

Объектно-ориентированное программирование ООТПиСП СТПМС **Пацей Наталья Владимировна**

кафедра Программной инженерии

n.patsei@belstu.by



327-1

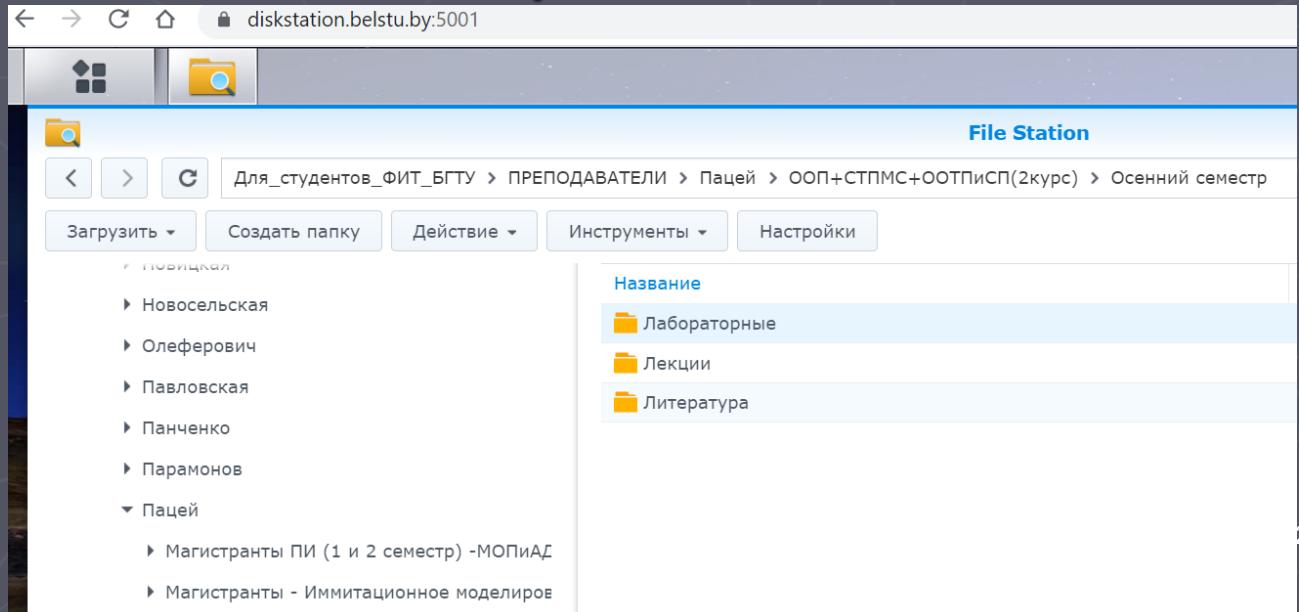
- ▶ 1 семестр - 1 л. + 1 лек – > экз
- ▶ 2 семестр – 1 л. + 1 лек – > экз + курс.

Необходимый материал

- 1) Лекции
- 2) Книги
- 3) Материалы

<https://diskstation.belstu.by:5001/>

Student
fitfit



Как будут выставляться оценки

40%

Оценка за
лаборатор.



Оценка за
аттестации

10%

Оценка за тесты
на лекции (теория)

50%

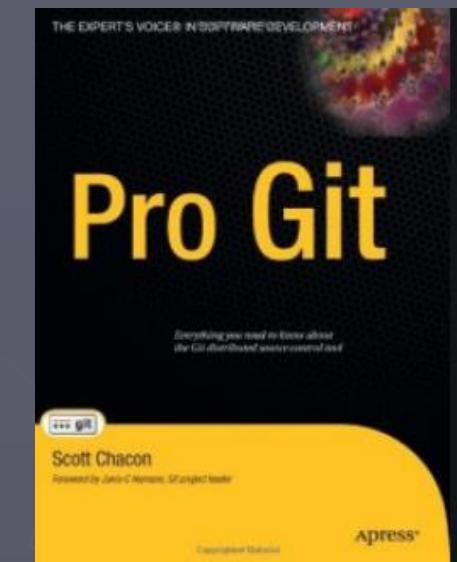
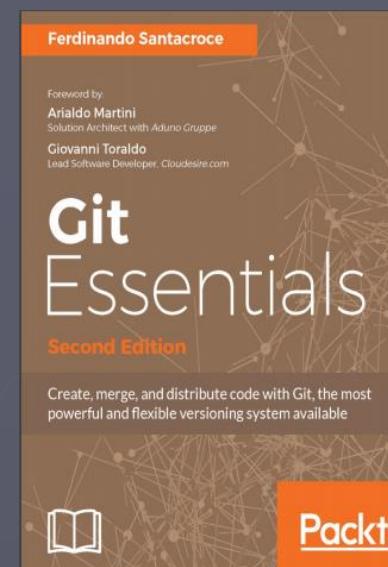
Экзамен

Системы контроля версий

назначение, типы, Git



Система управления версиями (от англ. **VersionControl System**, VCS или **Revision Control System**) — программное обеспечение для облегчения работы с изменяющейся информацией



<https://git-scm.com/book/ru/v2>

► <https://elearn.epam.com>

The screenshot shows a search results page for courses on the eLearn.EPAM platform. The search query is "git".

SHOW ALL 5 VENDORS

RATING
★ ★ ★ ★ ★ and above

CLEAR

We have found 50 items

Vendor	Title	Level	Rating	Language	Students
DesignSpot School	Version Control with Git	Intermediate level	4.7	ENG	869
DesignSpot School	Introduction in Testing	Basic level	4.6	RUS	10
DesignSpot School	Basic level	Basic level	4.6	ENG	5
DesignSpot School	Basic level	Basic level	4.6	ENG	5

5. Внесение изменений

СПОНСОР:



Цели

- Научиться отслеживать состояние рабочего каталога

1. Подготовка
2. Финальные приготовления
3. Создание проекта
4. Проверка состояния
5. Внесение изменений
6. Индексация изменений
7. Индексация и коммит
8. Коммит изменений
9. Изменения, а не файлы
10. История
11. Алиасы
12. Получение старых версий
13. Создание тегов версий
14. Отмена локальных изменений (до индексации)
15. Отмена проиндексированных изменений (перед коммитом)
16. Отмена коммитов
17. Удаление коммитов из ветки
18. Удаление тега oops
19. Внесение изменений в коммиты
20. Перемещение файлов
21. Подробнее о структуре
22. Git внутри: Каталог .git
23. Git внутри: Работа непосредственно с объектами git
24. Создание ветки
25. Навигация по веткам
26. Изменения в ветке master
27. Просмотр отличающихся веток
28. Слияние
29. Создание конфликта
30. Разрешение конфликтов
31. Перебазирование как альтернатива слиянию
32. Сброс ветки style
33. Сброс ветки master
34. Перебазирование
35. Слияние в ветку master
36. Несколько репозиториев
37. Клонирование репозиториев
38. Просмотр клонированного репозитория
39. Что такое origin?
40. Удаленные ветки
41. Изменение оригинального репозитория
42. Извлечение изменений
43. Слияние извлеченных изменений
44. Извлечение и слияние изменений
45. Добавление ветки наблюдения
46. Чистые репозитории
47. Добавление удаленного репозитория
48. Отправка изменений
49. Извлечение общих изменений
50. Размещение ваших git репозиториев
51. Расширение репозиториев

Спасибо!

Измените страницу «Hello, World»

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла на:

ФАЙЛ: hello.html

```
<h1>Hello, World!</h1>
```

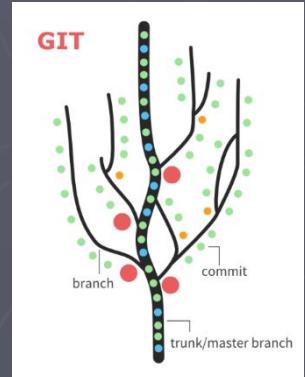
Проверьте состояние

Теперь проверьте состояние рабочего каталога.

<https://githowto.com/ru/setup>

Зачем ?

- ▶ Бэкап - Синхронизация кодовой базы
- ▶ Кто, когда и зачем сделал изменения
- ▶ Совместная работа
- ▶ Bug трекинговая система



1 проект
1 файл

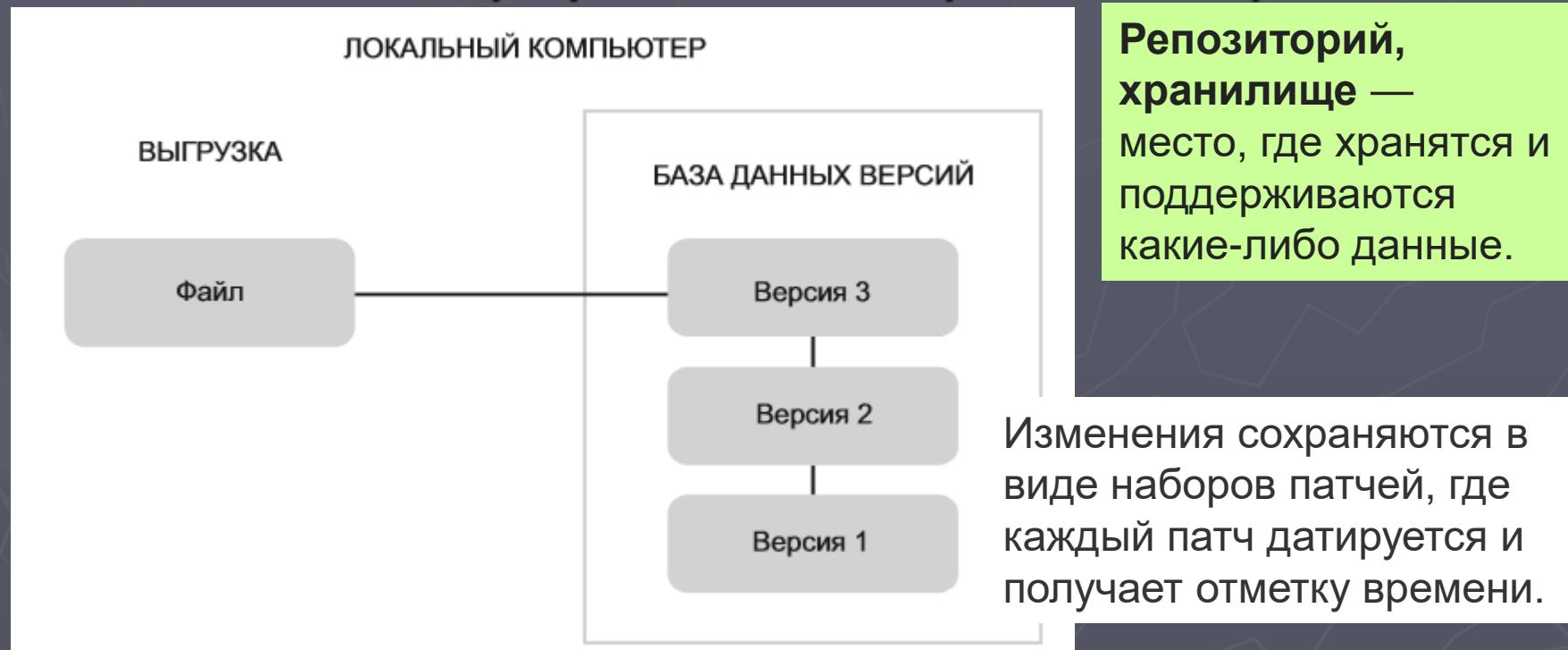


Разновидности

- ▶ Version Control System, VCS
- ▶ локальные системы контроля версий (Revision Control System, RCS)
- ▶ централизованные системы контроля версий (Centralized Version Control System, CVCS)
- ▶ распределенные системы контроля версий (Distributed Version Control System, DVCS)

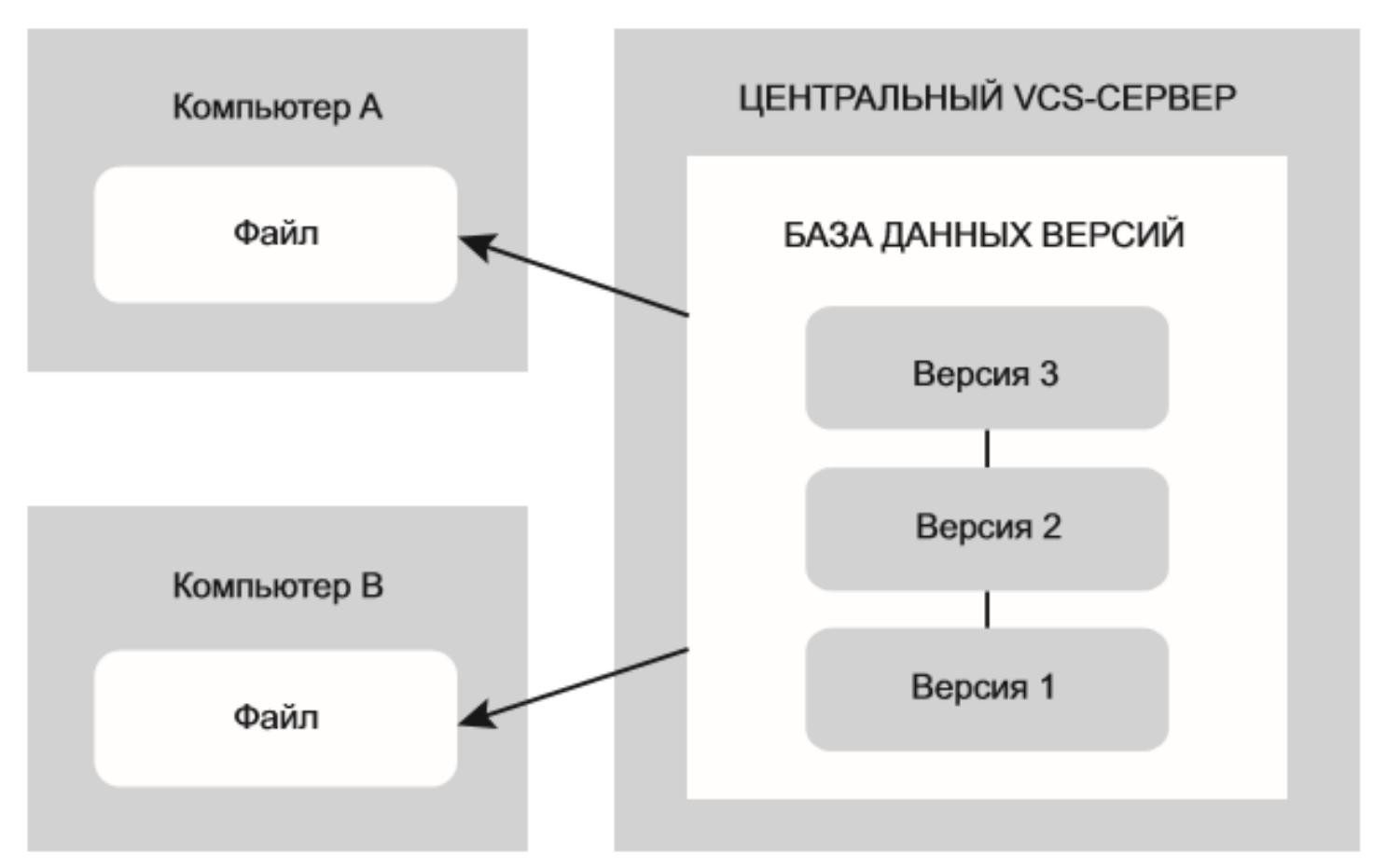
Локальные системы контроля версий

записывает историю изменения файла или набора файлов, чтобы в будущем была возможность вернуться к конкретной версии



