, о чём писать в главе «Реализация».

> **Важно.** В главе про архитектуру предлагается описать систему целиком, ориентируясь на целостные атрибуты, в том числе на её структуру, то в главе «Реализация» следует уделить внимание деталям.

Вот примеры таких деталей.

- Для компонент, выделенных в архитектуре, тут можно представить детальное описание. Но сделать это не для всех компонент, а только для нетривиальных. Это описание может содержать диаграммы классов для выделенных компонент.
- В данной главе могут быть изложены какие-то отдельные алгоритмы системы.
- Можно также описать самые разные нетривиальные аспекты реализации, например, как было достигнуто нужное быстродействие системы, или как удалось сделать быстрым поиск по тем или иным данным.

Для того, чтобы сделать изложение связным, можно перечислить в начале главы излагаемые дальше нетривиальные особенности реализации, а потом дать описание

каждой из них в отдельном разделе. Такой обзор (перечень) может быть полезным, поскольку иначе человек что-то описывает, но без объяснений, почему он описывает именно это, и нужно ли описывать что-то ещё. Профессор А.Н.Терехов в ответ на вопрос о том, что писать в главе «Особенности реализации», отвечал так: «Пиши о том, над чем думал больше, чем пять минут, когда писал код».

> **Важно.** То, чего не нужно делать в главе про реализацию, так это дотошно описывать ваш код: все классы, их поля и методы и пр. Даже говоря о реализации, нужно мыслить концептуально, т.е. описывать проблемы, и использовать русский язык для описания способов их решения. И лишь в редких случая допускаются вставки фрагментов кода, но только если там содержится что-то действительно существенное, о чём трудно сказать словами.

Ещё один существенный момент. Данную главу часто именуют традиционно следующим образом: «Особенности реализации», поскольку книги и статьи с таким названием часто выходили на русском языке в 60-е – 70-е годы, например, «Особенности реализации Алгола 68». С другой стороны, вам надлежит показать, что вы являетесь готовым для индустрии специалистом, который способен разрешать именно практические, реальные задачи — и вот вы описываете разнообразные проблемы, которые вы решали и решили во время разработки дипломного проекта: например, вы столкнулись с отсутствием документации, или с трудностями отладки/тестирования, или с «ползущими» требованиями и так далее. Здесь можно дать небольшую зарисовку о том, как все происходило, как исходная ситуация была сложна и как вы её разрешили. Такие моменты, будучи нетривиальными, интересны и характеризуют вас как реального специалиста, которые уже сталкивался с неакадемическими трудностями на производстве и сумел их решить. Включайте такого рода материал в особенности реализации. Но тут важно не перестараться, не написать слишком много или не начать писать уж совсем разговорным языком истории из жизни. В общем, в этом вопросе спросите совета в вашем университете — у научного руководителя, куратора дипломов на кафедре и так далее.

Контрольные вопросы

1) Каков смысл главы с описанием архитектуры в вашем дипломе?
2) Чем архитектура отличается от реализации?
3) Что такое целостные атрибуты системы?
4) Приведите примеры целостных атрибутов системы.
5) Что может входить в технологии разработки вашей системы?
6) Расскажите о структуре компонент как о целостном атрибуте системы.
7) Расскажите о главном сценарии работы системы как о целостном атрибуте системы.
8) Что на ваш взгляд, можно выделять в компоненты системы при составлении схемы с описанием её архитектуры?
9) Объясните, с какой целью в данной лекции предлагается создавать компонентную модель системы?
10) При создании компонентной модели системы в стиле, предлагаемом в данной лекции, требуется ли писать дополнительный код?
11) Нужно ли описывать программные код системы, которую вы реализовали, в дипломе? Ответ обоснуйте.
12) Расскажите, какими могут быть особенности реализации, описываемые в соответствующей главе.

Список литературы

1) Architecture Analysis and Design Language (AADL). AEROSPACE STANDARD AS5506C, 2016-03.
2) Д.В. Кознов. Основы визуального моделирования. М.: БИНОМ, 2008.

Лекция 7. Практическая полезность

В лекции даётся рассматривается вопрос о практической полезности задачи, решённой в рамках дипломной работы по программированию. Подробно рассматриваются варианты доказательства практической полезности, готовности продукта для использования – результаты тестирование продукта, апробация (пилотный проект или пользовательское тестирования), внедрение системы, экспериментальное исследование (исследование и измерение различных параметров работы системы).

Кому кроме тебя нужна эта работа?
Профессор А.Н. Терехов, из диалогов на предзащитах
студентов кафедры системного программирования СПбГУ

Вопрос о практической полезности вашей работы крайне важен для аттестационной комиссии и прочих читателей вашего текста, поскольку дипломы по программированию посвящены именно прикладной области. Эта предметная область не виртуальна, а в высшей степени реальна: обучаясь программисткой специальности студенты получают настоящую профессию, с помощью которой они могут зарабатывать деньги – рынок вакансий по программированию разнообразен и содержит много предложений с достойной заработной платой. И аттестационную комиссию справедливо интересует, сколь полезную, практически востребованную вещь студент создал в рамках своего диплома.

Важно понимать, что практическая полезность работы не следует автоматически из описания основных результатов вашего диплома – у вас все может быть круто, актуально, ново. Но этого недостаточно! Можно ли легко установить вашу систему и начать ее использовать? Имеются ли у вашей системы реальные пользователи, и если да, то кто они и каково их мнение о вашей системе? Возможно ли реализовать ваш теоретический метод или алгоритм и начать его применять на практике? Какова степень готовности ваших результатов для практического применения?

Ответы на эти и подобные им вопросы должен дать специальный результат вашей работы, описанный в отдельной главе. Ниже я предлагаю различные варианты этого результата.

- *Тестирование* продукта: выполненные вами различные виды тестирования (например, модульное тестирование), количество разработанных тестов, информация о покрытии тестами вашего кода, созданная вами инфраструктура тестирования (возможно, осуществляющая автоматическую генерацию тестов или тестовых данных), результаты тестирования (количество найденных и исправленных ошибок).
- *Апробация* – опытная эксплуатация вашего продукта в рамках пилотного проекта или пользовательского тестирования.
- *Внедрение* вашего продукта в реальный производственный проект или успешные продажи клиентам. Здесь интересно описание процедуры внедрения, если эта заказная система, а также реальное количество пользователей вашего продукта.
- *Экспериментальное исследование* – это измерение и исследование различных динамических параметров созданной программной системы и/или практическое сопоставление вашего продукта/метода/алгоритма с аналогами. Экспериментальное исследование может потребовать значительных усилий, включая следующее:

- создание концептуальной инфраструктуры: исследовательские вопросы, тестовые данные, используемые метрики, выбор аналогичных разработок;
- реализацию специфичной программной инфраструктуры экспериментального исследования: единой экспериментальной среды, доступа к тестовым данных для аналогов, вывод и представление результатов и пр.;
- собственно, выполнение самих экспериментов (а это может требовать мощных серверов и значительного времени);
- а также анализ результатов: тут нужно думать и рассуждать – широко, интересно, полемически.

Хочу отметить, что в рамках своей работы вы можете выполнить различные комбинации этих пунктов, например:

- выполнить тестирование и апробацию, или
- тестирование и внедрение, или
- апробацию и внедрение.

Тот путь, который вы выбрали для обоснования практической полезности совей работы, должен быть адекватно отражён в названии соответствующей главы, а также в описании задачи/результата.

Тестирование. Очень часто автор дипломной работы вообще не производил тестирования своей системы. Некоторое количество юнит-тестов и ещё что-то незначительное … Вот что именно? Если автор писал часть большой системы, то эта его часть должна как-то интегрироваться в общую систему. И в этом месте, возможно, и выполнялось тестирование. Возможно также, что тестирование выполнялось в составе всей системы – не автором, но тестировщиками проекта. Какие из этих тестов приходились на его систему? Какое количество исправлений сделал автор – при интеграции его подсистемы, при использовании её функционала пользователем, по запросам от тестеров, по запросам от пользователей…

Одним словом, в это место надо всмотреться внимательно – если система делалась в составе индустриального проекта, то тестирование должно было происходить. Но бывает, что проект до него еще не добрался, или студент реализовал некоторый демо-функциональность, которая пока особо никому не нужна (обидно звучит, но именно так часто бывает) – и потому эта подсистема не тестировалась.

Одним словом, вопрос с тестированием очень показателен. Но его не надо болезненно замалчивать — не стоит огорчатся, если система получилась не вполне реальной (то есть обошлась без тестирования). Диплом будет засчитан, ибо нет требования обязательного наличия тестирования у системы, описываемой в дипломе.

Но вот если тестирование было, но автор системы ничего про него сказать не может, то это оказывается нелепо…

Апробация. Для многих программных решений важным этапом жизненного цикла является опытная эксплуатация или, другими словами, пилотное внедрение. Оно позволяет понять, на сколько решение удобно и удовлетворяет реальные потребности пользователей и их начальства.

Попутно отмечу, чем решение отличается от коробочного продукта: решение предназначается для конкретных пользователей – внутри компании-разработчика или для некоторой другой компании, оно должно быть встроено в определенный контекст, поддержать некоторый бизнес-процесс и заказчик, как правило, участвует в разработке решения (отвечает на вопросы разработчиков, согласует требования, принимает промежуточные версии решения).

Коробочный продукт ориентирован на существенно более широкий круг пользователей, заранее известны лишь их профессиональные характеристики (например, если речь идёт о создании коробочных средств разработки, то их пользователями будут программисты, но трудно заранее сказать, в каких компаниях они будут работать). Количество инсталляций коробочного продукта неограниченно (собственно, инсталляции и продаются), войти в контакт с пользователями до окончания разработки затруднительно.

Но и в случае коробочного продукта имеются средства для организации его пилотного использования. Например, известный коробочный продукт – многоязыковая среда разработки компании JetBrains под названием IDEA – имеет общедоступную, открытую версию (Community Edition), которую каждый желающий может бесплатно установить и ознакомиться с новыми возможностями продукта, а также выдать компании-производителю обратную связь[4]. Эта информация оказывается крайне важной для компании, так как используется при проектировании новой функциональности продукта.

Отмечу, что я не стал детально рассматривать ещё один распространённый вид программной системы — Интернет-сервис. С одной стороны, он поход на решение, так как работает в рамках некоторой компании — установлен на её серверах, взаимодействует с её данными. С другой стороны, сервис имеет черты коробочного продукта, поскольку его пользователи находятся за пределами компании и их число заранее не ограничено. Мне кажется, что апробация сервиса не имеет принципиальных отличий от апробации коробочного продукта, точно также ориентируясь на некоторую специальную выборку пользователей.

> **Апробация** созданной вами программной системы подразумевает её *тестирование и опытную эксплуатацию* в рамках пилотного проекта или организованного иным способом тестового использования *реальными пользователями*.
>
> **Пилотный проект** – это реальное использование вашего решения/системы сторонними пользователями, но – в рамках пробного, некритичного проекта.

Таким образом, пилотный проект отличается от ситуации, когда несколько сторонних пользователей (то есть не вы сами или ваше ближайшее окружение) всего лишь «поиграли» с системой. Пилотный проект – это реальное внедрение в рамках реального бизнес-процесса заказчика с тем лишь ограничением, что взяты некритичные пользовательские данные, или создана «искусственная» ситуация, или выполнено использование в реальном, но не критичном проекте пользователей. Здесь важно, что пользователи и ситуация — настоящие, но не «боевые».

Экспериментальное исследование. Немного особняком в списке возможных вариантов практической полезности стоит экспериментальное исследование – сравнение ваших результатов с имеющимися аналогами. Оно целесообразно в первую очередь, для научных или исследовательских дипломов. В рамках же производственных дипломов (да и индустриальных проектов в целом) обычно нет времени и сил на проведение трудоёмких экспериментов для сравнения разработанной системы с аналогами. В этом случае достаточным является приемлемое качество продукта как такового (и это достигается тестированием), а также позитивная реакция пользователей. Необходимость дополнительных экспериментов в индустрии возникает сравнительно редко и связана она,

[4] Следует отметить, что дело обстоит несколько сложнее: Community Edition среды разработки от компании JetBrains предназначается для тестирования многих новых экспериментальных возможностей, которые в итоговую поставку не входят. Более того, имеется несложный механизм, с помощью которого пользователи могут включить в продукт свои собственные возможности – и они войдут в Community Edition, но, разумеется, не войдут в базовую поставку. Таим образом, Community Edition является средством взаимодействия с сообществом пользователей продукта.

как правило, с исследованием отдельных критических аспектов продукта, например, его производительности, или с поиском трудно обнаружимых ошибок, например, состояний гонок по данным в многопоточных приложениях.

Отмечу также трудность сопоставления реальных промышленных программных систем. Как правило, сопоставляются алгоритмы, реализующие некоторый функционал, и оказывается, что такие алгоритмы, такой функционал выделить из целого продукта очень непросто. Бывает также, что вклад в итоговый функционал оказывает, помимо алгоритма, также операции ввода/вывода, которые могут быть неразрывно соединены с самим алгоритмом (например, ввод/вывод делается не только в начале и конце, а во время всей работы алгоритма), и отделить алгоритм от этих операций, не меняя кода, невозможно.

Но в рамках научных или исследовательских дипломов продукт реализует именно некоторый алгоритм или определенную идею, являясь прототипом, единственная цель которого заключается в том, чтобы показать, продемонстрировать, на сколько эффективной является предложенная идея и/или её реализация, а пользоваться этим продуктом не представляется возможным. Поэтому его легко можно, а поэтому и нужно сопоставить с аналогами – такими же прототипами, ссылки на которые часто содержатся в научных статьях.

Отмечу также, что в случае научного диплома экспериментальное исследование может служить доказательством валидности полученных научных результатов, особенно в ситуации отсутствия формальных доказательств. Как я писал выше, доказуемость является очень важным критерием научности. А как доказать, если в научной работе не использован формальный математический аппарат? Только с помощью экспериментов![5]

В экспериментальных исследованиях мы стараемся уйти от следующей простой ситуации: мы де погоняли наш алгоритм на некоторых тестах – и все оказалось хорошо, наш алгоритм оказался эффективным. Когда я читаю такое описание экспериментов, то возникают следующие вопросы.

- Как составлялись тесты и почему их достаточно, чтобы сделать позитивные выводы?
- Если при этом использовались какие-то данные, то как они были получены, и почему их также достаточно?
- Как был организован эксперимент?
- С какими аналогами вы сравнивались, и почему были выбраны именно эти аналоги? И может ли быть такое, что имеются другие аналоги, с которыми вы не сравнивались, но которые лучше, чем ваш результат?
- Наконец, что означает, что «всё хорошо», «алгоритм оказался эффективным» и прочие подобные утверждения?

Следует понимать, что экспериментальное исследование является именно исследованием, и поэтому должно бы имеет свои четко поставленные задачи. Также нужно отметить, что оно весьма трудоёмко, а его сложность студенты (и не только они!) часто недооценивают, поэтому эксперименты запаздывают, оказываются не до конца объективными, представительными, да и просто незаконченными. Таким образом, их надо тщательно проектировать.

Экспериментальное исследование можно разделить на следующие шаги:

1) проектирование экспериментов;
2) выполнение экспериментов;
3) описание экспериментов в тексте вашего диплома.

[5] Тут следует отметить, что я не согласен с мнением, встречающемся во многих учебных заведениях России, в силу которого научным диплом является лишь в том случае, если в нём есть формулы и теоремы, т.е. использован математический аппарат. В частности, в программной инженерии как науке (а программную инженерию можно рассматривать и как промышленную деятельность, и как область обучения) математический аппарат используется крайне редко и весьма незначительно.

Казалось бы, в рамках данного курса следовало бы ограничиться третьим пунктом – описанием экспериментов в тексте диплома. Однако на практике эксперименты часто выполняются параллельно с написанием текста диплома (как я указывал выше – они запаздывают из-за недооценки их сложности, а также из-за слабой структуры и хаотичности). Кроме того, будучи ненадлежащим образом спроектированными и выполненными, эксперименты не могут быть адекватно описаны в тексте. Поэтому рассмотрим в этой лекции, наряду с описанием экспериментов, также их проектирование и выполнение.

Проектирование экспериментов. Основными китами эксперимента являются следующие аспекты:

- вопросы экспериментального исследования,
- тесты и/или данные,
- сравниваемые аналоги (с кем вы сравнивайтесь).

Эта триада представлена на рисунке 10.

