VisualStudio2010 β1 を使ってみた

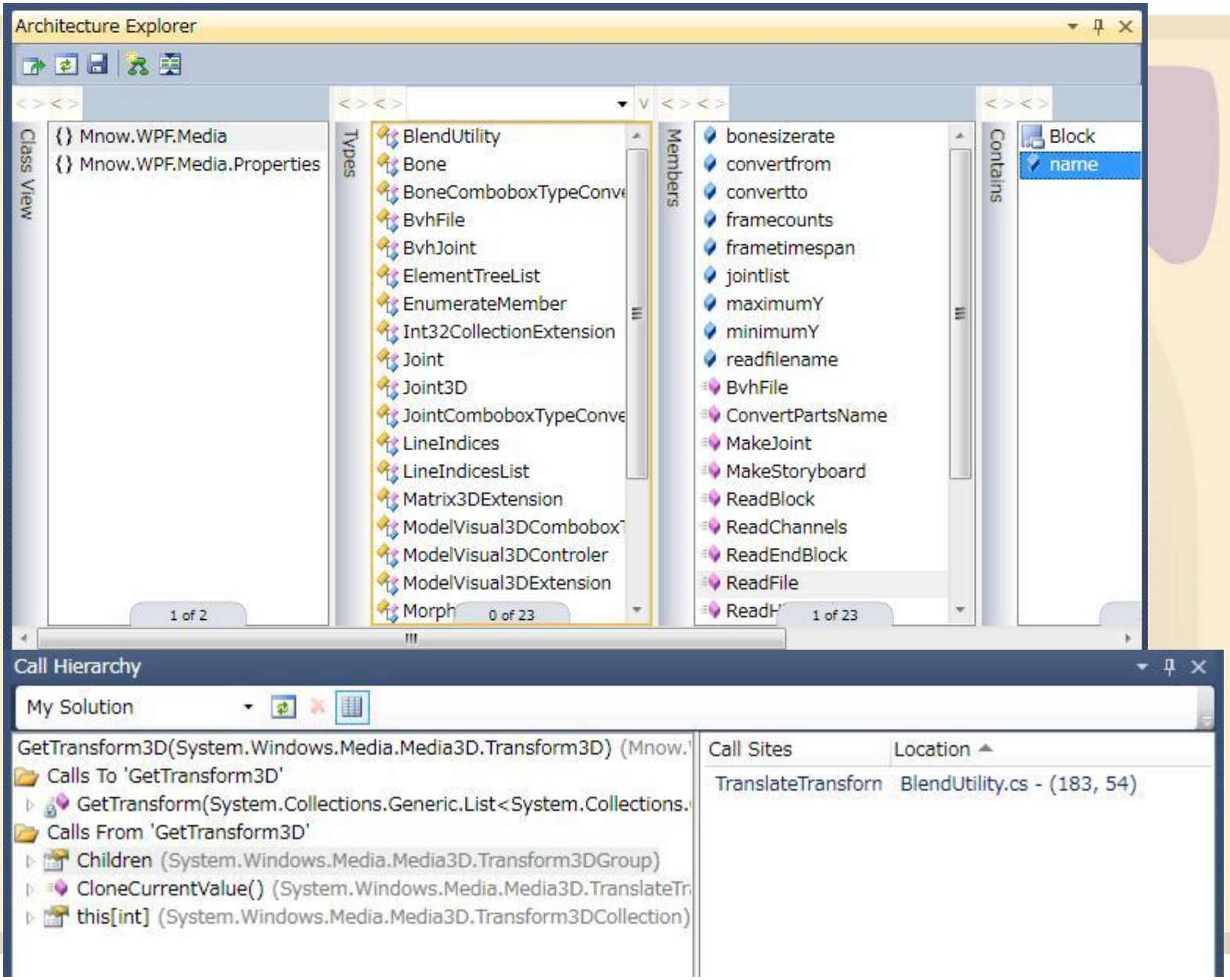
えムナウ（児玉宏之）
http://mnow.jp/
http://mnow.wankuma.com/
http://blogs.wankuma.com/mnow/
http://www.ailight.jp/blog/mnow/

アジェンダ

- 開発環境
- 言語
- フレームワーク
- パラレル
- UML
- まとめ

開発環境

- ドキュメント ウィンドウ
  - IDEの外部にフローティング
  - CTRL+ホイールで拡大縮小
  - 呼び出し階層の表示(Call From/To)
  - アーキテクチャーエクスプローラー
  - オブジェクト ブラウザで .NET Frameworkを検索
  - 検索単語を選択して、編集-移動で検索し移動
  - 画面やスコープ内の選択した変数やクラスを強調表示
  - VBでは If - End If などのキーワードペアを強調表示
  - 未定義クラス・メンバーからクラス・メンバーを生成
  - IntelliSense 使う、参考に表示する、の切り替え


