ソフトウェアシンポジウム2009@札幌
モデリングWG

太田　寛
エンベデッド エバンジェリスト
マイクロソフト株式会社

Agenda

守

・バックグラウンドの巻

破

・マイクロソフトの
モデリングテクノロジーの巻

離

・最新動向の巻

その前に・・・

- 自己紹介
  - マイクロソフトの組込み系エバンジェリスト
    - 極小～極大まで
    - 様々な業種の組込み機器・システム
    - 産学官の各種活動
  - 前職は、OA機器メーカー
    - ソフトウェアエンジニア
    - 開発プロセス改善
    - 社外活動

守

Microsoft

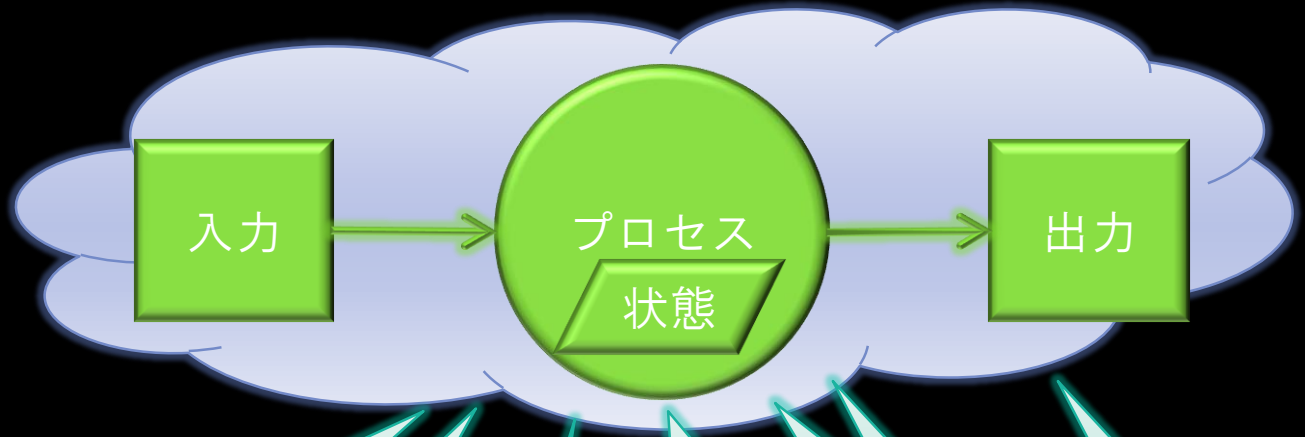
私の関心事

- より早く、より簡単に、望みのシステムを作り上げる方法は？
  - ビジネスとしてのソフトウェア開発
  - 主に組込み系
  - 中、大規模開発

モデル

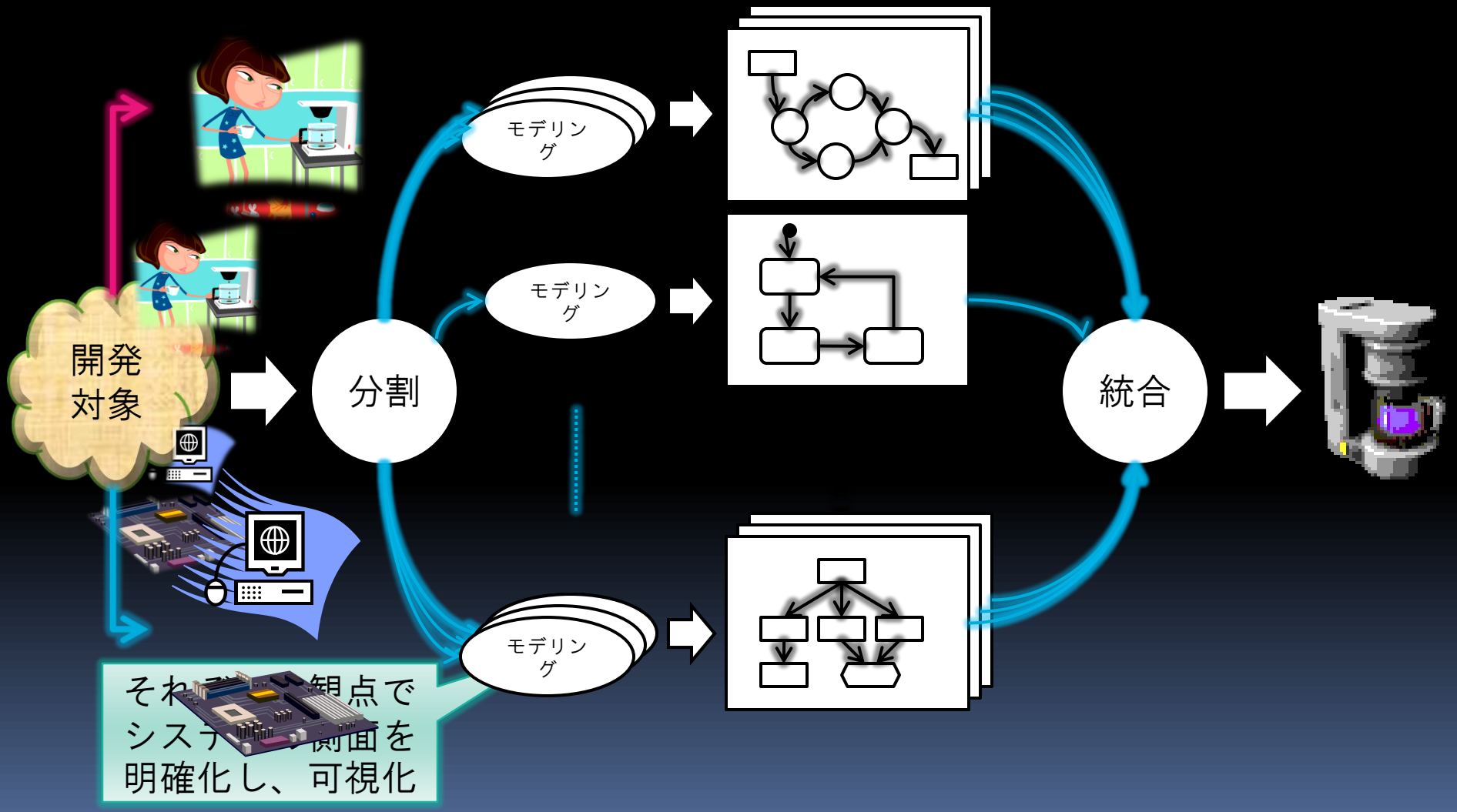
- アイデアやコンセプトを
  可視化し、
  一定の形式で、
  形式的に表現
  したもの