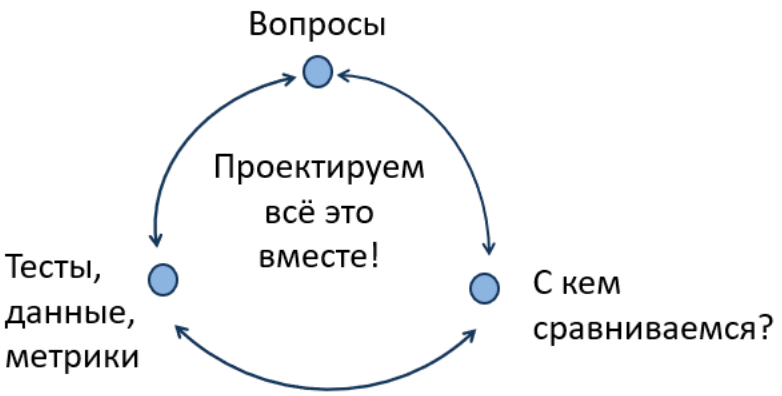


Рис. 10. Триада, составляемая при проектировании экспериментов

Что такое вопросы экспериментального исследования? Оказывается, что недостаточно просто проверить эффективность разработанного алгоритма/метода/системы. Нужно уточнить, что именно понимается под эффективностью, а также представить разные аспекты этой эффективности. Ниже приведены классические вопросы экспериментального исследования для некоторого нового (созданного в рамках вашего диплома) алгоритма.

1. Какой вклад в эффективность алгоритма вносят его отдельные шаги? Смысл этого вопроса в том, что вы можете представить сложных алгоритм с большим количеством разных шагов — и не ясно, сколь они существенно каждый из них улучшает общую производительность алгоритма, и не может ли так оказываться, что, отказавшись от какого-нибудь из них (например, про который написано больше всего теорем и текста), производительность алгоритма не изменится… Эксперименты для ответа на этот вопрос подразумевают отключение отдельного шага алгоритма и проверку оставшегося алгоритма на тех же данных/тестах с тем, чтобы оценить вклад отключенного шага.
2. Как предложенный алгоритм соотносится с существующими аналогами?

Могут понадобиться и другие вопросы. При этом для ответа на разные вопросы бывает необходимо поставить разные эксперименты, но может оказаться, что один эксперимент способен ответить на несколько или даже на все вопросы.

Вопросы экспериментального исследования конкретизируют понятие эффективности алгоритма/метода/системы, а также формулируют различные аспекты этой эффективности. Для ответа на такие вопросы может понадобиться больше, чем один эксперимент. Эти вопросы формулируются в начале исследования, и потом, при обсуждении результатов экспериментального исследования, позволяют структурировать это обсуждение.

Теперь рассмотрим вопрос о том, с какими аналогами целесообразно сравнивать ваш алгоритм. Рассмотрим первый вариант вопроса экспериментального исследования – вклад в эффективность алгоритма вносят его отдельные шаги. В этом случае важно уметь отключать тот или иной отдельный шаг алгоритма и сравнивать его работу в полном составе с одним отключенным шагом, потом – с отключенным другим шагом и так далее. В этом случае не требуется сравнения с аналогами. Такой вариант является достаточным, если вы надстраивали известный эффективный алгоритм – вы его улучшили, но и он исходно был хорошо. Поэтому сравниваться с чем-то ещё избыточно. Но если у вас иная ситуация, то аналоги нужны.

В этом случае нужно искать запускаемые аналоги (лучше всего, если для них будет также доступен и исходный код), создавать общую программную инфраструктуру для автоматизации эксперимента, и многократно запускать все найденные вами аналоги вместе с вашим алгоритмом на одних и тех же тестах/данных. Важный вопрос – где взять такие алгоритмы?

Лучше всего иметь стороннюю программную реализацию сравниваемых алгоритмов. В крайнем случае можно выполнить собственную реализацию, используя описание алгоритма в некоторой статье. Но второй вариант менее предпочтительнее, поскольку по необъяснимой причине ваша реализация *чужого* алгоритма показывает менее эффективные результаты, чем ваш собственный алгоритм.

Тесты/данные, а также метрики являются последним элементом базовой триады экспериментального исследования. Начнем с тестов/данных. Тесты оказываются важными, когда вы исследуете эффективность некоторого алгоритма, обрабатывающего программы на некотором языке, например, алгоритм символьного исполнения. Данные требуются, фактически, во всех остальных случаях. Очевидно, что и тесты, и данные означают, по существу, одно и тоже, поэтому ниже не будем их различать и поговорим только о данных.

Очень хорошо найти стандартные наборы открытых, общедоступных данных (benchmarks). Такие наборы имеются в разных областях. Это важно, поскольку такие наборы являются известными, другие исследователи их активно использовали и результатам экспериментов, выполненных на этих наборах, научное сообщество доверяет. Однако, может оказаться, что хоть стандартные наборы данных и имеются, но извлечь из них нужные вам данные является непростой задачей. Тогда их надо извлечь. Приведу пример.

Существуют открытые данные, содержащие информацию, изъятую из систем управления ошибок для ряда известных открытых проектов за последние 10–20 лет. Это очень ценная информация для различных новых алгоритмов – на ней можно провести показательные эксперименты. В частности, описания некоторых ошибок, представленных там, содержат stack traces, и если вы разработали новый алгоритм по анализу stack traces, то его целесообразно проверить на этих данных. Но для этого stack traces следует извлечь из этих данных и представить в нужном для вас формате. Кроме этого, их ещё также может быть целесообразным разбить на группы по сходству. Всё это может выполнить специально написанная вами программа (скрипт) — вручную этого не сделать в виду большого объёма данных.

Помимо стандартных наборов данных в экспериментальных исследованиях можно использовать данные вашей компании. Вряд ли вам разрешат их выложить в открытый

доступ вместе с кодом вашего алгоритма (даже в обезличенном виде), но если ваша компания является известной (например, Huawei, Яндекс или JetBrains), то использование таких данных сильно повысит солидность вашей работы.

Наконец, можно использовать так называемые *синтетические данные* – то есть данные, которые вы создали сами. Такой вариант доступен всегда, но он наименее предпочтителен, так как совпадают авторы алгоритма и тестируемых данных – известно, что самопроверки менее эффективны, чем проверки независимой стороной.

Скажу несколько слов о метриках. Со времён И. Ньютона количественный подход стал одним из столпов науки – научное знание стало выражаться в численных измерениях. Напротив, до этого была распространена натурофилософия, которая подходила к изучению природы, фактически, сугубо на основе размышлений и философствования, создавая умозрительные теории; эксперименты же и численные измерения либо вовсе не делались, либо играли незначительную роль.

> **Важно.** Не будьте натурофилософом, занимаясь экспериментами и формулируя результаты экспериментального исследования, например, следующим образом: я проверил свой алгоритм на тестах, и он показал хороший (или даже отличный результат). Постарайтесь объективно выразить результаты работы вашего алгоритма с помощью *измерений*, трактуя последние с помощью определённых *числовых метрик*.

Простейшей метрикой является время работы алгоритма (в секундах или миллисекундах), а также потребляемая память. Более редкой метрикой является покрытие процент покрытия тестом кода. В последние годы прочно вошли в обиход исследователей метрики из информационного поиска. Вот простейшие из них:

- точность вашего алгоритма — это процент правильных ответов в выдаче (Precision);
- полнота результатов, то есть на сколько в выдаче вашего алгоритма присутствуют все имеющиеся правильные результаты (Recall).

Имеются и более сложные метрики, такие как F-мера, Accuracy, ROC AUC и пр. Я не буду подробно останавливать на этом вопросе, поскольку он является содержанием целого курса по эффективной статистке. Интересующиеся могут почитать литературу, содержащуюся в списке, приложенном к данной лекции.

> **Важно.** Старайтесь без нужды не изобретать своих метрик, а используйте стандартные; при этом, как минимум, следует знать, что такое точность и полнота (вдруг это вам подойдёт?). вам наверняка об этом рассказывали, но вот теперь пришло время приложить эти знания к практике.

Контрольные вопросы

1) Как вы понимаете практическую полезность диплома по программированию?
2) Как следует реализовать в рамках дипломного проекта обоснование практической полезности ваших результатов? Какие здесь возможны варианты?
3) Как в тексте диплома следует оформить описание практической полезности?
4) Что такое тестирование?
5) В каких случаях тестирование может отсутствовать?
6) Что вы можете сказать про апробацию?
7) Что такое пилотный проект?
8) В чём особенность апробации решения и коробочного продукта?
9) Объясните, что такое экспериментальное исследование.
10) Расскажите о внедрении. Чем оно отличается от апробации?
11) В чем, на ваш взгляд, особенность экспериментального исследования?
12) В каких случаях экспериментальное исследование необходимо, а когда без него можно обойтись?
13) Расскажите о связи экспериментального исследования с доказательством валидности полученных научных результатов.
14) Перечислите шаги экспериментального исследования.
15) Расскажите о том, ка выбираются аналоги для сравнения.
16) Какую дополнительную программистскую работу приходится делать, имея россыпь реализаций других алгоритмов (сравниваемых аналогов)?
17) Что такое стандартные наборы данных (benchmarks) и зачем ни нужны?
18) Что такое синтетические данные?
19) Зачем нужны метрики экспериментального исследования?
20) Что такое точность алгоритма?
21) Что такое полнота алгоритма?

Список литературы

1) R. Van Solingen, V. Basili, G. Caldiera, & H.D. Rombach. Goal question metric (gqm) approach //Encyclopedia of software engineering. John Wiley & Sons 2002.
2) Х. Шенк. Теория инженерного эксперимента/ Пер. с англ. М. 1972.
3) М. Кристофер, Ш. Хайнрих, Р.Прабхакар. Введение в информационный поиск/ Пер. с англ. Изд-во Вильямс. 2020.

Лекция 8. Про объекты, встроенные в текст. Часть I

В этой лекции даётся определение объекта, встроенного в текст, обсуждается механизм работы таких объектов при чтении текста – нежелательный («по умолчанию») и желательный. Подробно рассматривается специальный вид объекта, встроенного в текст – рисунок и схема. Приводится шаблон для описания структуры системы с помощью UML. Приводятся примеры из реальных студенческих работ.

Алиса считала, что книга без картинок … неинтересная.

Льюис Кэрол. Алиса в стране чудес.

Здорово, когда в научной работе есть какое-нибудь волшебство!

Д. В. Луцив, к. ф.-м.н., ст. препод. кафедры системного программирования СПбГУ

В этой лекции предлагается приступить к следующей фазе проектирования текста вашего диплома — созданию основных встроенных объектов. Речь идет о рисунках, графиках, таблицах, формулах и пр. Эти объекты являются определённой квинтэссенцией содержания вашего текста, появление их неслучайно, а место расположения в тексте тщательно выверяется. Одним словом, эти объекты целесообразно проектировать и создавать до написания текста, позже «оборачивая» в текст. Этот процесс может служить методом написания текста, когда трудно писать и не ясно, с чего начать.

Что такое объекты, встроенные в текст. Итак, в текст диплома вы можете вставлять следующие объекты, которые мы будем называть объектами, встроенными в текст:

- рисунки и схемы,
- графики,
- таблицы,
- листинги кода,
- математические формулы.

Эти объекты не являются ни предложениями русского языка, ни их частью, то есть, строго говоря, они оказываются нетекстом. Однако с этими объектами не нужно бороться, поскольку они способны сделать текст вашего диплома более содержательным и, что немаловажно, более понятным и доступным для читателя. Тем не менее, следует разобраться, как правильно создавать эти объекты, а также встраивать их в текст.

Сначала обсудим механизмы «работы» этих объектов — то, как оно часто бывает и чего лучше избегать, а также покажем то, к чему следует стремиться.

На рисунке 11 показана часто встречающаяся ситуация — что получается, так сказать, по умолчанию, если не прикладывать специальных усилий. На этом рисунке изображена ситуация, когда для того, чтобы понять рисунок или таблицу, имеющиеся в тексте, следует как следует разобраться в содержании самого текста и потом снова вернуться к этой таблице/рисунку. В таком случае встроенный объект не способствует повышению понятности текста, а напротив, препятствуют, являя собой дополнительный артефакт, с которым читателю, помимо основного текста, нужно специально разбираться.

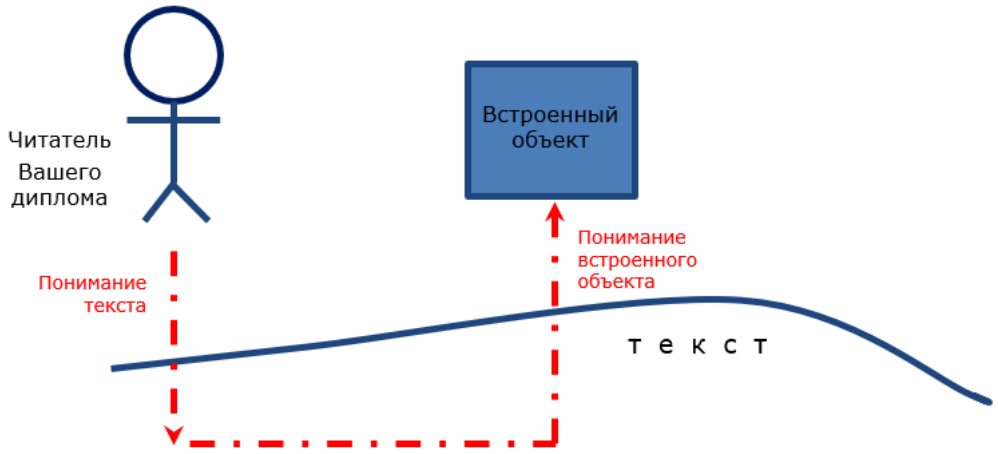


Рис. 11. Объекты, встроенные в текст: ситуация «по умолчанию»

На рисунке 12 представлен другой возможный механизм «работы» объекта, встроенного в текст. О чём же тут речь? О независимости и самодостаточности этих объектов при восприятии читателем текста вашего диплома, то есть в этом случае объект о том, что они должны быть понятны сами по себе, без чтения всего текста. И что читатель может пролистать ваш диплом, посмотреть на картинки и схемы, на графики и таблицы, на листинги кода и, наконец, на формулы – и составить себе некоторое впечатление о самом тексте. Конечно, весь текст ему так не понять (хотя, глубокому специалисту, быть может, этого будет и достаточно), но вход в вашу работу он получит. И потом, при дальнейшем чтении, восприятие таких объектов будет работать упреждающе, помогая ему схватывать суть и после её уточнять при внимательно прочтении соответствующего раздела, главы и т.д. Обозначим этот второй механизм в качестве желанного и дальнейшее изложение материала данной лекции будем строить на его основе.

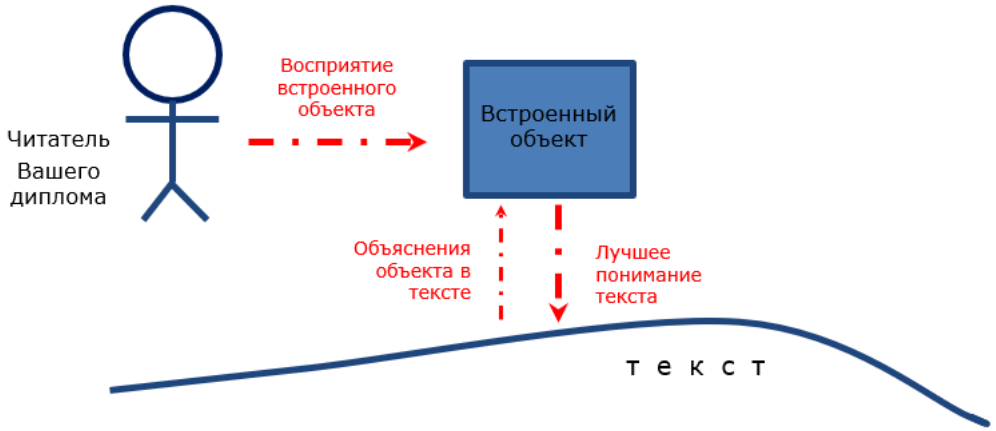


Рис. 12. Объекты, встроенные в текст: к чему можно стремиться

Рисунки и схемы. Зачем вообще в тексте нужны рисунки? Рисунки обладают наглядностью по сравнению с текстом, таблицами и т.п., и задействуют зрительное восприятие и связанную с этим образность, т.е. активизируют правополушарные процессы читателя. Рисунки создают определенные метафоры, ассоциативные ряды, помогая

погрузить читателя в предлагаемую информацию максимально полно. И в конечном счёте способствовать его лучшему и более быстрому понимаю излагаемой информации.

Лично мне при восприятии сложной информации всегда помогало наличие ёмких образов, таких, например, как рисунок семиуровневого сетевого открытого протокола. Такие образы хорошо запоминаются, аккумулируя в себе содержательную информацию о предмете. Правда, такие образы (представления) не всегда имеют визуальный ряд, зачастую оставаясь некоторым компактными, но бесформенными представлениями. Ряд таких представлений я составил себе ещё в студенческие годы во время освоения курсов по телекоммуникациям, архитектуре ЭВМ, базам данных и т.д. и успешно пользуюсь ими всю жизнь. И в новых областях, с которыми мне приходится сталкиваться (например, машинное обучение, символьное исполнение), я стремлюсь также сформировать такие представления для аккумуляции основных идей и подходов предметной области. Конечно, когда начинаешь погружаться в предмет глубоко, работая в выбранной области изо дня в день, годами, то оказываются востребованы другие механизмы восприятия информации, ориентированные на оперирование большим числом деталей, скорость принятия решения, на масштаб восприятия задач (как возможность видеть принципиально больше двух решений рассматриваемой задачи).