Architecture Explorer
Class View
{} Mnow.WPF.Media
{} Mnow.WPF.Media.Properties
1 of 2
Types
BlendUtility
Bone
BoneComboboxTypeConve
BvhFile
BvhJoint
ElementTreeList
EnumerateMember
Int32CollectionExtension
Joint
Joint3D
JointComboboxTypeConve
LineIndices
LineIndicesList
Matrix3DExtension
ModelVisual3DControler
ModelVisual3DExtension
0 of 23
Members
bonesizerate
convertfrom
convertto
framecounts
frametimespan
jointlist
maximumY
minimumY
readfilename
BvhFile
ConvertPartsName
MakeJoint
MakeStoryboard
ReadBlock
ReadChannels
ReadEndBlock
ReadFile
1 of 23
Contains
Block
name
Call Hierarchy
My Solution
GetTransform3D(System.Windows.Media.Media3D.Transform3D)
Calls To 'GetTransform3D'
GetTransform(System.Collections.Generic.List<System.Collections.
Calls From 'GetTransform3D'
Children (System.Windows.Media.Media3D.Transform3DGroup)
CloneCurrentValue() (System.Windows.Media.Media3D.TranslateTr
this[int] (System.Windows.Media.Media3D.Transform3DCollection)
Call Sites
Location
TranslateTransform
BlendUtility.cs - (183, 54)

開発環境

- デバッグ
  - ブレークポイント一覧の検索機能
  - ブレークポイントのXML保存と読み出し
  - スレッド ウィンドウで各スレッドのスタック確認
  - Parallel Tasks/Stacks で並列動作の確認
  - DataTips ソース上に付箋紙みたいに変数ウォッチを貼る
  - デバッグ中にダンプファイル出力
  - パフォーマンス分析のマルチCPUの分析結果
  - デバッグ履歴機能

フレームワーク

- テスト
  - 手動テストの記録と再生
    - コード化されたUIテストビルダーで操作した内容が記憶されメソッドを作成でメソッドを自動生成
    - アサーションの追加でどのコントロールのどのパラメータをチェックするか Spy みたいな UI で指定可能
    - Microsoft.VisualStudio.TestTools.UITesting 名前空間
    - UserControls.cs に使ったコントロール毎に制御するクラスを自動作成
    - RecordedMethods.cs に操作がメソッド呼び出しの形で記録
  - 自動の UI テスト
  - 仮想マシン環境での本番環境に近いテスト

言語

- C#
  - Dynamic
  - 名前付き引数、省略可能な引数
  - Office 相互運用機能
  - type equivalence interface project （異なるバージョンの Microsoft Office を使用)
  - ジェネリックの共変性と反変性(delegateは.NET Framework 2.0)

言語

- VB
  - 自動実装プロパティ
  - コレクション初期化子
  - 複数行にまたがるステートメント
  - 複数行のラムダ式
  - type equivalence interface project (異なるバージョンの Microsoft Office を使用)
  - Dynamic
  - ジェネリックの共変性と反変性

言語

- VB
  - 複数行にまたがるステートメント
    - カンマ “,” の後。
    - “(“ の後、”)” の前。
    - “{“ の後、”}” の前。
    - XML リテラルの中の “<%=“ の後、”%>” の前。
    - 文字列結合演算子の “&” の後。
    - 代入演算子（=, &=, +=, <<= など）の後。
    - 二項演算子（+, ?, Mod, <, <=, And, AndAlso など）の後。
    - “Is” と “IsNot” の後。
    - メンバ名の “.” の後、メンバ名の前。ただし、With ステートメントや初期化リストの中では “_” が必要とかなんとか。
    - XML リテラルの中の “.”、”.@”、”…” の後。ただし、With キーワードの中では “_” が必要とかなんとか。
    - 属性を示す “<” の後、”>” の前。ただし、アセンブリレベルとモジュールレベルの属性のときは “_” が必要。
    - LINQ の “From”、”Order By”、”Select” などの前後。ただし、”Order By” などを途中で改行してはダメ。
    - For Each ステートメントの In の後。
    - コレクション・イニシャライザの From キーワードの後。

言語

- Visual C++
  - **ユーザー エクスペリエンス**
    - 応答性の高いIntelliSense
    - MSBuild ベースのビルド環境
  - MFC の改善
    - Office 2007 や Windows の Look & Feel
  - 再起動マネージャをサポート
  - Parallel Pattern Library
  - コンパイラ
    - ラムダ式、auto キーワード新関数、decltype 演算子、rvalue への参照宣言 <T&&>、static_assert 宣言

言語

- Visual F#
  - マルチ パラダイム プログラミング言語
    - 関数型プログラミング
    - オブジェクト指向プログラミング
  - 純粋関数型言語で内部状態をもたない
  - 厳密型と型推論、リストやTupleが基本
  - パターンマッチ、ラムダ式
  - 遅延評価がデフォルト＝＞非正格性
  - 関数に引数を一部だけ渡して新たな関数を作る＝＞カリー化
  - 関数をネストにすることで評価順を固定＝＞モナド

フレームワーク

- .NET Framework 4
  - System.Diagnostics.Contracts
    - Debug.Assert から考えて一歩進んだ考え方
    - コンパイル時にワーニングが出る
    - ガード句・戻り値・オブジェクトが利用可能か をチェック
  - クラスライブラリ
    - BigInteger　理論的に値が上限または下限の境界を持たない大きな整数を表す型
    - SortedSet<T>　並べ替えられた順序で管理されているオブジェクトのコレクション
    - Tuple<T1> ~ Tuple<T1,T2,T3,T4,T5,T6,T7,TRest>　種類を明示的に指定しなくても特定の組のオブジェクトをインスタンス化、使い捨てクラスとしても利用可能

