



Инструмент для сравнения .NET сборок в интегрированной среде разработки Rider

Автор: Мирошников Владислав Игоревич, 19.Б11-мм

Научный руководитель: доцент кафедры СП, к.т.н. Ю.В. Литвинов

Консультант: технический руководитель Rider, к.т.н. И.Е. Мигалёв

Санкт-Петербургский Государственный Университет
Кафедра системного программирования

30 апреля 2022 г.

- Платформа .NET – платформа компании Microsoft, входит в тройку наиболее популярных фреймворков
- Сборка в .NET – это набор типов и ресурсов, которые образуют логическую функциональную единицу
- **Мотивация:** .NET пользователи время от времени хотят посмотреть на разницу двух сборок
- Rider – кроссплатформенная среда разработки для платформы .NET от компании JetBrains
- Предлагается добавить возможность сравнения .NET сборок в Rider

Цели и задачи

Цель: добавить инструмент сравнения скомпилированных .NET сборок в Rider

Задачи:

1. Провести обзор различных видов .NET сборок, а также обзор структуры и содержания .NET сборки
2. Провести обзор существующих решений, позволяющих сравнивать .NET сборки
3. Интегрировать в Rider инструмент для сравнения .NET сборок, обеспечивающий сравнение исходного кода, представленного в виде декомпилированного кода на C#
4. Провести апробацию решения с целью получения обратной связи от пользователей

Содержание и различные виды .NET сборок

Виды .NET сборок:

- Исполняемые сборки (*.exe)
- Сборки в виде файла библиотеки динамической компоновки (*.dll)

А также:

- Однофайловые
- Многофайловые

Содержание сборки:

- Манифест сборки
- Метаданные типов
- IL-код
- Ресурсы

Существующие решения

- Использование декомпилятора и инструмента для сравнения текста
- Полноценный инструмент
 - BitDiffer
 - JustAssembly