Revision Control System, RCS

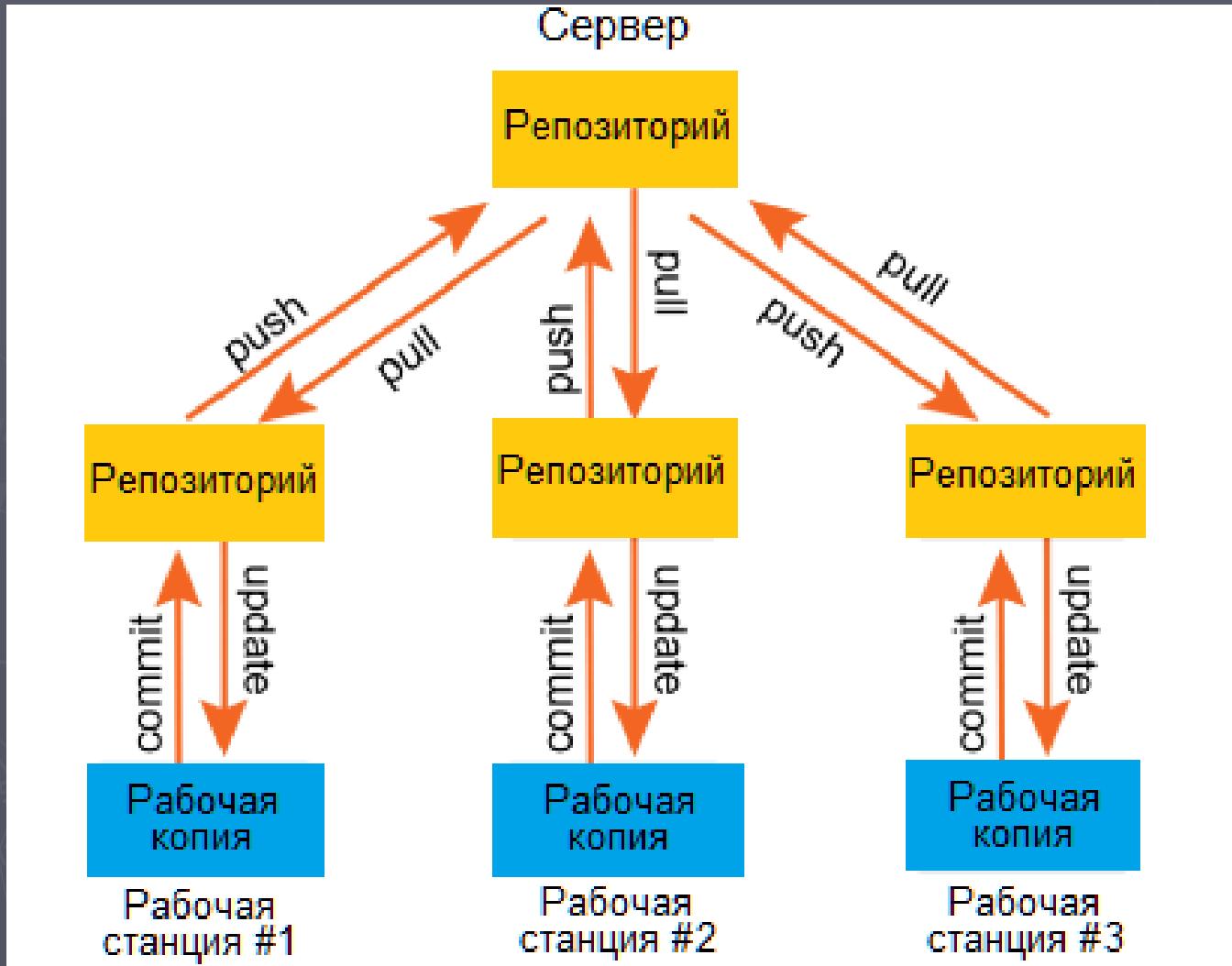
Централизованные системы контроля версий



► Недостаток:

- Единая точка отказа

Распределенные системы контроля версий



Сравнение решений

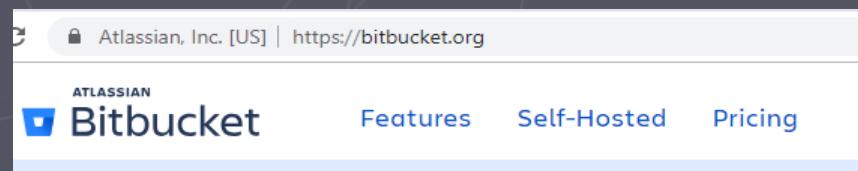
- ▶ Система одновременных версий (CVS) (80e)



- ▶ SVN (Subversion)



- ▶ Git
- ▶ Mercurial
- ▶ BITBUCKET



Git

Распределённая
система контроля
версий (SC, RC.....)

► В 2005 г.
BitKeeper

- Основные свойства :
- Быстродействие и размер
 - Безопасность и целостность (хэш SHA)
 - Достоверность
 - Гибкость (нелинейные рабочие процессы – слияние, ветвление)
 - Производительность (простое ветвление)
 - Функциональность

The screenshot shows a web browser displaying the URL <https://git-scm.com/about/branching-and-merging>. The page has a light green header with the Git logo and the tagline "git --distributed-even-if-your-workflow-isn't". A search bar is on the right. The main content area has a white background. On the left, there's a sidebar with "About" and "Documentation" sections. The "About" section lists: Branching and Merging, Small and Fast, Distributed, Data Assurance, Staging Area, Free and Open Source, and Trademark. Below this are "Downloads" and "Community" links. To the right, a large box is titled "About" and features a red circle with "Branching and Merging" and several grey circles labeled "Small and Fast", "Distributed", "Data Assurance", "Staging Area", "Free and Open Source", and "Trademark". Below this box is a section titled "Branching and Merging" with text about its unique branching model.

GitHub

- ▶ сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git



- +Обучение (глобальный поиск)
- +Реклама (резюме)
- +контроль доступа
- + багтрекинг
- + управление задачами
- + вики для каждого проекта

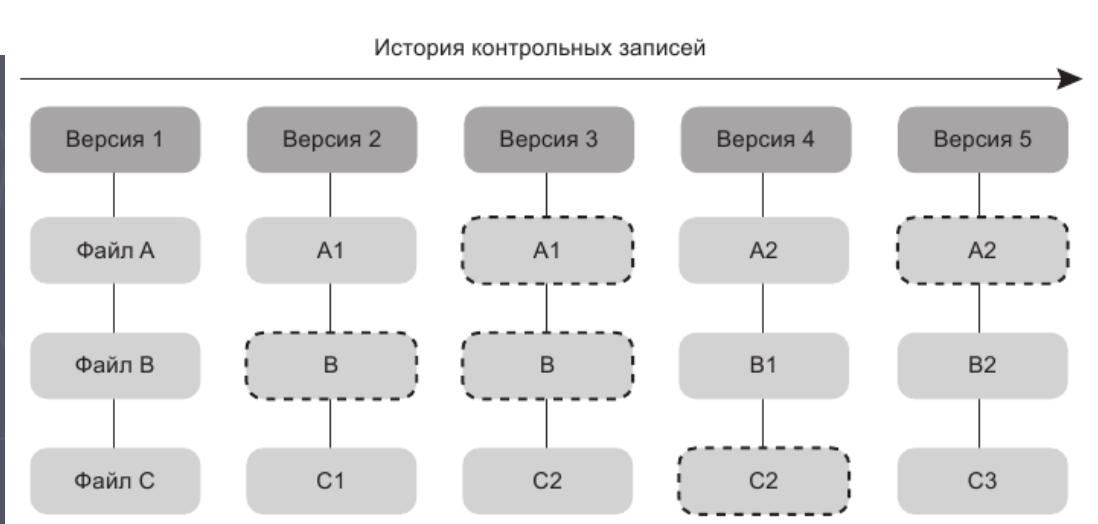
Git — это инструмент, позволяющий реализовать распределённую систему контроля версий, а GitHub — это сервис для проектов, использующих Git.

Git

1) Хранит снимки состояний, а не изменений



набор снимков состояния
миниатюрной файловой системы -
stream of snapshots



2) Локальность операций



3) Целостность Git (вычисление контрольных сумм – хеш SHA-1)

Хеш - строка из 40 символов, включающая в себя числа в шестнадцатеричной системе (0–9 и a–f) и вычисляемая на основе содержимого файла или структуры папки в Git.

24b9da6552252987aa493b52f8696cd6d3b00373

```
nrats@DESKTOP-R4P17F3 MINGW64 /c/Natallia/JavaProject/2019IBA/IBAJava_1 (master)
$ git log
commit 9617b068a940250e876070ae58124249f1b8ff25 (HEAD -> master)
Author: pnvttest <n.patsei@belstu.by>
Date:   Mon Feb 11 22:13:04 2019 +0300

    secaond

commit 39f9d1731d5688112f598339b63cbd0c720c0841
Author: pnvttest <n.patsei@belstu.by>
Date:   Mon Feb 11 22:11:49 2019 +0300

    second

commit d4287ab4ff0ce9d4a4483603adee5c279d828d98
Author: pnvttest <n.patsei@belstu.by>
Date:   Mon Feb 11 19:57:25 2019 +0300

    first commit

nrats@DESKTOP-R4P17F3 MINGW64 /c/Natallia/JavaProject/2019IBA/IBAJava_1 (master)
$ |
```

4) Три состояния

выполняется выгрузка одной из версий проекта

файл хранящий информацию о том, что именно войдет в следующую операцию фиксации

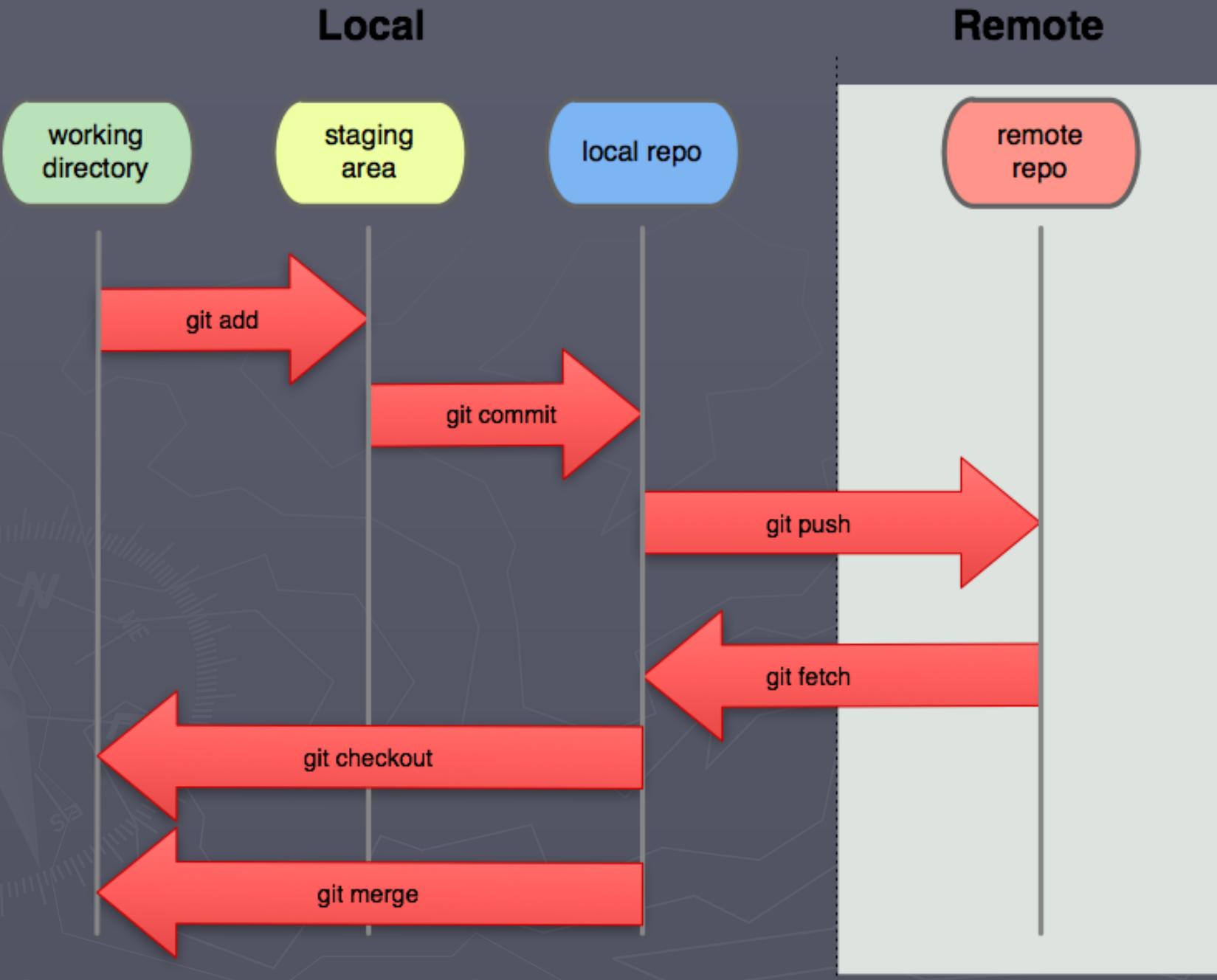
метаданные и объектная база данных проекта



