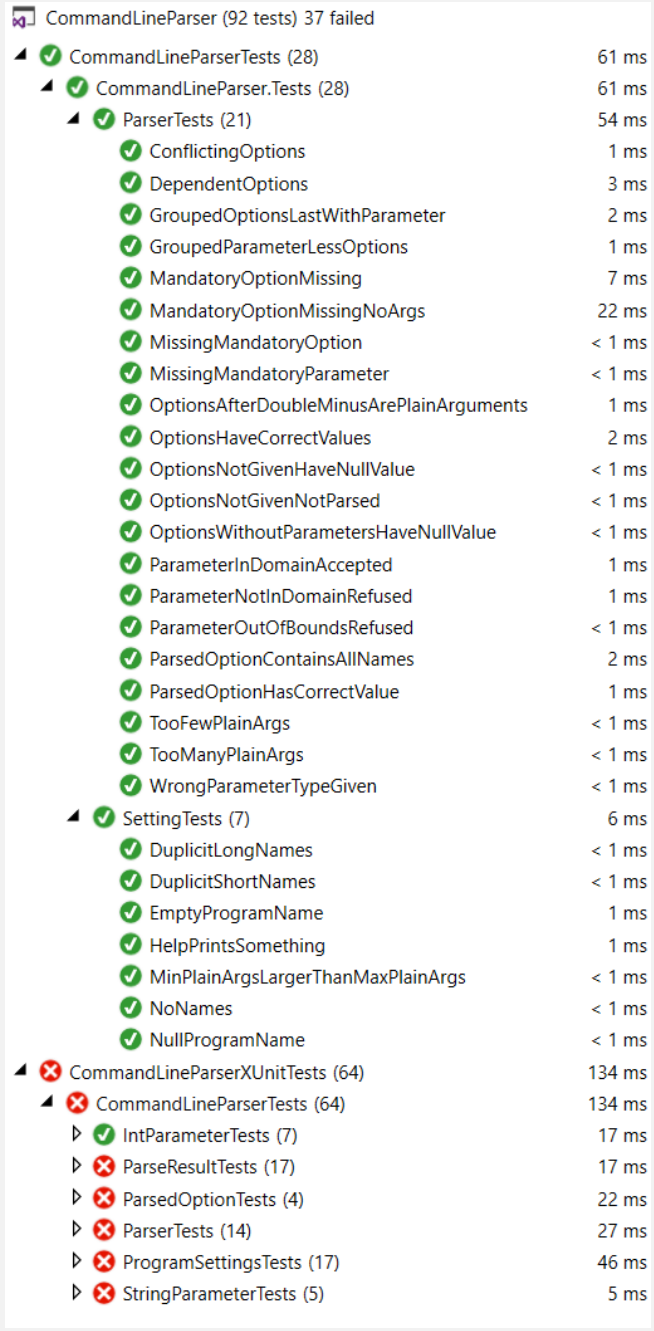
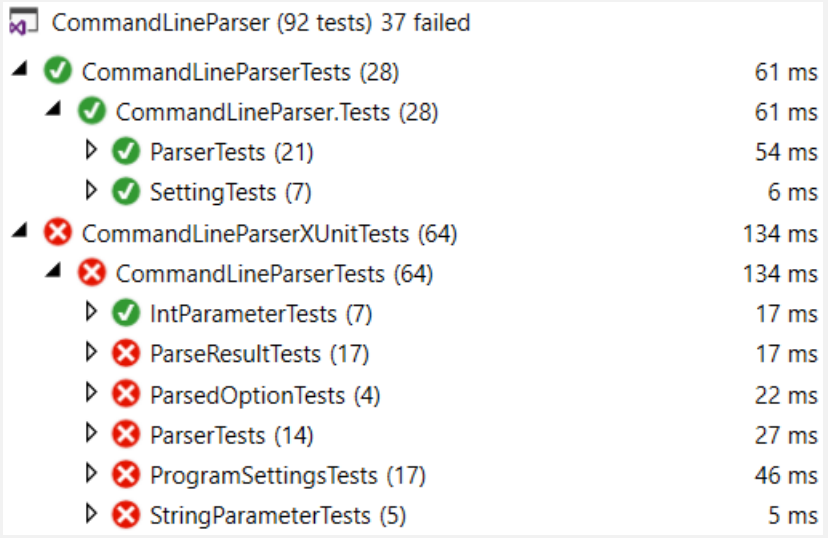
COMMAND LINE PARSER