ソフトウェアって・・・

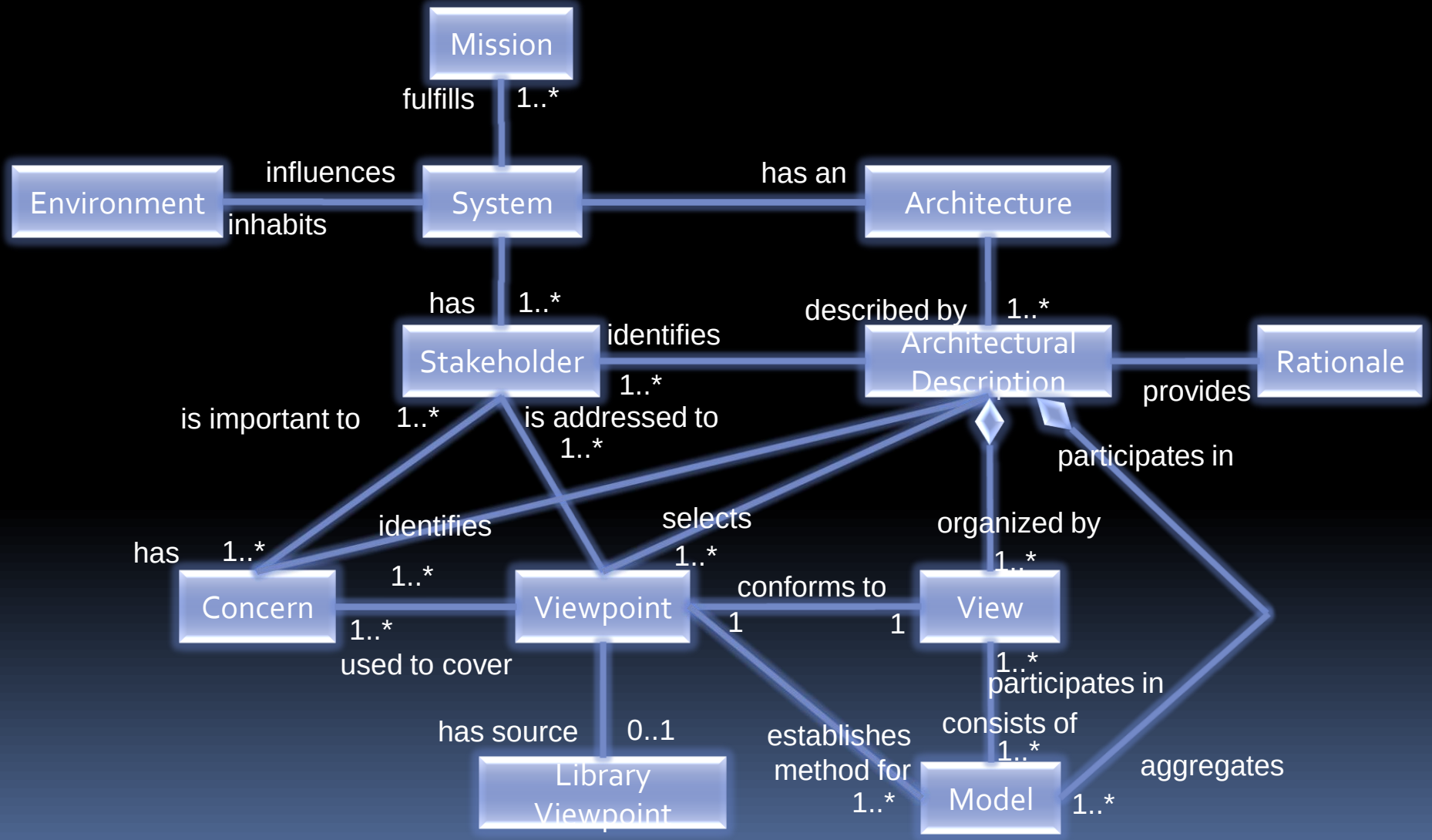


- コンテキスト
- ビジネス要求
- 製品ロジック
- 操作方法
- 各種要素技術
- システムアーキテクチャ
- 開発プロセス・開発環境

モデル駆動型開発

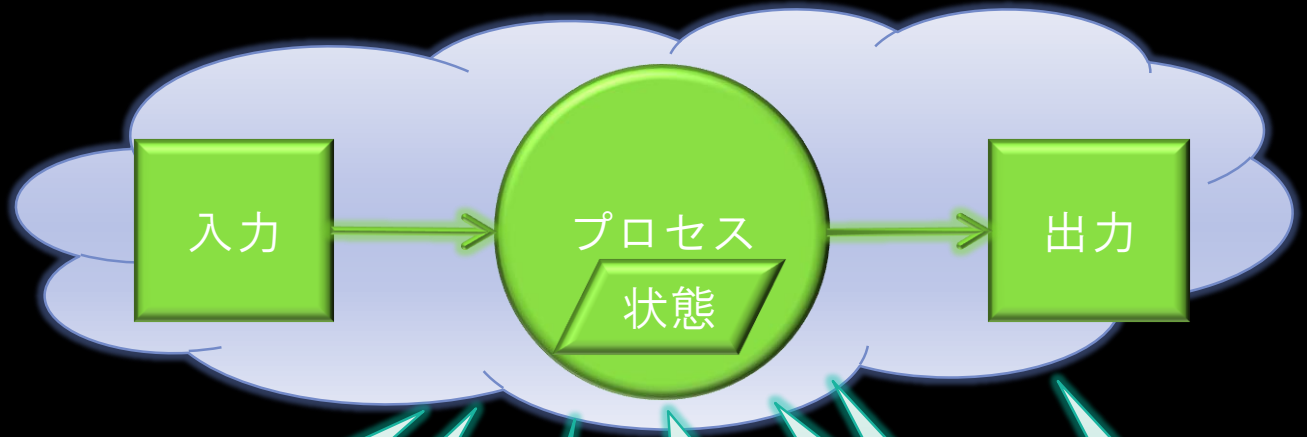


IEEE1471-2000



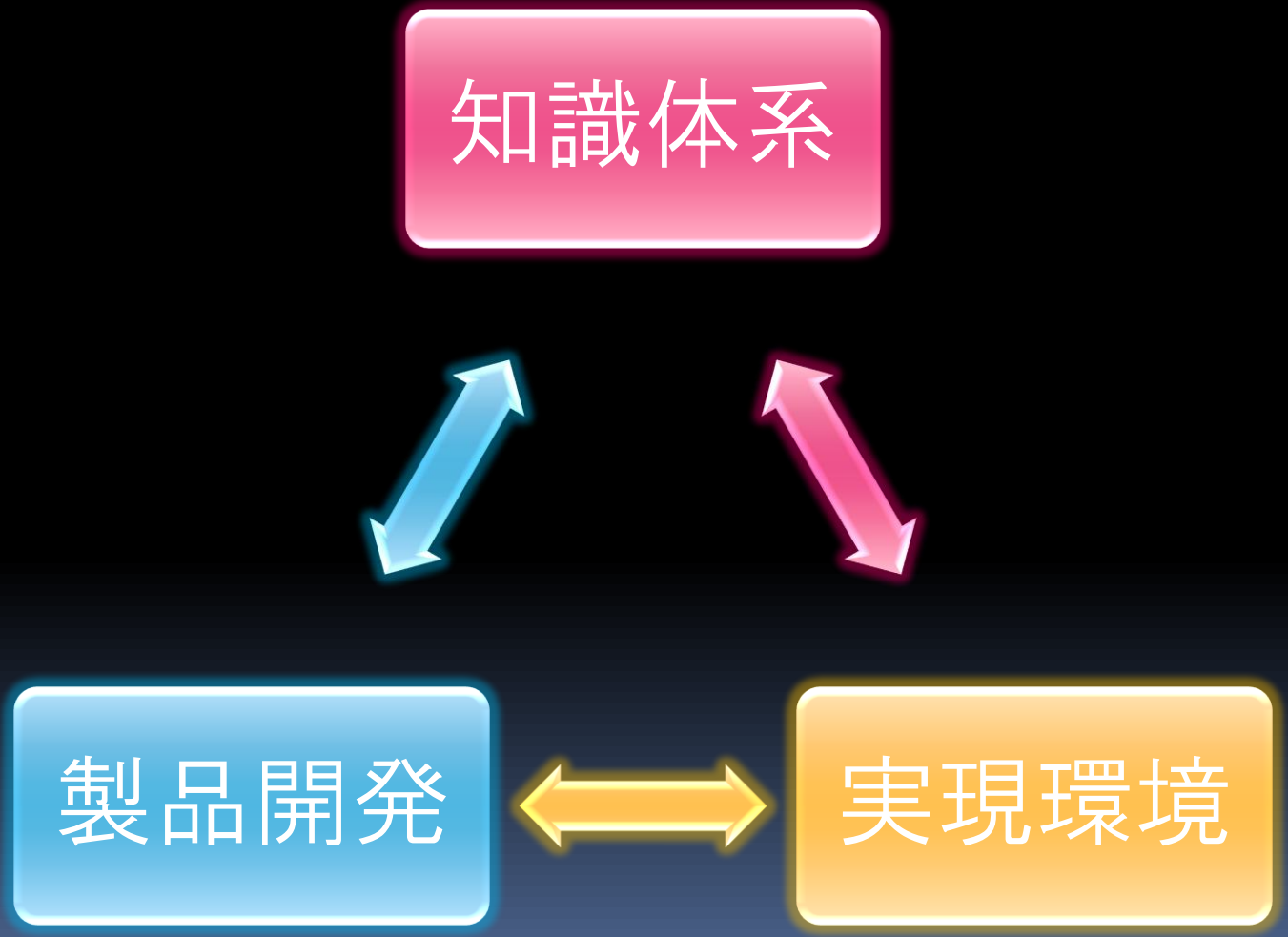
出典 : IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