Модифицированное
(modified)

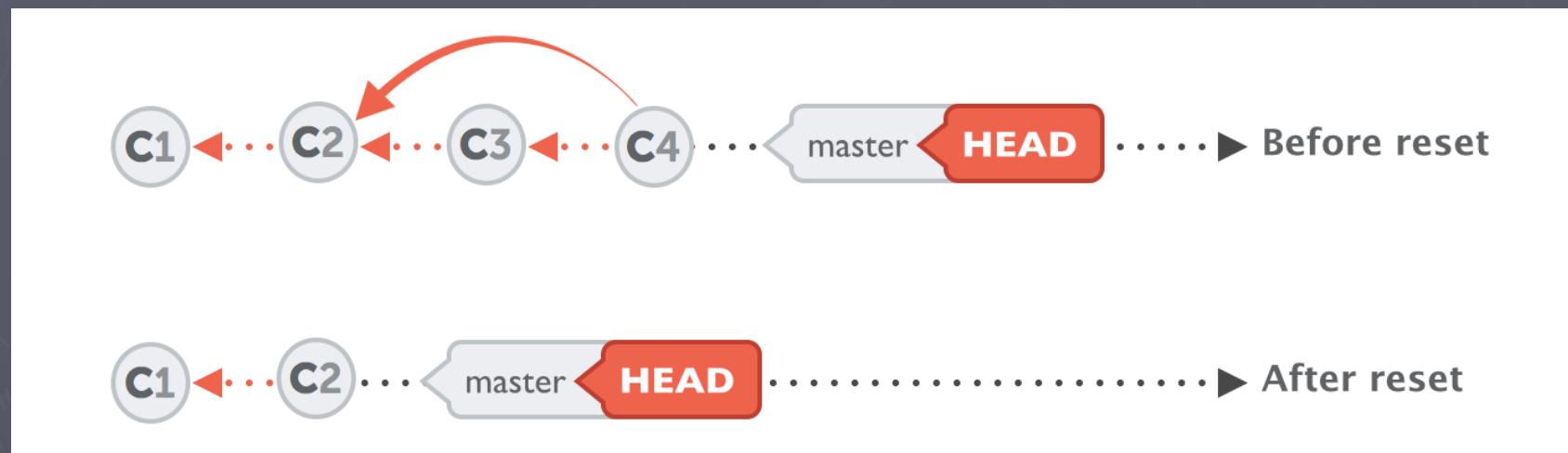
Индексированное
(staged)

Зафиксированное
(committed)

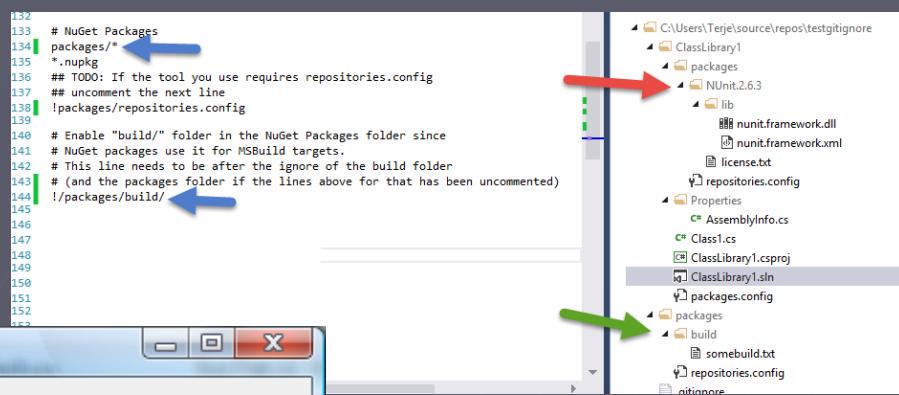
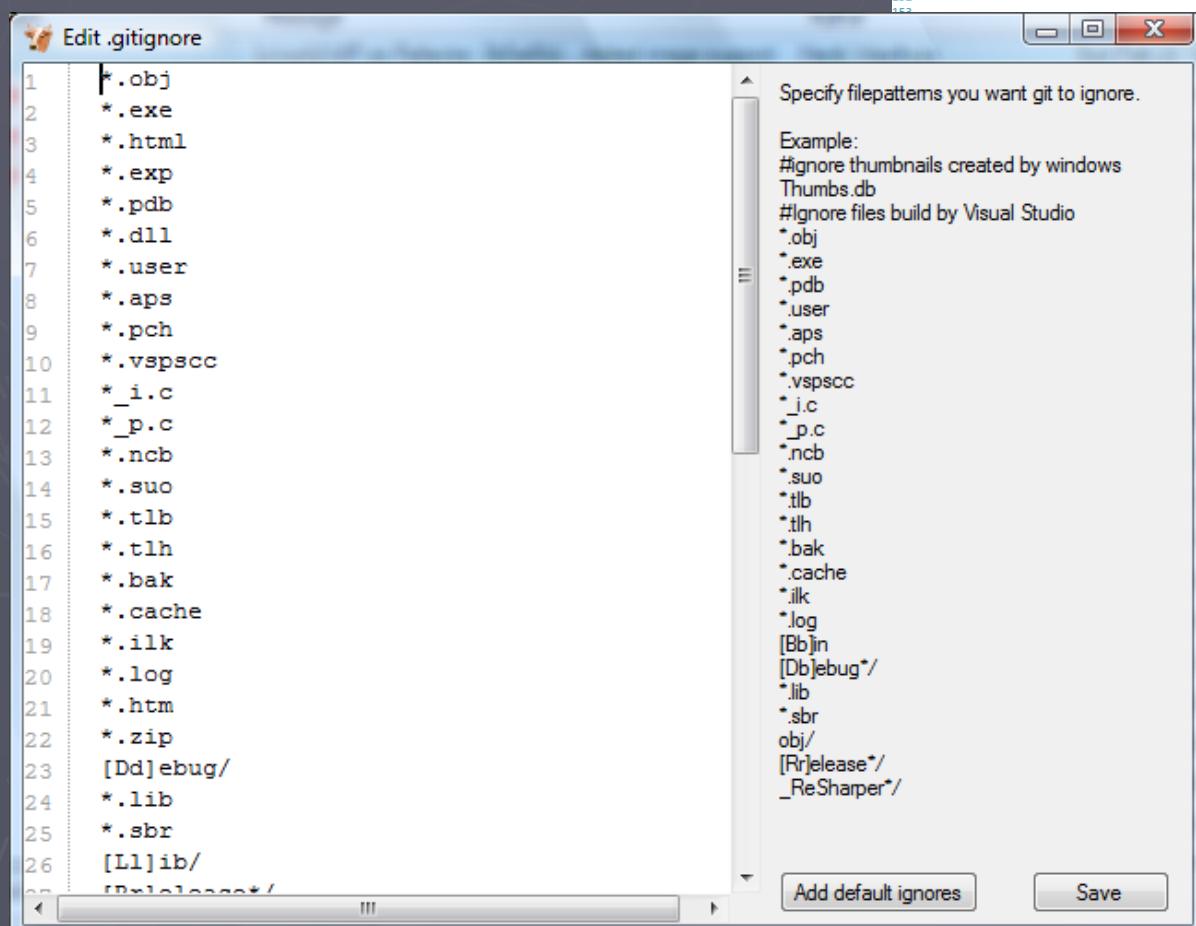


commit

reset

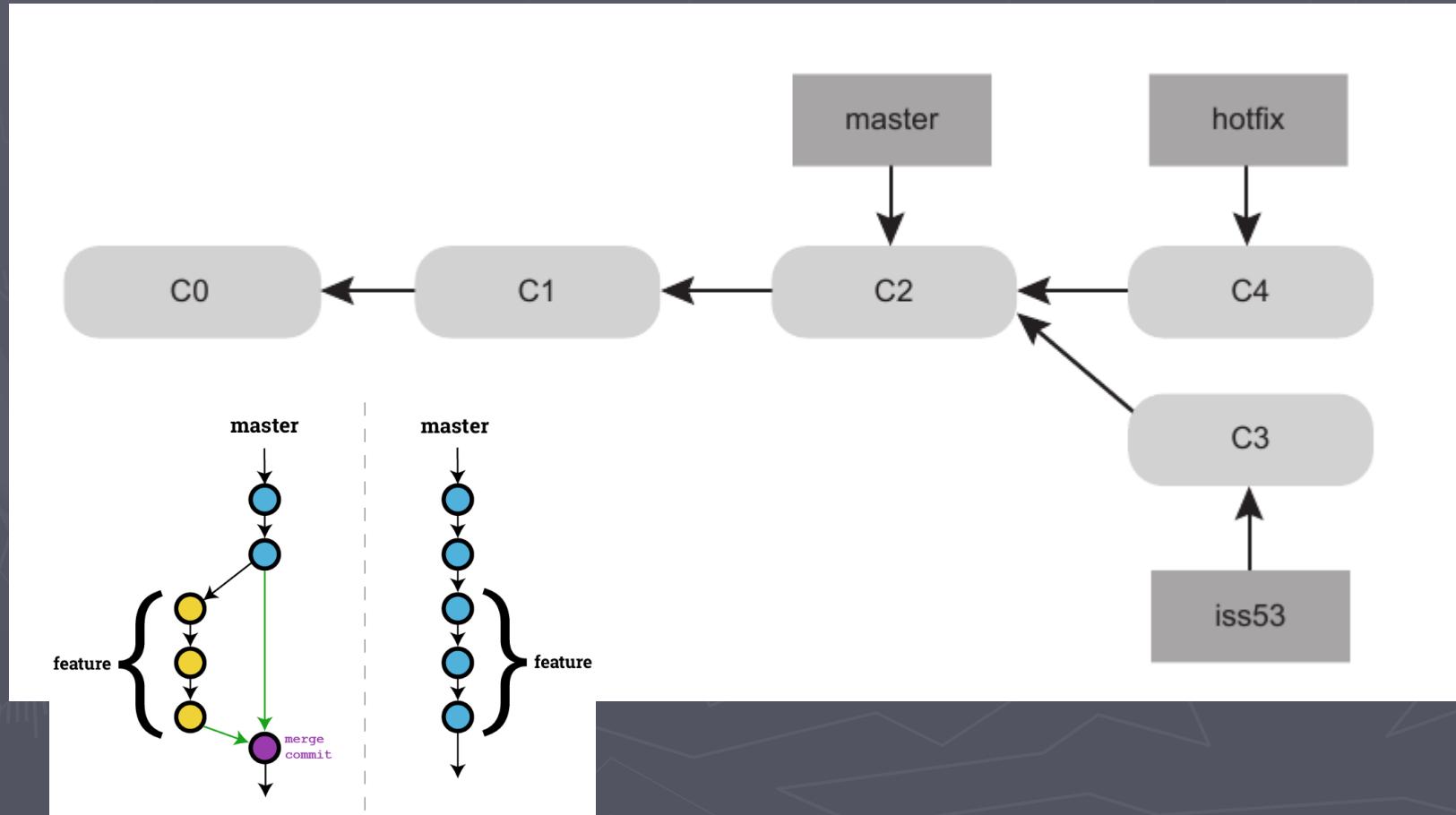


► Игнорирование файлов gitignore

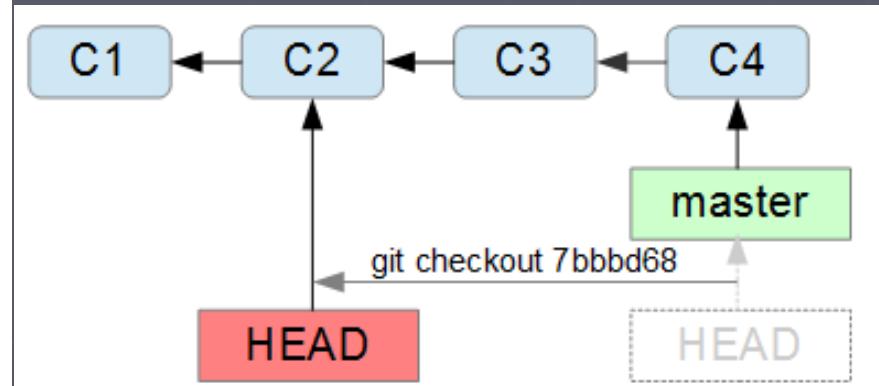
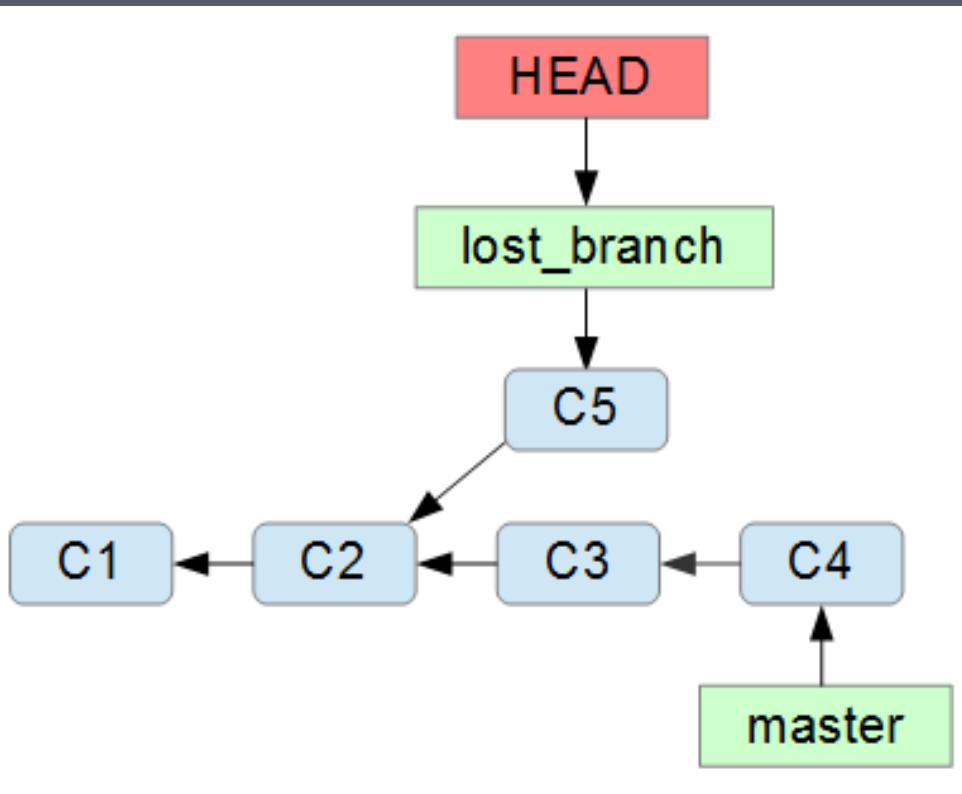