PROGRESS PRESENTATION

29 April 2019



- MSTest Test Project and xUnit Test Project



```
public void PrintHelp(System.IO.TextWriter writer)
{
    printHelpHeader(writer);
    writer.WriteLine();

    printGNUOptions(writer);
    writer.WriteLine();

    printGNUStandardOptions(writer);
    writer.WriteLine();

    //TODO documentation for plain arguments

    writer.Flush();
}
```

```
private void printHelpHeader(TextWriter writer)
{
    writer.WriteLine(indentSpace + programName + " [options]
}

private void printGNUStandardOptions(TextWriter writer)
{
    writer.WriteLine(indentSpace + "GNU Standard Options");

    writer.WriteLine(indentSpace + indentSpace + "--help Pri
    writer.WriteLine(indentSpace + indentSpace + "- V, --ver
    writer.WriteLine(indentSpace + indentSpace + "Print vers
    writer.WriteLine(indentSpace + indentSpace + "--Terminat
}

private void printGNUOptions(TextWriter writer)
{
    writer.WriteLine(indentSpace + "GNU Options");

    foreach (Option opt in Options)
    {
        printOption(writer, opt);
    }
}
```

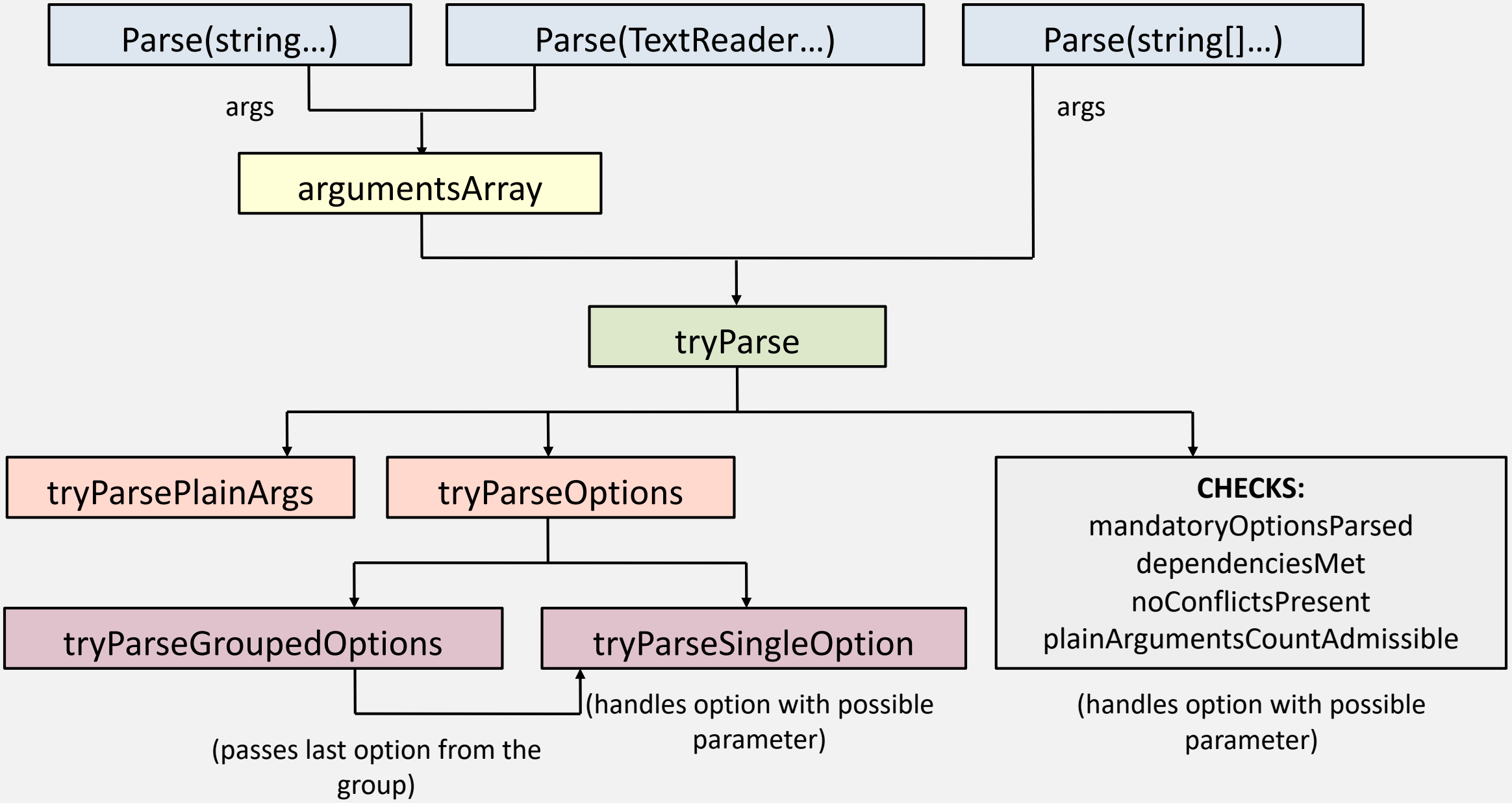
C:\Program Files\dotnet\dotnet.exe

```
time [options] command [arguments...]

GNU Options
    -f FORMAT, --format=FORMAT
        Specify output format, possibly overriding the format specified in the
    -p, --portability
        Use the portable output format.
    -o FILE, --output=FILE
        Do not send the results to stderr, but overwrite the specified file.
    -a, --append
        (Used together with -o.) Do not overwrite but append.
    -v, --verbose