フレームワーク

- .NET Framework 4
  - ディレクトリのLinq対応
    - Directory, DirectoryInfo クラスがLinqに対応したので、ファイルやディレクトリの列挙が簡単
  - メモリーマップドファイル
    - 非常に大きなファイルの一部のメモリ マップ ビューを作成・操作
  - 分離ストレージ
    - WebやSilverlightなど制約のある状態でのファイル保存は分離ストレージを利用
    - ファイル システム内で特定のパスを指定せずに、分離ストレージ ファイル システムの部分へ独自にデータを保存

フレームワーク

- ADO.Net Entity Framework
  - POCO（plain old CLR objects)サポート
    - EntityObjectクラス ではなく、通常のクラスにに出力して加工して書き戻す形での変更を管理
  - 遅延読み込み
    - 今までは関連するオブジェクトにアクセスする前にInclude で読みだす必要があったが、DeferredLoadingEnabled=True で不要
  - モデルファースト
    - Entityモデルを先に作成し、それをもとに論理モデルや実際のテーブルを作成
  - モデルブラウザー
    - 概念モデル・論理モデルの構造をツリー上に表示

フレームワーク

- WPF
  - データソースからDrag＆DropでコンテナやコントロールにまでバインドできるWindows Formsより進んだデータバインド
  - WPF Tree visualizer で階層表示(Snoopぽい)
  - DatePicker, DataGrid サポート
- WPFとSilverlight
  - 使いやすいカラーピッカー、データバインド画面

フレームワーク

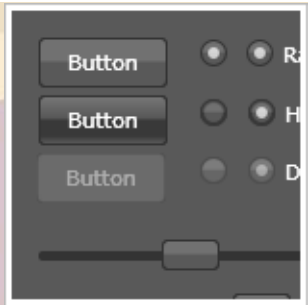
- WPF Toolkit
  - WPF Toolkit March 2009
    - VisualStateManager, WPF Ribbon, WPF Chart
    - Shader Effects BuildTask and Templates
    - WPF Splash Screen Item Templates
    - WPF Model-View-ViewModel Toolkit
  - WPF テーマ
    - Expression Dark, Expression Light, Whistler Blue, Shiny Red, Shiny Blue, Bureau Blue, Twilight Blue, Bubble Creme, Bureau Black

フレームワーク

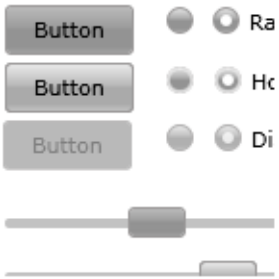
- Silverlight
  - Controlをドラッグドロップして画面を作成
  - データソースはIDE上はオブジェクトとサービスだけ
  - WPFに比べてないもの、DatePicker, DockPanel, DocumentViewer, Expander, Frame, GroupBox, ListView, Menu, Separator, StatusBar, ToolBar系, TreeView, ViewBox, WindowsFormsHost, WrapPanel
  - DataGrid はある。

フレームワーク

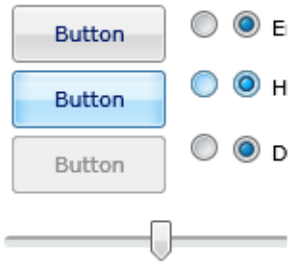
- Silverlight Toolkit
  - Silverlight 2 Toolkit March 2009
  - Silverlight 3 Toolkit March 2009
    - TreeView, DockPanel, WrapPanel, Label, HeaderedContentControl, HeaderedItemsControl, ButtonSpinner, Charting, Expander, ImplicitStyleManager, NumericUpDown, Spinner, UpdownBase, Viewbox, AutoCompleteBox, NumericUpDown, Accordion, DomainUpDown, LayoutTransformer, TimePicker, TimeUpDown, TransitioningContentControl
  - 色々なテーマ
    - Expression Dark, Expression Light, Whistler Blue, Rainier Orange, Rainier Purple, Bureau Black, Shiny Red, Shiny Blue, Bureau Blue, Twilight Blue, Bubble Creme



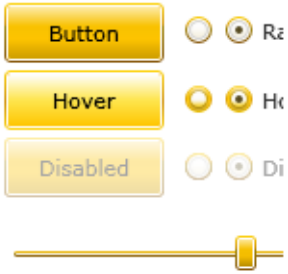
Expression Dark



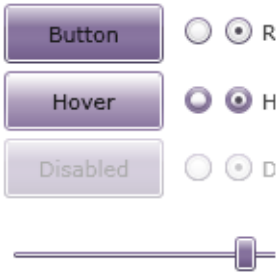
Expression Light



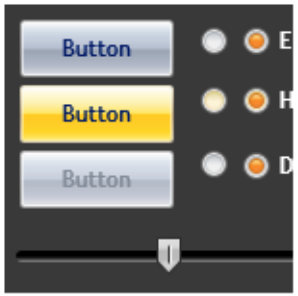
Whistler Blue



Rainier Orange



Rainier Purple

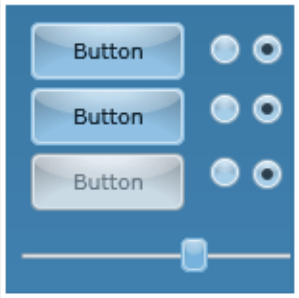


Bureau Black

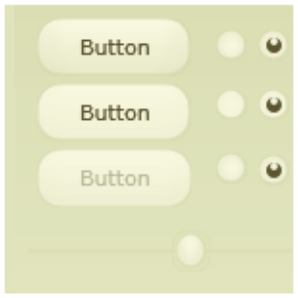
Shiny Red

Shiny Blue

Bureau Blue



Twilight Blue



Bubble Creme

フレームワーク

- Web 開発
  - コード スニペット
  - JScript 大幅に強化された IntelliSense
  - 簡単な Web 配布
- グラフ表示
  - Web と Forms のグラフ表示