5) Ветвления

ветвление (branching) означает отклонение от основной линии разработки, после которого работа перестает затрагивать основную линию

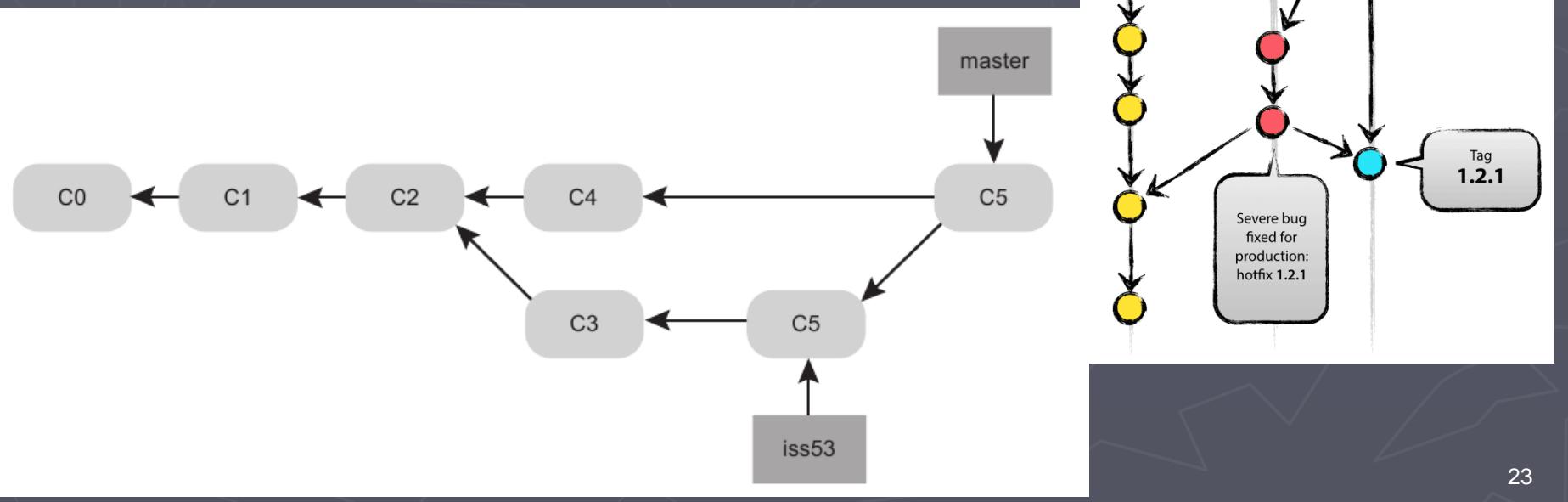
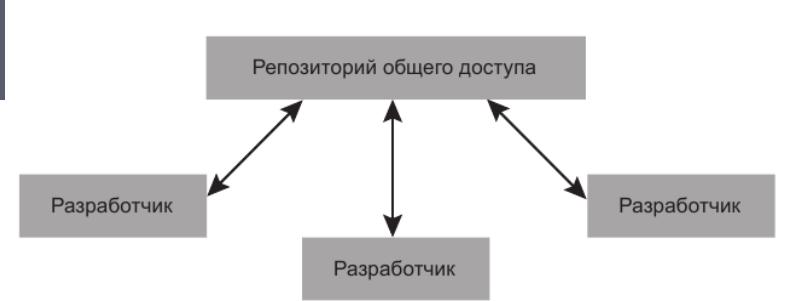
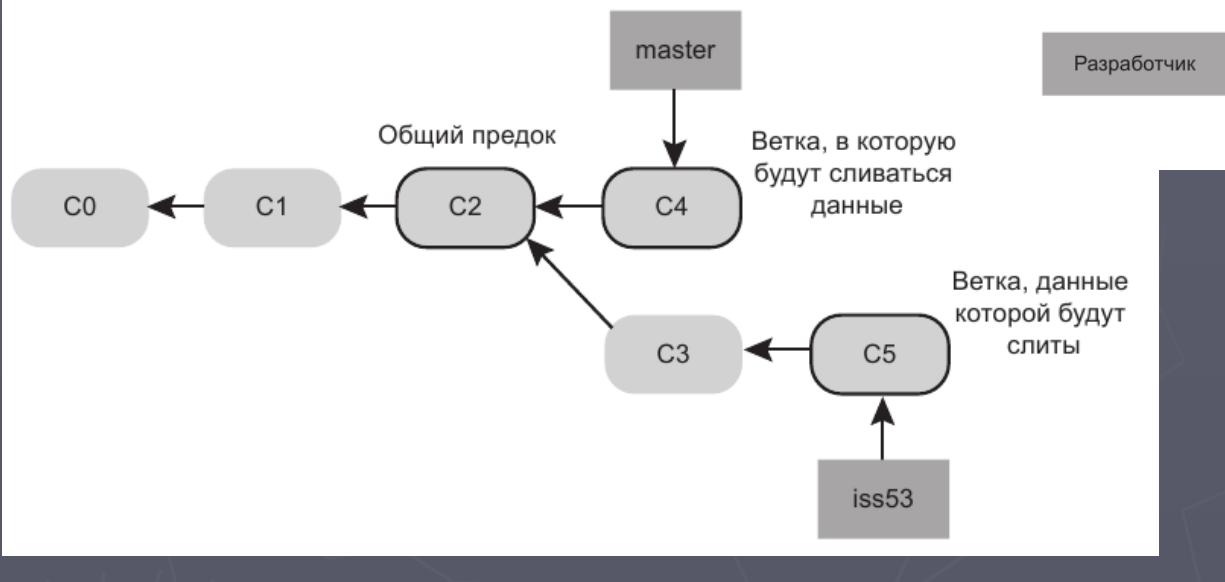


► Понятие HEAD

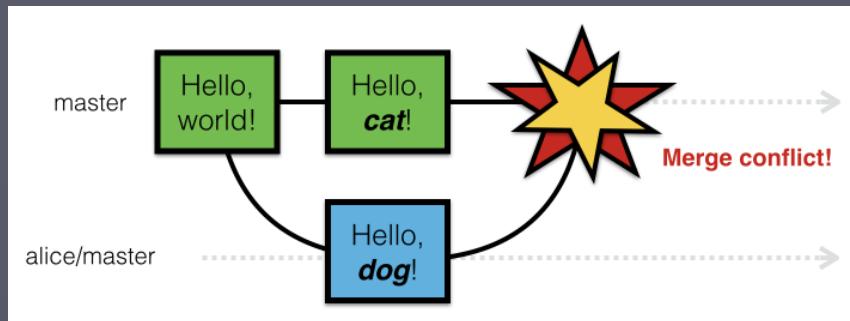


СМЕНА ВЕТОК МЕНЯЕТ ФАЙЛЫ В РАБОЧЕЙ ПАПКЕ

6) слияния



► Конфликты при слиянии

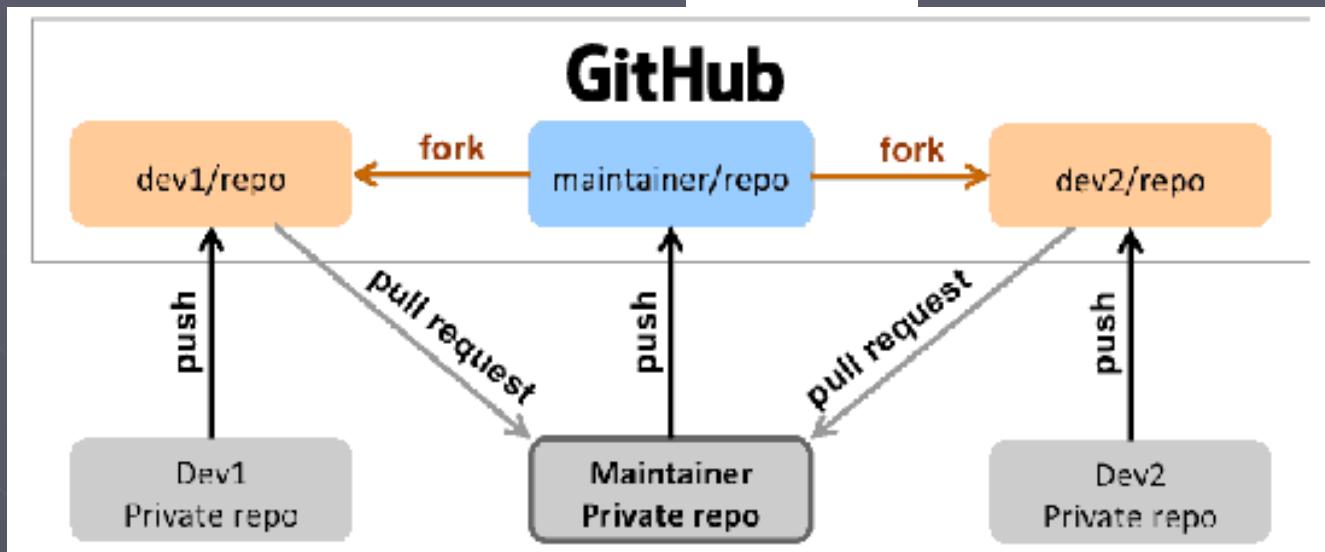


A screenshot of a code editor window titled "SimpleApplication.java". The code is a Java class with several methods. The editor highlights regions of code with different colors and styles, indicating conflicts between different branches. Specifically, it shows regions from line 10 to 14 and line 18 to 22, which are marked with magenta highlights and conflict markers (<<<<< HEAD, >>>>> refs/heads/new_idea).

```
3 public class SimpleApplication {  
4     /**  
5      * @param args  
6      */  
7     public static void main(String[] args) {  
8         // TODO Auto-generated method stub  
9         <<<<< HEAD  
10        // Fix a core bug  
11        =====  
12        // Alternate core fix  
13        >>>>> refs/heads/new_idea  
14    }  
15  
16    public static void uiFunction() {  
17        <<<<< HEAD  
18        // Fix a UI bug  
19    }  
20  
21    public static void addSecurity() {  
22        // Adds the needed security  
23        =====  
24        // Alternate UI fix  
25        >>>>> refs/heads/new_idea  
26    }  
27  
28  
29  
30}
```

```
23  
24 class String  
25 <<<<< HEAD:lib/jekyll/core_ext.rb  
26   def cutoff(desired = 5)  
27   =====  
28     def cutoff(desired = 400)  
29 >>>>> conflicts:lib/jekyll/core_ext.rb  
30     return self if self.length <= desired
```

ветвление проектов



Created Assigned Mentioned Review requests

is:open is:pr author:pnvtest archived:false

0 Open 0 Closed

Visibility Organization Sort

!