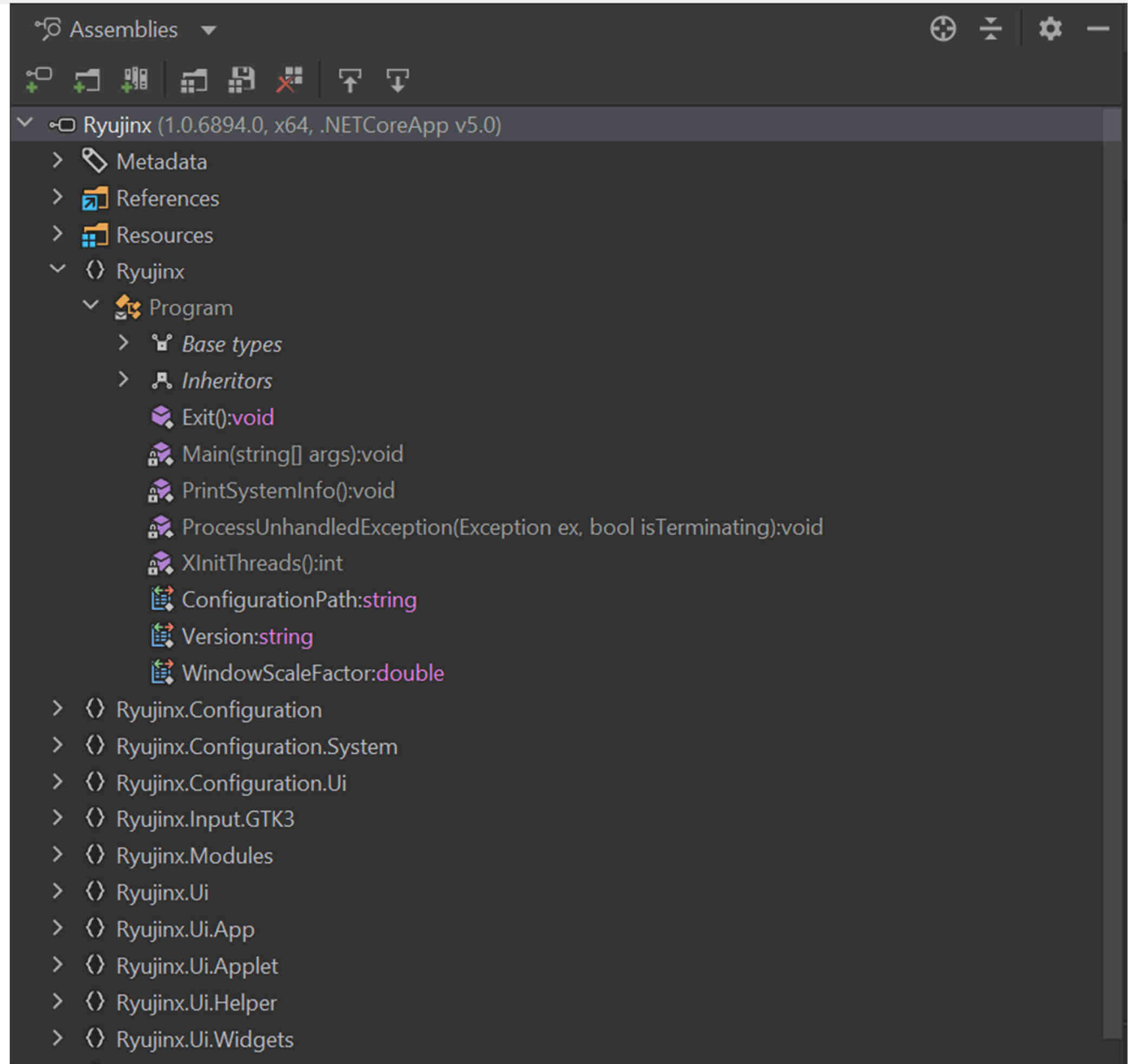
Декомпозиция задачи реализации

- «Разобрать» сборку и найти объекты, которые различаются и содержат исходный код
- Декомпилировать найденные объекты обратно в код на C#
- Посчитать и отобразить разницу

Подсистема Assembly Explorer

Дерево сборки:

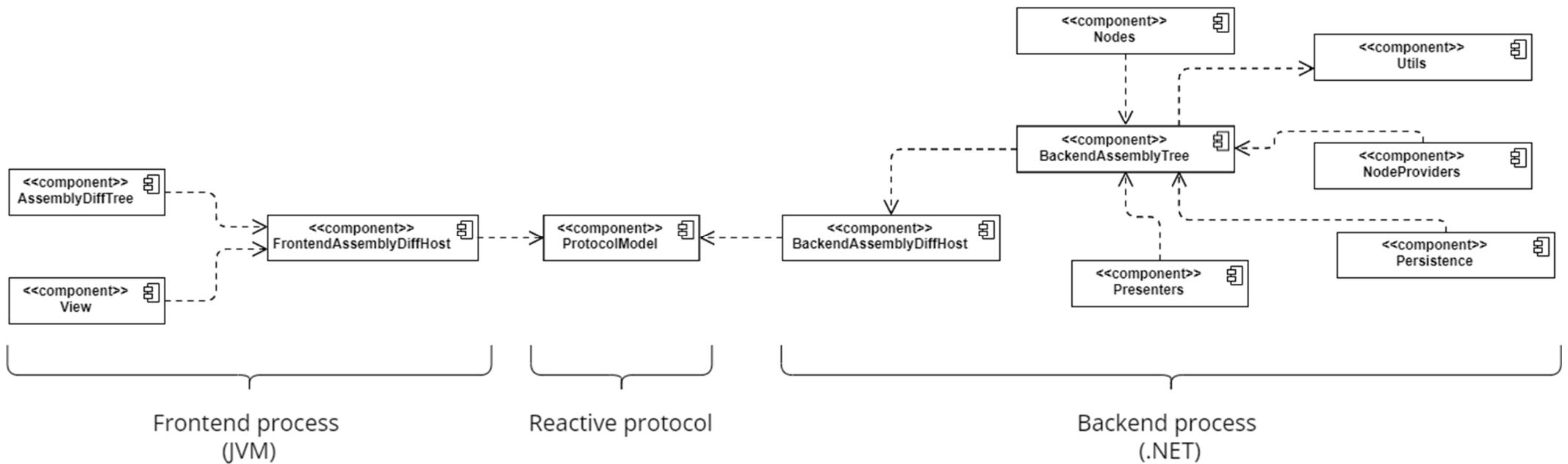
- Метаданные
- Зависимости от других сборок и модулей
- Ресурсные файлы
- Пространства имен и вложенные типы



