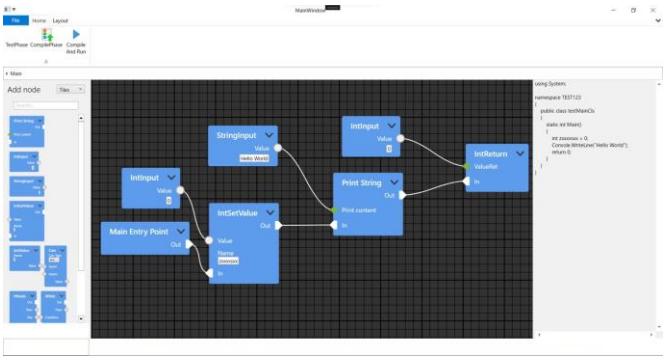
Node-GraphとRoslynを活用した
プログラミングアプリケーションの開発

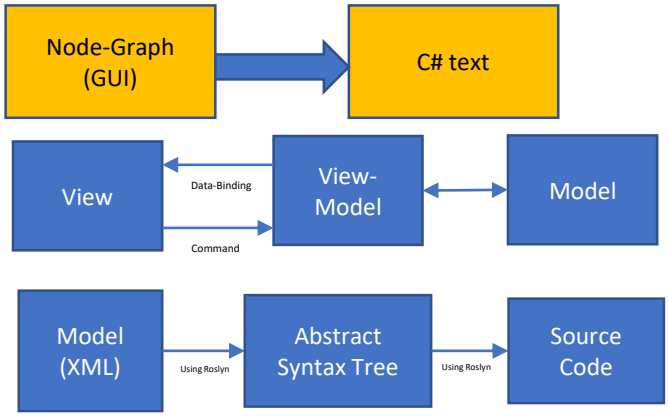
福島県立福島高等学校2年　　根本 晃希
板橋 勇慶

1.概要

このソフトウェアでは、Node-Graphを用いて
プログラミングをすることが可能である。
また、拡張性を持たせることで、
幅広いアプリケーションが作れるように工夫
している。



2.システム構成



UIアーキテクチャはMVVMを用いた。
また、Nodeをソースコードに変換する過程では、Roslynライブラリを使用。
このライブラリはソースコードとASTの相互変換が可能である。
アプリケーション側でNodeをASTに変換。
それをRoslynでソースコードに変換することで、変換時の文法ミスを軽減できる。
このような構成により拡張性と安定性の
向上が実現している。

このように、テンプレート化できるところ
が存在する。
そこで、Roslynを使用することで簡略化を
可能にし、実装時のコスト削減に成功した。

```
[ModelAyano]
private IfNodeModel _model = new IfNodeModel();
```

選択箇所がマーカー部分

```
using System.Reactive;
using System.Reactive.Linq;
using System;
namespace [NSName]
{
    /// <summary>
    /// [CLSName] のViewModel簡略化用ロジック
    /// </summary>
    public partial class [CLSName]
    {
        string XMLSymkun=XMLModelSymbol.ToDisplayString();
        string XMLNodeXYSymbolkun = XMLNodeXYSymbol.ToDisplayString();
        foreach (IFieldSymbol symf in fields)
        {
            ProcField(src, symf, XMLSymkun, CLSName, XMLNodeXYSymbolkun);
        }
        src.Append("] ]");
        return src.ToString();
    }
    [ 個の参照
    private void ProcField(StringBuilder src, IFieldSymbol fSymbol, string XMLModelSymbol, string CLSName, string XMLNodeXYName)
    {
        string fName=fSymbol.Name;
        [TypeSymbol fType = fSymbol.Type;
        if (fName == "model") return;
        src.Append("$@"
    }
    /// <summary>
    /// Model
```

ジェネレーター(抜粋)

3.実装の工夫

```
this._model.TYPE = typeof(IfNodeViewModel).AssemblyQualifiedName;
this._model.MODELTYPE = this._model.GetType().AssemblyQualifiedName;
this.UUIDChanged.Subscribe(newvalue =>
{
    _model.UUID = newvalue;
});
this.NameChanged.Subscribe(newvalue =>
{
    _model.Name = newvalue;
});
this.PositionChanged.Subscribe(newvalue =>
{
    _model.Points = new FatNoder.Serializer.Node.Xml.XMLNodeXY()
    {
        X = newvalue.X,
        Y = newvalue.Y
    };
});
```

```
this._model.TYPE = typeof(ForNodeViewModel).AssemblyQualifiedName;
this._model.MODELTYPE = this._model.GetType().AssemblyQualifiedName;
this.UUIDChanged.Subscribe(newvalue =>
{
    _model.UUID = newvalue;
});
this.NameChanged.Subscribe(newvalue =>
{
    _model.Name = newvalue;
});
this.PositionChanged.Subscribe(newvalue =>
{
    _model.Points = new FatNoder.Serializer.Node.Xml.XMLNodeXY()
    {
        X = newvalue.X,
        Y = newvalue.Y
    };
});
```

4.今後の展望

- プラグインの追加
- アプリケーションの埋め込み
- スマホ版の実装
- バグ修正