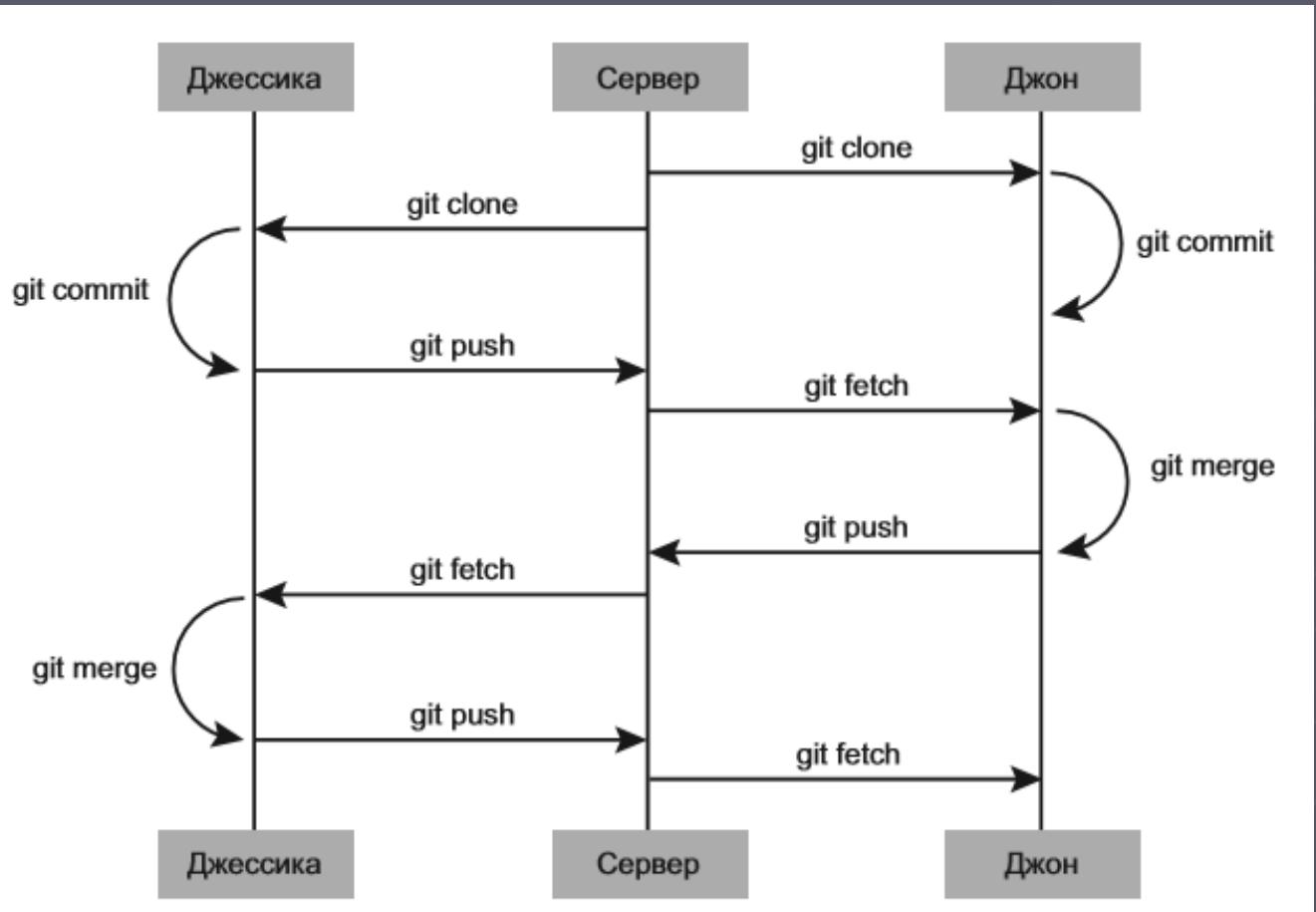
No results matched your search.

You could search [all of GitHub](#) or try an [advanced search](#).

ProTip! Type [g](#) [i](#) on any issue or pull request to go back to the issue listing page.

Распределенная разработка

► Centralized Workflow

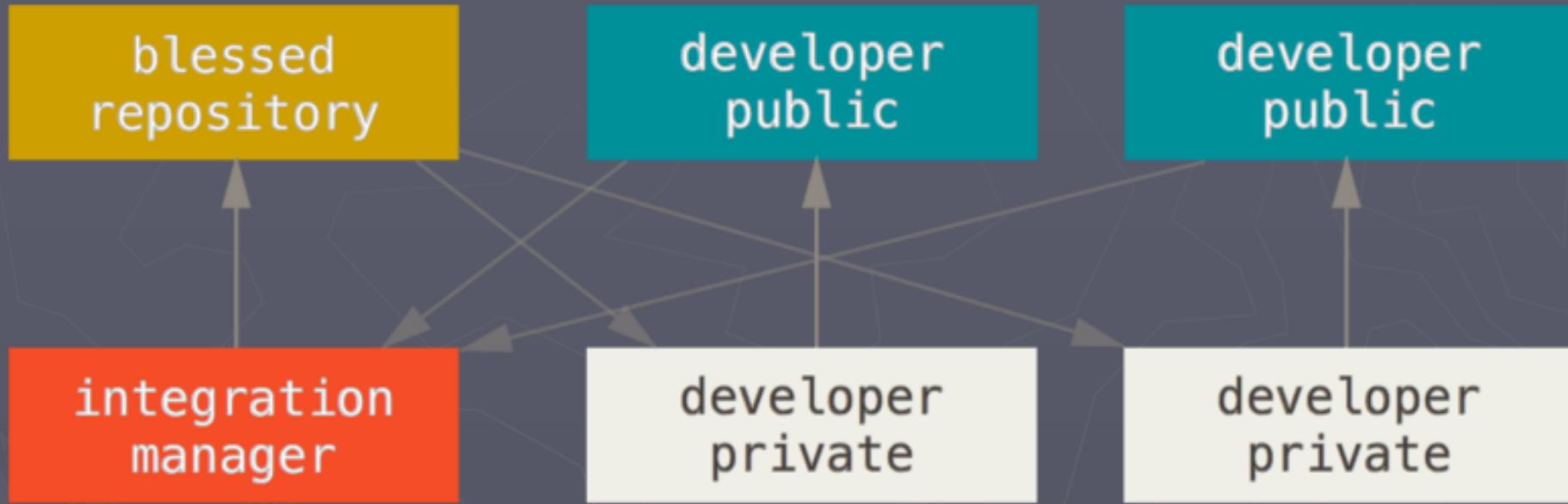


developer

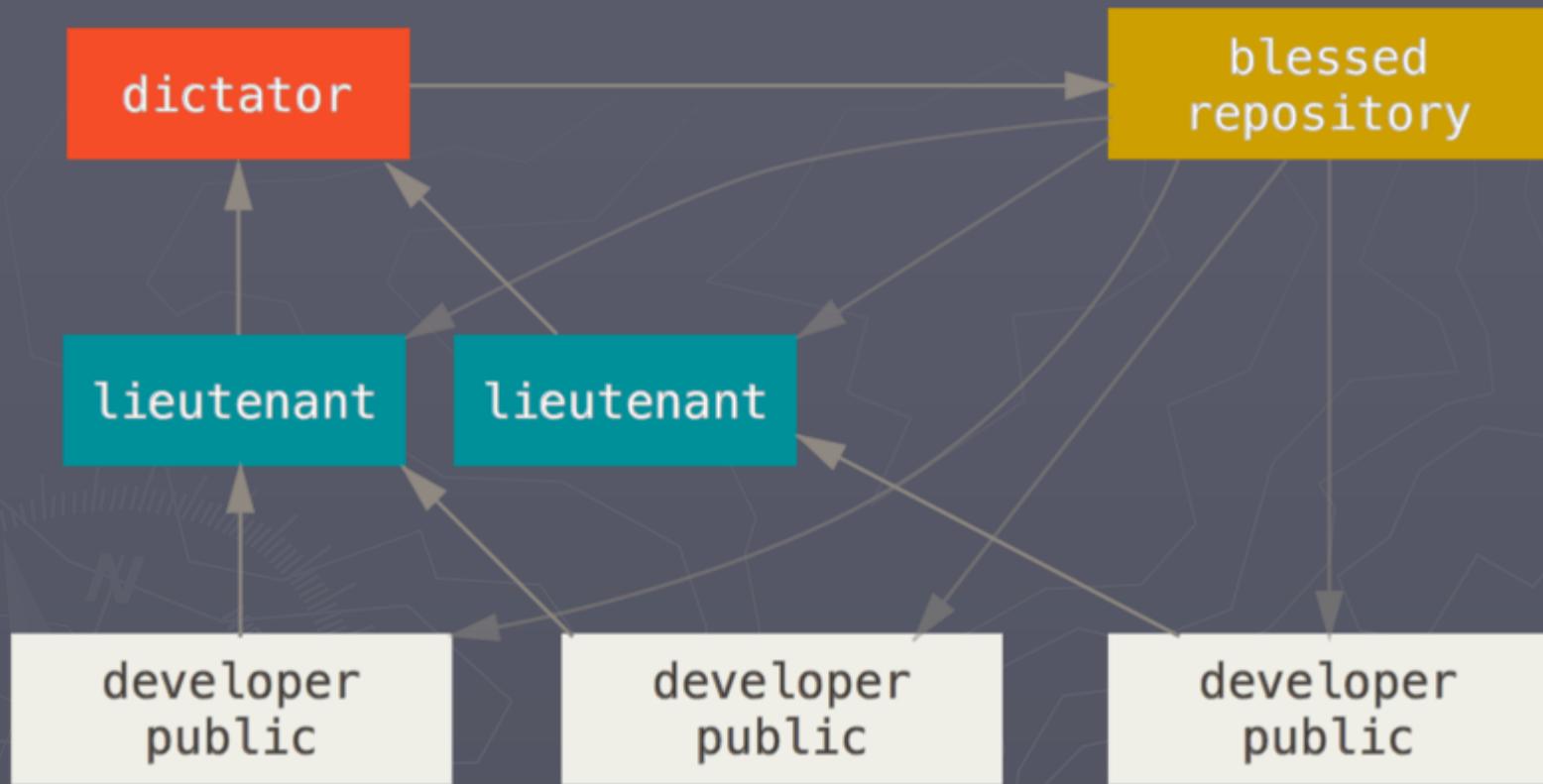
developer

26 developer

► Integration-Manager Workflow



► Dictator and Lieutenants Workflow



7) хостинг Git-репозиториев

<https://git-scm.com/book/ru/v2/>

Протоколы - HTTP, SSH (Secure Shell) и Git

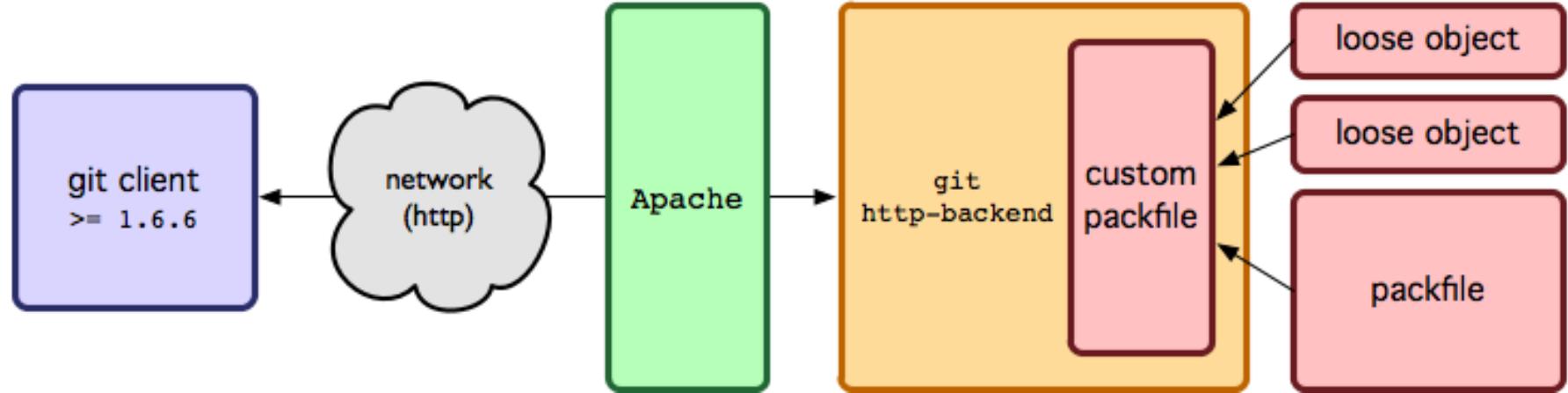
Приложение GitLab

Сторонний хостинг

<https://git.wiki.kernel.org/index.php/GitHosting>
[\(GitHub \)](#)

the user

your server



8) Управление организацией

The screenshot shows the GitHub organization creation process. At the top, there is a navigation bar with various icons and links like Pull requests, Issues, Marketplace, and Explore. A search bar is also present. On the right, a dropdown menu is open with options: New repository, Import repository, New gist, and **New organization** (which is highlighted in blue).

The main area is titled "Sign up your team" and contains three steps:

- Step 1: Set up the organization** (represented by a user icon)
- Step 2: Invite members** (represented by a group icon)
- Step 3: Organization details** (represented by a gear icon)

Below these steps, there is a section titled "Create an organization account". It includes fields for "Organization name" (marked with a red asterisk) and "Billing email" (also marked with a red asterisk). A note below the name field states: "This will be your organization name on <https://github.com/>". Another note below the email field states: "We'll send receipts to this inbox". To the right of these fields, a descriptive text explains: "Organization accounts allow your team to plan, build, review, and ship software — all while tracking bugs and discussing ideas."

At the bottom, there is a section titled "Choose your plan". It offers three options:

- Team For Open Source** (green circle with a checkmark): "Advanced collaboration and management tools for open source teams"
- Team** (white circle): "Advanced collaboration and management tools for teams"
- Enterprise** (white circle): "Security, compliance, and deployment controls for organizations". Below this, there is a note in Russian: "Активация Windows. Чтобы активировать Windows, перейдите в раздел 'Параметры'." (Activation of Windows. To activate Windows, go to the 'Settings' section.)

The URL at the bottom left is <https://github.com/organizations/new>.