Общий принцип работы:

- Инициализация фронтенд дерева и дополнительных компонентов, асинхронный запрос на бэкенд
- Инициализация и построение внутреннего бэкенд дерева
- Построение дерева с применением специальных алгоритмов
- Обмен данными при помощи протокольной DSL модели
- Вычисление разницы и отображение пользователю при нажатии на узел дерева

Архитектура инструмента сравнения



Пример работы

Ryujinx old.dll - Ryujinx.d... x

Ryujinx (1.0.4522.0, x64, .NETCoreApp v3.1) - Ryujinx (1.0.6894.0, x64, .NETCoreApp v5.0)

- References
- Resources
- Ryujinx
 - Program
 - Base types
 - Inheritors
 - Exit():void
 - Glib_UnhandledException(UnhandledExceptionArgs e):void
 - Main(string[] args):void**
 - PrintSystemInfo():void
 - ProcessUnhandledException(Exception ex, bool isTerminating):void
 - XInitThreads():int
 - ConfigurationPath:string
 - Version:string
 - WindowScaleFactor:double
 - Ryujinx.Configuration
 - Ryujinx.Configuration.System
 - Ryujinx.Configuration.Ui
 - Ryujinx.Input.GTK3
 - Ryujinx.Modules
 - Ryujinx.Ui
 - Ryujinx.Ui.App
 - Ryujinx.Ui.Applet
 - Ryujinx.Ui.Helper
 - Ryujinx.Ui.Widgets
 - Ryujinx.Ui.Windows

Side-by-side viewer | Do not ignore | Highlight words | 25 differences

```
namespace Ryujinx
{
    internal class Program
    {
        public static string Version { get;
        public static string ConfigurationPa
        private static void Main(string[] ar
        {
            Toolkit.Init(new ToolkitOptions()
            {
                Backend = (PlatformBackend) 1,
                EnableHighResolution = true
            });
            Program.Version = Assembly.GetEntri
            Console.Title = "Ryujinx Console "
            string environmentVariable = Envir
            Environment.SetEnvironmentVariable
            // ISSUE: method pointer
            ExceptionManager.UnhandledExceptio
            ConfigurationState.Initialize();
            LoggerModule.Initialize();
            DiscordIntegrationModule.Initializ
            string path1 = Path.Combine(AppDor
            string str = Path.Combine(Environr
            string path2 = Path.Combine(str, '
            internal class Program
            {
                public static double WindowScaleFacto
                public static string Version { get;
                public static string ConfigurationP
                private static extern int XInitThre
                private static void Main(string[] a
                {
                    string path1 = (string) null;
                    string str1 = (string) null;
                    bool flag1 = false;
                    for (int index = 0; index < args.Le
                    {
                        string str2 = args[index];
                        if (str2 == "-r" || str2 == "--
                        {
                            if (index + 1 >= args.Length)
                            {
                                Logger.Log? error = Logger.
                                ref Logger.Log? local = ref
                                if (local.HasValue)
                                {
                                    Logger.Log valueOrDefault =
```



















Алгоритм построения дерева сравнения сборок

1. Загрузка сборок и создание корневого узла дерева
2. Поиск «точек входа» и создание первого уровня дерева: поддеревья зависимостей, ресурсов и типов сборки
3. Три группы сущностей для каждого уровня дерева: узлы, презентеры и провайдеры. Провайдер $n-1$ -го уровня порождает узлы n -го уровня
4. Презентеры составляют отображение конкретного узла

