* **~~Закончить переделывание интерефейса утилиты ScanX, под нужды Searcher-a.~~**
* ~~Доработать тесты Engine (после UI)~~
* ~~Доработать события Engine(после UI)~~
* ~~Переделать appContext под спринги~~
* Запуск тестов в режиме изоляции(=юнит тестирование).
* Закинуть в папку docs диаграммы компонентов и классов.
* ~~Выделить вм-ки в отдельную библиотеку~~
* 1)      Манифест требовал запуска под админом
* 2)      Сканирование выводит невнятное сообщение об ошибке и ничего более не происходит.
* 3)      В целом проект сырой. Комментариев практически нет.
* 4)      Трудно отделить собственный код от заимствованного UI-приложения ScanX.
* 5)      UI не вычищен от старого кода. Оставлено много мёртвого кода.
* 6)      Реализация рекурсий не продумана.
* 7)      Дизайн плох в целом и в куче частностей.
* 8)      Изобретен велосипед в виде весьма слабого IoC-контейнера (static AppContext, IObjectFactory)
* 9)      Знание WPF неуд. При таком подходе проще было консольную программу написать.
* 10)   Знание MVVM на 2+ (наблюдается один ObservableCollection, и тот не для результатов сканирования)
* 11)   Сборки из каталога Plugins загружаются по маске \*.\* (ошибки загрузки «левых» файлов заглушены пустым catch)
* 12)   7 вхождений слова aBaTaPbl4
* 13)   Поиск по маске выполнен через GetFileName(fileName).Contains(pattern)
* 14)   16 тестов на всё
* 15)   Класс FileSystem содержит кучу несвязных методов, в том числе для поддержки Xml и Assembly, месиво
* 16)   Многопоточное сканирование реализовано через ThreadPool.QueueUserWorkItem, который нигде не конфигурируется. Более того предварительно строится список всех каталогов рекурсивно в основном потоке.
* Клонировать репозитарий с убунты. Вычистить ненужные файлы упаковать и передать