.NET, CLR, C#



Microsoft®

МАСТЕР
КЛАСС

CLR via C#

ПРОГРАММИРОВАНИЕ НА ПЛАТФОРМЕ
MICROSOFT .NET FRAMEWORK 4.5
НА ЯЗЫКЕ C#

4-е издание Джейффи Рихтер 



Professional

C# 7 and
.NET Core 2.0

O'REILLY®

7th Edition
Covers .NET Standard 2

C# 7.0 in a Nutshell

THE DEFINITIVE REFERENCE

Joseph Albahari & Ben Albahari

Язык
программирования
C# 7 и платформы .NET
и .NET Core

8-е издание

Эндрю Троелсен
Филипп Джепик

ДИАЛЕКТИКА
www.williamspublishing.com

Apress®

- ▶ Пацей, Н. В. Объектно-ориентированное программирование на C++/C#: учеб.-метод. пособие . В 2 ч., Ч. 1/ Н. В. Пацей – Минск.: БГТУ, 2014. – 191 с.
- ▶ Книги в папке на diskstation
- ▶ <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/>
- ▶ <https://github.com/dotnet/csharplang/blob/master/spec/README.md>
- ▶ professorweb.ru
- ▶ <https://metanit.com/sharp/general.php>

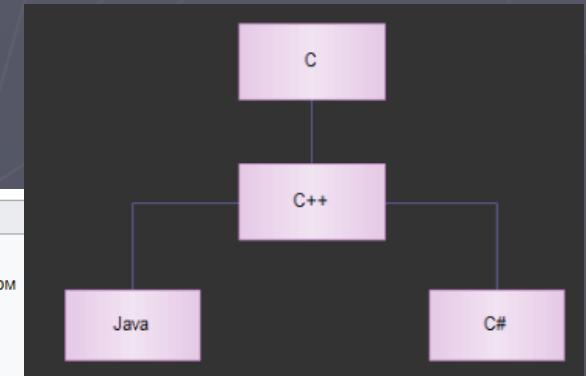
C# - 2000г. А́ндре́с Хэ́йлсберг

Версия	Спецификация языка			Дата	.NET Framework	Visual Studio
	ECMA	ISO/IEC	Microsoft			
C# 1.0	Декабрь 2002	Апрель 2003 (недоступная ссылка)		Январь 2002	.NET Framework 1.0	Visual Studio .NET (2002)
C# 1.2				Октябрь 2003	.NET Framework 1.1	Visual Studio .NET 2003
C# 2.0	Июнь 2006	Сентябрь 2006		Сентябрь 2005 ^[15]	.NET Framework 2.0	Visual Studio 2005
C# 3.0		Отсутствует ^[16]		Август 2007	.NET Framework 3.5	Visual Studio 2008
C# 4.0				Апрель 2010	.NET Framework 4	Visual Studio 2010
C# 5.0	Декабрь 2017	Отсутствует ^[16]		Июнь 2013	.NET Framework 4.5	Visual Studio 2012
C# 6.0		Отсутствует		Июль 2015	.NET Framework 4.6	Visual Studio 2015
C# 7.0		Отсутствует		Март 2017	.NET Framework 4.6.2	Visual Studio 2017
C# 8.0		Отсутствует		Сентябрь 2019	.NET Framework 4.8	Visual Studio 16.3.0

► ECMA-334 и ISO/IEC 23271

Общая информация по версиям

	C# 2.0	C# 3.0	C# 4.0	C# 5.0	C# 6.0	C# 7.0 ^[13]
Новые возможности	<ul style="list-style-type: none"> Частичные типы Обобщённые типы (<i>generics</i>) Итераторы и ключевое слово <i>yield</i> Анонимные методы Оператор <i>null</i>-объединения <i>Nullable</i>-типы 	<ul style="list-style-type: none"> Запросы, интегрированные в язык (<i>LINQ</i>) Инициализаторы объектов и коллекций Лямбда-выражения Деревья выражений Неявная типизация и ключевое слово <i>yield</i> Анонимные типы Методы 	<ul style="list-style-type: none"> Динамическое связывание и ключевое слово <i>dynamic</i> Именованные и опциональные аргументы Обобщенная covariance и contravariance Библиотека <i>TPL</i>, концепция задач и классы <i>Task</i>, <i>Parallel</i> Класс <i>MemoryCache</i> Классы параллельных коллекций 		<ul style="list-style-type: none"> Компилятор как сервис Импорт членов статических типов в пространство имён Фильтры исключений <i>await</i> в блоках <i>catch / finally</i> Шаблон <i>TAP</i> Асинхронные методы <i>async</i> и <i>await</i> Сведения о вызывающем объекте 	<ul style="list-style-type: none"> <i>out</i>-переменные Сопоставление с шаблоном Шаблоны с <i>is</i> Шаблоны и выражение <i>switch</i> Кортежи Распаковка кортежей (деконструktоры) Локальные функции Улучшения литералов Локальные переменные и возвращаемые значения по ссылке Расширение списка типов, возвращаемых асинхронными методами



Спецификация CLI

- ▶ ***CLI (Common Language Infrastructure)*** – спецификация общеязыковой инфраструктуры. Определяет архитектуру исполнительной системы и набор представляемых сервисов.

Стандарты: ECMA-335 и ISO/IEC 23271.

ISO/IEC 23271

- ▶ Концепция и архитектура. Здесь определены понятия: ***CTS (Common Type System), VES (Virtual Execution System) и CLS (Common Language Specification)***
- ▶ Метаданные и семантика.
- ▶ Инструкции CIL. ***CIL (Common Intermediate Language)***.
- ▶ Библиотеки и профили. ***BCL (Base Class Library)***. Описание библиотеки поставляется в виде xml-файла ***CLILibraryTypes.xml***.
- ▶ Формат взаимодействия с отладчиком
- ▶ Приложения

.NET Framework

- Microsoft.NET (.NET Framework) – программная платформа. Содержит следующие основные компоненты:

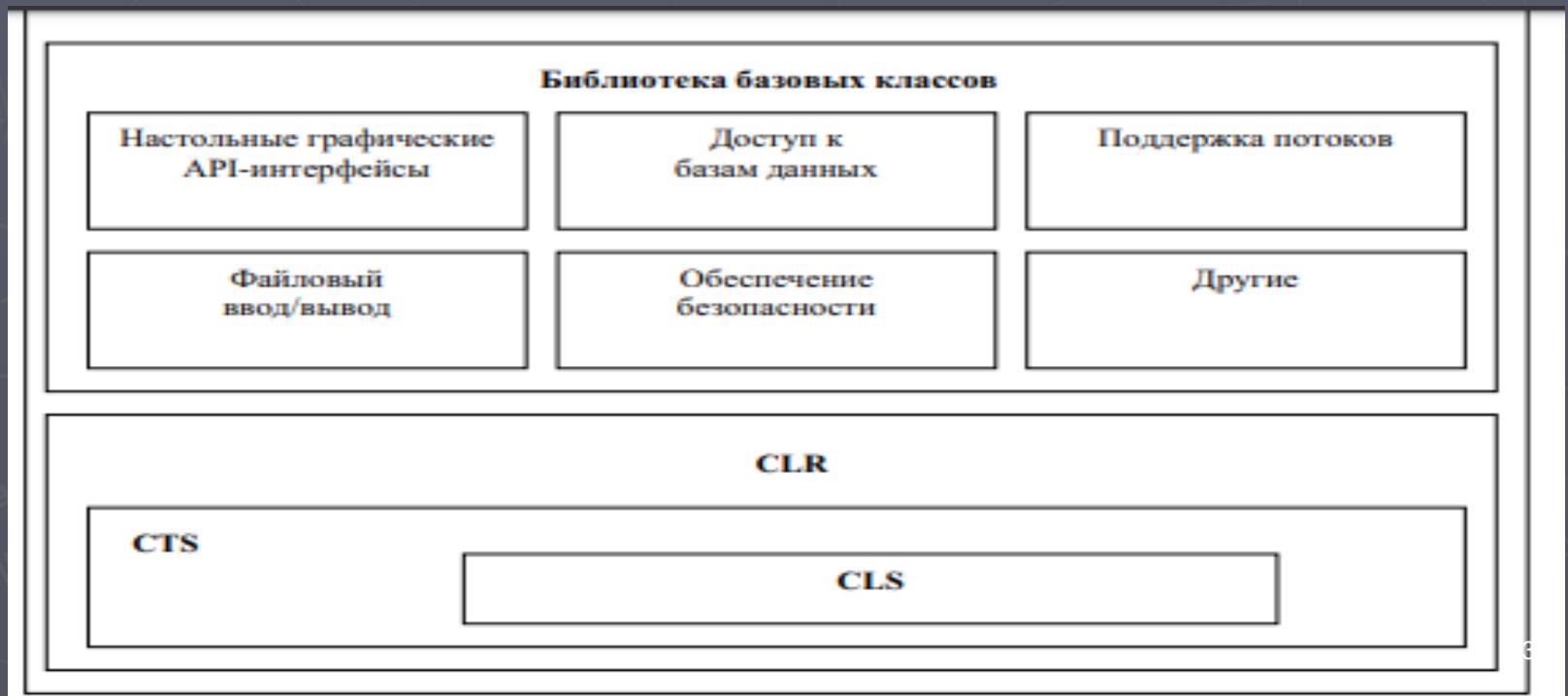
обеспечивает совместное использование разных языков программирования, а также безопасность, переносимость программ и общую модель программирования для платформы Windows

CLR (Common Language Runtime) – общеязыковая среда исполнения, виртуальная машина на которой исполняются все приложения, работающие в среде .NET. Реализация CLI VES компанией Microsoft. Компилятор **JIT (Just in Time)**.

MSIL (Microsoft IL) – реализация CLI CIL компанией Microsoft.

FCL (Framework Class Library) – реализация CLI BCL компанией Microsoft. Можно рассматривать, как API CLR.

- ▶ CLR (Common Language Runtime) – Среда Времени Выполнения или Виртуальная Машина. Обеспечивает выполнение сборки (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация)
- ▶ FCL (.NET Framework Class Library) – соответствующая CLS спецификации объектно-ориентированная библиотека классов, интерфейсов и системы типов (типов-значений)



Традиционные Windows-приложения

.NET-приложения
Базовые типы:
Windows Application, Console Application Class Library

Visual Studio .NET

Языки программирования Microsoft (VB, C/C++, C#, J#, JScript) и независимых поставщиков

Common Language Specification

Типы .NET-приложений (Console, Windows Forms, Components, ASP .NET, Web Services и пр.)

.NET Framework

Библиотеки классов

Базовые классы .NET Framework

Дополнительные классы .NET

Связь с COM-объектами

CLR (Common Language Runtime)

Windows

Сервисы операционной системы (Win32 API)

.NET FRAMEWORK – решение следующих проблем

- ▶ 1. Интеграция языков программирования.
- ▶ **CLS (Common Language Specification)** – общеязыковая спецификация, предназначенная для разработчиков компиляторов. → **CTS (Common Type Systems)**- спецификацию типов, которые должны поддерживаться всеми языками ориентированными на CLR. Microsoft выпустил несколько компиляторов соответствующих этой спецификации: C++/CLI (C++ с управляемыми расширениями), C#, VB .NET, JScript.

► 2. Работа на многих платформах.

- При компиляции кода компиляторы .NET Framework генерируют код на промежуточном языке (**CIL**, **Common Intermediate Language**). При исполнении CLR транслирует CIL-код в команды соответствующего процессора. В принципе, однажды .NET Framework-приложение должно работать везде, где установлены и работают CLR и FCL.

► 3. Упрощенное повторное использование кода.

- CLR позволяет типы разработанные на одном языке использовать в других языках.

► 4. Автоматическое управление памятью.

- CLR автоматически отслеживает использование ресурсов. Сборщик мусора.

► 5. Проверка безопасности типов.

- При работе в CLR практически исключена возможность записать (стереть) данные в область памяти, которая для этого не предназначена. Нет возможности передать управление в произвольную точку.

- ▶ **6. Единый принцип обработки сбоев.**
- ▶ Один из самых неприятных моментов в Windows-программирование – это отсутствие единой системы обработки ошибок и сбоев: возврат функций, коды состояний, HRESULT, исключения и т.п. Для обработки ошибок и сбоев в CLR используется только механизм исключений.

► 7. Взаимодействие с существующим кодом.

- Поддерживаются функции Win32 DLL – библиотек.

► 8. Проблемы с версиями.

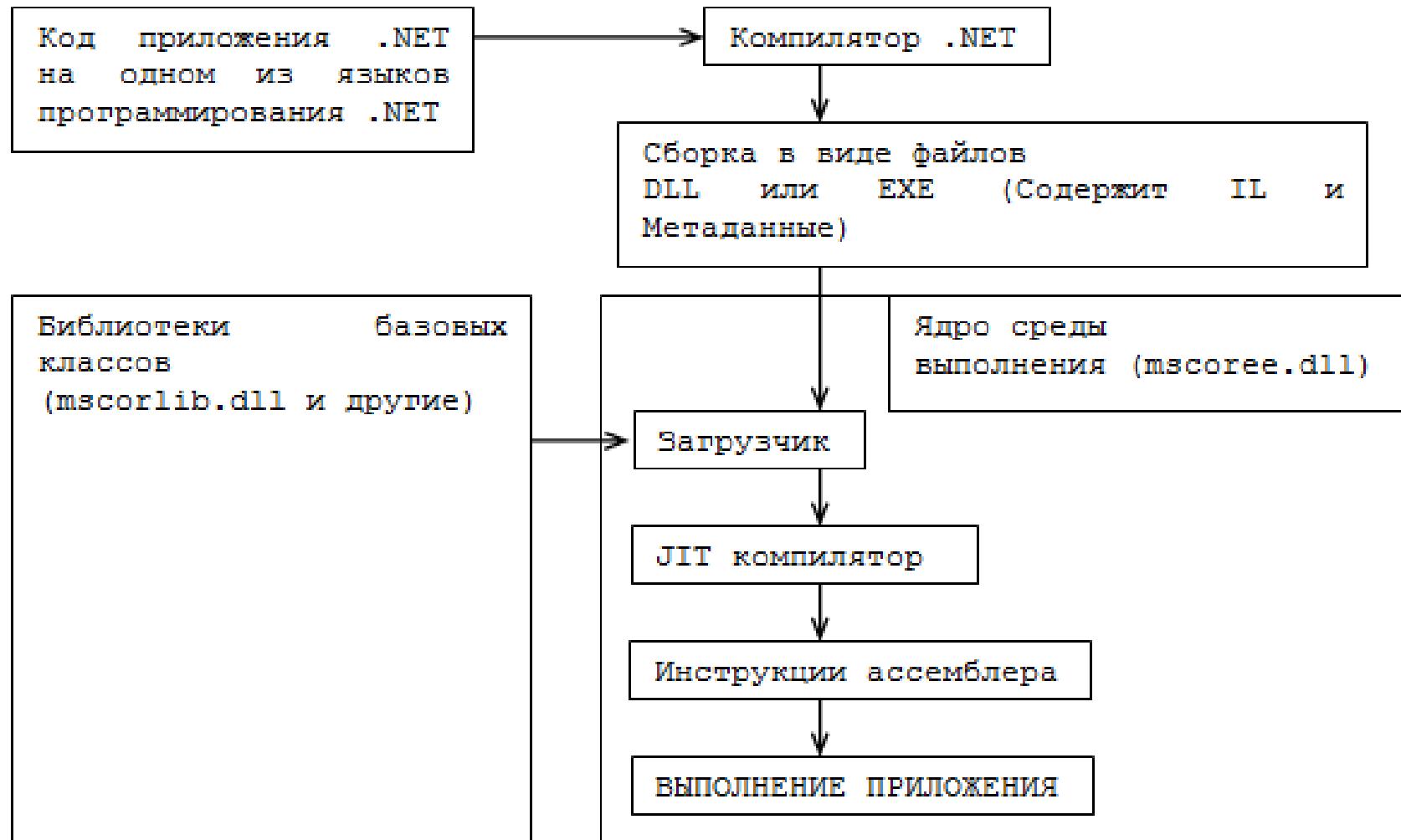
- В Windows возникают проблемы связанные с совместимостью DLL-библиотек. .NET Framework приложение всегда работает с компонентами с которыми компилировалось и тестировалось приложение.