При формировании таких представлений удачные рисунки играют значительную роль. Например выше, стремясь объяснить роль встроенных в текст объектов, я нарисовал два рисунка. При этом я хотел наглядно выразить идею связи встроенного объекта и текста при чтении текста. Мне кажется, что у меня не плохо получилось, но, конечно, финальное суждение остаётся за вами, уважаемый читатель.

Таким образом, рисунок может появиться в той части вашего текста, где вы хотите донести до читателя некоторую важную идею, и где вам удаётся её изобразить подходящим способом. Вовлечение правополушарной активности читателя при чтении вашего диплома делает акт восприятия читателем вашего текста более полным, притом активизируя зрительный канал. Несмотря на то, что этот канал, вроде бы и так задействован — ведь читатель читает ваш текст глазами, — тем не менее такое зрительное восприятие является весьма механическим и не раскрывает всех возможностей зрения. В качестве дальнейшей отсылки в этом вопросе можно рекомендовать классически труд известного психолога Д.Гибсона, посвященный экологическому зрению. Гибсон представляет зрительный акт как целостное явление, в которое, в идеале, человек вовлечён весь, целиком! При этом важно помнить, что зрение у современного человека является доминирующим каналом восприятия, поэтому максимальное задействование этого канала способствует и более качественному восприятию, и синхронизации информации, воспринятой, имеющейся у разных людей. В качестве примера для последнего утверждения отмечу популярность чертежей в инженерно-строительных разработках – с помощью них проектировщики и инженеры передают свой замысел рабочим и монтажникам, т.е. чертежи оказываются способом профессионального общения.

> **Рисунки** в тексте помогают фокусировать внимание читателя на узловых идеях работы, активизируют правополушарные процессы читателя и способствуют более полному, более глубокому понимаю текста.

В качестве специального случая рисунка я выделяю схему (чертеж). Какова отличительная особенность схемы по сравнению с рисунками? Эта особенность заключается в том, что на схеме используется фиксированный визуальный язык, основанный на чертежном проектировании и использующий, преимущественно, метафору графа. Этот язык может быть очень простым и быть либо разновидностью, частью стандартных языков проектирования программного обеспечения, таких как UML, BPMN, SysML и др., либо модификацией этих стандартных языков. Пример схемы представлен на рисунке 13.

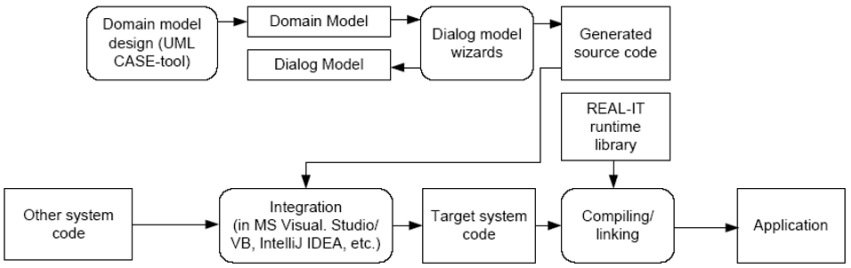


Рис. 13. Пример схемы[6]

На этой схеме представлена технология разработки информационных систем REAL-IT и язык этой схемы таков: овалами изображаются действия (осуществляемые как в самой технологии, так и за её пределами), а прямоугольниками – информационные артефакты (спецификации, модели, программы и пр.), которые подаются на вход действиям и/или порождаются ими. Обратите внимание, что эта схема сбалансирована в смысле верх/низ – верх более лёгкий, а низ – более массивный. Эта метафора оказывается естественной для человеческого восприятия, так как основывается на силе земного тяготения – если у конструкции верх тяжелее, чем низ, то она развалится, не сможет стоять вертикально. Апелляция к таким естественным метафорам сильно облегчает восприятие схемы.

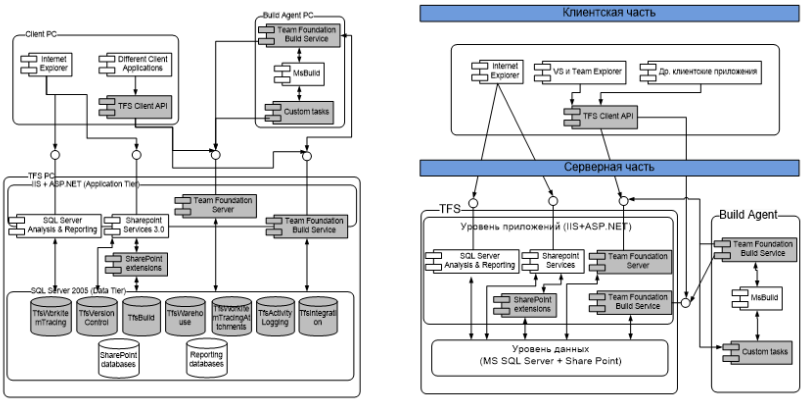


Рис. 14. Ещё один пример схемы

На рисунке 14 представлено две схемы, описывающие технологию компании Microsoft под названием Visual Studio Team System. Левый рисунок показывает первую версию схемы, правый – последнюю. Видно, что в итоговом варианте четче прослеживается структура – вся схема визуально поделена на клиентскую и серверную части, а в серверной части выделен основной сервер под названием TFS и вспомогательный узел Build Agent. Кроме того, в итоговой схеме главная серверная компонента TFS разбита на две – «Уровень приложений» и «Уровень данных», – и во второй скрыты составляющие её сущности, поскольку они не являются ключевыми для рассматриваемой технологии.

[6] Эта схема взята из статьи A. Ivanov, D. Koznov. REAL-IT: Model-Based User Interface Development Environment. Proceedings of IEEE/NASA ISoLA 2005 Workshop on Leveraging Applications of Formal Methods, Verification, and Validation. Loyola College Graduate Center Columbia, Maryland, USA, 23–24 September 2005: 31–41.

Схемы отличаются от произвольных рисунков тем, что для их создания используется специальный язык – некоторый ограниченный набор фигур и символов, каждый из которых имеет фиксированную семантику и синтаксис (т.е. правила связей друг с другом).

Последнее обстоятельство, а именно, наличие серии схем в тексте, является важным. В качестве примера могут служить рисунки из лекции 3, с помощью которых я объяснял, что такое управляющая структура диплома, или рисунки 11 и 12, на которых я пытался отразить нежелательный и предпочтительный принципы работы встроенных в текст объектов. В сложных дипломных работах эта серия может включать в себя до десяти и более схем, и в этом случае особенно важно наличие единого визуального языка этих схем. В этом случае представленные в тексте схемы складываются в единую спецификацию и могут восприниматься более-менее независимо от основного текста. И это может оказаться очень ценным в том случае, если вы, например, описываете сложную систему, в которой имеются разные «слои» – компонентная архитектура, функциональность системы, размещение (deployment) системы на оборудовании и так далее. И для всего этого могут использоваться различные схемы, которые, между тем, должны быть между собой согласованы — они используют одни и те же сущности и имена.

Пример такой серии схем представлен на рисунках 15–19[7]. На рисунке 15 в максимально общем виде представлена технология PEToDA, которая является специализированным framework для создания инструментария по анализу производительности, удовлетворяющего специфическим требованиям конкретного проекта/информационной системы. В силу того, что эти требования различаются, а также из-за реализационных особенностей, требуется гибкая технология, позволяющая учесть всю эту специфику. На рисунке показано, что на вход PEToDA получает целевую систему для анализа производительности, а также требования для выполнения этого анализа. Также показаны основные блоки технологии – модель типового инструментария, среда для разработки целевого инструментария, средства интеграции инструментария с другими методами/средствами анализа производительности, а также платформа (run time), на которой инструментарий будет выполняться.

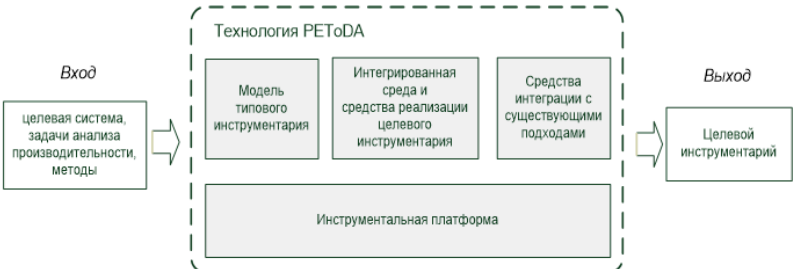


Рис. 15. Обзор технологии PEToDA

На следующем рисунке показана верхнеуровневая функциональная модель целевого инструментария, который создаётся с помощью предлагаемой технологии. Показано, что первым функциональным блоком является планирование, управление и контроль экспериментов по анализу производительности, и здесь подразумеваются специальные средства, включая пользовательский интерфейс и инструменты для создания необходимого специализированной функциональности. Второй функциональный блок посвящён генерации нагрузки для целевой информационной системы и выполнению

[7] Эти примеры взяты из работы Евгения Рачинского «Анализ производительности клиент-серверных систем» (дисс. на соискание степени кандидата физико-математических наук, СПбГУ, 2010).

измерений результатов. Наконец, третий функциональный блок посвящён анализу результатов измерений – статистическому анализу и генерации необходимых отчётов.

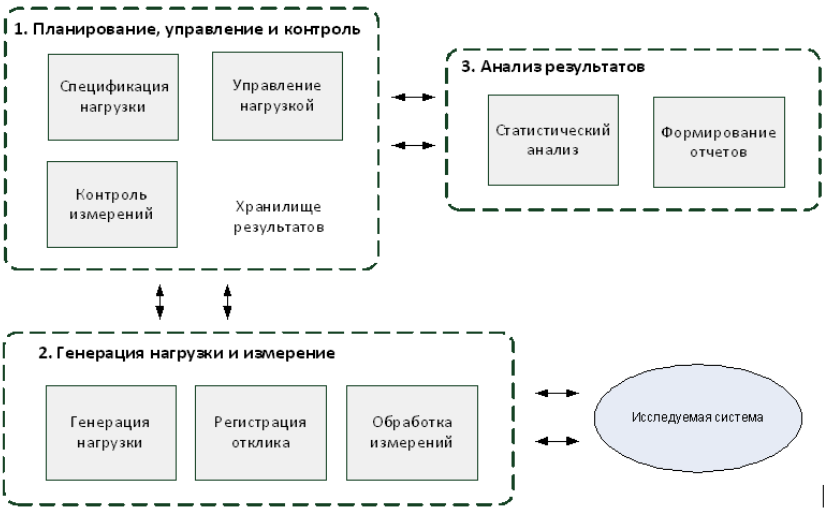


Рис. 16. Функциональная модель типового инструментария

На следующем рисунке представлена базовая схема работы целевого инструментария, создаваемого с помощью технологии PEToDA. Сначала выполняется компонент инструментария, далее запускается сеанс работы (старт эксперимента). После этого исполнение разбивается на две параллельные ветки – работа генератора нагрузки и выполнение управление его работой, включая сбор поступающей информации.

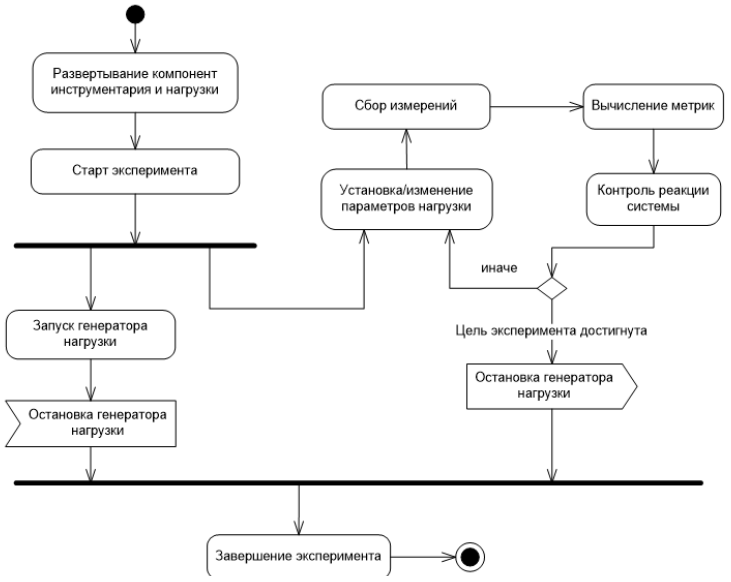


Рис. 17. Базовая схема работы целевого инструментария

На этом рисунке представлена компонентная модель технологии PEToDA. Серым цветом показаны компоненты, которые не зависят от конкретного целевого инструментария, реализованы в PEToDA её авторами и должны использоваться as is. Компоненты, окрашенные в белый цвет, или присутствуют в PEToDA в виде примеров и должны быть соответствующим образом модифицированы, или вовсе отсутствуют и должны быть реализованы в рамках создания целевого инструментария, так сказать, «с нуля», но соблюдая определённые для них интерфейсы.

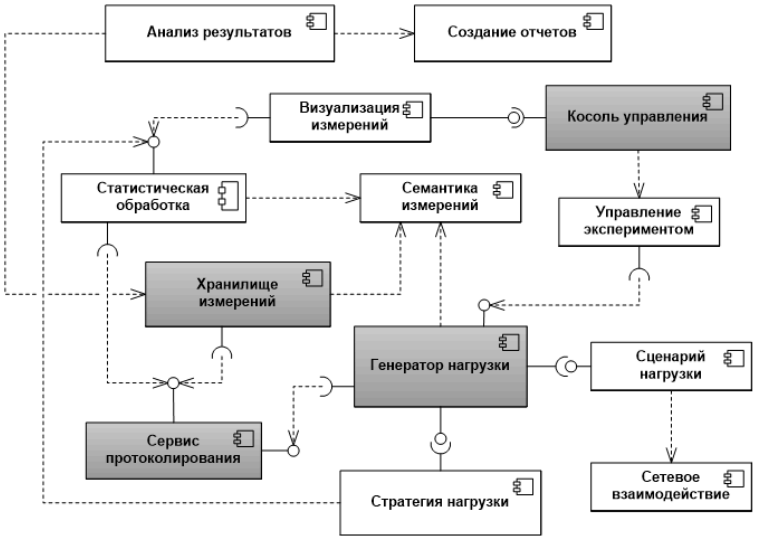


Рис. 18. Компонентная модель целевого инструментария

На этом рисунке показано размещение компонент целевого инструментария на вычислительных ресурсах. Имеется одна рабочая станция для управления экспериментом, на которой находится консоль управления, хранилище измерений, и с которой оператор выполняет контроль эксперимента и анализ результатов. И далее имеется несколько рабочих станций (удалённые PC-агенты), которые выполняют генерацию нагрузки на исследуемую систему. Разделение по разным компьютерам управления экспериментом и генерации нагрузки выполнено в связи с ресурсоёмкостью последней, причём может быть организовано даже несколько серверов для генерации нагрузки. Наконец, удаленные PC-агенты соединяются напрямую с исследуемой информационной системой.

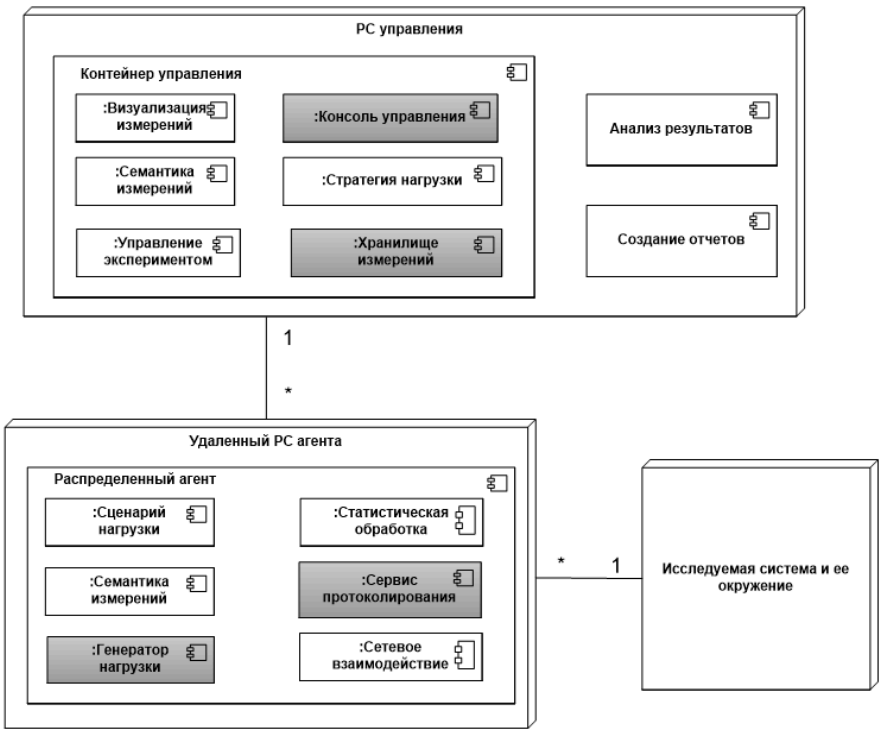


Рис. 19. Диаграмма развёртывания целевого инструментария на вычислительных ресурсах

Таким образом я продемонстрировал, как можно создать набор различных схем и описать их в тексте – хотя должно быть очевидно, что приведённые мною описания являются краткими. Можно было их расширить детальным описанием каждой компоненты с рисунка 18, а также различными другими деталями. Отмечу также, что две последних схемы используют одно и то же множество компонент (но, однако, на последней схеме автор забыл упомянуть компоненту «Статистическая обработка»). Бывает очень странно, когда в таких случаях имеются различные наборы компонент. Также отмечу, что в этих схемах использовался UML, который был несколько расширен. Рисунок 16 не являются диаграммами UML, рисунок 17 является диаграммой активностей UML, рисунок 18 – диаграммой компонент UML, рисунок 19 – диаграммой развертывания UML.