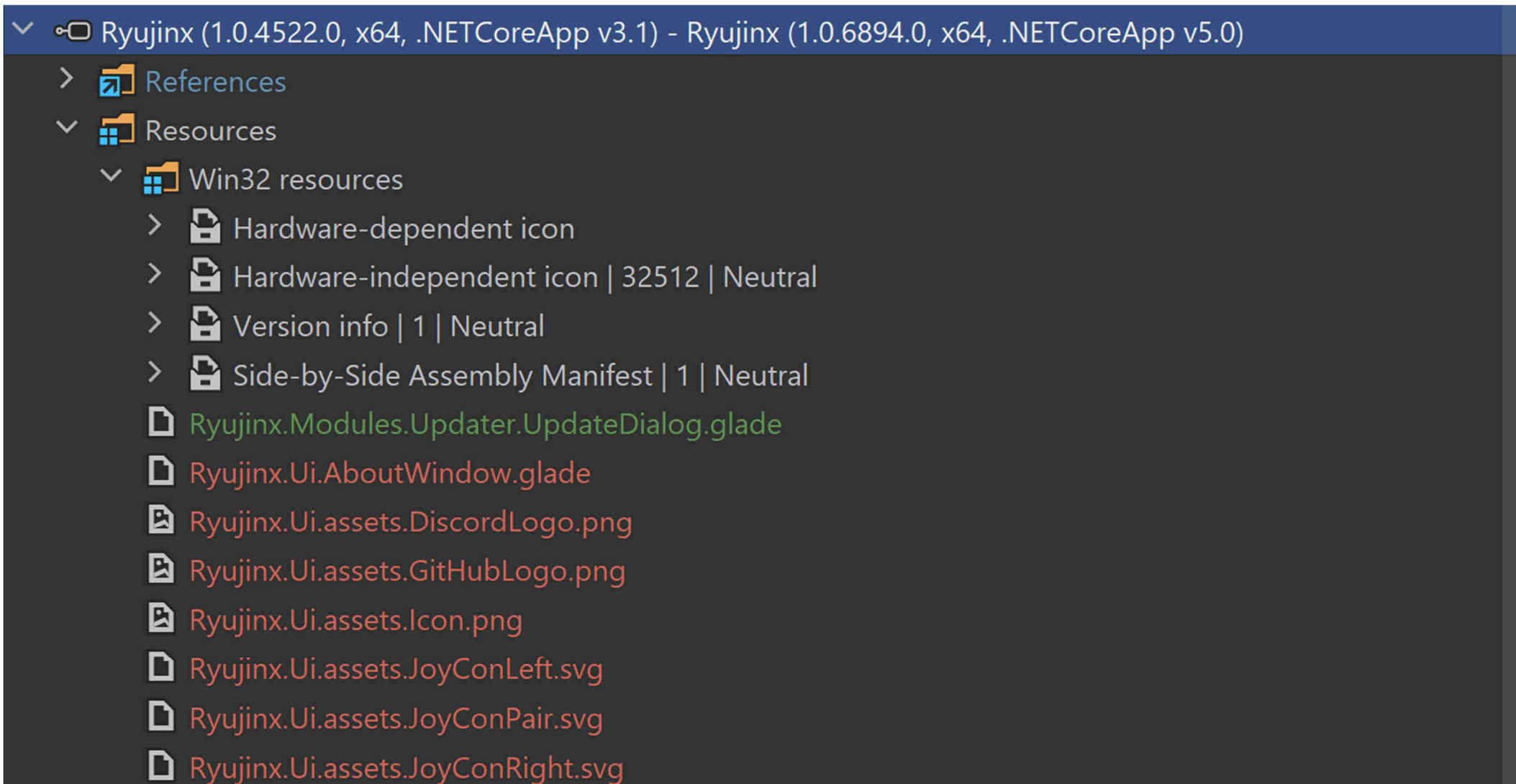
Поддерево зависимостей

▼  RyuJinx (1.0.4522.0, x64, .NETCoreApp v3.1) - RyuJinx (1.0.6894.0, x64, .NETCoreApp v5.0)