ソフトウェアの開発って・・・

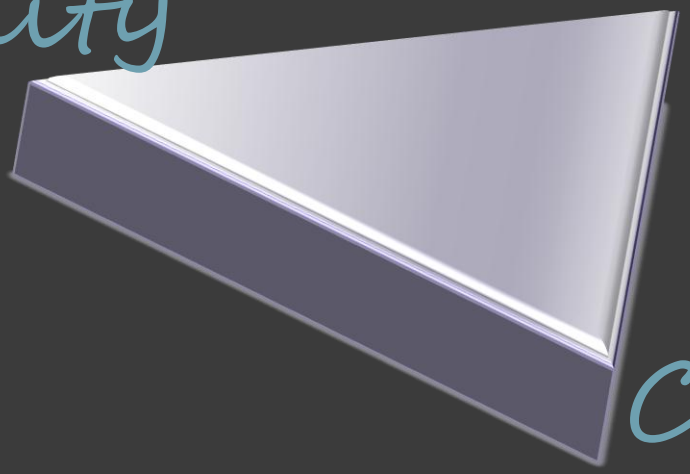


- コンテキスト
- ビジネス要求
- 製品ロジック
- 操作方法
- 各種要素技術
- システムアーキテクチャ
- 開発プロセス・開発環境

皆さんの話をお聞きして



QCDのトレードオフ



より少ないコストで、より高い価値を

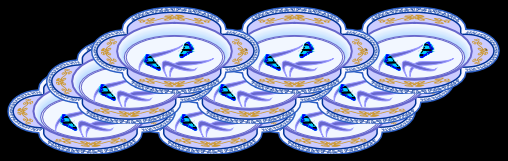
コスト

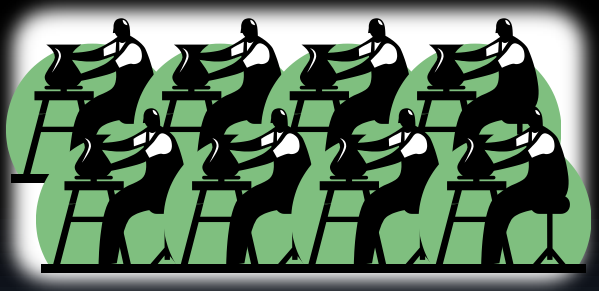
コスト=環境＋部品＋人件費

人件費∝N

規模の問題

1日に1枚
皿を作る



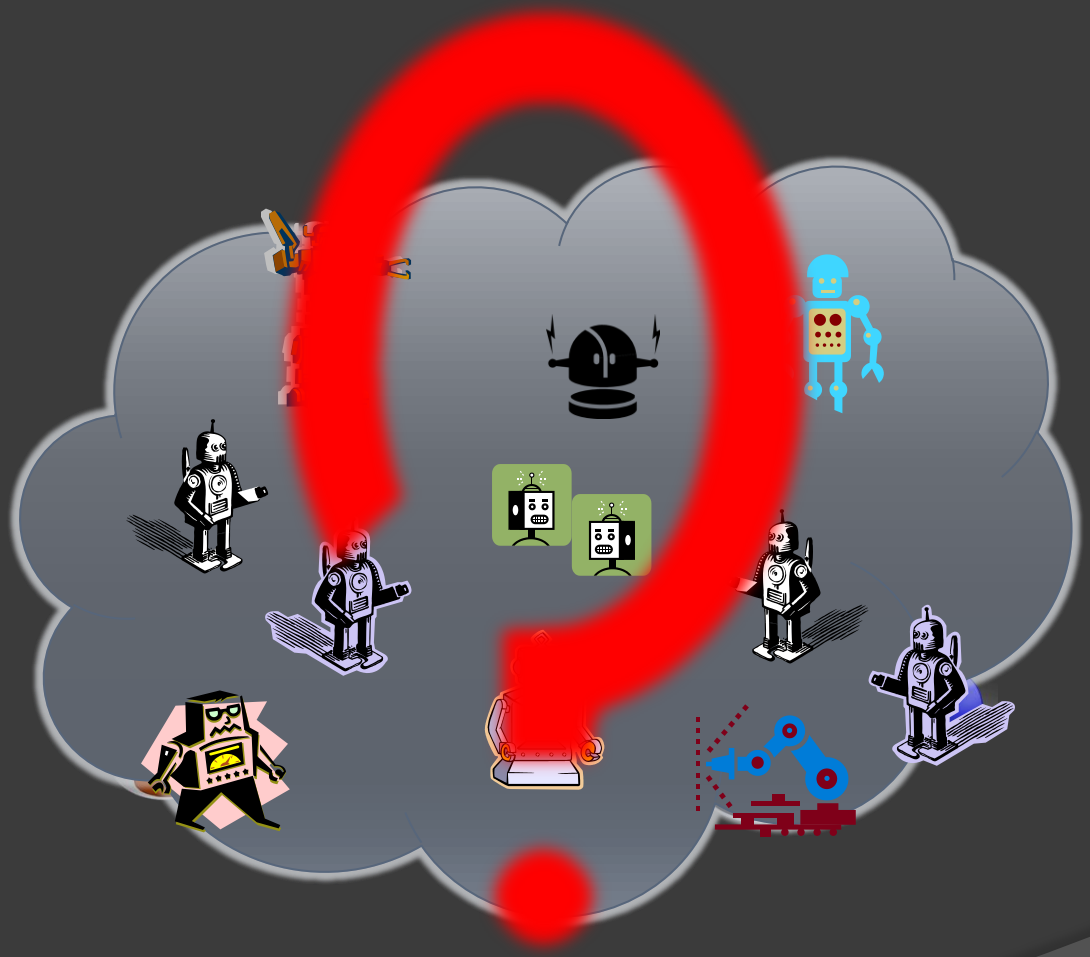


- 作るための技術が全く違う（Innovation）
- 皿を作るための本質は再利用
- 定型作業の自動化
- 知的作業の集約化

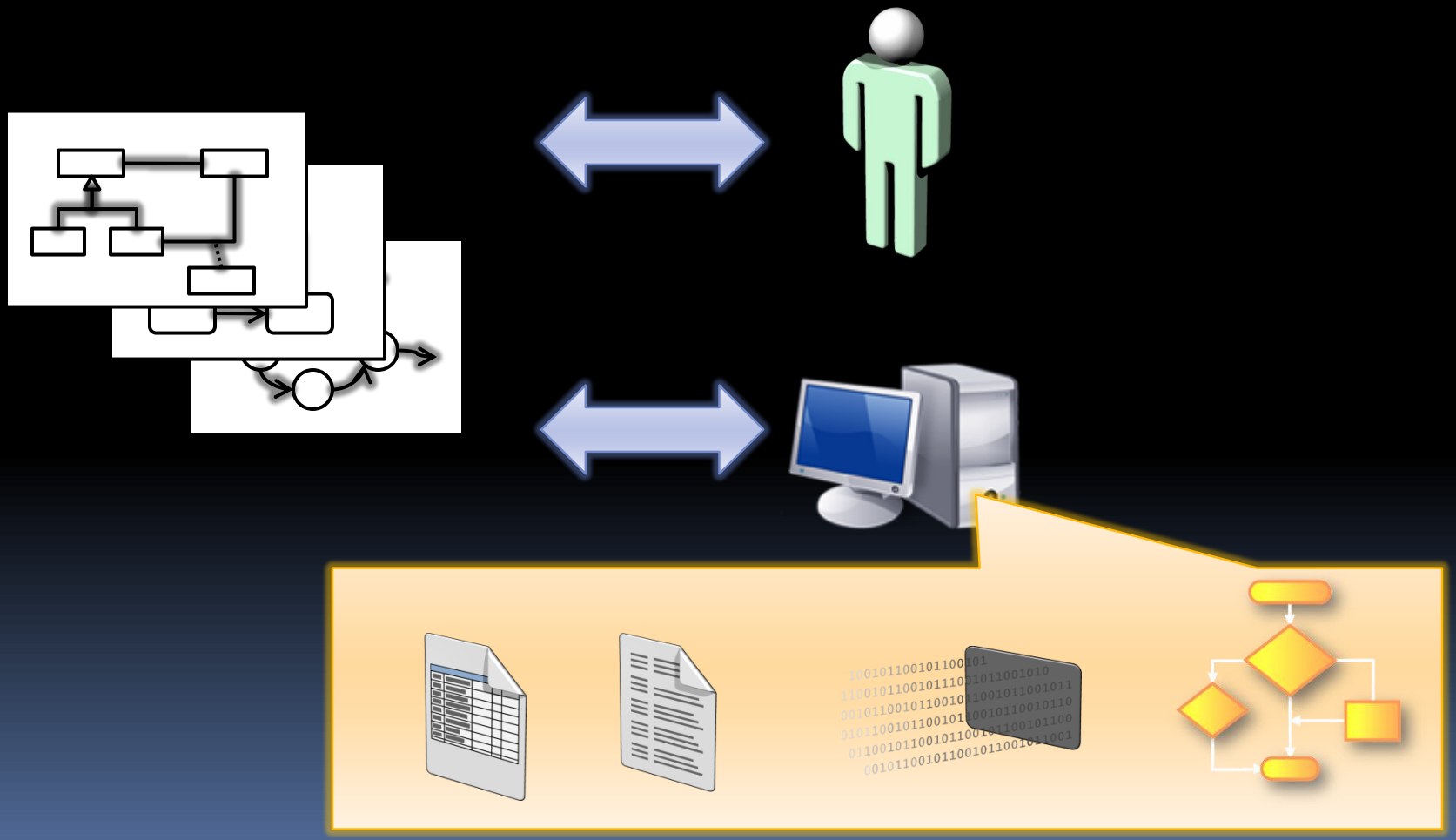