В заключении отмечу, что не нужно вставлять в текст диплома слишком много рисунков и схем – их нужно ровно столько, сколько нужно для достижения ваших целей. Ведь в принципе, используя UML и имея определенный навык, можно весь диплом изложить только с помощью схем и рисунков. Или создать, уже безо всяких навыком, очень много скриншотов интерфейса своего инструмента. И тогда получится, что уже текст на русском языке нужно будет встраивать в последовательность рисунков (видимо, в виде комментариев). И у нас получится нетекст.

Но мы стремимся писать текст. Поэтому рисунки и схемы целесообразно создавать лишь для каких-то важных, узловых аспектов работы, при этом каждая схема/рисунок должна иметь оптимальный размер. Также, если вы вставляете скриншоты, то их не должно быть слишком много (а очень легко и как-то само собой в тексте оказывается большое количество скриншотов интерфейса вашей системы!), и хорошо, когда эти скриншоты свободны от большого количества деталей, для которых отсутствуют пояснения в вашем тексте.

Контрольные вопросы

1) Какие объекты, встроенные в текст, вы можете назвать?
2) Опишите схему работы объекта, встроенного в текст, по «умолчанию».
3) Опишите желаемую схему работы объекта, встроенного в текст.
4) В каких случая стоит создавать рисунок?
5) Чем именно помогают рисунки в тексте?
6) Чем схема отличается от рисунка?
7) Назовите примеры языков для создания схем в области проектирования программного обеспечения.
8) Каковы особенности создания серии схем?
9) Каковы особенности встраивания в текст схем?

Список литературы

1) М. Фаулер, К. Скотт. UML в кратком изложении: Применение стандартного языка объектного моделирования / Пер. с англ. М.: Мир, 1999.
2) Д.В. Кознов. Основы визуального моделирования. М.: БИНОМ, 2008.
3) Д. Гибсон. Экологический подход к зрительному восприятию. М.: Прогресс, 1988.
4)

Лекция 9. Про объекты, встроенные в текст. Часть II

В этой лекции продолжается обсуждение объектов, встроенных в текст. Рассматриваются таблицы — их назначение и структура, — графики, листинги кода (включая псевдокод) и математические формулы.

Продолжим обсуждение объектов, встроенных в текст. Таким образом, у нас остались не рассмотренными:

- таблицы,
- графики,
- листинги кода и
- математические формулы.

Таблицы. Хорошо спроектированные и оформленные, понятные, ёмкие таблицы существенно облагораживают текст, позволяя собрать и представить различную справочную или иную *однородную* и *атрибутивную* информацию. Однородность информации означает, что мы имеем набор информационных элементов определённого вида – программных продуктов определённого класса (например, различные СУБД), алгоритмов, персон и так далее. Атрибутивность означает наличие у каждого информационного элемента имеется набор одних и тех же атрибутов, которым могут соответствовать различные значения, и в таблице атрибуты выражаются с помощью разных колонок таблицы (один атрибут соответствует одной колонке). Зачастую однородную информацию трудно удачно представить в тексте другим способом, разве что, с помощью графиков (но последнее не всегда удобно).

> **Таблицы** являются структурированным представлением однородной и атрибутивной информации, позволяя суммировать, собрать информацию для того, чтобы сделать аргументированные выводы или структурировать описание в тексте однородных объектов.

Вот некоторые варианты ситуаций, кода таблицы могут оказаться полезными в тексте вашего диплома.

- В виде таблицы удобно делать резюме в том или ином разделе текста, так сказать, «собирать» информацию.
- С помощью таблицы оказывается естественным упорядочить различные однородные данные по апробации, по рассмотренным технологиям/алгоритмам/системам и пр.
- Таблицы являются хорошим способом для представления большого объёма данных — например, помещённые в приложения к дипломному тексту.

Остановимся подробнее на структуре таблицы. Она включает в себя короткий и «говорящий» заголовок, возможно, подзаголовки, а также имена столбцов и строк, и иногда — примечания. На рисунке 20 представлен пример таблицы[8]. В этом примере обратите также внимание на дизайн таблицы — он выполнен тщательно и прихотливо: соседние строки даются разными оттенками серого, отсутствуют разделительные линии у столбцов, используются разные шрифты и прочее.

[8] Данная таблица взята из статьи Thomas M. Annesley. Bring Your Best to the Table, Clinical Chemistry October 2010, 56 (10): 1528–1534.

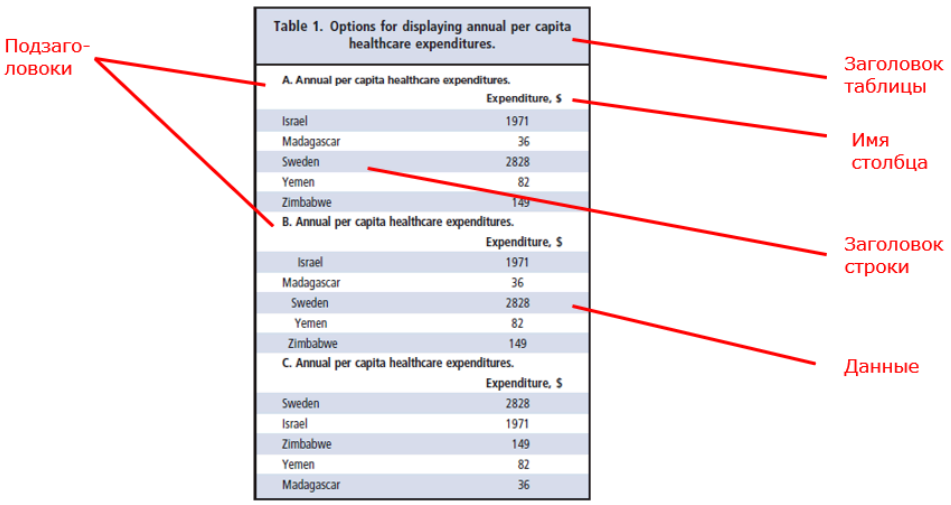


Table 1. Options for displaying annual per capita healthcare expenditures.

A. Annual per capita healthcare expenditures.	
	Expenditure, $
Israel	1971
Madagascar	36
Sweden	2828
Yemen	82
Zimbabwe	149
B. Annual per capita healthcare expenditures.	
	Expenditure, $
Israel	1971
Madagascar	36
Sweden	2828
Yemen	82
Zimbabwe	149
C. Annual per capita healthcare expenditures.	
	Expenditure, $
Sweden	2828
Israel	1971
Zimbabwe	149
Yemen	82
Madagascar	36

Рис. 20. Пример таблицы

	Tool	Developed by	Used	Released
T1	CaptainFeature	Fachhochschule Kaiserlautern in Germany.	A	2002
T2	Pure::Variants	Pure-Systems' company in Germany.	I	2003
T3	FeatureIDE	Otto-von-Guericke-University Magdeburg in Germany.	A	2005
T4	RequiLine	Research Group Software Construction in Germany.	B	2005
T5	XFeature	An association of P&P Software Company with the Swiss Federal Institute of Technology	B	2005
T6	MOSKitt	gvCASE project by the Valencian Regional Ministry of Infrastructure and Transport in Spain.	A	2008
T7	Feature Modeling Tool	Research Group of GIRO at the University of Valladolid and Burgos in Spain.	A	2008
T8	Feature Model DSL	Gunther Lenz and Christoph Wienands in Practical Software Factories in .NET book.	A	2008
T9	CVM Tool	European project ATESST in Germany.	B	2009

Рис. 21. Пример таблицы[9]

Графики. Остановимся на одном частном случае рисунков — на графиках. Графики визуально и наглядно показывают динамику изменения некоторых связанных величин, например, график движения некоторого тела в зависимости о времени, или распределение положительных и отрицательных ответов относительно разделяющей гиперплоскости. Таким образом, график, фактически, визуализирует некоторую математическую формулу. При этом, как правило, используются двухмерные зависимости и, преимущественно, в декартовой системе координат (иногда, впрочем, используются логарифмические или полярные координаты).

> **График** является специальным видом рисунка и предназначен для визуального и наглядного отображения некоторой значимой математической зависимости между определёнными данными.

[9] Эта таблица взята из статьи M. Dammagh, O. De Troyer. Feature Modeling Tools: Evaluation and Lessons Learned, Advances in Conceptual Modeling. Recent Developments and New. LNCS, 2011, vol. 6999: 120–129.

Однако, обратите внимание на то, что если у вас имеются зависимые данные, то из этого автоматически не следует, что их обязательно нужно помещать на графики, а эти графики помещать в текст вашего диплома. Следует отметить, что график должен быть *наглядным* — визуализация некоторой зависимости между данными сама по себе не совершает волшебства. Если, например, все значимые точки на графике собрались в одной небольшой области, а большая часть графика оказывается пустой, и изменением масштаба ситуацию не решить, то такой график лучше не помещать в текст работы. В качестве примера приведем три графика, представленных на рисунке 22. На этих графиках показаны результаты сравнения двух инструментов — KLEE и KLEE+LI+OPT — по трём параметрам: времени работы, используемой памяти, покрытию кода. Даже не вдаваясь в детали очевидно, что графики получились совсем неинтересными.

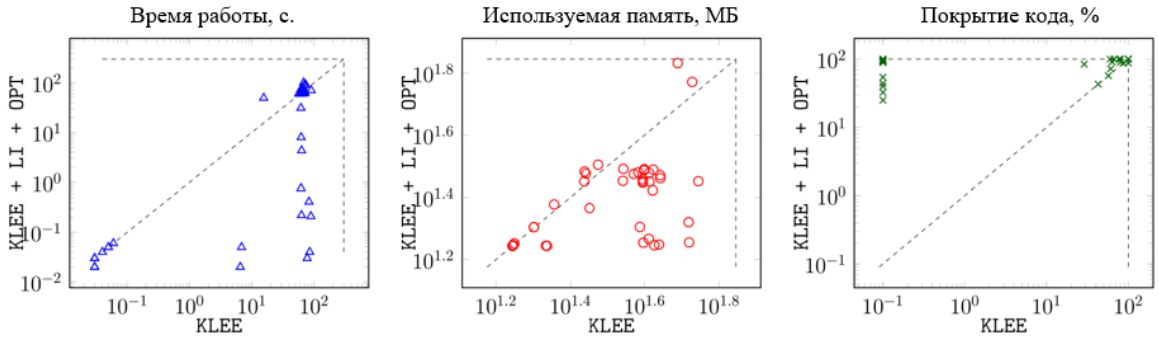


Рис. 22. Пример неудачных графиков

В подобных ситуациях вместо графиков можно представить таблицы, которые, не претендуя на особую наглядность, строго представят однородные данные. Так, вместо графиков с рисунка 22 была создана таблица, фрагмент которой представлен ниже (смотрите рисунок 23).

Тестовые С-программы	Время работы, с		Оперативная память, МБ		Процент покрытого кода,%	
	KLEE	KLEE+ LI+OPT	KLEE	KLEE+ LI+OPT	KLEE	KLEE+ LI+OPT
abs.c	0.03	0.04	20	20	100	100
aggregate.c	0.02	0.02	17.3	17.4	100	100
array_equality.c	0.02	0.02	17.3	17.3	100	100
array_sum.c	0.02	0.02	17.4	17.4	100	100
arraylist.c	3	0.05	20.5	17.5	0	88
avl_balance.c	62.4	77.7	42	32.31	0	29
avl_find.c	72	90	44.6	33.4	66.7	83
avl_height.c	68.8	78.5	47.9	26.3	0	76
avl_insert.c	61	98	32	40	60.9	69
binomial_heap.c	69.7	61.5	43.4	30	0	95
boolean.c	0.04	0.04	17.3	17.4	100	100
get_sign.c	0.03	0.03	17.3	17.4	100	100
integer_series.c	7.8	56.1	21.3	23.6	0	100

Рис. 23. Вместо графиков – таблица

Однако полный размер этой таблицы оказался большим, заняв несколько страниц (тут представлен лишь её фрагмент). Тогда ее поместили в приложение к статье, а в основной

текст вставили укрупненную таблицу, представленную ниже (смотрите рисунок 24)[10]. Таким образом был решен вопрос с неинформативными графиками. Возможны и другие варианты, например, изменение метрик, — возможно, что в этом случае удастся получить более наглядный график. Например, в экспериментах с алгоритмами машинного обучения можно измерять результаты с помощью целого спектра всевозможных метрик, и среди них можно выбрать те, которые выпукло отразят ваши результаты[11].

Продолжая тему наглядности графика, отмечу, что целесообразно делать его «игрушечными», то есть демонстрировать некоторую *идею*, а не показывать, как обстоят дела на самом деле. Ведь на самом деле всё может быть непросто и многомерно, и попытка с помощью одного графика представить некоторые реальные индустриальные данные в большинстве случаев оказывается неинформативной. Ну разве что у вас будет идея показать, что чего-то очень много: например, что для индустриальной системы существует и хранится очень много аварийных стеков вызовов (stack traces), или что их количество в определенный моменты жизненного цикла системы (сразу после выпуска новой версии и передачи её пользователям) очень быстро возрастает.

Инструмент	Время работы, с.			Оперативная память, МБ			Среднее покрытие кода, %
	Мин.	Макс.	Средн.	Мин.	Макс.	Средн.	
KLEE	0.02	98.8	42.3	17.3	106.5	41	48.3
KLEE+LI+OPT	0.02	128.1	43.2	17.3	120.1	28.4	86.9

Рис. 24. Пример обобщающей таблицы

Отмечу также ещё один важный момент про графики. Хорошо, если в графике имеется некоторая *интрига*, особенность в зависимости между данными, которая на нём отчётливо видна. И важно притом, чтобы эта особенность была бы интересной, положительно характеризовала ваши эксперименты и так далее. Фактически, наличие это интриги, особенности, во многом и оправдывает график — иначе можно было бы ограничиться таблицей. Обратимся к примеру, приведённому на рисунке 25. Идея этого графика заключается в том, что год от года на федеральном портале российской Федерации накапливается все больше открытых данных. При этом на графике есть интрига: в 2016 году был большой скачок, то есть одномоментно было выложено очень много данных. Этот скачок был связан с тем, что к федеральному порталу подключились региональные порталы, и все данные, которые были накоплены на них, стали доступны на федеральном портале.

> **Важно.** График должен быть наглядным, иметь некоторую *идею* и содержать *интригу*, то есть быть интересным.

Как и прочие объекты, встроенные в текст, график следует описать в основном тексте вашей работы. Нужно описать те данные и ту зависимость между ними, которая изображается. Также следует пояснить используемую систему координат. Можно также

[10] Этот пример и соответствующие таблицы были взяты из статьи А.В. Мисонижник, А.А. Бабушкин, С.А. Морозов, Ю.О. Костюков, Д.А. Мордвинов, Д.В. Кознов «Автоматическое тестирование LLVM-программ со сложными входными структурами данных» (Иванниковские чтения 2022), в написании которой я принимал активное участие.

[11] Но подходящую метрику можно и не найти. Так, в статье TraceSim: a Method for Calculating Stack Trace Similarity. (R.Vasiliev, D. V. Koznov, G. A. Chernishev, A. Khvorov, D. V. Luciv, N. Povarov, MaLTeSQuE@ESEC/SIGSOFT FSE 2020: 25–30) мы не только отказались от графиков, но даже, представляя результаты экспериментов в таблицах, мы ввели новую, собственную метрику Error Redaction. Проблема была в том, что наш алгоритм давал совсем небольшое численное преимущество по сравнению с существующими. Однако как наши, так и существующие результаты довольно близко подходили к 100%, то смысл новой метрики был в том, что изменение эффективности с 90% до 92% — это не то же самое, что с 50% до 52%. В первом случае мы уменьшили оставшуюся неопределенность в 10% на 1/5 (т.е. на 20%), а во втором случае — на 1/25, т.е. на 4%. Соответственно, первый случай оказывается существенно более значимым достижением, чем второй случай.

