# Постановка задачи

## VBA

## Терминология

# Основные требования к требуемому программному продукту

1. Возможность создания расширений для конечных пользователей.
2. Простота использования.  
   Не требует SDK и другого ПО для создания расширений. Вся работа происходит в среде разработки (IDE).
3. Возможность работы как под x86, так и под x64 архитектурами.
4. Возможность отладки расширения.
5. Доступ расширения к объектам расширяемого приложения, реакция на его события.  
   При изменении удалении или добавлении объекта в хост приложении, расширение должно автоматически обновлять информацию о своем окружении.
6. Доступ к библиотеке классов .NET Framework.
7. Поддержка большого числа расширений и взаимодействие их друг с другом.
8. Возможность разрешения зависимостей между расширениями.
9. Удобные инструменты для написания кода (Аналогично IntelliSense).
10. Возможность сохранения расширений по усмотрению пользователя в базу данных, архив, папку и т. д.
11. Динамическая загрузка и выгрузка расширений из адресного пространства хост приложения.
12. Компиляция и перезагрузка расширения «на лету» (Не требуется перезапуск хост приложения)
13. …

# Обзор средств для построения расширяемых приложений на платформе .NET

## Плагины

[http://en.wikipedia.org/wiki/Plug-in\_(computing)#Purpose\_and\_examples](http://en.wikipedia.org/wiki/Plug-in_(computing)#_blank)

### Существующие разработки для .NET

Крупные и средние проекты

1. AL Platform (<http://www.agile-sys.com/productsA.php>)
2. MEF (<http://mef.codeplex.com/>)
3. Plux.NET (<http://ase.jku.at/plux/index.html>)
4. Mono.Addins (<http://monoaddins.codeplex.com/>)
5. Compact Plugs & Compact Injection (<http://compactplugs.codeplex.com/>)

Маленькие, учебные проекты, тестовые примеры

1. .NET Based Add-in/Plug-in Framework with Dynamic Toolbars and Menus (<http://www.codeproject.com/KB/macros/Net_AddinProjFrmwork.aspx>)
2. AddIn Enabled Applications (<http://www.codeproject.com/KB/dotnet/AddInModel.aspx>)

Отдельно стоит упомянуть

1. SDA (<http://laputa.sharpdevelop.net/AnnouncingSharpDevelopForApplicationsSDA.aspx>)  
   SDA (SharpDevelop for applications), Набор инструментов, позволяющий построить собственное приложение на основе ядра SharpDevelop. Это приложение может получить «в наследство» многие возможности самого SharpDevelop, в том числе и возможность написания плагинов. (SharpDevelop Addins)

Все вышеперечисленные инструменты, платформы и библиотеки не решают поставленную задачу, а именно предоставление создания расширений *конечному пользователю* приложения. Их основная цель – дать разработчику ПО возможность сделать свое приложение более гибким, но в то же время не перегруженным функционалом, который для многих пользователей окажется попросту не нужным. Яркий пример подобных приложений – интернет браузеры Mozilla Firefox и Google Chrome. Изначально довольно аскетичные, они могут при помощи плагинов получать функционал, удовлетворяющий даже самого требовательного пользователя.

## Разработки, позволяющие встраивать в приложение возможность расширения базового функционала конечными пользователями

1. VSTA ([http://www.microsoft.com/download/en/details.aspx?id=5804#overview](http://www.microsoft.com/download/en/details.aspx?id=5804#_blank))  
   Visual Studio Tools for Applications – мощный набор инструментов, пришедший на замену VBA и имеющий схожий набор возможностей.
2. VSTO ([http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24263#overview](http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24263#_blank))  
   То же, что и VSTA, используется как замена VBA в поздних версиях Microsoft Office.

Рассмотрим эти решения подробнее.

### VSTO

Внутренняя разработка компании Microsoft. Ее использование в сторонних приложениях невозможно. Для этих целей существует второй проект – VSTA. Подробно рассматривать возможности VSTO в рамках этой работы не имеет смысла.

### VSTA

Поначалу именно VSTA была выбрана, как замена VBA.