CLR

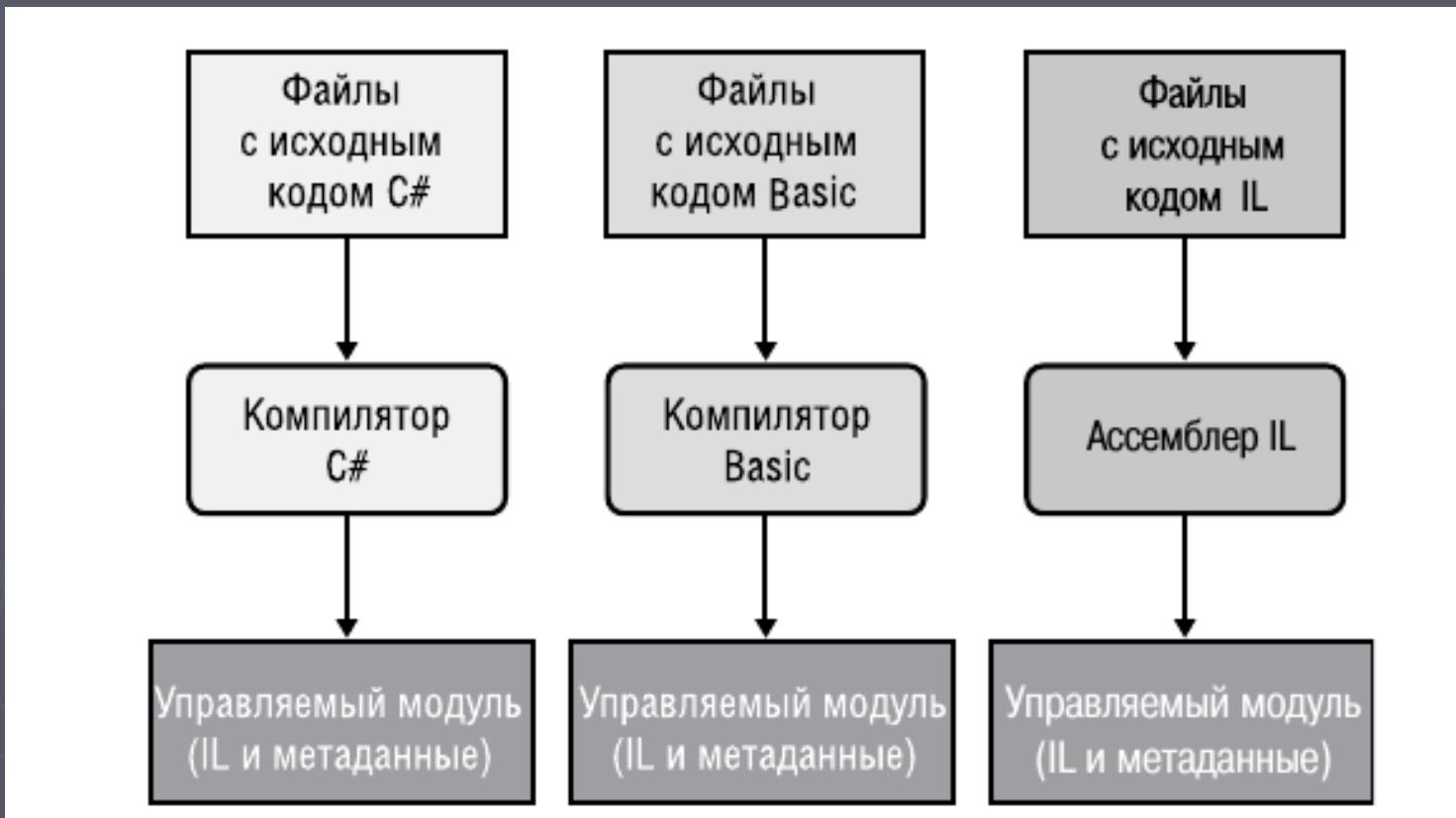
C++/CLI, Visual Basic, F#, Iron Python,
Iron Ruby и ассемблер Intermediate
Language (IL)

Ada, APL, Caml, COBOL, Eiffel, Forth, Fortran,
Haskell, Lexico, LISP, LOGO, Lua, Mercury,
ML, Mondrian, Oberon, Pascal, Perl, Php,
Prolog, RPG, Scheme, Smalltalk

Структура среды выполнения CLR



1) Компиляция исходного кода в управляемые модули



assembly

2) Выполнения кода в среде CLR

CLR (Common Language Runtime – общязыковая исполняющая среда)

IL

оперативная
компиляция
(Just In Time
– JIT).



машинный код

IL-код
компилятора

машинный
JIT-код

Оптимизация
/optimize и /debug

NGen.exe IL->машинный код

IL
объектно-ориентированный машинный
язык не зависящий от процессора

ILAsm.exe – ассемблер

ILDasm.exe - дизассемблер IL

Компиляция

csc.exe /out:Program.exe /t:exe
/r:MSCorLib.dll Program.cs

csc.exe Program.cs



PE (portable executable)

Управляемы модуль - portable executable (PE)

Заголовок PE32 или PE32+

Заголовок CLR

Метаданные

Код Intermediate Language (IL)

заголовок PE (32/64),

заголовок CLR (версия CLR, точки входа модуля, размеры и месторасположение ресурсов и метаданных),

метаданные (специальные таблицы, содержащие исходный код типов и членов данных);

код IL (код который CLR компилирует в команды процессора).

Метаданные

двоичный набор таблиц данных:

типы и их члены

портируемые типы и их члены

Назначение:

- 1) устраняют необходимость в заголовочных файлах (прототипы);
- 2) используются в VS для подсказок;
- 3) используется при верификации кода на предмет безопасных операций;
- 4) можно сериализовать объект одной машине и восстановить состояние объекта на другой машине;
- 5) используются при сборке мусора.

Таблицы определений:

- ▶ **ModuleDef** – одна запись, идентифицирующая модуль (версия, имя, GUID).
- ▶ **TypeDef** – запись для каждого типа, определенного в модуле.
- ▶ **MethodDef** – запись для каждого метода (сигнатура, смещение в модуле кода MSIL, ссылка на **ParamDef**).
- ▶ **FieldDef** – запись для каждого поля.
- ▶ **ParamDef** – запись для каждого параметра.
- ▶ **PropertyDef** – запись для каждого свойства.
- ▶ **EventDef** – запись для каждого события.

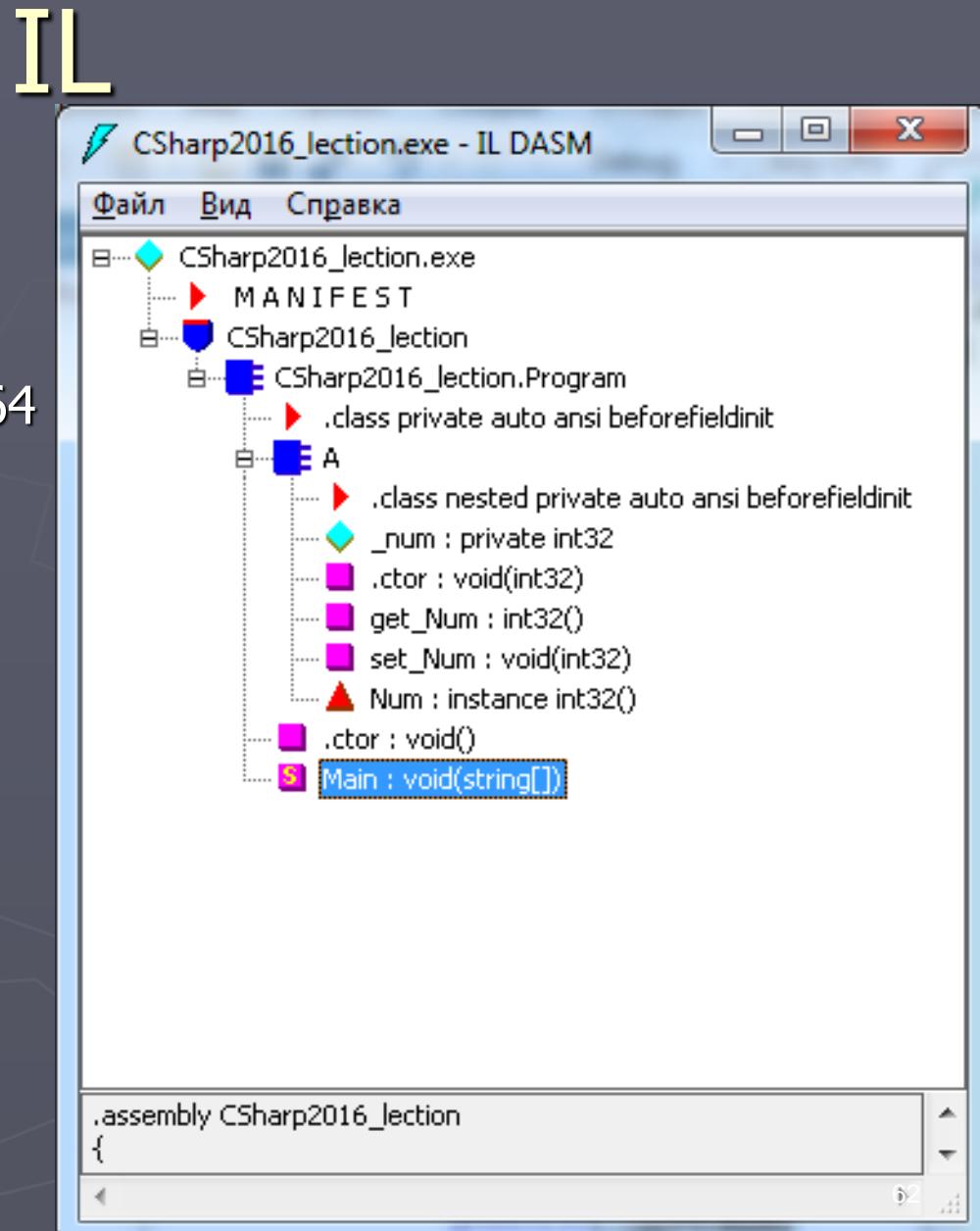
► Таблицы ссылок

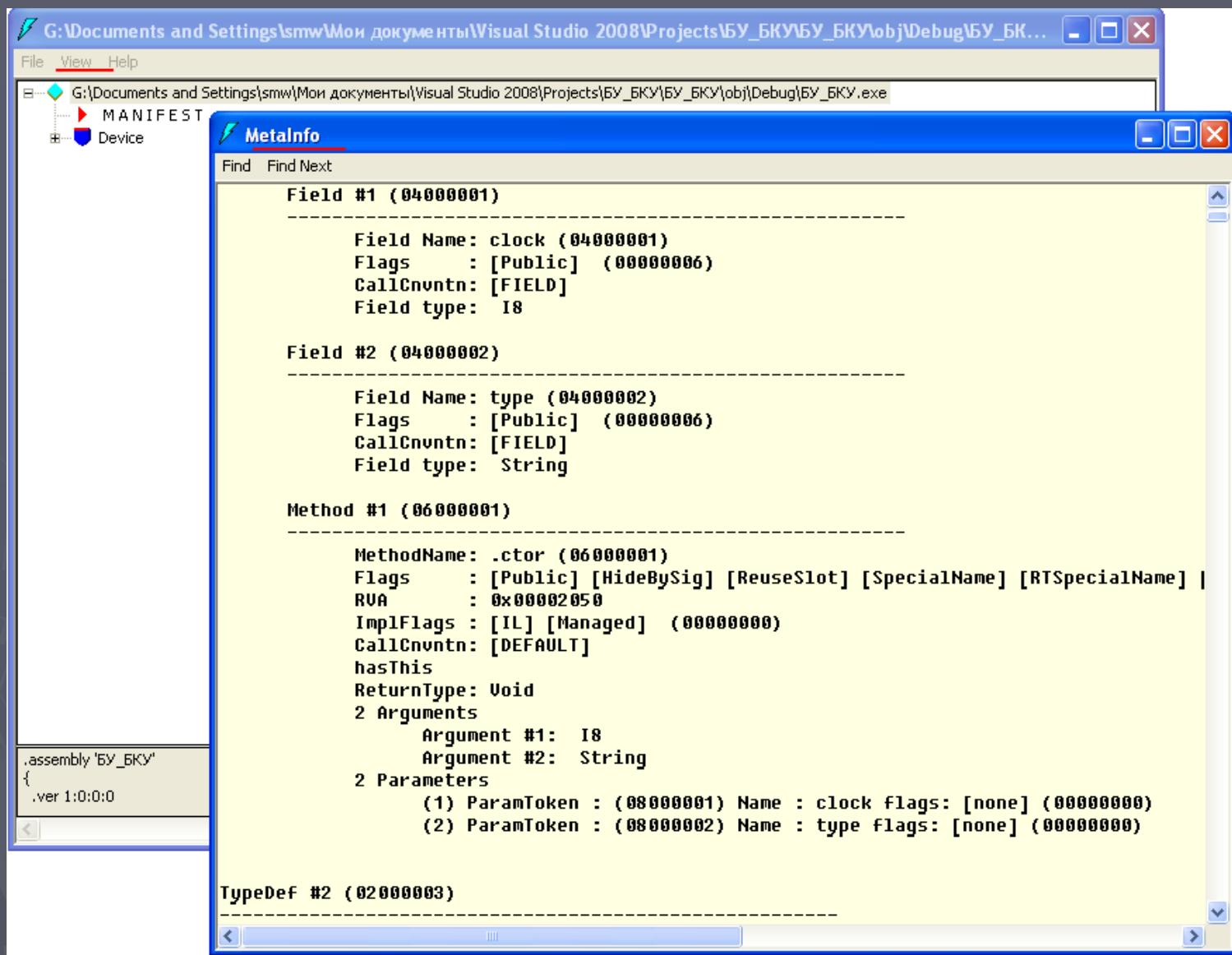
- **AssemblyRef** – запись для каждой сборки на которую ссылается модуль.
- **ModuleRef** – запись для каждого PE-модуля, на типы которого ссылается код модуля.
- **TypeRef** – запись для каждого типа на который ссылается модуль.
- **MemberType** - запись для каждого члена, на который ссылается модуль.