パラレル

- Task Parallel Library (TPL)
  - Parallel.For
    - Parallel.For(startIndex, endIndex, (currentIndex) => DoSomeWork(currentIndex));
  - Parallel.ForEach
    - Parallel.ForEach(sourceCollection, item => Process(item));
  - Parallel.Invoke

```
Parallel.Invoke(
        () => DoSomeWork(),
        () => DoSomeOtherWork());
```

パラレル

- Task Parallel Library (TPL)
  - Task<Tresult>

```
var task = Task<int>.Factory.StartNew(()->DoSomeWork());
int i = task.Result;  //ここで終了を待つ
```

  - Task.WaitAll

```
Task[] tasks = new Task[3]  { Task.Factory.StartNew(() => MethodA()),
                              Task.Factory.StartNew(() => MethodB()),
                              Task.Factory.StartNew(() => MethodC()) };
Task.WaitAll(tasks); //ここで終了を待つ
```

  - AggregateException
    - InnerExceptions Task 処理中のすべてのException

パラレル

- Task Parallel Library (TPL)
  - Parallelの中断
    - Exception、ParallelLoopState クラス、Break、Stop、ShouldExitCurrentIteration

```
try
{
    Parallel.For(1, 100000, (i, loopState) =>
    {
        //  if (i == 20) throw new Exception("例外");
        //  if (i == 20) loopState.Break();
        if (i == 20) loopState.Stop();
        if (loopState.ShouldExitCurrentIteration) return;
        Console.WriteLine(i);
    });
}
catch (AggregateException e)
{
    Console.WriteLine(e);
}
```

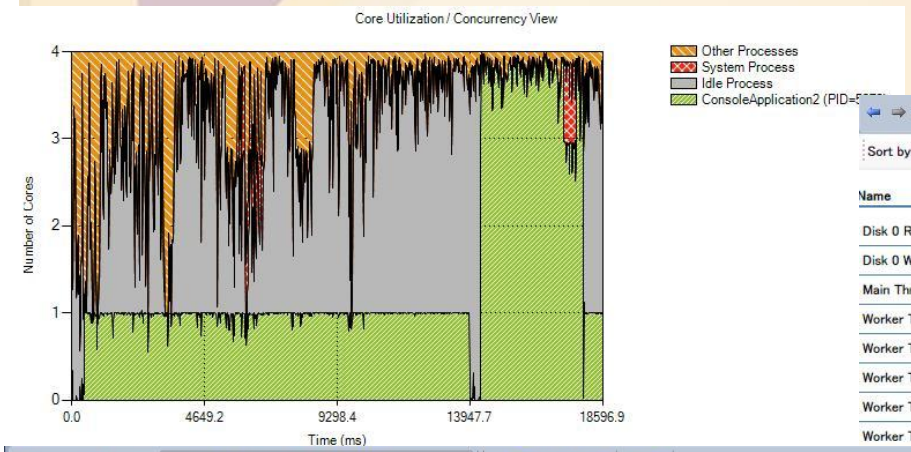
パラレル

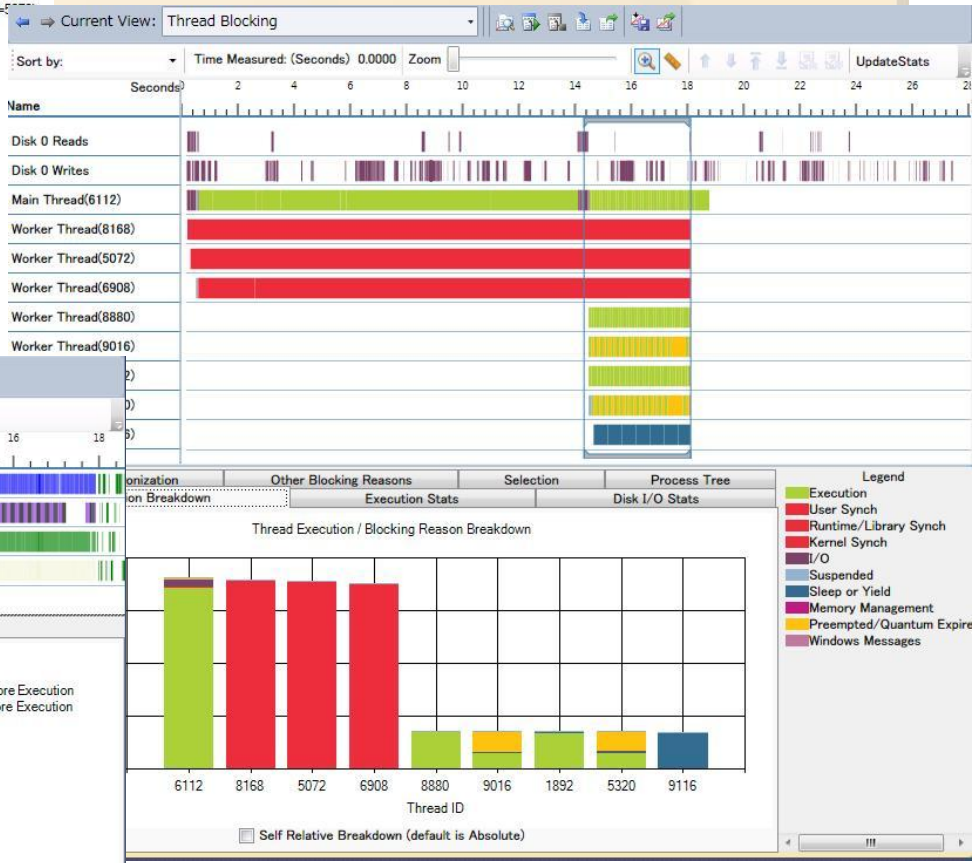
- PLINQ
  - AsParallel() パラレル化、AsOrdered() 昇順化
  - ParallelEnumerable