より早く、より簡単に、望みのシステムを作り上げる方法は？

- 機械でできることは機械にまかせ、人間は人間しかできない創造的な仕事に注力
- 源流での問題解決

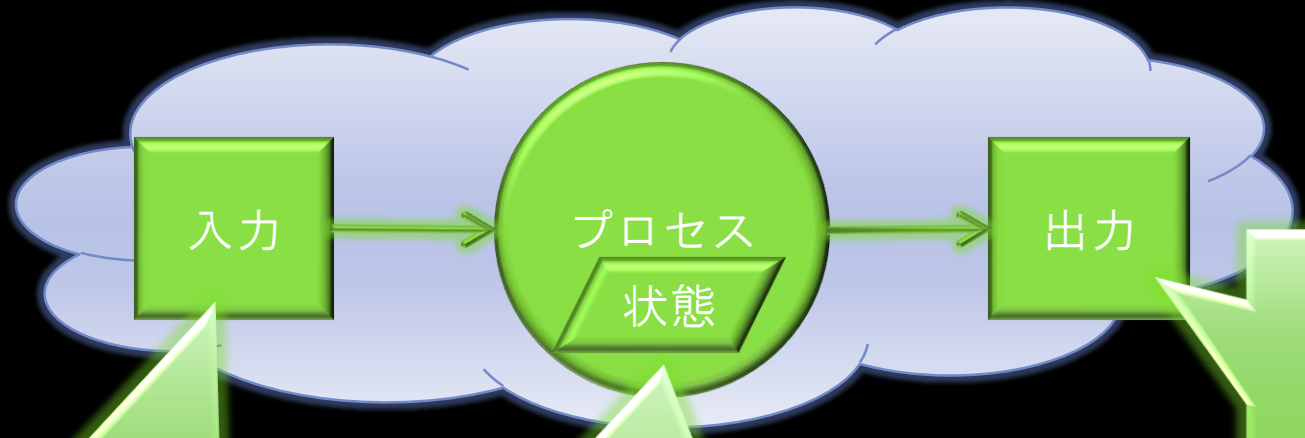
自動化の対象

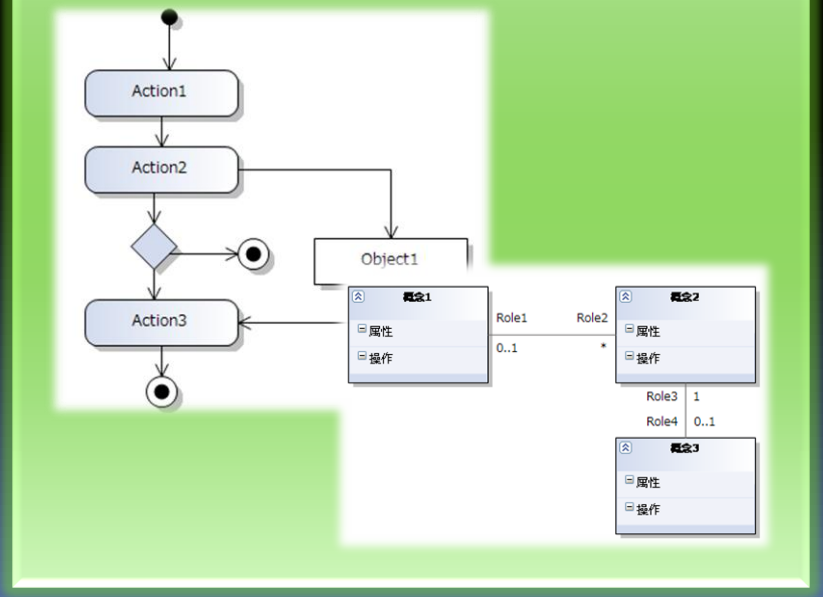


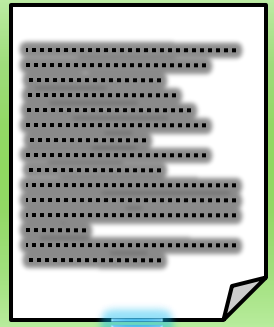
モデル駆動型開発における モデルの使い方



ソフトウェアの開発って・・・

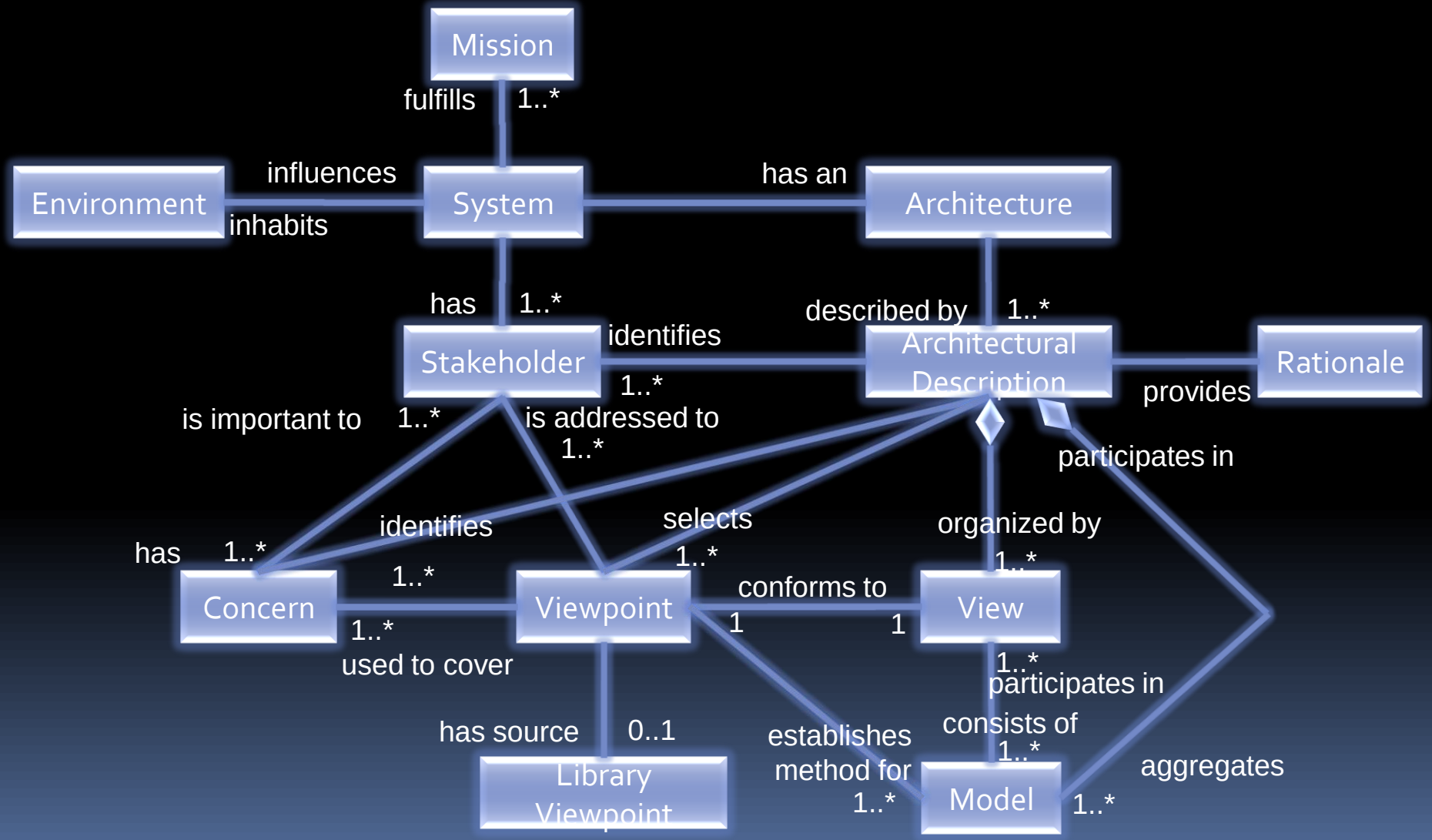






Platform

IEEE1471-2000



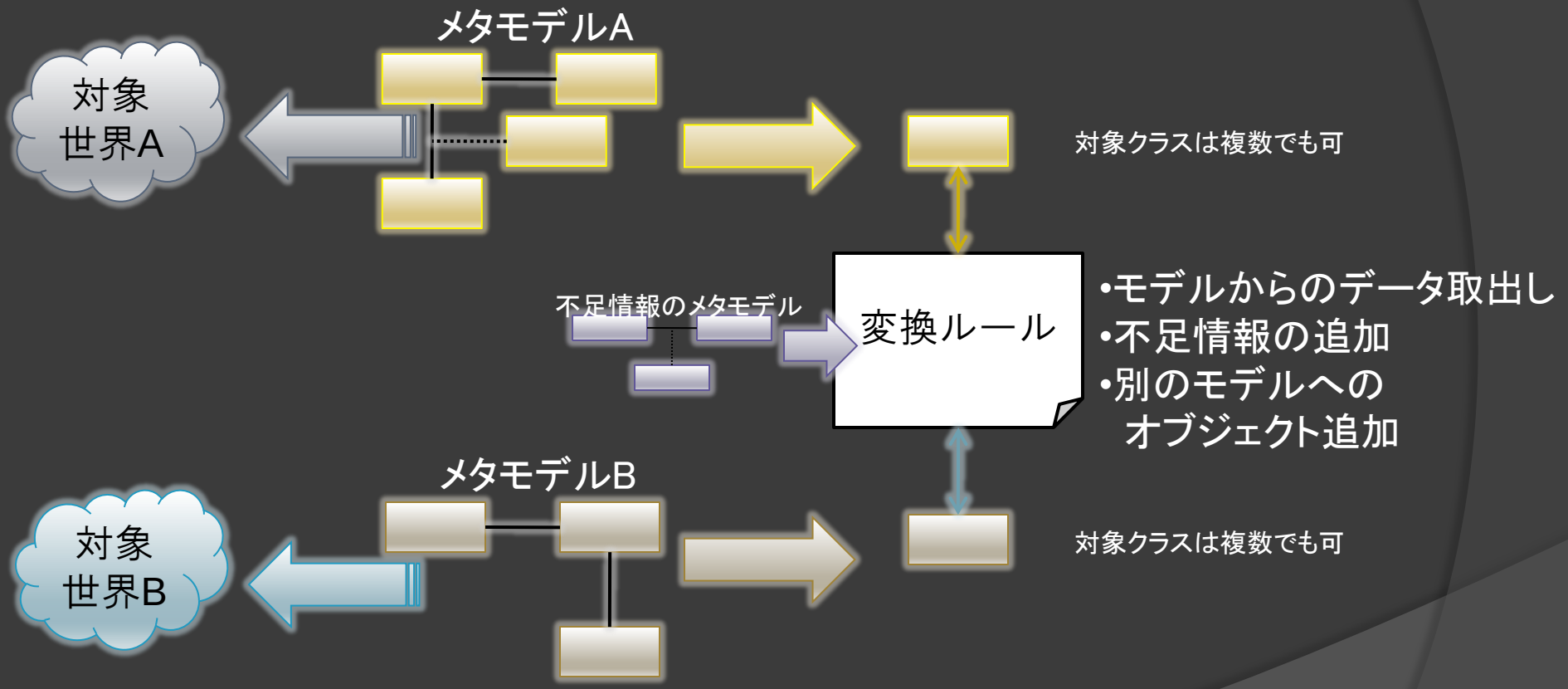
出典 : IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