обратить внимание читателя на идею графика и обязательно следует описать интригу, возможно, добавив в текст некоторую дискуссию.

Также следует выполнить тщательный дизайн графиков —– выбрать подходящим образом связываемые величины, обозначить их на осях, выбрать стиль визуализации, подобрать цвета, шрифты и прочее.

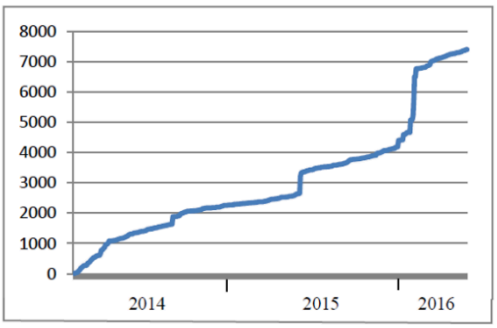


Рис. 25. Пример удачного графика[12]

Листинги. Перейдём теперь к листингам. Что это такое? Речь идёт о специально оформленном и вставленном в текст фрагменте исходного кода или псевдокоде. Сложность этого вида объектов, встроенных в текст, заключается в том, чтобы создать компактный, содержательный объект, который находится в гармонии с основным текстом. Это бывает непросто, потому что кода обычно много, алгоритм кажется довольно сложным и нужно постараться, чтобы создать для него компактный псевдокод. Также может оказаться весьма объемной созданная вами грамматика, какая-нибудь система правил, доказательство теоремы на языке Coq и прочее. Но слишком большой листинг оказывается нетекстом, претендуя на некоторую самостоятельную роль. Например, некоторые авторы считают, что код или формальная спецификация важны сами по себе и читателю будет интересно ознакомиться с ними непосредственно. Если это так, то я рекомендую выносить такие листинги в приложение к диплому, а если их объем оказывается через чур велик даже для приложения (например, по размерам такой листинг оказывается сопоставим с тестом всего диплома или превосходит его), то выложите ваш код/спецификацию в Интернет и укажите в тексте на него ссылку.

> **Листинг** — это специальный объект, встроенный в текст, который содержит фрагмент исходного кода вашей системы, псевдокод алгоритма или некоторую формальную спецификацию (грамматику созданного вами языка программирования, некоторую систему логических правил и так далее).

Пример листинга представлен ниже, на рисунке 26[13].

[12] Этот пример взят из статьи D.Koznov, O.Andreeva, U.Nikula, A.Maglyas, D.Muromtsev, I.Radchenko. A Survey of Open Government Data in Russian Federation. 8th International Conference on Knowledge 8th International Conference on Knowledge Management and Information Sharing (KMIS 2016), 9–11 November, Porto, Portugal Management and Information Sharing (KMIS 2016), 9–11 November, Porto, Portugal.

[13] Этот листинг взят из статьи R.Vasiliev, D. V. Koznov, G. A. Chernishev, A. Khvorov, D. V. Luciv, N. Povarov: TraceSim: a method for calculating stack trace similarity. MaLTeSQuE@ESEC/SIGSOFT FSE 2020: 25–30.

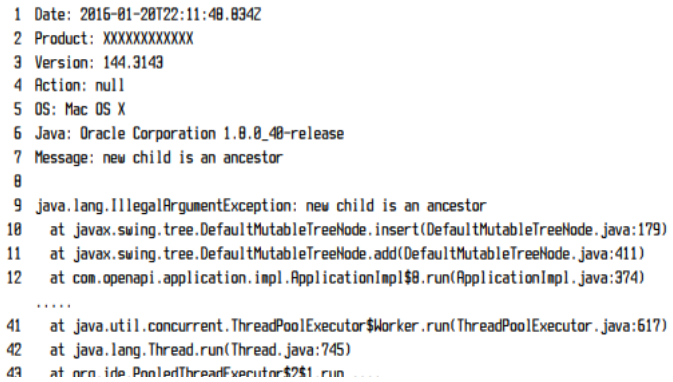


Рис. 26. Пример листинга — описание аварийного стека вызова

На этом листинге приведен пример состояние стека вызова Java-программы после того, как программа аварийно завершилась. Представлена вложенная динамическая цепочка Java-методов, которая составляет контекст программы на момент авариного завершения. Кроме того, в начале листинга приведена некоторая справочная информация: дата происшествия, номер продукта, версия и пр.

Итак, в листингах может содержаться очень разная информация. Рискуя повториться, напомню, что важнейшее правило при создании листингов заключается в том, что они не должны быть большими, при этом листинги исходного кода целесообразно создавать как иллюстрации определенных идей. Таким образом, вам следует ясно понимать, что же именно вы хотите сказать листингами, на что обратить внимание, то есть вам нужно понимать, как они «работают» в вашем тексте, и организовать эту «работу» нужным для вас способом.

Листинг, занимающий одну и более страниц недопустим в основном тексте. Несколько подряд идущих листингов по полстраницы и более также нежелательны. Можно заметить, что в подобных случаях писатели пытаются заменить текст нетекстом, не просто иллюстрируя, а адресуя читателя к коду за деталями и, таким образом, перенося в этот код нить повествования. Разумеется, что в таком сценарии листинг остаётся без текстовых комментариев. Так вот, всё это неправильно!

> **Важно.** Нет нужды прикладывать к диплому полные исходные коды вашей системы (даже в приложениях!). Если ваш код является свободно распространяемым, то можете указать ссылку в Интернет на соответствующий репозиторий. Также как нет нужды и детально описывать код в тексте вашей дипломной работы.

Правда, в некоторых университетах для защиты диплома требуют представить разработанный студентом исходный код. Однако это требование порой трудно выполнить, поскольку код этот, как правило, является собственностью компании, где студент работает (а почти все дипломники-программисты в России уже работают). Открыть такой код, даже частично — только для членов комиссии по защите дипломов — требует отдельных затрат, в том числе и административных.

Остановимся подробнее на псевдокоде. Что это такое?

> **Псевдокод** — это лаконичное (не более половины страницы) описание вашего алгоритма, выполненное на тщательно выверенном уровне абстрактности с помощью некоторого упрощённого алгоритмического языка (псевдоязыка).

Псевдокод, псевдоязык — в этих словах приставка «псевдо» означает, что и код, и язык в некотором смысле нереальны: то есть псевдокод невозможно исполнить на вычислителе, поскольку он может содержать кванторы (т.е. элементы исчисления предикатов) и производные от них операторы, а также слова на неформальном языке; псевдоязык не является настоящим языком программирования, поскольку предназначен для создания псевдокода.

Важным отличием псевдокода от листинга исходного кода заключается в том, что код в листинге почти всегда является фрагментом, а описание вашего алгоритма (псевдокод) должно быть замкнутым. Но для этого нужно как следует постараться.

Итак, зачем нужен псевдокод? В научных работах требуется, во-первых, выделить формальный алгоритм из инструмента или программной системы, в рамках которой он реализован, и именно последний предъявить в качестве научного результата. Во-вторых, описание этого алгоритма не должно быть слишком громоздким, то есть требуется опустить различные реализационные детали. Дело в том, что наука имеет дело с реальностью, которая хорошо поддаётся ёмким, компактным обобщениям, которые иногда называют законами. А деталей программной реализации обычно много, и научного интереса они не представляют: для научного сообщества важен лишь факт, что алгоритм реализован. Такой алгоритм нужен как формальный артефакт, описывающий ваш научный результат. Пример псевдокода для алгоритма поиска нечетких повторов в тексте по образцу представлен ниже[14] — смотрите рисунок 27.

```
/* D, p, k — Входные данные
/* R — Результат
1  W_1 ← ∅                          /* начало фазы 1 (сканирование)
2  for ∀ w_1 : w_1 ∈ D ∧ |w_1| = L_w
3      if d_di(w_1, p) ≤ k_di
4          add w_1 to W_1
5  W_2 ← ∅                          /* начало фазы 2 («усушка»)
6  for w ∈ W_1
7    w'_2 ← w
8    for l ∈ I
9      for ∀w_2 : w_2 ⊆ w ∧ |w_2| = l
10       if Compare(w_2, w'_2, p)
11           w'_2 ← w_2
12       add w'_2 to W_2
13 W_3 ← Unique(W_2)                /* начало фазы 3 (фильтрация)
14 for w_3 ∈ W_3
15   if ∃w'_3 ∈ W_3 : w ⊂ w'_3
16       remove w_3 from W_3
17 R ← W_3
```

Рис. 27. Пример листинга: псевдокод алгоритма поиска повторов по образцу

Вопрос о языке псевдокода является открытым. Мне неизвестно такого стандартного языка, и каждый автор создаёт свою версию. Важным здесь является следующее. Язык псевдокода должен быть в значительной степени декларативным — в нём можно использовать кванторы и логические связи, неформальный текст и логические операторы для условий циклов и прочее. При этом лучше не пользоваться процедурами, определенными в таком же псевдокоде — как правило, достаточно, что эти процедуры имеют «говорящие» имена, а также контекст, и, соответственно, по этим признакам понятно, какую функциональность они реализуют. Наконец, целесообразно стараться

[14] Этот алгоритм взят из работы Д.В. Луцив. Поиск неточных повторов в документации программного обеспечения (диссерт. на соискание уч. ст. к.ф.-м.н., СПбГУ, 2018).

убирать лишние императивные (реализационные) детали — счетчики цикла, вспомогательные переменные и пр.

Ваш первый псевдокод может быть сильно похож на реальный код, поэтому следует выполнить несколько итераций и добиться его компактности и выразительности. Также имеет смысл посмотреть разные варианты псевдокода в научных статьях — не обязательно, кстати, близких к вашей области!

Псевдокод обязательно нужно описывать в основном тексте диплома. Я рекомендую описывать каждую строку, некоторые строки можно объединять в группы и описать группу целиком. В этом описании вы будто проговаривайте исполнение вашего алгоритма вместе с читателем. Хочу предостеречь от распространенного заблуждения, что код есть лучшая документация. Если у вас имеется текст на естественному языке и в него вставлен фрагмент кода — нестоящего или псевдокода — этот код нужно соответствующим образом описать в тексте, «привязать» его к тексту. И тогда он начинает что-то пояснять, расширять, дополнять и прочее. А иначе он будет чужеродным артефактом в тексте вашего диплома.

Математические формулы. Многие дипломные работы по программированию содержат формулы. Например, работы по многопоточности, поиску гонок и пр. используют формальную семантику многопоточности (так называемые модели памяти). И в контексте этих работ частно выполняются формальные доказательства различных свойств представленных алгоритмов. Также много математики в работах, посвященных символьному исполнению, функциональному и логическому программированию и так далее. Необходимо отметить, что математические формулы часто переплетаются с листингами кода, и такая ситуация требует особенной аккуратности.

Я часто замечал, что если формулами специально не заниматься, то в тексте работы они очень быстро превращаются в нетекст.

- Формулировки теорем состоят исключительно из формул, а иногда и вовсе из одной единственной формулы, и эти формулировки не включают слов на русском языке, то есть не содержат предложений.
- Доказательства теорем оказываются формальными и очень объёмными, особенно при использовании популярной системы автоматического доказательства Coq. И это сплошной нетекст.
- Нет надлежащих текстовых описаний формул — авторы считают, что формулы понятны сами по себе и самодостаточны. Таким образом, текст начинает тяготеть к чистому набору формул.
- вынесенные формулы вплетаются в предложения по нескольку штук, так что одно предложение может занимать одну страницу или даже больше.
- Не соблюдаются знаки препинания при оформлении формул, что нарушает предложения русского языка.
- Формулы вставляются в перечисления (с булетами или с цифрами), что делает последние очень большими, громоздкими, а это, в свою очередь, перегружает структуру текста.

Вариант такого нетекста приведен ниже.

> К примеру, атом $P(x)$ в дизъюнкте
>
> $$\phi \wedge P(x) \rightarrow P(x')$$
>
> рекурсивен, а в дизъюнкте
>
> $$\phi \wedge P(x) \rightarrow \bot$$
>
> нерекурсивен.

При этом и цепочка вынесенных формул в одном предложении может быть существенно длиннее, равно как и каждая формула в отдельности — значительно больше.

Конкретно в этом случае я бы рекомендовал не выносить каждую формулу на отдельную строку, и тогда бы это предложение целиком уместилось в одну строку и было бы полностью корректным. А если формулы хочется иметь вынесенными, тогда нужно разбивать предложение на несколько (в данном случае я бы рекомендовал два отдельных предложения.).

Итак, как же оформлять формулы, вынесенные из текста? Ниже приведен простой пример.

Таким образом, тело любого дизъюнкта C, имеющее ограничение φ, можно записать следующим образом:

$$\varphi \ \wedge \ \bigwedge NRec\,(C) \ \bigwedge\bigwedge \ Rec\,(C)\,.$$

Важно, что перед вынесенной формулой идет двоеточие, и тогда в конце формулы ставится точка. Таким образом, вынесенная формула корректно встраивается в предложение и не нарушает его границ.

Более сложный пример, в котором после вынесенной формулы продолжается предложение, представлен ниже.

Произведение неинтерпретированных атомов $R_1(x_1), \ldots, R_m(x_m)$ определим следующим образом:

$$\prod_{i=1}^{m} R_{i(x_i)} = \ R\left(x_{j_1}, \ldots, x_{j_m}\right),$$

где $x_{j_1}, \ldots, x_{j_m}$ — такая перестановка чисел $1, \ldots, m$, что $R_{j_1} \leq \ldots \leq R_{j_m}$.

Можно заметить запятую в конце вынесенной формулы, потом с отдельной строки идёт слово «где», после которого продолжается предложение. Заметьте, что красная строка для придаточного предложения, начинающегося с «где», отсутствует!

Правила оформления вынесенных формул можно кратко сформулировать так.

1. Одно предложение может включать в себя не более одной вынесенной формулы.
2. Используйте всегда двоеточие перед формулой и надлежащим образом «закруглять» текст перед двоеточием. Но можно завершить предваряющее вынесенную формулу предложение и точкой — например, дать пояснения к представленной далее формуле, заканчивая их словами «как представлено ниже.»
3. После формулы ставить либо точку, либо запятую, последнюю в том случае, если за формулой продолжается предложение.

Хотелось бы заметить, что, например, в знаменитом трёхтомном труде по математическому анализу Георгия Михайловича Фихтенгольца, который я, во многом, принимаю за образец, правила оформления формул несколько отличаются — они чуть более сложные. Но важно, что правила там имеются, и поэтому текст выглядит аккуратно и представительно.

> **Важно.** Выносите объёмные доказательства, большие формальные спецификации (например, систему формальных правил) в приложения к диплому. Если же формальные выкладки всё равно оказываются очень объёмными, то можно выносить их ещё дальше — на внешние Интернет-ресурсы, указывая в работе соответствующую ссылку. Однако в основном тексте требуется объяснять основные идеи доказательства, сделать обзор.

Контрольные вопросы

1) Какова задача использования таблиц в тексте?
2) Опишите ситуации, когда таблицы могут быть полезными в тексте вашего диплома.
3) Что такое график?
4) Опишите требования к хорошему графику.
5) Что вы можете сказать о наглядности графика?
6) Что такое идея графика?
7) Может ли график быть безыдейным, и что это означает?
8) Что такое интрига в графике и почему она важна?
9) Что вы можете сказать об описании графика в тексте диплома — нужны ли нет такие описания, сколь тщательными и детальными они должны быть, если бывают хотя бы иногда нужны?
10) Что такое листинг?
11) Приведите пример того, когда листинг начинает играть самостоятельную роль в тексте. Объясните, почему это нежелательно.
12) Что следует предпринять, если хочется адресовать читателя к большому листингу, который не помещается в текст?
13) Нужно ли прикладывать к тексту диплома полные исходные коды созданной системы?
14) Что такое псевдокод?
15) О чём в данном случае символизирует приставка «псевдо»?
16) В чём отличие листингов кода и псевдокода?
17) Каковы требования к языку псевдокода, другими словами, что следует делать, а что нет, создавая псевдокод?
18) Расскажите, как следует описывать псевдокод в основном тексте вашего диплома?
19) Опишите признаки нетекста при оформлении математических формул.
20) Что такое вынесенная из текста формула?
21) Для чего, на ваш взгляд, выносят формулы из основного текста?
22) Опишите рекомендуемые правила по оформлению вынесенных формул.