▼  References

-  ARMeilleure (1.0.6894.0)
-  CairoSharp (3.22.25.128)
-  DiscordRPC (1.0.150.0 / 1.0.175.0)
-  GdkSharp (3.22.25.56 / 3.22.25.128)
-  GLibSharp (3.22.25.56 / 3.22.25.128)
-  GLWidget (1.0.2.0)
-  GtkSharp (3.22.25.56 / 3.22.25.128)
-  ICSharpCode.SharpZipLib (1.3.0.8)
-  LibHac (0.11.1.0 / 0.12.0.0)
-  Microsoft.Win32.Primitives (4.1.2.0)
-  Mono.Posix.NETStandard (1.0.0.0)
-  Newtonsoft.Json (12.0.0.0)
-  OpenTK (1.0.5.12)
-  OpenTK.Core (4.0.0.0)
-  OpenTK.Graphics (4.0.0.0)
-  OpenTK.Mathematics (4.0.0.0)
-  PangoSharp (3.22.25.56 / 3.22.25.128)
-  RyuJinx.Audio (1.0.4522.0 / 1.0.6894.0)

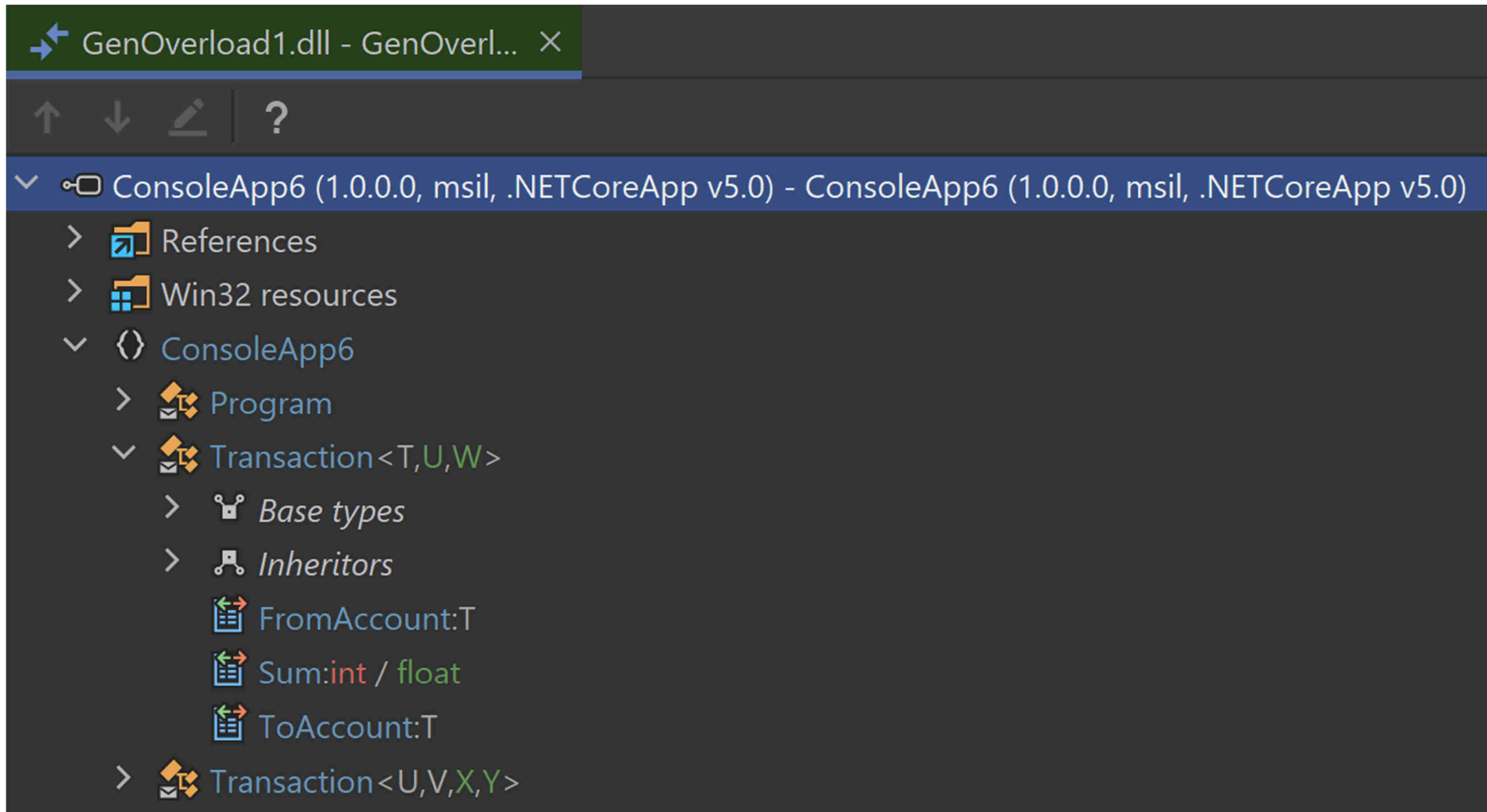
Поддерево ресурсов



Поддерево типов сборки – алгоритм

1. Поиск пространств имен и группировка по CLR QualifiedName, быстрая и полная проверка для определения состояния узла
2. Построение «детей» для каждого пространства имен:
 - a) Быстрая проверка – сравнение сигнатур
 - b) Полная проверка – рендер IL кода
3. На следующих уровнях аналогично быстрая и полная проверка, в случае наличия перегруженных типов – применяется алгоритм семантического сравнения и сопоставления .NET-типов
4. Также применяется алгоритм вычисления разницы сигнатур

Поддерево типов сборки – пример



Алгоритм семантического сравнения .NET-типов

```
class OverloadListing
{
    public void Add(int id, int age)
    {
        // Method body
    }
    // Other methods ...
    public void Add(string name, string email)
    {
        // Method body
    }
}
```

Старая сборка

```
class OverloadListing
{
    public void Add(string name, string[] emails)