ビューポイント（あるいは、ドメイン）

- 各ビューポイントは、意味的に独立している（相互に無関係）
  - 各ビューポイントは、モデルで記述される
- システムを実現するには、相互に無関係なものをマップする必要がある
  - モデル⇔モデル マッピング
  - モデル⇒モデルへの変換

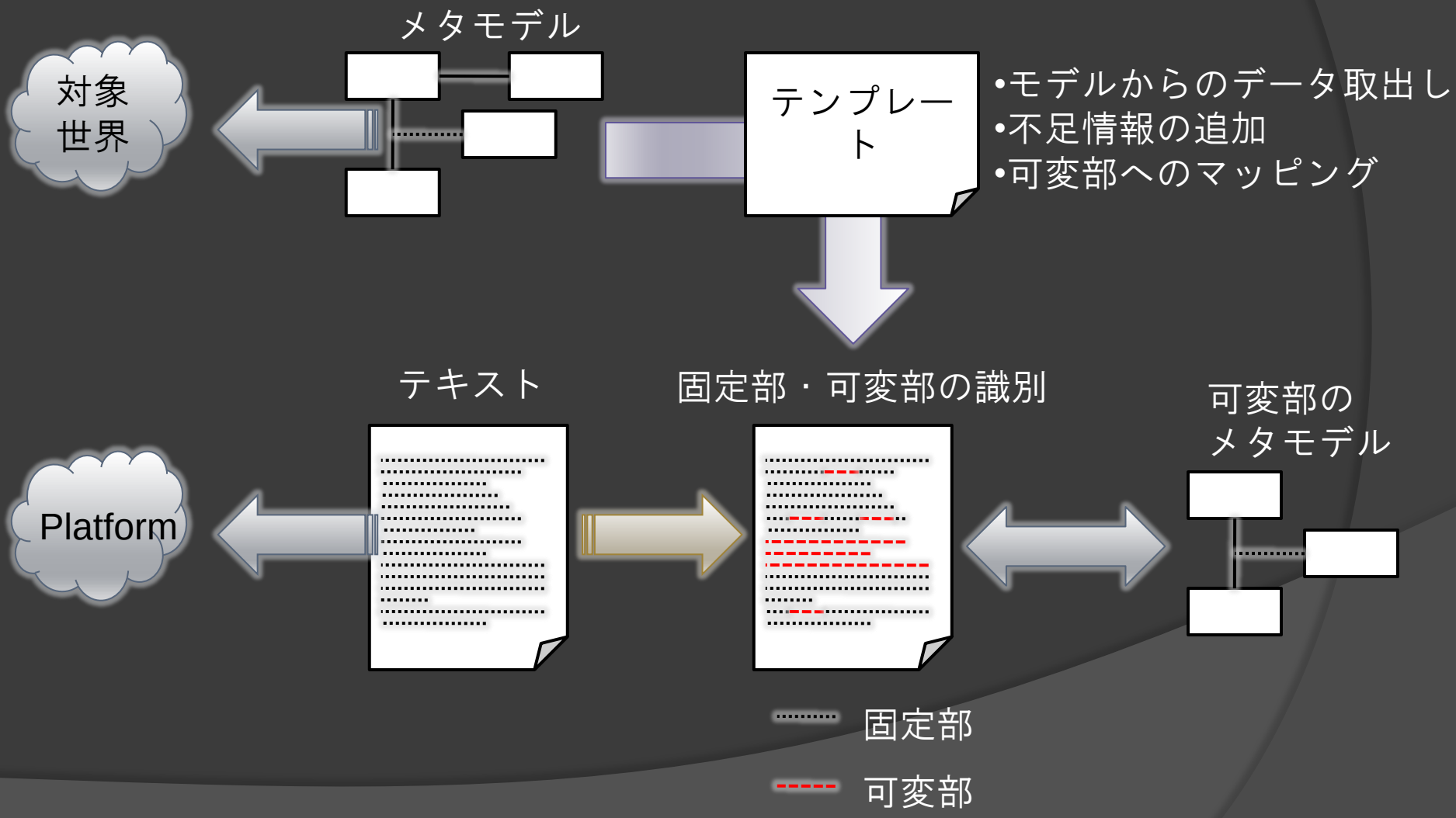
- ※システムを実現するには、ビューポイント相互に制約を与える
  - もちろん物理的に実装が不可能なものは不可能

Model ⇒ Model

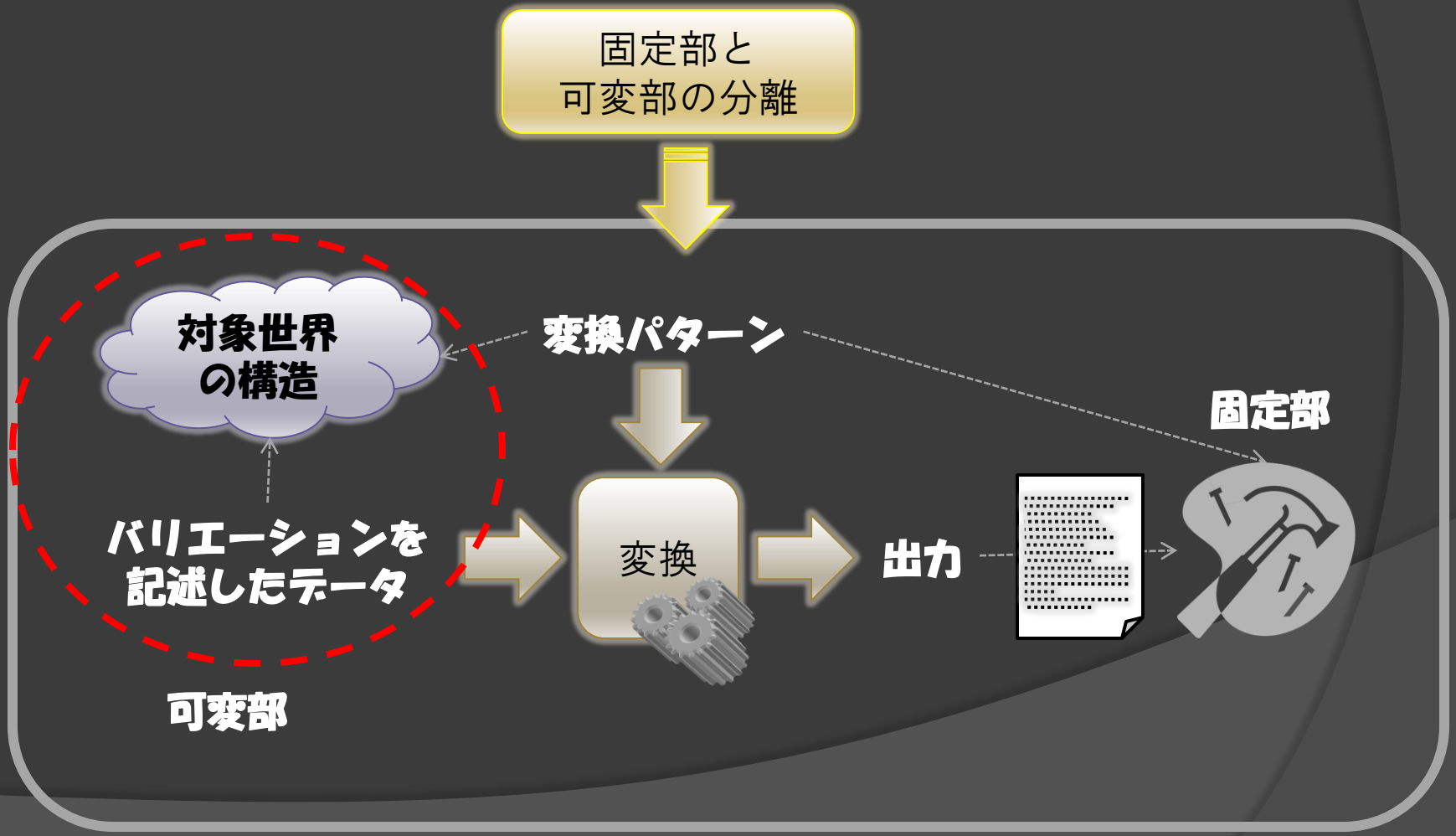


メタモデルA⇔メタモデルBの間の双方向変換は、水平方向変換の時にやりやすく、垂直方向変換では難しい

Model ⇒ Text


メタモデル
対象
世界
テンプレート
•モデルからのデータ取出し
•不足情報の追加
•可変部へのマッピング
テキスト
固定部・可変部の識別
可変部の
メタモデル
Platform
固定部
可変部