Список литературы

1) T.M. Annesley. Bring Your Best to the Table, Clinical Chemistry October 2010, 56 (10):1528–1534.
2) Г.М. Фихтенгольц. Основы математического анализа. В 3-х томах. Издание 14-ое, стереотипное, 2022, М.: Изд-во Лань – Прогресс.

Лекция 10. О русском языке

Эта лекция посвящена некоторым особенностям русского языка, важным при написании текста диплома. Рассматривается, чем письменная речь отличается от устной с предостережением использования последней в текстах, рассказывается про плакатный стиль, научные тексты и техническую документацию. Даются рекомендации по стилистике текста, детально обсуждаются правила оформления перечислений. Рассматриваются типовые ошибки русского языка, встречающиеся в дипломах по программированию.

Про устную и письменную речь. Поговорим теперь о том, чем письменный язык отличается от разговорного, а также какова специфика плакатного стиля (последнее важно для составления презентаций). Не стоит всё это путать, ибо при подготовке и защите диплома вам понадобиться и нормально произнесённая речь на защите (которую, кстати, можно специально репетировать), и специфический язык для составления презентаций (я называл его *плакатным стилем*, поскольку он схож например, с языком плакатов, вывесок, объявлений, стенгазет), ну, и, собственно, письменная речь для написания текста дипломной работы.

Сначала остановимся на отличиях устной речи и письменной речи. Неформально. Устная речь — это тот язык, которым люди пользуются в непосредственном общении друг с другом, а письменная речь применяется при написании текстов. Сегодня, в связи с развитием электронных средств общения (социальных сетей, смс-сообщений, мессенджеров и прочего) грань между устной и письменной речью стирается: люди пишут также, как говорят. Однако, если речь идёт о целостных произведениях — постах, статьях, документах и так далее — то тут же возникает разница между письменной и разговорной речью.

Не претендуя на полноту обсуждения вопроса, тем не менее рассмотрим подробнее, чем отличается устная и письменная речь.

- Устный язык часто сопровождается эмоциональной подачей, жестами, мимикой, голосовым интонированием — и всё это служит дополнительными, так сказать, невербальными средствами передачи информации, а также способом расстановки акцентов, «склеивая» речь в связное и выразительное сообщение. В письменной речи все эти дополнительные средства отсутствуют, поэтому приходится полагаться только на слова, следовательно, текст должен быть связным и последовательным. Более того, он должен иметь структуру, что, кстати, бывает нелишним и в устной речи, например, когда вы произносите доклад или отвечаете на экзамене.
- Устная речь допускает вольности в структуре предложений — можно пропустить подлежащее, заменив его указующим жестом, можно сделать красноречивую пазу вместо сказуемого и т.д. Синтаксис письменной речи должен быть корректным. Более того, когда мы говорим, то не заботимся о запятых, тире и двоеточиях (всё это заменяется интонациями) — а когда пишем тексты, то заботимся и должны знать правила их использования.
- В устной речи может использоваться лексика из очень широкого диапазона, различные варианты жаргонов (например, программистский), собственные изобретённые слова и словоформы. В письменной речи используются только

признанные в языке слова, то есть то, что можно найти в словарях, а также словоформы, образуемые строго по правилами русского языка.

Рассмотрим письменную речь более пристально и заметим, что она бывает разной. Есть, например, язык, которым пишется художественная литература, и он очень богат, его сложно определить однозначно, можно лишь наметить границы. А есть язык публицистический, которым пользуются журналисты и пишут статьи в газетах и журналах. Своеобразен и малопонятен неспециалистам юридический язык (попробуйте заказать юридическое заключение по какому-нибудь интересующему вас вопросу!). Имеется также научный язык, используемый в монографиях, учебниках и статьях. Следует упомянуть про язык технической документации, который используется в различных инженерных дисциплинах, в том числе в области программирования.

Нас интересуют последние два варианта письменной речи — научный язык и язык технической документации, — поскольку диплом по программированию можно писать и на том, и на другом.

Про научный язык и научные тексты. Особым видом письменной речи является язык научных текстов. Перечислим ряд особенностей этого языка.

- Особая структура текстов. Как правило, в любом научном тексте присутствует введение с обзором и постановкой задачи и заключение с выводами, а также реферат (abstract) для краткого представления текста (и часто — отдельно от самого текста, например, в оглавлении конференции). Другие разделы тщательно выверяются по их содержанию и позиционированию в общем тексте. При этом принято отдельно выносить обзоры, отделяя таким образом, «чужое» от «своего», предлагаемого авторами.
- Научные тексты как правило, являются инкрементальными — они надстраивают фрагмент знания к уже имеющемуся зданию, поэтому для неспециалистов они часто бывают труднодоступны и неинформативны.
- В научных текстах тратятся большие усилия на позиционирование предлагаемых автором результатов, на обоснование используемого подхода, на оттаивание новизны результатов, на дискуссию и сопоставление. Все это оказывается рассыпанным по всему тексту и затрудняет восприятие материала. Читая научные тексты по разным специальностям, порой даже кажется, что авторов интересует не столько суть вопроса, сколько отстаивание своего мнения, позиционирование и всякая такая вот правильность. Кроме того, полифония связей предмета описания научного текста с разными контекстами в научных текстах редко бывает богатой — учёные боятся голословных утверждений, боятся оказаться неспециалистами и пр.
- Научные тексты пишутся специфическим языком, широко используя безличные предложения — «рассматривается проблема…», «известно, что…», «вычисление этих параметров организовано следующим образом…». Иногда предложения строятся от первого лица во множественном числе: «рассмотрим…», «возьмём…», «сделаем следующий вывод…». В этом случае автор как-бы вовлекает в повествование читателя (или слушателя лекции) — он вместе с ним «рассматривает», «берёт», «делает выводы».
- Научным текстам не свойственна эмоциональность, они представляют ровное, спокойное течение мысли.

Про техническую документацию. Здесь речь идёт о языке, которым пишутся различные документы, описывающие технические системы (сопроводительные документы по проекту, руководства пользователя и пр.), программное обеспечение, а также регламентные документы в крупных учреждениям.

- Если научные тексты неэмоциональны, то техническая документация и вовсе пишется сухим языком. Её цель — изложить функциональную информацию: как работает некоторый агрегат, как действовать в тех или иных обстоятельствах, что допустимо, а что нет, из каких классов и функций состоит данное программное обеспечение и так далее.
- Такие тексты имеют мелкозернистую структуру, то есть значительное количество небольших разделов и подразделов.
- В технических текстах много повторов — дословно одни и те же фрагменты имеются как в разных документах компании, так и в одном документе, поскольку одна и та же информация повторяется для удобства чтения, а также используются одни и те же обороты речи, сходная структура документов. Более того, организации даже тратят специальные усилия на стандартизацию своих текстов (документации).
- Большое количество повторов в технической документации также связано с тем, что часто она не предназначена для последовательного чтения — к тому или иному разделу обращаются в режиме справочника, чтобы найти ответ на конкретный вопрос.
- Сухой, описательный характер технической документации — в отличии от научных текстов здесь представлен минимальный объем обоснований или мотиваций тех или иных технических решений, не встречается дискуссий и прочего. Вместо всего этого каждый документ имеет ясно определенное назначение, фокус и контекст, а также соответствующее содержание.
- Стандартизованное и вычищенное оформление текста — отступы, подписи к рисункам и т.д.

Про язык вашего диплома. Сразу замечу, что тут имеются варианты, хотя в целом, в качестве базовых, нам интересны научный язык, а также язык технической документации.

Итак, ваш текст может быть написан публицистическим языком, быть живым, задорным и текучим. Но вы также можете написать формальный строгий текст. Вы также можете писать сугубо в стиле технической документации, занудно описывая различные аспекты вашей программной системы и лишь немного разбавляя сухость и узость технических текстов (главным образом, во введении). И, наверное, можно найти ещё варианты — всё это зависит от вас, от склада вашего характера, от взглядов, желания и трудолюбия, а также от многих других факторов (сочетания звёзд на небе, вашего знака зодиака и т.д.). Старайтесь найти свой стиль, но оставаясь в рамках предмета! Собственный стиль сделает вашу работу действительно вашей, поможет самовыразиться, а также позволит сделать процесс работы над текстом максимально интересным.

Важно избежать разговорного языка, как это обсуждалось выше. Сформулирую признаки разговорного языка, часто встречающиеся в текстах студентов.

- Излишняя эмоциональность, с помощью которой автор часто стремится сделать дополнительные акценты. Например, вместо следующего текста «другое крайне важное свойство…» можно просто написать «следующее свойство».
- Голословные и резкие утверждения, например, «обычно все в этом случае пишут так:» — и дальше следует фрагмент программы на некотором языке. Или утверждение про общеизвестность определённого далеко неочевидного свойства некоторого языка программирования. С одной стороны, такие утверждения могут являться свидетельством независимого мышления автора, а могут быть наивными и часто — не вполне верными. Тут нужна столь любимая в античности мера. Например, можно избегать чрезмерных обобщений (особенно если вам они свойственны), а там, где вы всё таки хотите обобщить, имеет смыл потрудится и сформулировать аргументы или поискать авторитеты — можно сослаться на блог известного специалиста, например, если вы что-то

утверждаете про язык C++, то таким специалистом может быть Бьерн Страуструп. Особенно много таких утверждений встречается во введении. Следует помнить, что современная научная письменная речь является плавной и спокойной.

- Использование пышных эпитетов или превосходных степеней прилагательных и наречий: «превосходный алгоритм», «были получены великолепные результаты», «наилучшим способом сделать это является ...». Многие подобные выражения уместны в научной публицистике или в устной научной речи (например, в выступлении оппонента на защите диссертации).

Старайтесь давать *определения* основным понятиям и явлениям, используемым в вашей работе. Это важно, поскольку в дипломах по программированию встречается большое количество понятий, неизвестных широкому кругу читателей. При этом отвечайте сначала на вопрос «что это такое?». Студенты часто не делают этого, сразу переходя к примерам. «Что это такое?» — «Это?... Ну, например…» Стремитесь определить феномен! Но, разумеется, не стоит впадать в излишний формализм, наводить тень на плетень и злоупотреблять формальными определениями. Следует понимать, что определить феномен вовсе не означает выдать кондовое, формальное определение. Определить можно неформально, но вполне точно.

> **Определения** важных понятий вашей работы составляют существенную часть хорошего текста. Тут важно понимать, что определения не обязательно должны быть формальными: гораздо важнее, чтобы они действительно определяли и делали для читателя понятными узловые, многозначные, сложные понятия вашей работы. Например, главу про архитектуру целесообразно начать с определения того, что вы в своей работе считаете архитектурой системы в виду многозначности этого термина.

Также полезной является практика писать тезисами: сначала вы нечто утверждаете или называете (кратко, ёмко), а после раскрываете, объясняете, описываете. Тут речь идёт именно о стиле написания, а не о каких-то отдельных значимых тезисах: и тезис, и его раскрытие могут уместиться в одном предложении, а могут растянуться на несколько. Тут важно не перепутывать разные мысли, не перескакивать с одной мысли на другую и так далее.

В качестве примера приведу следующее предложение: «Ответ на третий вопрос позволит оценить, как именно изменяется производительность в худшем случае на искусственных примерах». В этом предложении неформально объясняется суть исследовательского вопроса при постановке эксперимента. Так вот, существенной информацией (тезисом) является первая часть этого предложения: «Ответ на третий вопрос позволит оценить, как именно изменяется производительность системы». А вот то, что использоваться будут искусственные тесты, а также будет применена метрика «в худшем случае», — это уже раскрытие данного тезиса, точнее, реализационные детали, которые лучше убрать из этого предложения.

Но помните, что если ваша тема сложна и интересна, то вам, скорее всего, не удастся создать строго линейный текст, в котором все детали раскрываются строго последовательно, все новые понятия вводятся в одном единственном месте и т.д. Поэтому неизбежны повторы, забегания вперёд, упоминания понятий без надлежащих определений и т.д. Учитесь при этом не повторять по многу раз один и тот же текст дословно, но обсуждайте одно и то же по-разному, разными словами, с разных точек зрения, с разным уровнем деталей. Учитесь также легко касаться понятия — но так, чтобы читателю было максимально понятно!

Плакатный стиль.

> **Плакат** — это разновидность прикладной печатной графики, наборно-шрифтовое или художественно-иллюстративное листовое крупноформатное печатное тиражное издание, содержащее в наглядно-компактном виде информацию рекламного, агитационно-пропагандистского, инструктивно-методического, учебного и другого характера. Лист плаката содержит броское изображение и броский заголовок или призыв. (Большая российская энциклопедия).

Соответственно, стиль использования текста в плакатах будем называть **плакатным стилем**. В нашем случае элементами плакатного стиля являются заголовки, а также текст в презентациях. Давайте коснемся этого стиля, чтобы грамотно его использовать там, где нужно, и (что очень важно), не использовать его там, где не нужно.

Для плакатного стиля характерно следующее:

- использование назывных предложений,
- метафоричность и эмоциональная составляющая,
- упрощенное использование знаков пунктуации.

В лекции про заголовки мы уже обсуждали назывные предложения. Напомню ещё раз, что это такое.

> **Назывным** называется предложение с подлежащим без сказуемого, притом подлежащее выражено существительным в именительном падеже.

Вот примеры назывных предложений:

- «Введение в язык C++.»,
- «Обзор встроенных сред разработки.»,
- «О проблеме анализа указателей при символьном исполнении».

Подобные назывные предложения следует использовать в названии и заголовках, а также в названии слайдов и в пунктах перечислений вашей презентации. Последний вариант — использование назывных предложений в перечислениях в презентации — хорошо подходит в том случае, когда вы намерены объяснять каждый соответствующий пункт голосом.

Метафоричность и эмоциональная составляющая плакатного стиля означают, что вы отступаете от сдержанности и бесстрастности научного стиля. Это можно сделать в заголовке работы, например: «You're Recommending What?»[15], «Think Before Your LFSRs Jump»[16] — но я бы не рекомендовал такие заголовки диплома, поскольку подобные вольности в российских университетах не приняты. Можно также создать рисунок для текста диплома или презентации, вставив туда короткую просторечную фразу (но делать это следует дозировано — не следует превращать текст диплома и презентацию в комикс).

Знаки препинания в плакатном стиле используются, так сказать, в «облегчённом» режиме: не ставятся точки в конце заголовков, не ставятся знаки препинания (точки, точки с запятой, запятые) после элементов перечисления, не ставятся двоеточия после верхнего пункта при переходе к его подпунктам. Всё это делается для того, чтобы придать тексту большую *визуальную* выразительность. То есть предполагается дополнительный (плакатный) визуальный ряд, или некий визуальный минимум — например, заголовки выделяются с помощью дополнительных шрифтов и/или «жирного» стиля. Кроме того,

---

[15] N. Chonacky. You're Recommending What? Computing in Science & Engineering, 2007, 9(3): p. 2.

[16] M.A. Wahla, M. Mohsin, M. Afzal. Think Before Your LFSRs Jump. International Conference on Computer and Information Technology, 2010, vol. 1: 1070–1074.

текст плакатного стиля предполагает дальнейшие пояснения — или оральные, если он входит в вашу презентацию, или тестовые, если мы имеем в виду заголовки.

> **Важно.** Не используете плакатный стиль в обычном тексте (кроме заголовков и названия своей работы), в частности, не используете его в перечислениях.

О предложениях русского языка и перечислениях. Вспомним уроки русского языка в школе, и посмотрим строго и формально на текст — из чего он состоит? Ответ очевиден — из предложений. Вот как определяет предложение знаменитый русский лингвист, теоретик, систематизатор и популяризатор русского языка Д.Э. Розенталь.

> **Предложение** — это минимальная единица человеческой речи, представляющая собой грамматически организованное соединение слов (или слово), обладающее известной смысловой и интонационной законченностью. Будучи единицей общения, предложение вместе с тем является единицей формирования и выражения мысли, в чем находит свое проявление единство языка и мышления. (Д.Э.Розенталь. Словарь лингвистических терминов).

Предложение может быть сложным и включать в себя причастные и деепричастные обороты, а также быть сложносочиненным, то есть иметь придаточные предложения, соединяемые вместе союзами, знаками препинания. Но сколь бы сложным ни было предложение, оно начинается с заглавной буквы и заканчивается точкой. Как указывалось выше, в этом состоит отличие письменной речи, используемой в текстах, от плакатного стиля.

Однако, в студенческих текстах это простое правило часто нарушается — ниже представлено несколько примеров из реальных текстов.

> Что:
> Приложение X.
>
> Дополнительные условия:
> Поле игры – тор.
>
> Особенности:
> Приятный интерфейс, позволяющий задавать начальное состояние путем выбора .....
>
> База индукции. Тривиально.