    {
        // Method body
    }
    // Other methods ...
    public void Add(int id, float age)
    {
        // Method body
    }
}
```

Новая сборка

Алгоритм семантического сравнения .NET-типов

- Для типов, которые могут быть перегружены
- Инвариант: методы с разными сигнатурами и одинаковым именем не считаются разными
- Вводится метрика, определяющая близость сигнатур

```
public int CompareByMostCommonParams(Element first,
    Element second)
{
    //...
    if (first.IsStatic == second.IsStatic)
        similarityCounter++;

    if (first.IsVirtual == second.IsVirtual)
        similarityCounter++;

    if (first.IsOverride == second.IsOverride)
        similarityCounter++;

    //other keywords checks (sealed, abstract, etc.)
}
```

- Применяется жадный алгоритм: сортировка пар и поиск наиболее близкой

- Сервис YouTrack для апробации и получения обратной связи
- EAP пользователи и разработчики
- Исправлены недочеты, например, некорректное отображение статуса узла при определенных сценариях работы

Результаты

1. Проведен обзор различных видов .NET сборок, структуры и содержания .NET сборки
2. Проведен обзор существующих решений, позволяющих сравнивать .NET сборки
3. Интегрирован в Rider инструмент для сравнения .NET сборок в виде дерева сравнения сборок
4. Проведена апробация решения с целью получения обратной связи от пользователей

Дальнейший вектор работы:

1. Расширить сравнение ресурсов сборки: добавить возможность просмотра разницы содержимого ресурсов
2. Интегрировать инструмент сравнения сборок в dotPeek и ReSharper