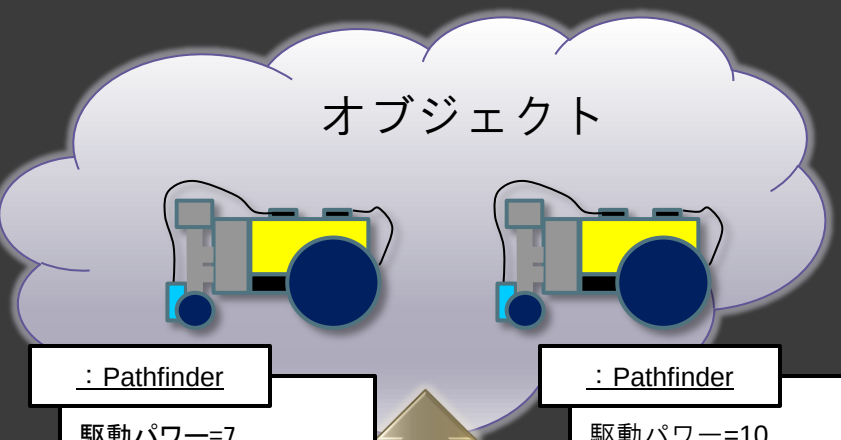
自動生成の仕組み



概念モデル

- 同じ特徴をもつ“モノ”（概念）の集合と、“モノ”の間の“関係”を記述
  - 同じ特徴
    - 同じ“データ”（変数）の組みで記述される
    - 同じ“振る舞い”（ライフサイクル）を持つ
- “クラス図”で記述
  - “クラス”＝“概念”
  - クラス図は、バリエーションの可能範囲
  - オブジェクト、リンクによりバリエーションを記述

クラスとオブジェクト



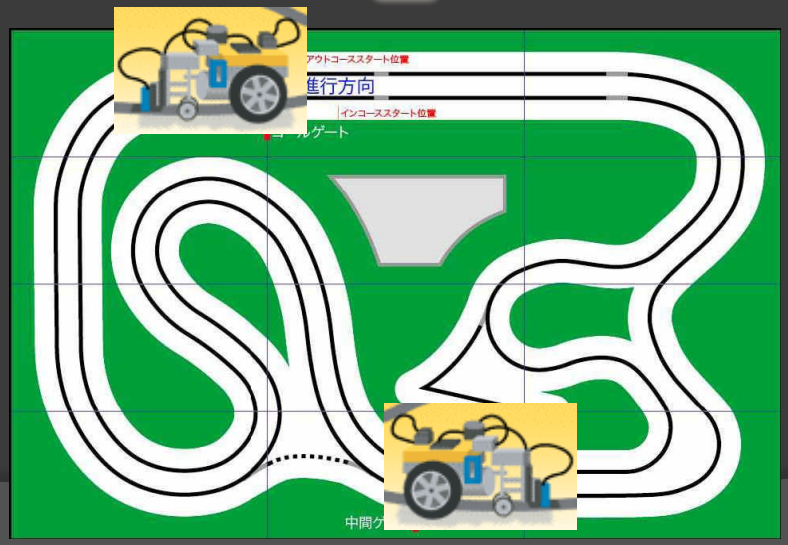
クラス

Pathfinder

クラスの名前

駆動パワー
ステアリング方向
ステアリングパワー
電池電圧

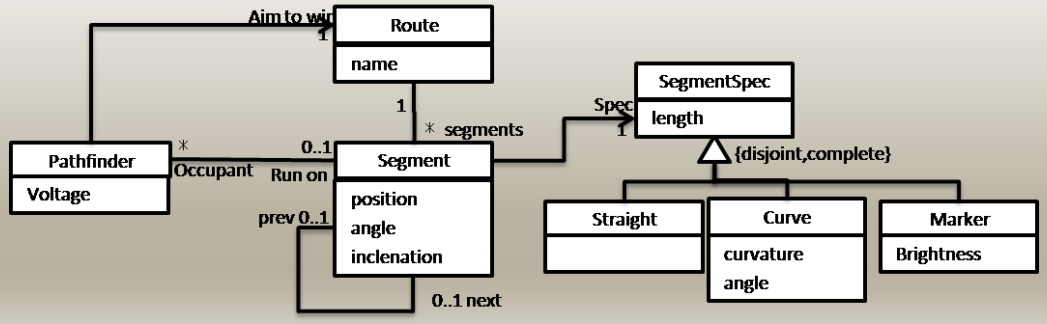
属性：
オブジェクトを特徴付けるパラメータセット

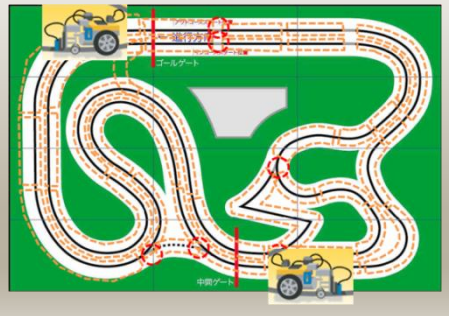


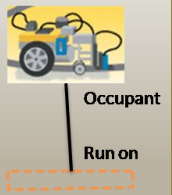
全てのオブジェクトは、クラスで定義された特徴、振舞をもつ

クラス抽出には絵を描くと良い

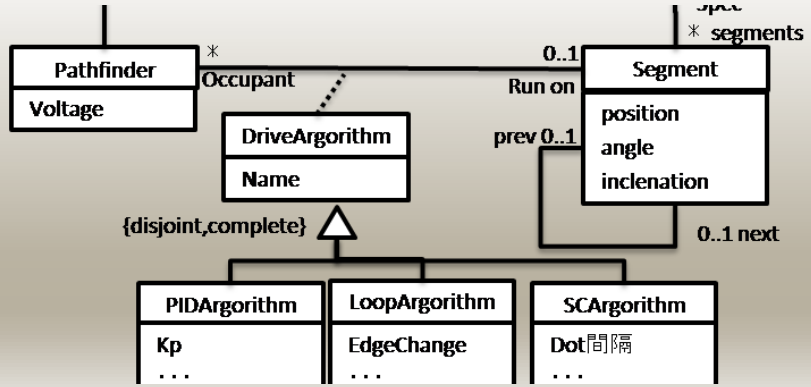
概念レベルのクラス図

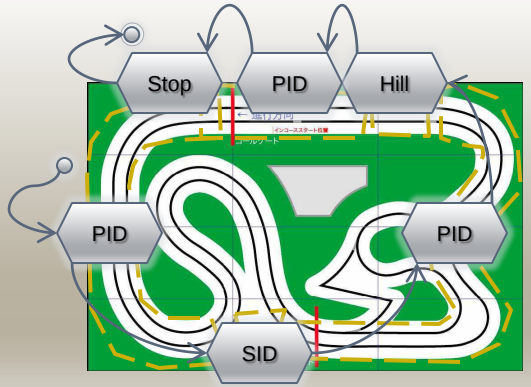




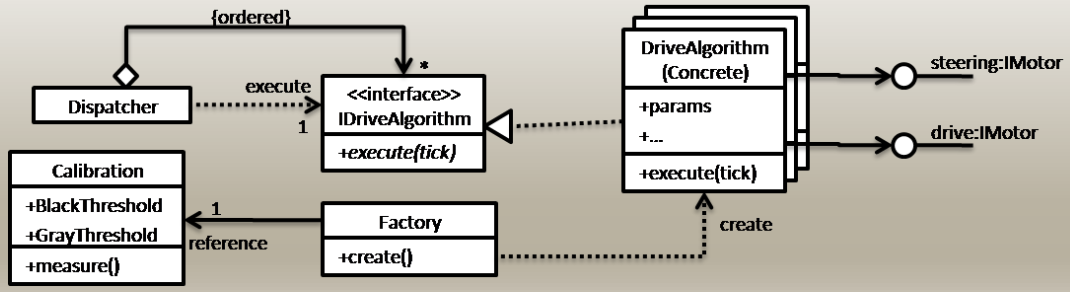


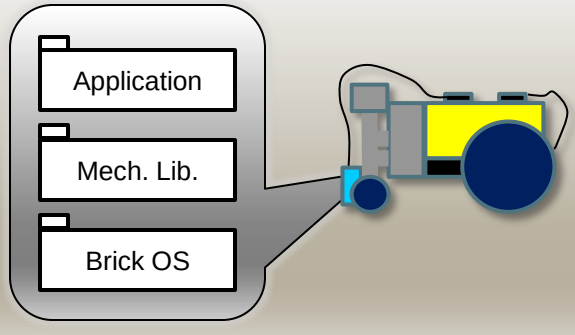
仕様レベルのクラス図





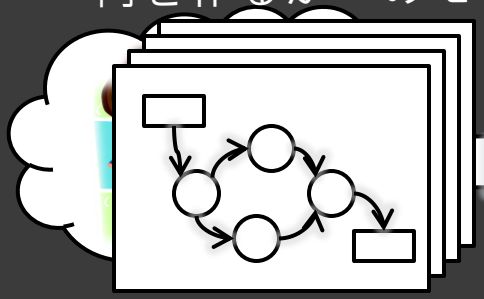
設計レベルのクラス図（Composite Structure Diagramを含む）