Замечу, что многие студенты-программисты склонны писать текст как комментарии к программному коду, а студенты-математики порой пишут очень лаконично, будто снабжая математические формулы небольшими пояснениями. При этом ломаются основные структурные единицы текста — предложения русского языка: последние не содержат тех или иных главных частей предложения (подлежащего и/или сказуемого), отсутствуют связи разных частей сложного предложения (например, двоеточие, после которого следует перечисление). В итоге пишущий едва ли осознаёт, где у него кончается одно предложение и начинается другое, а это не является столь же загадочным явлением, как превращение вторника в среду у Льюиса Кэрола в «Алисе в стране чудес».

Нарушение структуры предложения особенно ярко проявляется в перечислениях, которые мы рассмотрим особо. Перечисление — это набор пунктов, каждый из которых начинается с новой строки, а весь список предваряется некоторым общим текстом. Такие перечисления важны для текста диплома, поскольку и задачи, и результаты оформляются

именно как перечисления. При оформлении таких перечислений, фактически, КАЖДЫЙ студент допускает ошибки.

Ниже представлен пример простого перечисления.

Структура имеет следующие поля:

- поле 1,
- поле 2,
- поле 3.

Заметьте, что перед перечислением стоит двоеточие, поскольку предложение русского языка продолжается! А раз так, то все элементы перечисления идут со строчной, не с заглавной, буквы. И в конце каждого пункта ни в коем случае нет точки — тогда бы предложение в этом месте и закончилось бы. Точка идёт в самом конце списка, после последнего пункта.

Рассмотрим более сложный вариант перечисления.

Структура имеет следующие поля:

- поле 1 – это имя;
- поле 2 является датой создания этой структуры;
- поле 3 – это комментарий.

Что тут усложнилось? Каждый из пунктов стал более объёмным и содержит теперь больше слов, а также дополнительные знаки препинания внутри (в данном случае это тире). Поэтому запятая в конце такого пункта уже не вполне уместна (хотя она была бы всё ещё корректна). И в этом случае мы в конце каждого пункта ставим точку с запятой.

Если мы хотим иметь в каждом пункте ещё больше текста, который состоял бы из нескольких предложений, то можно поступить следующим образом.

Структура имеет поля, перечисленные ниже.

- Поле 1 – это имя структуры. Имя должно состоять только из букв или цифр, спецсимволы не могут входить в имя.
- Поле 2 является датой. Дата представляется в формате ДД.ММ.ГОД. При этом ДД и ММ занимают по две ячейки, а ГОД – четыре ячейки.
- Поле 3 содержит комментарий. Он не может быть длиннее, чем 100 символов.

Посмотрим, что же изменилось в этом списке. Во-первых, исчезло двоеточие: поскольку у нас внутри пунктов получаются отдельные предложения, то и весь список не следует пытаться оформить как одно предложение. В связи с этим мы убираем двоеточие, заменяя его точкой. Но кроме этого мы ещё также надлежащим образом переделываем предложение, которое предваряет перечисление. Также все пункты перечисления идут с заглавной буквы, так как теперь в этом месте всегда начинается новое предложение. И каждый пункт заканчивается точкой, поскольку теперь в этом месте предложение заканчивается.

Ну а как же быть с нумерованными списками? А зачем они, кстати, вообще нужны? Хороший вопрос…. Если у вас нет на него ответа, то используйте обычные списки, как было показано выше. А нумерованные списки нужны, если в последующем тексте вы ссылаетесь на номера пунктов, например, так: «в первом случае..., во втором.., в третьем…». Ещё, конечно, многие полагают, что нумерация полезна сама по себе, чтобы подчеркнуть упорядоченность списка. Но мне кажется, что это является очередной попыткой сделать из текста нетекст, то есть программу — там то, понятное дело, порядок строк очень важен.

Итак, нумерованный список может быть оформлен следующим образом.

Структура имеет поля, перечисленные ниже.

1. Поле 1 – это имя структуры. Имя должно состоять только из букв или цифр, спецсимволы не могут входить в имя.
2. Поле 2 является датой. Дата представляется в формате ДД.ММ.ГОД. При этом ДД и ММ занимают по две ячейки, а ГОД – четыре ячейки.
3. Поле 3 содеожит комментарий. Он не может быть длиннее, чем 100 символов.

Что тут важно? Использование номера в виде цифры с точкой создаёт в нашем списке предложения вида «1.» и прочие номера. Из этого следует, что весь список не может быть одним предложением, а должен состоять из нескольких. А раз так, то двоеточием перед списком пользоваться нельзя, поскольку оно используется именно для того, чтобы соединить весь список в одно предложение. Если же есть желание использовать нумерацию в том случае, когда каждый пункт перечисления очевидно не образует самостоятельного предложения, то можно сделать так, как показано ниже.

Структура имеет следующие поля:

1) поле 1 – это имя;
2) поле 2 является датой создания этой структуры;
3) поле 3 – это комментарий.

Маркеры элементов списка «1)», «2)» и так далее не используют точек и поэтому могут быть интегрированы в одно предложение.

Заканчивая обсуждать вопрос с оформлением списков в тексте, отметим еще один момент. Например, часто в студенческих работах можно встретить следующий «текст»:

- Пункт 1
  - Подпункт 11
    - Подпункт 111
    - Подпункт 112
    - Подпункт 113
  - Подпункт 12
- Пункт2
  - Подпункт 21
  - Подпункт 22
- Пункт3
  - Подпункт 31
  - Подпункт 32

То есть пишущий строит сложную классификацию как иерархию, в виде большого списка, имеющего при этом глубокий (больше двух) уровень вложенности. Строго говоря, такие перечисления можно грамотно оформить, и они будут корректны с точки зрения русского языка. Но они превращают текст в нетекст, а, например, в описание сложной структуры данных в программе, в формальную онтологию и тому подобное. Ведь важно непросто корректно представить информацию в тексте вашего диплома, а представить её читаемым образом. А это, в свою очередь, означает соблюдение неписанных правил — ведь текст является, кроме формально правильных предложений русского языка, ещё и информацией, изложенной на русском языке, а не замаскированным представлением сложных структур данных. Важно, чтобы при написании текста вы представляли себе не формальные математические выкладки или фрагменты программ, но были бы проникнуты флюидами русского языка, его ритмами, плавностью изложения, связным течением мысли и речи.

Часто встречающиеся ошибки. В заключении я перечислю и прокомментирую ряд часто встречающихся ошибок в текстах дипломных работ по программированию.

- Иногда встречается обилие лишних, ненужных притяжательных местоимений, например, «файлы, хранящие *в себе* ...», «метод популярен из-за *его* высокой производительности», «механизм ленивой инициализации и описанные в данном разделе улучшения *для него*». Выделенные местоимения можно опустить без потери ясности.
- Использование англицизмов, например, *датасет, коммит, хеш, кэш, сетап, алиасинг, инстанциирование, сериализация/десериализация, фреймворк, тул, утилита, коммент, засабмитеть, пинганути, зарегать*. В этих и подобных случаях предлагается искать русские аналоги. Буду откровенен — это не всегда удаётся (например, я сам использую слово «фреймворк»), но стараться обязательно нужно. В частности, вместо слова «кэш» можно использовать слово «кэш-память»: определяющее русское слово оправдывает использование англицызма через дефис.
- Часто можно встретить злоупотребление англоязычными названиями программных систем, пакетов, технологий и библиотек без русских определяющих слов, даже при первом упоминании в тексте. Например, далеко не все читатели поймут, что такое «qla2xxx», а вот «драйвер qla2xxx» выглядит существенно читабельнее. Кроме того, если вам нужно склонять или спрягать английский термин в силу его определённого положения в предложении, то русское определяющее слово для этого тоже подходит — именно оно и будет склоняться/спрягаться. Например, можно встретить такие варианты «с помощью Spacer'а можно выполнить следующие задачи…». Вместо этого рекомендую следующий вариант: «с помощью *решателя* Spacer можно выполнить следующие задачи…». Слово «решатель» является примером русского определяющего слова.
- Не стоит начинать главу или раздел со ссылки на рисунок, например, со слов «На рисунке 1 представлен …». В начале главы/раздела нужен полноценный текст, поскольку глава/раздел вряд ли посвящены одному рисунку. Например, целесообразно в самом начале обозначить, о чём будет эта глава/раздел.
- Не следует в самом начале главы или раздела повторять заголовок, например, раздел «Среда разработки IntelliJ IDEA» не стоит начинать так: «Среда разработки IntelliJ IDEA предназначена для…». Следует перестроить первое предложение, например, так: «*Рассмотрим* среду разработки IntelliJ IDEA, *которая* предназначена для…».
- Многие слова по-русски не стоит использовать во множественном числе, подобно их аналогам в английском языке, например, «*поведения* программы», «*чтения* и *записи*», «*наследования*» и так далее, хотя прямой запрет на образование множественного числа для них отсутствует. Вместо этого лучше использовать дополнительное определяющее слово, которое будет удобно склонять, в том числе ставить в множественное число, например: «*сценарии* поведения», «*операции* чтения и записи», «*отношения* наследования».
- Часто можно встретить ситуацию, когда заголовок и следующий за ним подзаголовок идут без текста между ними. В подобных случаях весь ваш текст начинает походить на формальную спецификацию, например, XML-спецификацию.
- Часто приходится сталкиваться с использованием неподходящих предлогов, игнорирующих естественную семантику слов русского языка. Например, рассмотрим следующее словосочетание «вычисление символьных условий *в* обратном символьном исполнении»: здесь предлог «в» лучше заменить на предлог «*при*». В следующем примере «это вычисление выполняется *на* строке 10» предлог «на» заменяем на предлог «*в*», а в «*на* втором состоянии …» — предлог «на» заменяем предлогом «*во*». Чувствование этой естественной

семантики подменяется кальками с английского, а также ассоциациями, навеянными текстами программ и формальными алгоритмами.

- Можно заметить также злоупотребление словами «является», «реализует», «данный», «этот» и некоторыми другими — они могут встречаться по нескольку раз в одном предложении, а также в подряд идущих предложениях. Предлагаю искать синонимы и перестраивать такие предложения.
- А вот использование синонимов для называния основных объектов диплома сильно запутывает изложение. Например, нецелесообразно называть систему, которую вы разработали, ещё и программой, и программным комплексом, а то и вовсе алгоритмом (последнее, как ни странно, часто встречается). Пусть она всегда будет системой, а в противном случае вы будете запутывать читателя.
- Часто в русской письменной речи начинают пропускать предлоги на манер английского языка, где нормальным считается набор слов, идущих подряд, в котором главным членом является последнее слово, а все предшествующие являются его характеристиками. В качестве примера приведем следующую фразу: «в рамках специальных шаблонов оформления статей». Вместо этого концовка фразы может выглядеть так: «*по* оформлению статей». Или «цикл проходит ячейки списка» меняем на «цикл проходит *по* ячейкам списка», а «поиск уязвимостей программ» можно заменить на «поиск уязвимостей *в* программах». Подобные наборы слов могут быть существенно длиннее приведённых примеров — пользуйтесь предлогами и другими возможностями русского языка!

Контрольные вопросы

1) Каково назначение письменной речи?
2) Какие черты устной речи невозможно использовать в письменной речи?
3) Дайте определение предложению русского языка.
4) Что такое плакатный стиль?
5) Перечислите случаи, когда в тексте диплома можно применять элементы плакатного стиля.
6) Каковы особенности научного стиля?
7) Что вы можете сказать об особенностях написания технической документации?
8) Почему в ней много повторов (повторяющихся фрагментов текста)?
9) Почему маркер «1.» является отдельным предложением, а маркер «1)» — нет?
10) В каких случаях нужно использовать нумерованные списки, а когда — не нумерованные?
11) В чем разница в использовании запятой и точки с запятой в конце элементов списка?
12) Можно ли использовать в конце разных элементов списка разные знаки препинания – например, то точку, то точку с запятой?
13) На сколько целесообразно пользоваться в одном тексте большим количеством разных маркеров для элементов списков — например, один список оформлен с помощью булетов, второй — с помощью галочек, третий — с помощью тире и так далее?

Список литературы

1) А. Мильчин, Л. Чельцова. Справочник издателя и автора. Изд. 5-ое. 2018, М.: 2018. Изд-во студии Артемия Лебедева.
2) Д.Э. Розенталь. Справочник по русскому языку: орфография и пунктуация. АСТ, 2020.
3) Д.Э. Розенталь, М.А. Теленкова. Словарь лингвистических терминов. М.: Изд-во Просвещение. 1976.
4) В.В. Колесов. Русская речь. Вчера. Сегодня. Завтра. Изд-во Юна. 1998.
5) Большая российская энциклопедия в 35 томах. М.: Изд-во «Большая российская энциклопедия», 2004–2017 / Т. 26. 2014.

Лекция 11. А теперь пишем текст

Эта лекция посвящена обсуждению самого процесса написания текста. Приводится ряд советов по входным требованиям, по наладке процесса и по его исполнению. Также приводятся эталонные временные затраты на дипломную работу и написание текста.

Вдохновляюсь порывно
И берусь за перо!

Игорь Северянин. Ананасы в шампанском

Все, о чем я писал выше, можно более-менее успешно выполнить. Но этого может оказаться недостаточно для того, чтобы написать хороший текст диплома по программированию. Потому что важно не только спроектировать текст, не только собрать информацию для введения, глав основной части и заключения, не только создать и оформить встроенные в текст объекты — графики, формулы, рисунки и прочее. Помимо всего этого нужно, собственно, написать сам текст. И это большая работа, которая может оказаться для вас очень непростой, ибо вы берётесь за неё впервые.

Ниже я привожу ряд советов, которые могут вам помочь при непосредственном написании текста диплома. Эти советы не являются исчерпывающими, однако они родились из моего собственного опыта работы над большими текстами, а также из многолетних наблюдений за студентами. Поэтому если какие-то из этих советов покажутся вам полезными и дельными — смело и решительно используйте их в своей работе!

Выделить достаточно времени и внимания. Непосредственно на саму работу по написанию текста диплома нужно выделить достаточно времени. Я рекомендую выделить один месяц в том случае, когда эта работа является у вас основной. Или два-три месяца, если вы совмещаете эту работу с другой деятельностью (например, с работой в индустриальной компании). Что я имею в виду под основной работой? Например, когда вы каждый день тратите на текст по три-пять часов ежедневно. Большой текст — это большая работа, и она требует достаточно времени.

Для успешной работы над текстом диплома вам целесообразно освободиться от текущих дел и обязанностей, то есть вызволить себя из привычной рутины. Например, если вы работаете в компании, то я рекомендую на это время взять отпуск (именно на написание текста!). Может оказаться (и для многих это так и есть), что именно написание текста диплома оказывается самой затратной частью дипломного проекта. Посмотрите ещё, от чего вы можете на это время освободиться, где у вас могут быть резервы — времени, сил и внимания. Особенно внимания — про это люди часто забывают.

> **Важно.** Написание текста очень часто является *самой **непривычной и затратной частью вашего дипломного проекта***. Именно на эту деятельность целесообразно взять отпуск на работе, если вы уже работаете в индустриальной компании.

Тут следует понимать, что свободного внимания, которое он может вложить в разные области, дела, отношения с людьми и пр. у человека имеется ограниченное количество. Вы не можете одновременно делать слишком много дел, читать более определенного

количества книг и так далее. И причина этого заключается не столько в том, что вам не хватает времени. Тут не хватает вас — в виде вашего внимания: не объять, не погрузиться. И это является границами вашего внимания. Какое-то количество дел, работы и прочего вы можете удерживать и полноценно осуществлять, а свыше — уже не получается. Поэтому важно на старте иметь нужные ресурсы.

> **Важно.** На старте работы над текстом диплома иметь нужное количество ресурсов — времени, сил, внимания. Это важно, поскольку это ваш первый большой текст, т.е. подобную работу вы делаете впервые и поэтому она чревата непредвиденными сложностями.

Теперь я скажу несколько слов про сроки всех работ по диплому. Ниже я привожу идеальное расписание работы над дипломом из расчета того, что сама защита диплома происходит в конце мая — начале июня.

1. Тему дипломной работы следует выбрать к началу-середине сентября.
2. Весь программный код и/или теоретическую часть диплома целесообразно закончить к декабрю-январю. Ни в коем случае не откладывайте на самый конец эксперименты — помните, что эксперименты всегда опаздывают!
3. Январь-февраль посвятить чтению, экспериментам, возможно, исправлению ошибок и некоторым доработкам, а также проектированию управляющей структуры текста диплома.
4. Март–апрель — это время для написания текста диплома.
5. Май — время подготовки разных официальных документов, общение с рецензентом, подготовка презентации и репетиция речи.