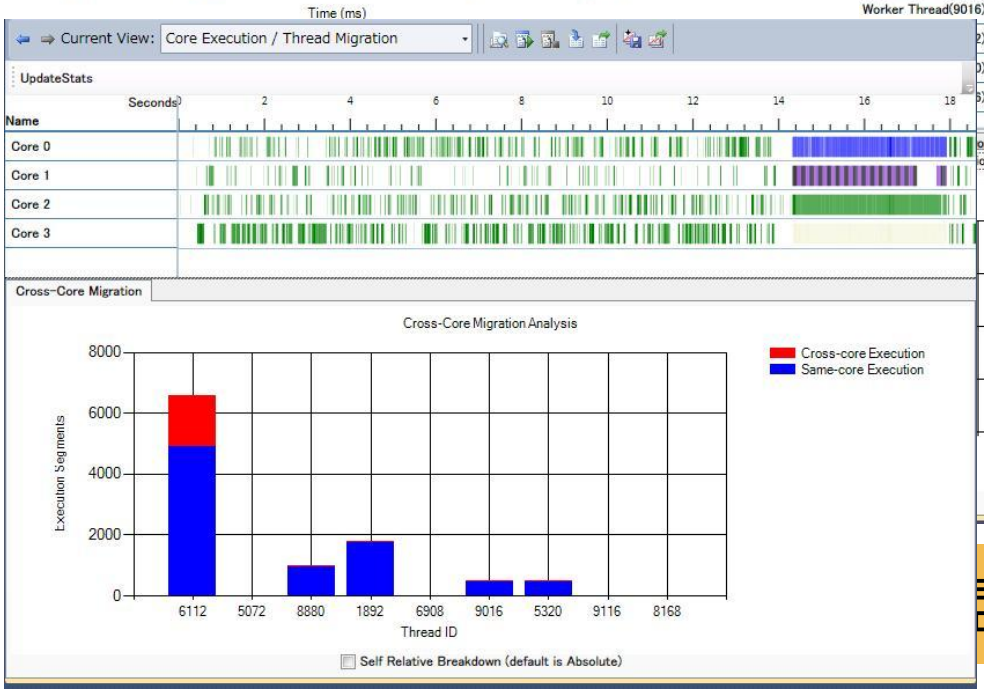
```
var q = from num in Enumerable.Range(10, 100).AsParallel()
        where num % 10 == 0
        select num;
q.ForAll((i) => Console.WriteLine(i)); Console.WriteLine();

var q2 = from num in ParallelEnumerable.Range(10, 100).AsOrdered()
        where num % 10 == 0
        select num;
foreach(int i in q2)
{
    Console.WriteLine(i);
}
```