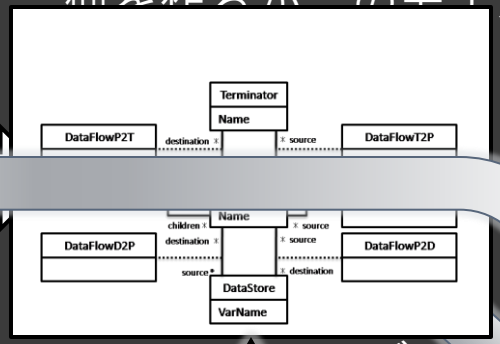


モデルの変換

“何を作るか”のモデル



“何を作るか”のモデル



これらのモデルには
実装のための情報は
ふくんでいない

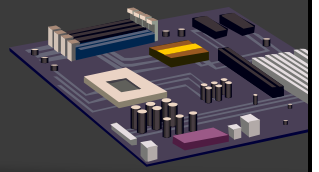
モデルの書き換え内容

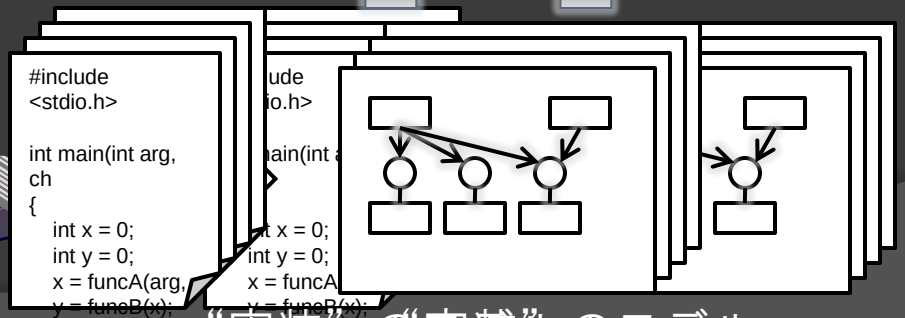
変換
ルール

変換
ルール

テンプレート変換エンジン

最終
プログ
ラム

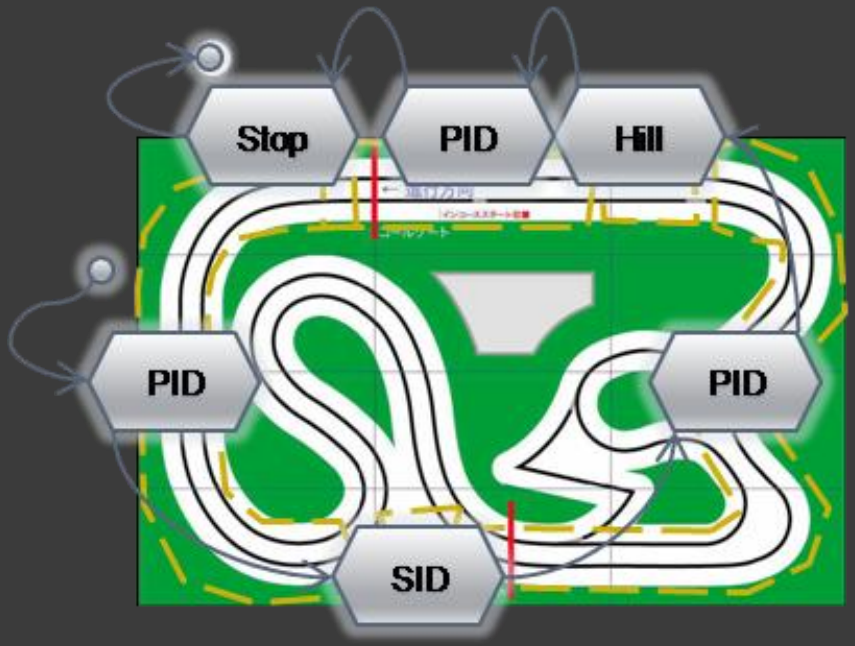




“実装”のモデル

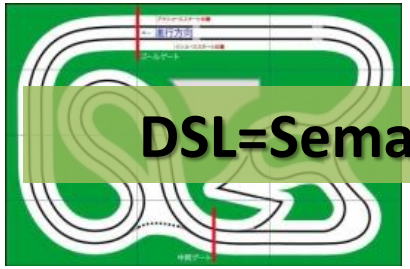
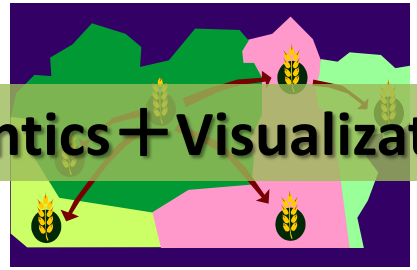
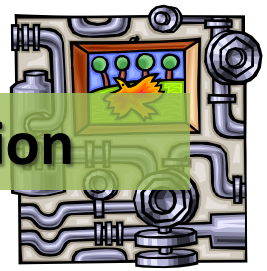
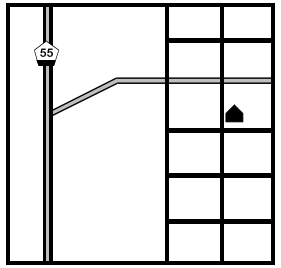
実装技術の使い方
・参照実装
・参照アーキテクチャ