Наладить процесс работы. Этот совет может показаться вам общими словами, однако процесс выполнения чего-либо, работы над чем-то — это вполне ощутимое явление. Так же как и его отсутствие. Когда процесс есть, вы чувствуете, что несмотря на трудности, вам есть на что опереться, и вы не начинаете работу каждый день как с чистого листа, и каждый рабочий сеанс связывается неразрывно с предыдущим, будто бы вы и не прерывались. У вас может появиться ощущение специфической наполненности этим действием — написанием текста, — вы будто всё время пишите. Вы привыкаете к этой работе как к явлению в вашей жизни, она оказывается естественной и постоянной. Вы целиком в этом, остальные активности, заботы отошли на задний план и не беспокоят вас. Вы полностью вовлечены в процесс несмотря на то, что у вас, безусловно, имеются в это время и другие дела. Но последние вы выполняете как бы между делом, не отвлекаясь на них вниманием. ваше внимание максимально полно сосредоточено на процессе написания текста. Вы можете полагаться на гармоничное, точное следование процессу и не беспокоиться о том, много или мало вы сделали сегодня. Если вы находитесь в рамках вами же созданного, запущенного процесса, то всё должно сойтись.

Наладка процесса обычно требует времени и сил, особенно, если то, что вы делаете, вы делаете впервые, как, например, создание текста диплома. Требуется потратить нужное количество времени, усилий — если их оказывается меньше некоторого порогового значения, то процесс так и не начнётся. Поскольку расчётных данных тут нет (всё сильно зависит от индивидуальных особенностей и ситуации), то тратьте сил и времени на текст диплома в начале работы особенно много. Сейчас важен не результат (сколько текста написал), а именно наладка процесса — именно на это должны быть нацелены ваш усилия в данный момент.

При наладке процесса важно также зарядить нужный темп работы, поскольку иначе вы можете инициировать процесс, который вас не приведёт к должной цели за имеющееся у вас время. На это нужно потратить необходимые усилия и посмотреть, например, в каких ритмах и сколь интенсивно вы в принципе можете работать — тут интересен диапазон и

сопутствующие обстоятельства. Например, вы знаете за собой, что можете включиться, собраться – но лишь в ночь перед экзаменом. А поступательно работать в течение долго времени вам непривычно и трудно. Но тут такое дело, которое за одну ночь не сделать! Значит, нужно трезво посмотреть на себя и найти способы для самоорганизации.

И вот вы установили процесс. У вашей работы имеется ритм: каждый день текст планомерно, непрерывно увеличивается, вы перестали дёргаться, беспокоиться и спокойно работаете. Хотя и в этот период вам может быть непросто.

Ни в коем случае не рвите процесс, то есть не отвлекайтесь — иначе придётся всё начинать сначала. Рвать процесс означает, например, сделать длинную паузу, и/или заняться каким-то другим делом, которое заберёт ваше внимание, отвлечёт вас, вовлечёт в иное: как я уже писал выше, свободного внимания у человека имеется фиксированное количество, и если в какое-то дело его прибыло, значит, из какого-то дела убыло. Например, вы внезапно вместо написания текста увлеклись компьютерной игрой и зависли на несколько дней – в начале казалось, что это безобидно, но вот, по прошествии нескольких дней вы видите, что контекст исчез и надо начинать все сначала.

> **Важно.** Не отвлекайтесь во время написания текста диплома — на компьютерные игры, развлечения, чрезмерное чтение новостей и блуждание по Интернету. Не прерывайте работу на долго, например, на несколько дней. Вернувшись, вы можете обнаружить, что нужно начинать сначала…

От целого к целому. Перед написание текста предыдущая фаза – создание управляющей структуры, в которую входит сбор и упорядочивание информации, а также проектирование непосредственно самого текста — должна быть полностью выполнена. Хуже нет, когда в процессе работы над текстом ты ещё параллельно создаёшь и его структуру, думаешь над содержанием, или доделываешь практическую/теоретическую часть работы (например, эксперименты, или хуже того — дописываешь код). Напротив, проектирование текста можно совмещать с выполнением практических работ по диплому. А процесс написания текста лучше с этим не совмещать. Таким образом приступая к написанию текста важно закончить все предыдущие работы по диплому, сесть — и на одном дыхании написать. Можно сказать, что данная книга как раз и посвящается тому, чтобы это было возможно сделать. В противном случае работа над текстом может превратиться в мучение — не забывайте, что это ваш первый большой текст!

> **Важно.** При написании текста у вас должна быть готова управляющая структура и собрана вся необходимая информация. Контрпродуктивно при работе над текстом сильно отвлекаться на сбор информации или создание/переделку оглавления и, хуже того, кроить текст «по живому», часто перенося те или иные абзацы из одной главы/раздела в другой и переделывая их при этом.

Работайте над текстом в идее «от целого к целому». На момент начала написания текста у вас имеется целостный проект — управляющая структура вашего будущего текста. И далее вы всё время создаёте целостный текст, не оставляя его надолго «разобранным». В частности, не увлекайтесь вставкой в текст комментария «todo», что означает, что автор намеривается вернуться в эту точку текста и доделать его определенным образом. Порой приходится видеть большие тексты, прямо испещрённые этим четырёхбуквием. И часть этих «todo» порой перекочёвывают в итоговый текст. Поэтому я рекомендую не фрагментировать работу над текстом, а наоборот, всё время интегрировать. Постоянно создавайте целостный текст, а не старайтесь собрать целое из многочисленных деталей. Сразу доделывайте то, что можно сейчас доделать. Например, если вы знаете, какую ссылку вы хотите вставить в это место текста — сразу вставляйте её, не стоит вместо этого помечать, что когда-нибудь сюда будет вставлена такая-то

ссылка. Работать таким образом гораздо приятнее, чем постоянно тонуть в отдельных мелочах и с ужасом думать о том, как много всего еще осталось делать.

Об обратной связи. Очень полезно давать прочитать ваш текст разным людям — научному руководителю, в первую очередь, и, возможно, ещё кем-нибудь. При этом требуйте, что люди читатели внимательно и выдали вам много замечаний. Часто бывает, что, например, люди из индустрии поверхностно читают тексты и остаются всем довольны. Не верьте им — так прочитанный текст вашей работы был прочитан зря! Но прося о замечаниях, предупредите читателей, что эти замечания должны быть локальными — ни о какой переделке структуры текста, задач и результатов, и уж тем более самой дипломной работы (например, изменении архитектуры системы или реализации дополнительных возможностей) речи быть не должно. Локальность замечаний важна, т.к. такие замечания вам будет по силам исправить. А глобальные замечания могут втянуть вас в непосильную и несвоевременную работу с различными неприятными сопутствующим обстоятельствами — бессонными ночами, недоделанной в срок работой, беспокойством, нервозностью и прочими «радостями». Отказывайтесь от исправления глобальных замечаний, даже если они и справедливы.

> **Важно.** Отдавайте себе отчёт в том, зачем вы просите научного руководителя, ваших товарищей прочитать ваш текст — для того, чтобы услышать, что вы молодец, или для получения замечаний, которые помогут вам улучшить ваш текст.

Но получив порцию локальных замечаний, обязательно реализуйте их! После того, как вам их выдали, очень важно занять правильную позицию, то есть не спорить с читателем, не пытаться ему что-то объяснить, не настаивать на своём в*и*дении (а студенты очень часто именно так реагируют на критику). Извлекайте из замечаний конструктивные зёрна — и тут же их реализуйте в своём тексте. Помните, не забывайте то, зачем вы отдали текст на прочтение. (Может быть, что вы отдали кому-либо прочитать свой текст с тем, чтобы получить одобрение — это бывает не лишним, но я сейчас говорю о другой цели — улучшить текст). Стремитесь из любого замечания, даже показавшегося вам на первый взгляд неважным, извлечь конструктивное зерно — такое стремление может поддержать вашу позицию по улучшению текста.

> **Замечания,** которые вы получили по тексту своего диплома, *обязательны к реализации* — все, которые являются локальными. Даже если вам кажется, что читатель вас не понял и советует что-то неадекватное, перепишите это место более понятным образом.
>
> **Важно.** Но не спешите реализовывать глобальные замечания, требующие переделки всего текста или, более того, всей дипломной работы. Даже если это кажется не очень сложным, вы можете ввязаться в большую работу, которая окажется неуместной в самом конце. Будьте мужественными, признавая, что в вашей работе могут остаться недостатки.

Пусть ваш текст настоится, как хорошее вино. Хорошо, когда у вас есть возможность дать тексту «отстоятся», то есть вы можете сделать небольшую паузу после того, как всё написали. Вернувшись к работе, вы локально что-то улучшаете, может быть, переписываете отдельные места целиком, или что-то добавляете. Но важно, чтобы этот перерыв был небольшим, и вы бы не успели выключиться из работы — иначе потом понадобиться снова включаться, а на это нужно время. В результате текст настаивается, набирает, как хорошее вино, приобретает обстоятельность, завершенность, вы его «доводите». Эта фаза является важной, и здесь можно получить специфическое

удовольствие, в том числе от того, что вы можете позволить себе тщательно, неспеша делать качественную работу. Это очень важный опыт, который впоследствии может вам помочь далеко не только при написании текстов.

Сделайте песню из работы над текстом диплома! Работайте над текстом интенсивно, включённо, продуктивно и радостно. Не нужно быть разобранным, несобранным, не следует постоянно отвлекаться. Будьте веселы и сосредоточены, полны энергии и увлечены. Не уставайте к концу дня, но наоборот, чувствуйте удовлетворение и подъём. Помните, что это ваш текст. И вы совершаете не тяжёлый труд, а радостную работу.

Контрольные вопросы

1) Что является вашими основными ресурсами – и вообще, и для написания диплома?
2) Что означает выделить достаточно времени и внимания на написание диплома?
3) Как вы поняли, что означает наладить процесс работы над текстом? Каковы признаки того, что вы наладили такой процесс?
4) Какие мотивации для получения обратной связи по вашему тексту являются, на ваш взгляд, конструктивными?
5) Какими ещё могут быть мотивации при передаче диплома на чтение третьи лицам?
6) Расскажите о локальных и глобальных замечаниях к тексту диплома. Как следует относиться к этим замечаниям в конце работы над текстом?
7) Что означает дать тексту «отстоятся»?
8) Что вы можете сказать о рекомендованном душевном настрое для работы над текстом? А каков ваш собственный душевный настрой?

Заключение

Итак, можно увидеть, что хороший текст рождается как сочетание многих и многих мелочей, всевозможных усилий и радений. Всё это объединяется, интегрируется в ясное, понятное и интересное повествование. В итоге читатель вашего диплома не задумывается над тем, над чем задумывались вы — писатель: над структурой оглавления, над тем, как материал распределен по главам, или о том, сколь удачно было выполнено введение в область исследования и так далее. Все эти вопросы, надлежащим образом решённые и вплетённые в общую канву, порождают у читателя естественный процесс восприятия изложенного вами материала. Естественность и лёгкость восприятия текста оказываются плодом значительной работы автора и надлежащего сочетания большого количества предпринятых им частных усилий. Целое рождается из многочисленных деталей, насыщаясь ими, и каждая из этих деталей важна и вносит в ваш текст свой тон, свою ноту.

И совсем уже в заключении хочу отметить, что я всегда считал, что высококвалифицированный программист — это непросто создатель хорошо работающего программного кода. А это — мыслитель, эксперт, который понимает и прозревает логику развития своей отрасли и всего программирования в целом, являясь специалистом, способным проектировать и создавать сложные программные решения в разных сферах человеческой деятельности. Он также способен участвовать в разных необычных, смелых, новых проектах, в проектах «на стыке», он способен иметь своё мнение и доносить его до разных людей. Ну и ещё он — образованный, культурный человек, с разнообразным кругом интересов, умеющий говорить, аргументировать, писать, находить продуктивные и красивые аналоги своим мыслям и идеям в литературе, науке… Именно из широкого масштаба могут рождаться новые, необычные, эффективные решения.

Список литературы

1) А.Ю. Андреев Российские университеты XVIII — первой половины XIX века в контексте университетской истории Европы. Издательство «Знак», 2009.
2) А.В. Ахутин. Понятие «природа» в античности и в Новое время. М. Наука. 1988.
3) Ф. Бэкон. Новый органон. Рипол Классик, 2018.
4) Большая российская энциклопедия в 35 томах. М.: Изд-во «Большая российская энциклопедия», 2004 – 2017/ Т. 26. 2014.
5) В.И. Вернадский. Избранные труды по истории науки. М.: Наука, 1988.
6) Н. Винер. Кибернетика, или управление и связь в животном и машине/ Пер. с англ. М.: Советское радио, 1968.
7) Б.Л. Ван Дер Варден. Пробуждающаяся наука I. Математика древнего Египта, Вавилона и Греции / Перевод с англ. М.: ГИФМЛ, 1959.; Пробуждающаяся наука II. Рождение астрономии / Перевод с англ. М.: ГИФМЛ, 1991.
8) К.И. Вигерс. Разработка требований к программному обеспечению/ Пер. с англ. Русская редакция, 2004.
9) Т.А. Гаврилова, Д. В. Кудрявцев, Д. И. Муромцев. Инженерия знаний. Модели и методы. Санкт-Петербург, 2016.
10) К.В. Гумбольдт. О внутренней и внешней организации высших научных заведений в Берлине/ Пер. с немец. в журн. «Неприкосновенный запас», N 2, 2002.
11) Т. Бьюзен. Супермышление/ Пер. с англ. Мн.: ООО «Попурри», 2003.
12) Д. Гибсон. Экологический подход к зрительному восприятию. Прогресс, 1988.
13) Л.Н. Гумилёв. Этногенез и биосфера земли. Гидрометеоиздат. 1990.
14) Дж.К. Джонс. Инженерное и художественное конструирование/ Пер. с англ. М.: Мир, 1976.
15) Дж.К. Джонс. Методы проектирования/ Пер. с англ. М.: Мир, 1983.
16) Д.А. Марка, К. МакГоуэн. Методология структурного анализа и проектирования SADT /Пер. с англ. М.: Мета Технология, 1993.
17) Д.В. Кознов. Основы визуального моделирования. БИНОМ, 2008.
18) В.В. Колесов. Русская речь. Вчера. Сегодня. Завтра. Изд-во Юна. 1998.
19) М. Кристофер, Ш. Хайнрих, Р. Прабхакар. Введение в информационный поиск/ Пер. с англ. Изд-во Вильямс. 2020.
20) А.Мильчин, Л.Чельцова. Справочник издателя и автора. Изд. 5-ое. 2018. М.: Изд-во студии Артемия Лебедева.
21) В.В. Налимов. Спонтанность сознания. Водолей Publisher, Томск-Москва. 2007.
22) И. Ньютон. Математические начала натуральной философии (Philosophiae Naturalis Principia)/ Перевод с лат. А. Н. Крылова. Собр. трудов. М. – Д., т. 7, 1936.
23) С.И. Ожёгов. Словарь русского языка. М. 1989.
24) Очерки истории информатики в России. Н.: Научно-исследовательский центр ОИГГМ СО РАН, 1998.
25) Д.Э. Розенталь. Справочник по русскому языку: орфография и пунктуация. АСТ, 2020.
26) Д.Э. Розенталь. М. А. Теленкова. Словарь лингвистических терминов. М.: Изд-во Просвещение. 1976.
27) Н.В. Ростиславлева. Теория и практика раннего либерализма в Германии (первая половина XIX в.). Диссертация на соиск. уч. степ. докт. истор. н-к. М., 2011.
28) Рекомендации по преподаванию информатики в университетах. Пер. с англ. Изд-во СПбГУ, 2002.

29) М. Фаулер, К. Скотт. UML в кратком изложении: Применение стандартного языка объектного моделирования/ Пер. с англ. М.: Мир, 1999.
30) Г.М. Фихтенгольц. Основы математического анализа. В 3-х томах. М.: 2022. Издание 14-ое, стереотипное. Изд-во Лань – Прогресс.
31) Х. Шенк Теория инженерного эксперимента/ Пер. с англ. Москва. 1972.
32) А.М. Цвелик. Жизнь в невозможном мире. Издательство Ивана Лимбаха. 2012.

33) S. Alam, S. Zardari, M. Bano. Software engineering and 12 prominent sub‐areas: Comprehensive bibliometric assessment on 13 years (2007–2019). IET Software, 2022, 16 (2): 125-145.
34) T.M. Annesley. The Title Says It All. Clinical Chemistry, March 2010, 56 (3): 357–360.
35) T.M. Annesley. “It was a cold and rainy night”: Set the Scene with a Good Introduction. Clinical Chemistry, May 2010, 56 (5): 708–713.
36) T.M. Annesley. Bring Your Best to the Table, Clinical Chemistry October 2010, 56 (10):1528–1534.
37) Architecture Analysis and Design Language (AADL). AEROSPACE STANDARD AS5506C, 2016-03.
38) J. Pontus, M. Ekstedt, and I. Jacobson. Where's the theory for software engineering? IEEE Software, 2012, 29 (5): 96-96.
39) Software Considerations in Airborne Systems and Equipment Certification. DO-178C. Federal Aviation Administration. 2012.
40) R. Van Solingen, V. Basili, G. Caldiera, & H. D. Rombach. Goal question metric (gqm) approach //Encyclopedia of software engineering. John Wiley & Sons 2002.