パラレル

- パフォーマンスウィザードでパラレルの分析





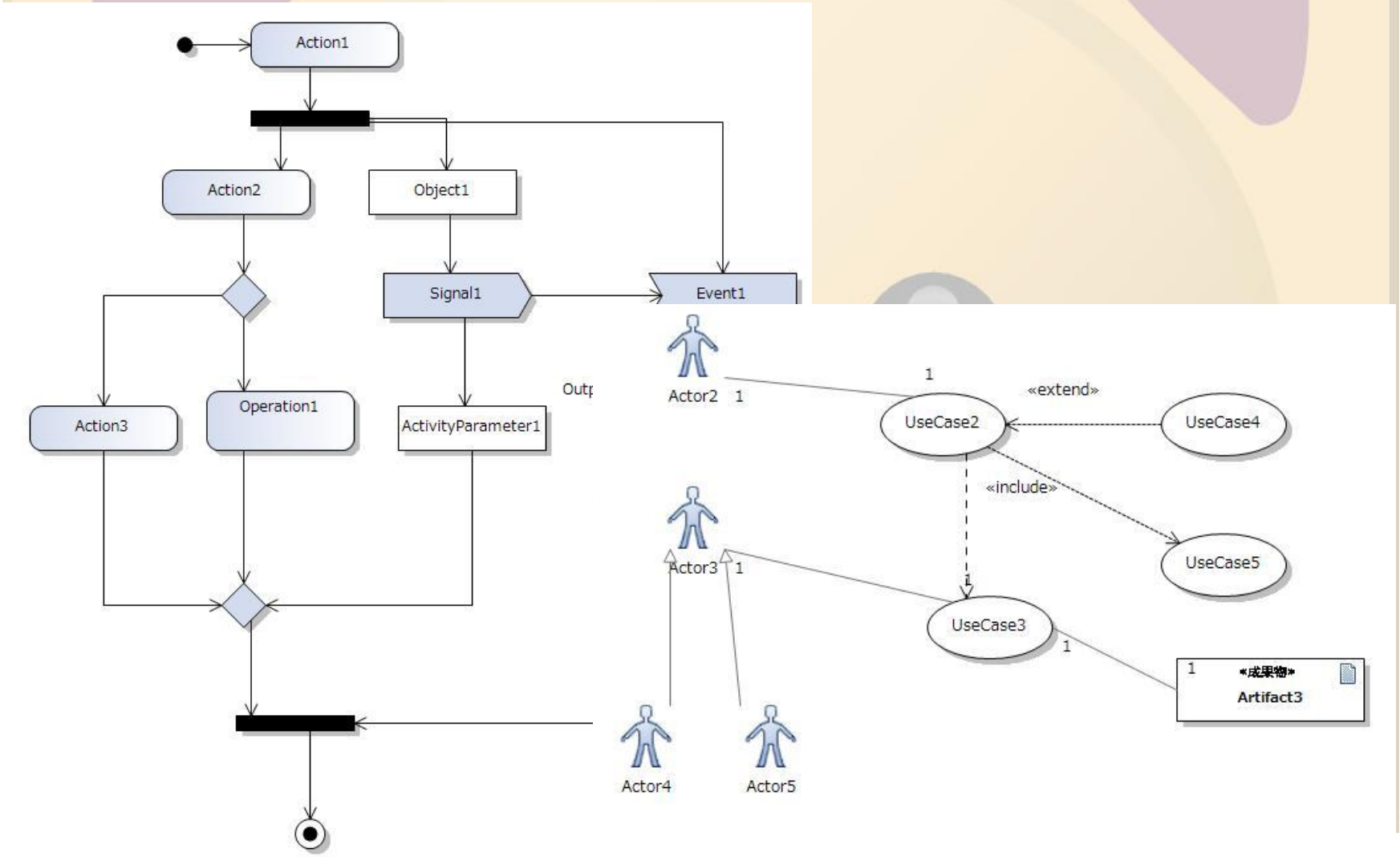


- 同時実行のコレクションクラス
  - 今までのコレクションはスレッドセーフではない、コレクションに結果を追加していくタスクでは使えない
    - BlockingCollection<*T*> クラス
    - ConcurrentBag<T> クラス
    - ConcurrentDictionary<TKey, Tvalue> クラス
    - ConcurrentQueue<T> クラス
    - ConcurrentStack<T>クラス
- 遅延初期化クラス
  - スレッドセーフな遅延初期化　Lazy<T>
  - スレッドローカルな遅延初期化　ThreadLocal<T>