Таблицы метаданных декларации

- ▶ **AssemblyDef** – идентифицирует сборку
- ▶ **FileDef** - по одной для каждого PE-файла и файла ресурсов.
- ▶ **ManifestResourceDef** – по одной для каждого файла ресурсов.
- ▶ **ExportedTypesDef** – записи для всех открытых (public) типов.

Program Files (x86)\
Microsoft SDKs\
Windows\v7.0A\Bin\x64
Ildasm.exe





Манифест

набор таблиц метаданных

файлы, которые входят в сборку,
общедоступные экспортимые

типы,

файлы ресурсов или данных

G:\Documents and Settings\smw\Мои документы\Visual Studio 200...

File View Help

G:\Documents and Device MANIFEST

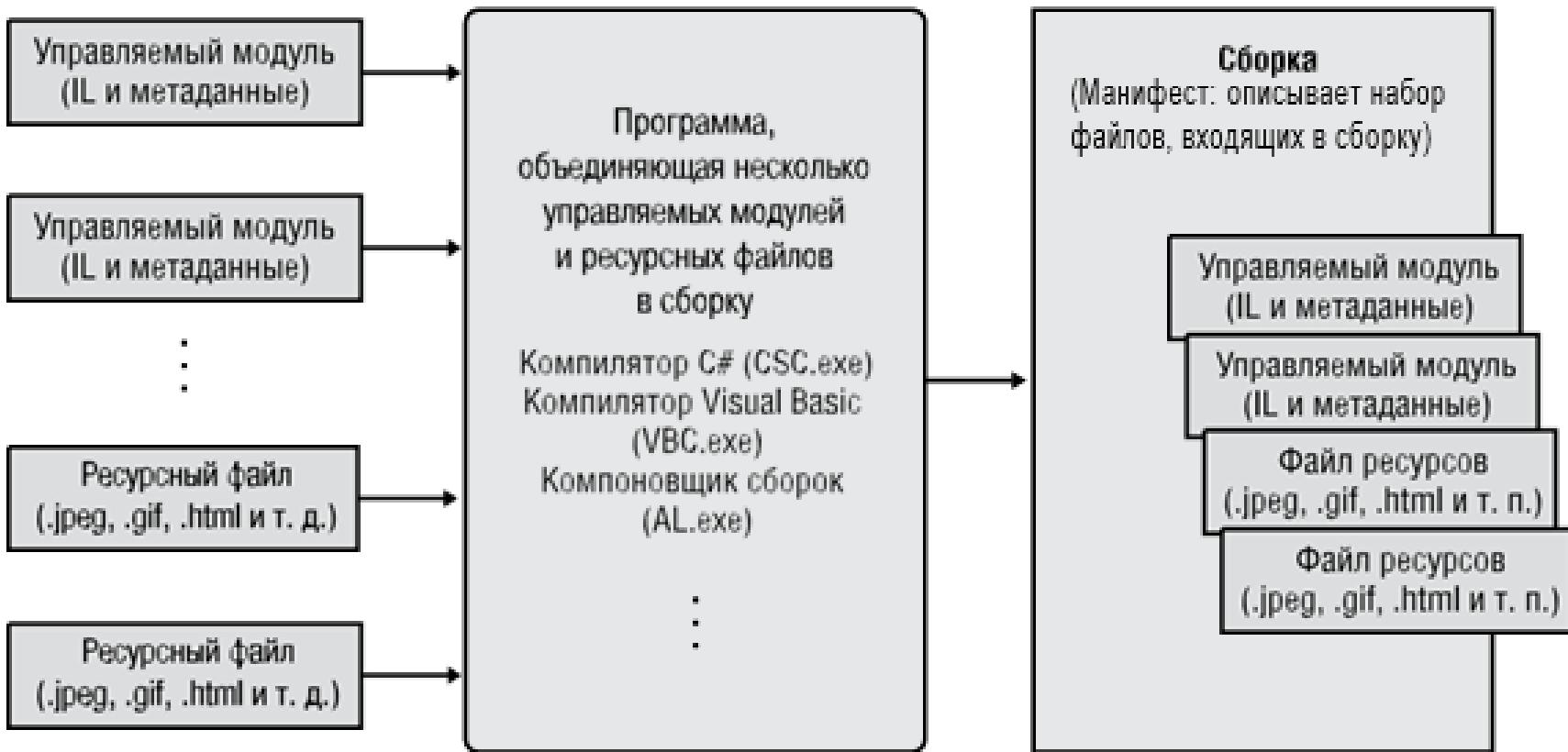
Find Find Next

```
// Metadata version: v2.0.50727
.assembly extern mscorel
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 2:0:0:0
}
.assembly extern System
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 2:0:0:0
}
.assembly extern System.Windows.Forms
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 2:0:0:0
}
.assembly extern System.Xml
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 2:0:0:0
}
.assembly extern System.Drawing
{
    .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A )
    .ver 2:0:0:0
}
.assembly extern System.Core
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 3:5:0:0
}
.assembly 'БУ_БКУ'
{
    .ver 1:0:0:0
}
.custom instance void [mscorlib]System.Runtime.CompilerServices
.custom instance void [mscorlib]System.Reflection.AssemblyP
.custom instance void [mscorlib]System.Runtime.CompilerServices
```

Сборки

- ▶ Сборка (assembly) — 1) это абстрактное понятие, для логической группировки одного или нескольких управляемых модулей или файлов ресурсов.
- ▶ 2) дискретная единица многократно используемого кода внутри CLR

Exe, dll



Исполнение сборки

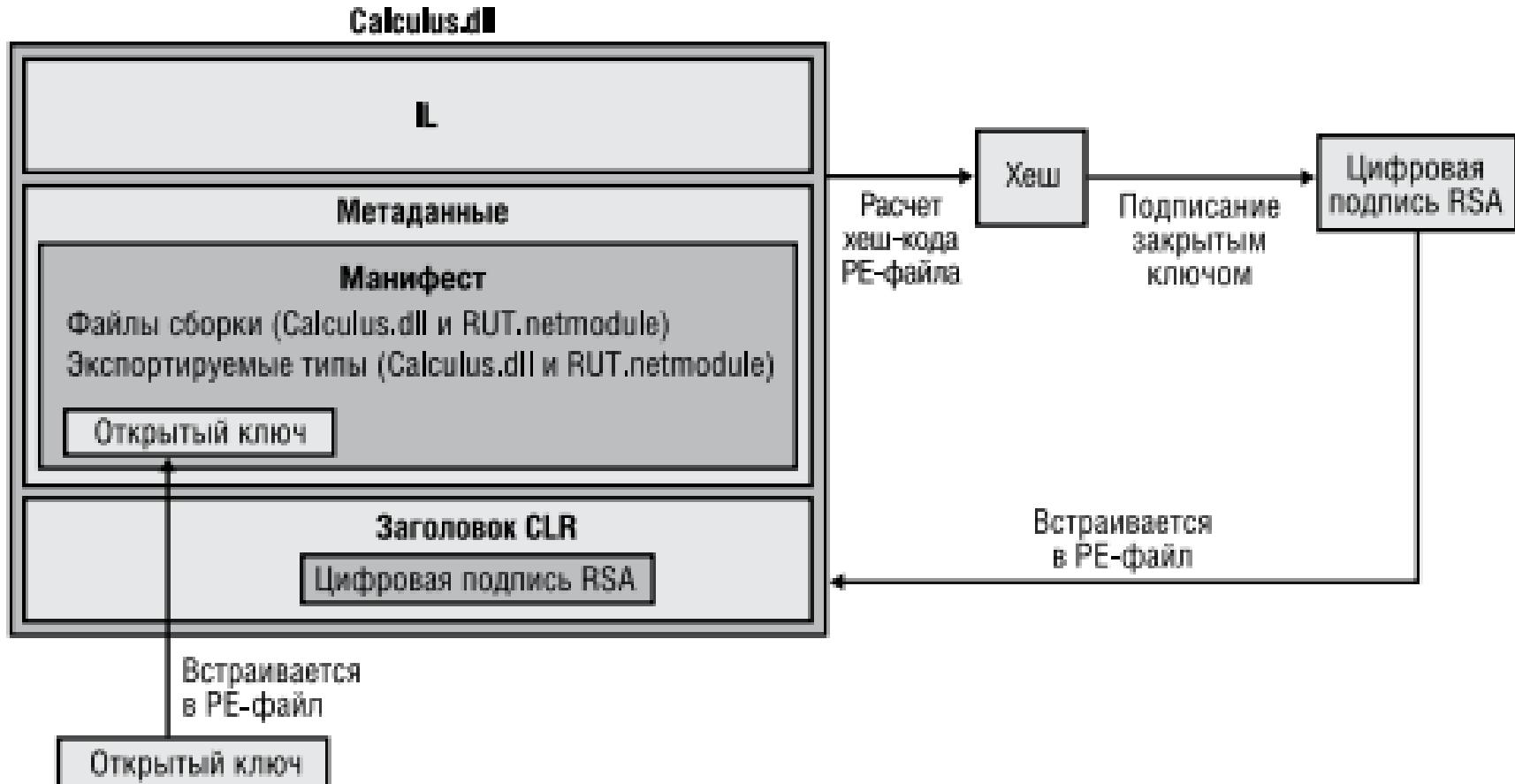
JIT-компилятор (Just-In-Time)

- 1) CLR ищет типы данных и загружает во внутренние структуры
- 2) Для каждого метода CLR заносит адрес внутренней CLR функции JITCompiler
- 3) JITCompiler ищет в метаданных соответствующей сборки IL-код вызываемого метода, проверяет и компилирует IL-код в машинные команды
- 4) Они хранятся в динамически выделенном блоке памяти.
- 5) JITCompiler заменяет адрес вызываемого метода адресом блока памяти, содержащего готовые машинные команды
- 6) JITCompiler передает управление коду в этом блоке памяти.

Типы сборок:

- ▶ с нестрогими именами (weakly named assemblies)
- ▶ со строгими именами (strongly named assemblies).
 - подписаны при помощи пары ключей, уникально идентифицирующей издателя сборки (безопасность, управление ее версиями, развертывание в любом месте пользовательского жесткого диска или в Интернете)
 - атрибуты: имя файла (без расширения), номер версии, идентификатор регионального стандарта и открытый ключ.

Подписание сборки



Развертывание сборки

► закрытым

- развертываются в базовом каталоге приложения или в одном из его подкаталогов. Для сборки с нестрогим именем возможно лишь закрытое развертывание.

► глобальным

%systemroot%\assembly

GAC – Global Assembly Cache

GACUtil.exe

The screenshot shows a Windows application window titled "Windows assembly". The interface includes a toolbar with icons for file operations, a breadcrumb navigation bar ("Windows > assembly"), and a search bar. On the left is a sidebar with categories like "Из" (Installed), "Би" (Bibliography), and "До" (Deployment). The main area is a grid table with columns: Имя сборки (Assembly Name), Версия (Version), Культура (Culture), Маркер открытого ключа (PublicKeyToken), and Архитектура ... (Architecture). The table lists numerous assemblies, many of which have their names partially obscured by ellipses (...). The entries include Accessibility, ADODB, AuditPolicyGPMan..., BDATunePIA, ComSvcConfig, cscompmgd, CustomMarshalers, dfsvc, ehCIR, ehexhost, ehexhost32, ehiActivScp, and ehiBmlDataCarousel. Most entries are MSIL or x86, with a few AMD64 entries. At the bottom of the table, it says "Элементов: 0".

Имя сборки	Версия	Культура	Маркер открытого ключа	Архитектура ...
Accessibility	2.0.0.0	b03f5f7f11d50a3a		MSIL
ADODB	7.0.330...	b03f5f7f11d50a3a		
AuditPolicyGPMan...	6.1.0.0	31bf3856ad364e35		x86
AuditPolicyGPMan...	6.1.0.0	31bf3856ad364e35		AMD64
BDATunePIA	6.1.0.0	31bf3856ad364e35		x86
BDATunePIA	6.1.0.0	31bf3856ad364e35		AMD64
ComSvcConfig	3.0.0.0	b03f5f7f11d50a3a		MSIL
cscompmgd	8.0.0.0	b03f5f7f11d50a3a		MSIL
CustomMarshalers	2.0.0.0	b03f5f7f11d50a3a		x86
CustomMarshalers	2.0.0.0	b03f5f7f11d50a3a		AMD64
dfsvc	2.0.0.0	b03f5f7f11d50a3a		MSIL
ehCIR	6.1.0.0	31bf3856ad364e35		MSIL
ehexhost	6.1.0.0	31bf3856ad364e35		MSIL
ehexhost32	6.1.0.0	31bf3856ad364e35		x86
ehiActivScp	6.1.0.0	31bf3856ad364e35		MSIL
ehiBmlDataCarousel	6.1.0.0	31bf3856ad364e35		MSIL
		31bf3856ad364e35		MSIL

Windows Installer (MSI)

► Читаем часть 1
С 1 -120 стр.