Была разработана методика внедрения инструментов VSTA в существующее приложение. В рамках работы были решены проблемы взаимодействия VSTA IDE и хост приложения, такие как программное управление, реакция на события, реализация отладки расширения. Так же была решены проблема использования механизмов VSTA без изменения объектной модели и архитектуры хост приложения.

В конце концов, Microsoft отказалась лицензировать VSTA, и закрыло этот проект для сторонних фирм-разработчиков ПО.

## Технологии, библиотеки и инструменты, позволяющие реализовать расширяемое приложение

1. System.Addin
2. .NET Reflection / Remoting
3. Mono Cecil (?)
4. Iron Python (?)

## Плагины или скрипты?( <http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA>)

Для написания пользовательских расширений могут использоваться как скрипты (в терминологии некоторых программ «[макросы](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%80%D0%BE%D1%81)»), так и [плагины](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) (независимые модули, написанные на [компилируемых языках](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F); в некоторых программах они могут называться «утилитами», «экспортёрами», «драйверами»).

Скриптовый язык удобен в следующих случаях:

1. Если нужно обеспечить [программируемость](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) без риска дестабилизировать систему. Так как, в отличие от [плагинов](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD), скрипты интерпретируются, а не компилируются, неправильно написанный скрипт выведет диагностическое сообщение, а не приведёт к системному краху;
2. Если важен выразительный код. Во-первых, чем сложнее система, тем больше кода приходится писать «потому, что это нужно» — см., например, *[Hello World#Маргинальные примеры](http://ru.wikipedia.org/wiki/Hello_World" \l ".D0.9C.D0.B0.D1.80.D0.B3.D0.B8.D0.BD.D0.B0.D0.BB.D1.8C.D0.BD.D1.8B.D0.B5_.D0.BF.D1.80.D0.B8.D0.BC.D0.B5.D1.80.D1.8B" \o "Hello World)*. Во-вторых, в скриптовом языке может быть совсем другая [концепция программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), чем в основной программе — например, [игра](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0" \o "Компьютерная игра)может быть монолитным однопоточным приложением, в то время как управляющие персонажами скрипты выполняются параллельно. В-третьих, скриптовый язык имеет собственный проблемно-ориентированный набор команд, и одна строка скрипта может делать то же, что несколько десятков [строк](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D1%81%D1%82%D1%80%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%B0) на традиционном языке. Как следствие, на скриптовом языке может писать программист очень низкой квалификации — например, [геймдизайнер](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%B9%D0%BC%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD%D0%B5%D1%80" \o "Геймдизайнер) своими руками, не полагаясь на программистов, может корректировать правила игры;
3. Если требуется [кроссплатформенность](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Хорошим примером является [JavaScript](http://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript) — его исполняют браузеры под самыми разными ОС.

У плагинов же есть три важных преимущества.

1. Готовые программы, оттранслированные в машинный код, выполняются значительно быстрее скриптов, которые [интерпретируются](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80) из [исходного кода](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) динамически при каждом исполнении. Поэтому скриптовые языки не применяются для написания программ, требующих оптимальности и быстроты исполнения. Но из-за простоты они часто применяются для написания небольших, одноразовых («проблемных») программ.
2. Полный доступ к любому аппаратному обеспечению или ресурсу ОС (в скриптовом языке для этого должен существовать написанный на машинном коде [API](http://ru.wikipedia.org/wiki/API)). Плагины, работающие с аппаратным обеспечением, традиционно называют [драйверами](http://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80).
3. Если предполагается интенсивный обмен данными между основной программой и пользовательским расширением, для плагина его обеспечить проще.

Также в плане быстродействия скриптовые языки можно разделить на языки *динамического разбора* ([sh](http://ru.wikipedia.org/wiki/Sh" \o "Sh), [command.com](http://ru.wikipedia.org/wiki/Command.com)) и *предварительно компилируемые* ([Perl](http://ru.wikipedia.org/wiki/Perl" \o "Perl)). Языки динамического разбора считывают инструкции из файла программы минимально требующимися блоками, и исполняют эти блоки, не читая дальнейший код. Предкомпилируемые языки транслируют всю программу в [байт-код](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) и затем исполняют его. Некоторые скриптовые языки имеют возможность компиляции программы «на лету» в [машинный код](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (т. н. [JIT-компиляция](http://ru.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F)).