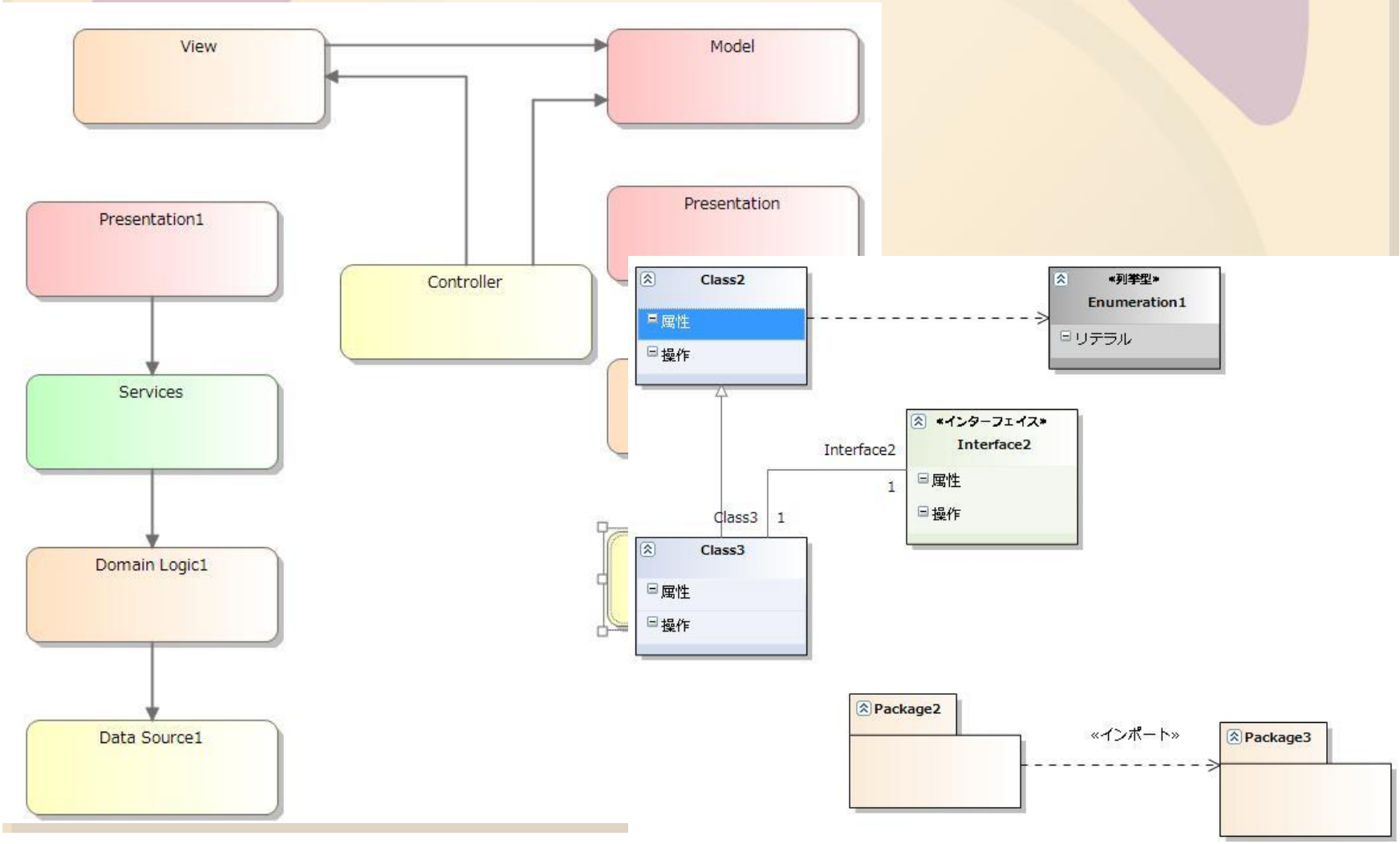
UML

- UML図が描ける
  - アクティビティ図
  - ユースケース図
  - レイヤー図
  - 論理クラス図
  - コンポーネント図
  - シーケンス図

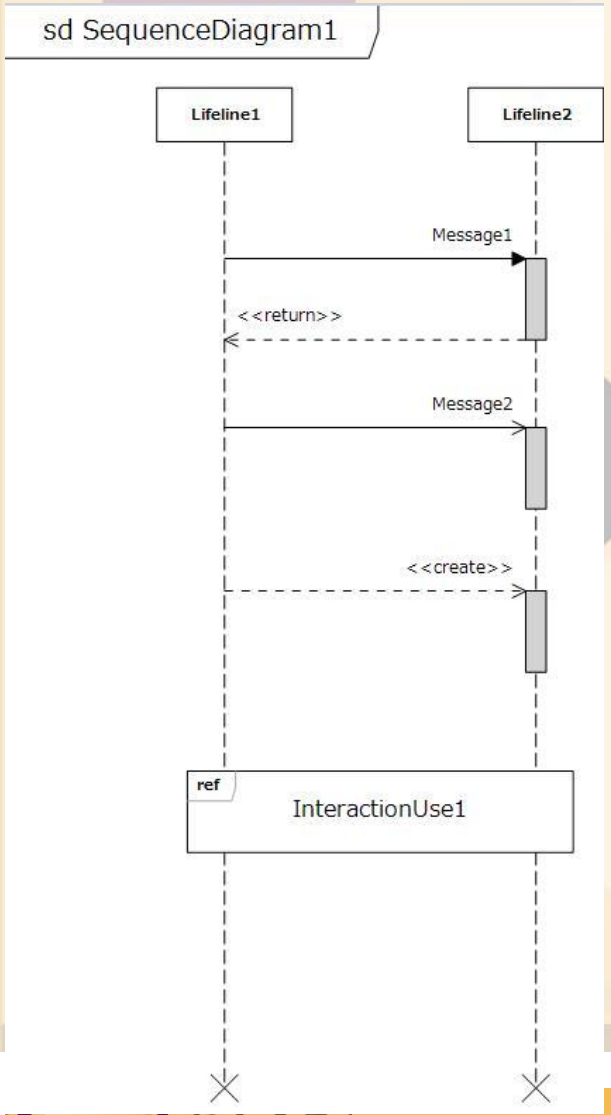
UML

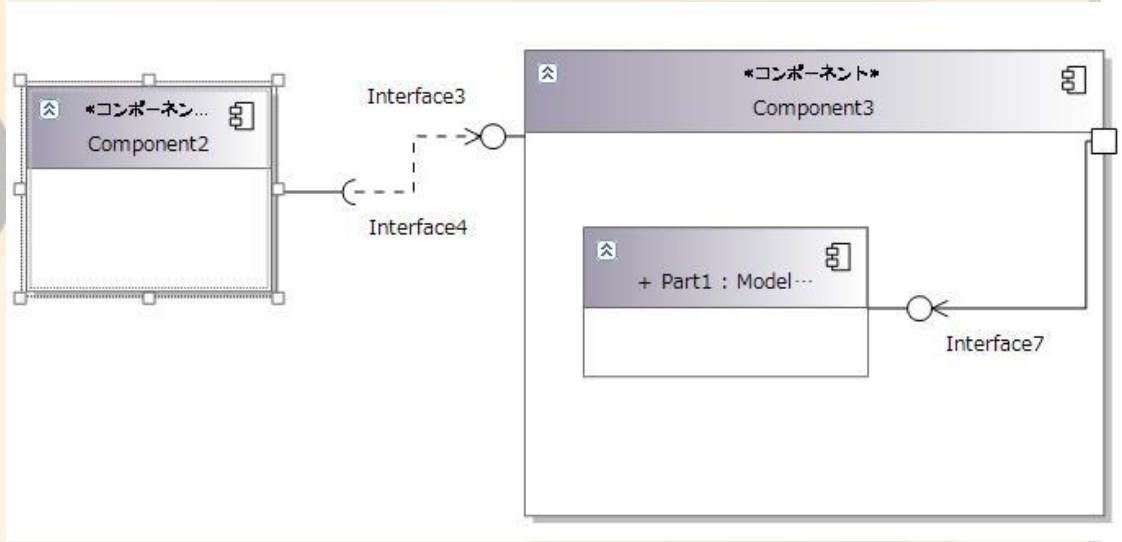

Action1
Action2
Object1
Signal1
Event1
Action3
Operation1
ActivityParameter1
Actor2
1
«extend»
UseCase2
UseCase4
«include»
UseCase5
Actor3
1
UseCase3
1
1
成果物
Artifact3
Actor4
Actor5