        Give very verbose output about all the program knows about.
    -V, --version
        Print version information on standard output, then exit successfully.
    --help
        Print a usage message on standard output and exit successfully.

GNU Standard Options
    --help Print a usage message on standard output and exit successfully.
    - V, --version
    Print version information on standard output, then exit successfully.
    --Terminate option list.
```

```
internal List<Option> MandatoryOptions = new List<Option>();
internal Dictionary<string, Option> OptionsDictionary = new Dictionary<string, Option>();
internal Dictionary<Option, Option> OptionDependencies = new Dictionary<Option, Option>();
internal List<List<Option>> OptionConflicts = new List<List<Option>>();
```


Parse(string…)
Parse(TextReader…)
Parse(string[]…)
args
args
argumentsArray
tryParse
tryParsePlainArgs
tryParseOptions
CHECKS:
mandatoryOptionsParsed
dependenciesMet
noConflictsPresent
plainArgumentsCountAdmissible
tryParseGroupedOptions
tryParseSingleOption
(handles option with possible parameter)
(handles option with possible parameter)
(passes last option from the group)

```
Regex singleOptionRegex = new Regex(@"^(\-{1}(?<optionName>[A-Za-z]{1})|\-{2}(?<optionName>[A-Za-z]{2,}))(={1}(?<paramString>[0-9]+)|(?<paramString>[0-9]*))$");

var singleOptionMatch = singleOptionRegex.Match(args[argsIterator])

string optionName;

string paramString;

if (singleOptionMatch.Success) {

 optionName = singleOptionMatch.Groups["optionName"].ToString();

 paramString = singleOptionMatch.Groups["parameterString"].ToString();

}
```