UML


View
Model
Presentation1
Presentation
Controller
Services
Domain Logic1
Data Source1
Class2
属性
操作
«列挙型»
Enumeration1
リテラル
«インターフェイス»
Interface2
属性
操作
Interface2
1
Class3
1
Class3
属性
操作
Package2
«インポート»
Package3

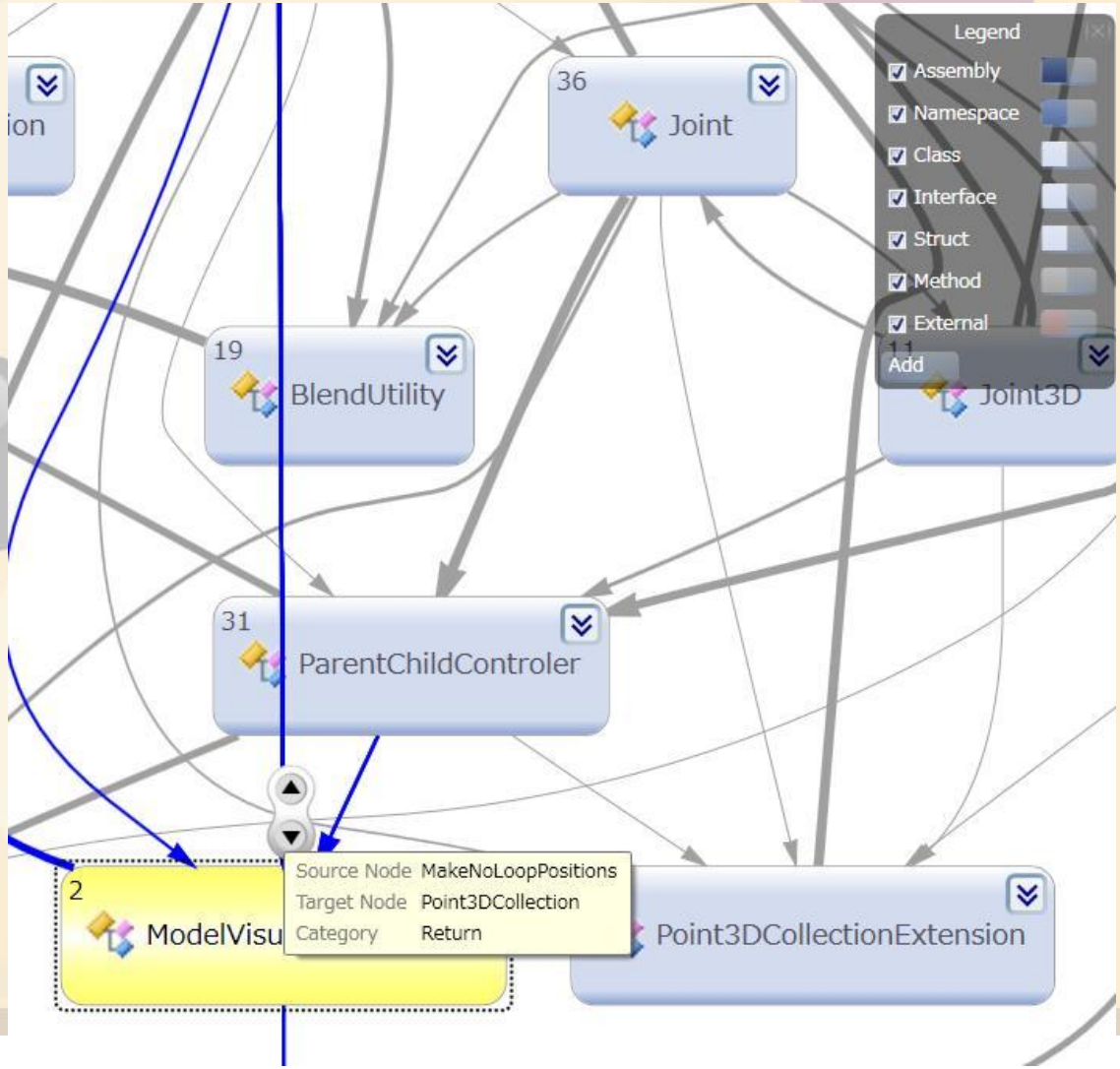
UML


sd SequenceDiagram1
Lifeline1
Lifeline2
Message1
<<return>>
Message2
<<create>>
ref
InteractionUse1


«コンポーネン...
Component2
Interface3
Interface4
«コンポーネント»
Component3
+ Part1 : Model
Interface7

UML

- クラス関係図


Legend
Assembly
Namespace
Class
Interface
Struct
Method
External
36
Joint
19
BlendUtility
11
Joint3D
31
ParentChildControler
2
ModelVisu
Source Node MakeNoLoopPositions
Target Node Point3DCollection
Category Return
Point3DCollectionExtension

まとめ

- VisualStudio2010はものすごい。
- 今日紹介した以外にも、Oslo や AjaxToolKit / WpfToolkit / SilverlightToolkit などでも取り込んでくるものがあるかもしれない。
- TeamSystemやテスト関係は充実すると思うのでデバッグもやりやすくなる。
- みなさんもβ版のバグ報告をしてよいVisualStudioを一緒に作っていきましょう。