例えば、入力モデルは



DSL : Domain Specific Language

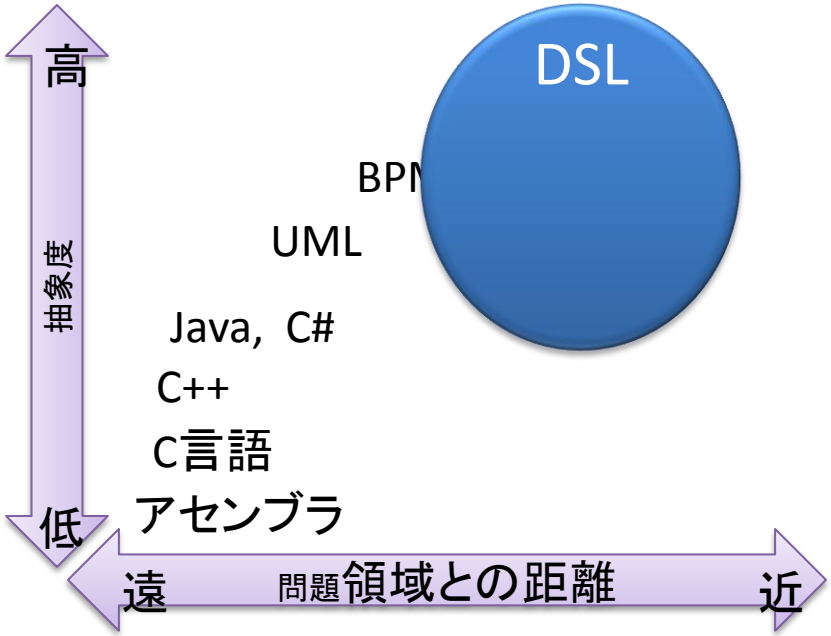
- 特定の範囲の問題を解決するために設計された専門の言語

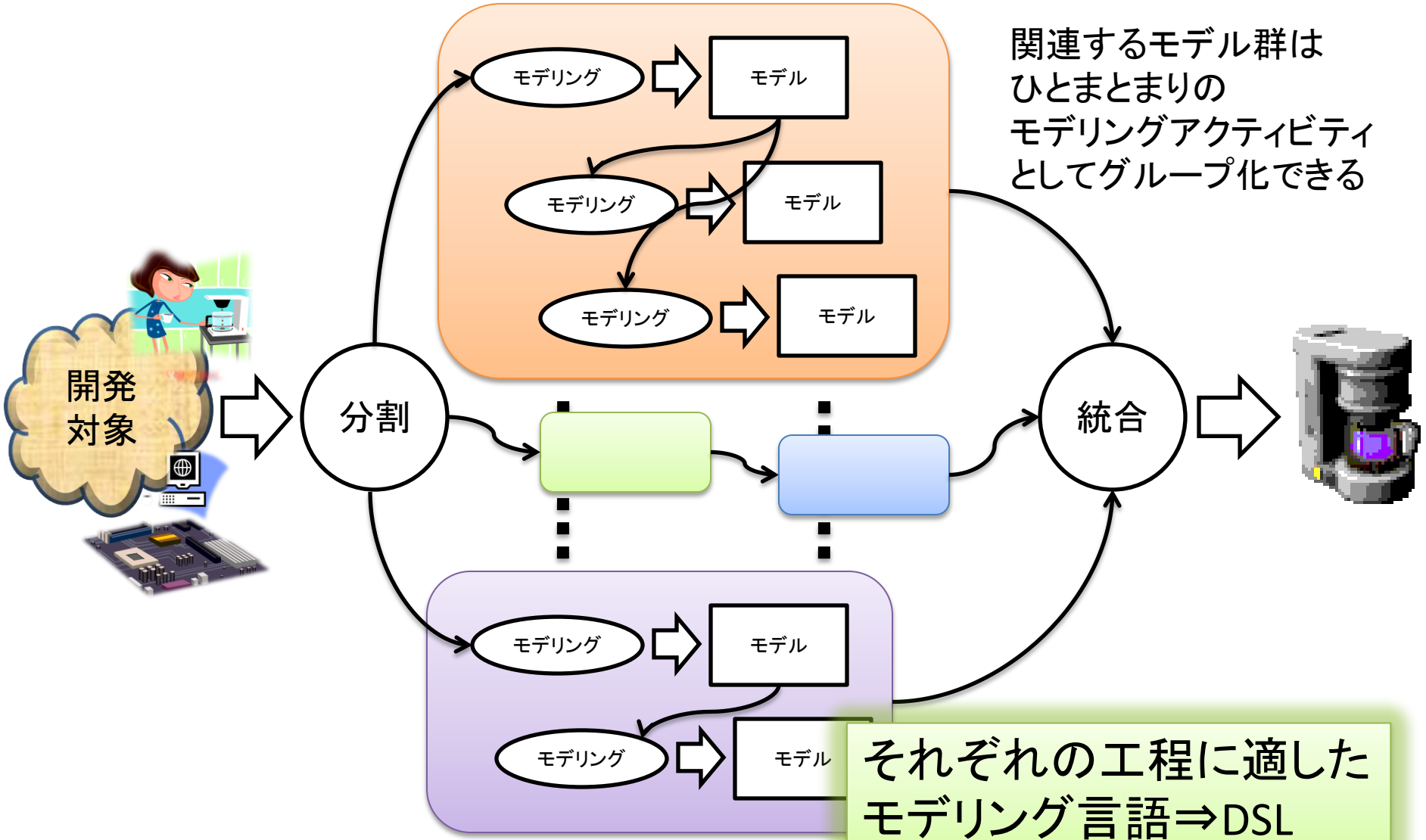
DSL=Semantics＋Visualization

DSLの例:

- HTML
- SQL
- 正規表現
- Windows Form Editor
- Wizard
- Executable UML



DSLを利用する


関連するモデル群は
ひとまとまりの
モデリングアクティビティ
としてグループ化できる
開発
対象
分割
モデリング
モデル
モデリング
モデル
モデリング
モデル
統合
モデリング
モデル
モデリング
モデル
それぞれの工程に適した
モデリング言語⇒DSL

DSLの定義

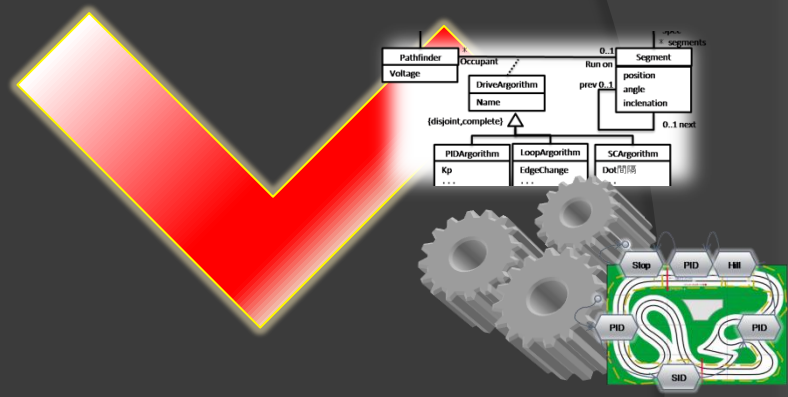
- DSL　　：＝モデル
- モデル　　：＝意味＋見栄え
- DSLを定義　　：＝意味と見栄えの定義

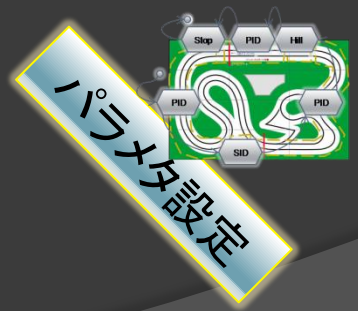
クラス図を使って、DSLの“意味”を定義する
このクラス図は、“メタモデル”と呼ばれる

流れを振り返ると…

従来の開発スタイル

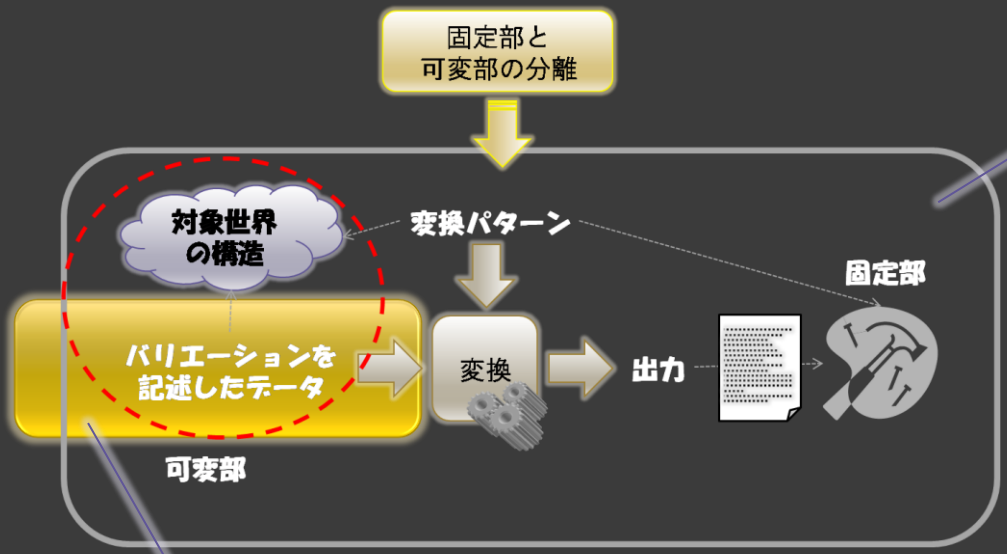
コード生成の仕組み構築





個々の条件への対応

開発環境への統合



モデルエディタ、
変換環境を
開発環境に統合

開発環境を
使って開発

再びコスト

- はたして、コストは下がるのか

- $\text{コスト} = \Sigma C_t + N \times C_v$
  - DSL、変換の仕組みを作るコストが小さく、かつ、バリエーションを記述するコストが十分小さければ、トータルコストは、劇的に下がる

モデル、モデル、モデル

- バリエーションを規定するモデル
  - UML
    - クラス図、アクティビティ図、ステートチャート
  - プログラミング言語
  - XML Schema
  - ・・・
- バリエーションを記述するモデル
  - UML
    - クラス図、シーケンス図、・・・
  - プログラム
  - XML
  - ・・・

バリエーション記述で重要なこと

- メタモデル（意味論）
  - 概念クラス図
- ビュー（フォーム）
  - 結局は、クラス⇔オブジェクトの関係だが、直感的に判りやすい表現
- オーサリング環境
  - Editor、Validator、Debugger
  - Model Data Access/Transformation

モデル駆動型開発を実践する上で重要なこと

- プラットフォームの成熟度
  - 利用可能な部品
  - 適用可能な範囲
  - 抽象度
- 開発環境の成熟度
  - 拡張性・カスタマイズ性
  - 適用可能な範囲

Microsoft®

Visual Studioに装備されているモデリング環境

- ウィザード
- プロジェクトテンプレート、アイテムテンプレート
- 各種エディタ
  - EDM
  - Form Designer
  - Workflow Designer
- UML Editor
- Architecture Explorer

モデル駆動型開発を実践するための開発環境拡張機能

- テンプレート
- アドイン
- DSL
- GAT/GAX
- Blueprints
- Software Factories

Microsoft®

Template

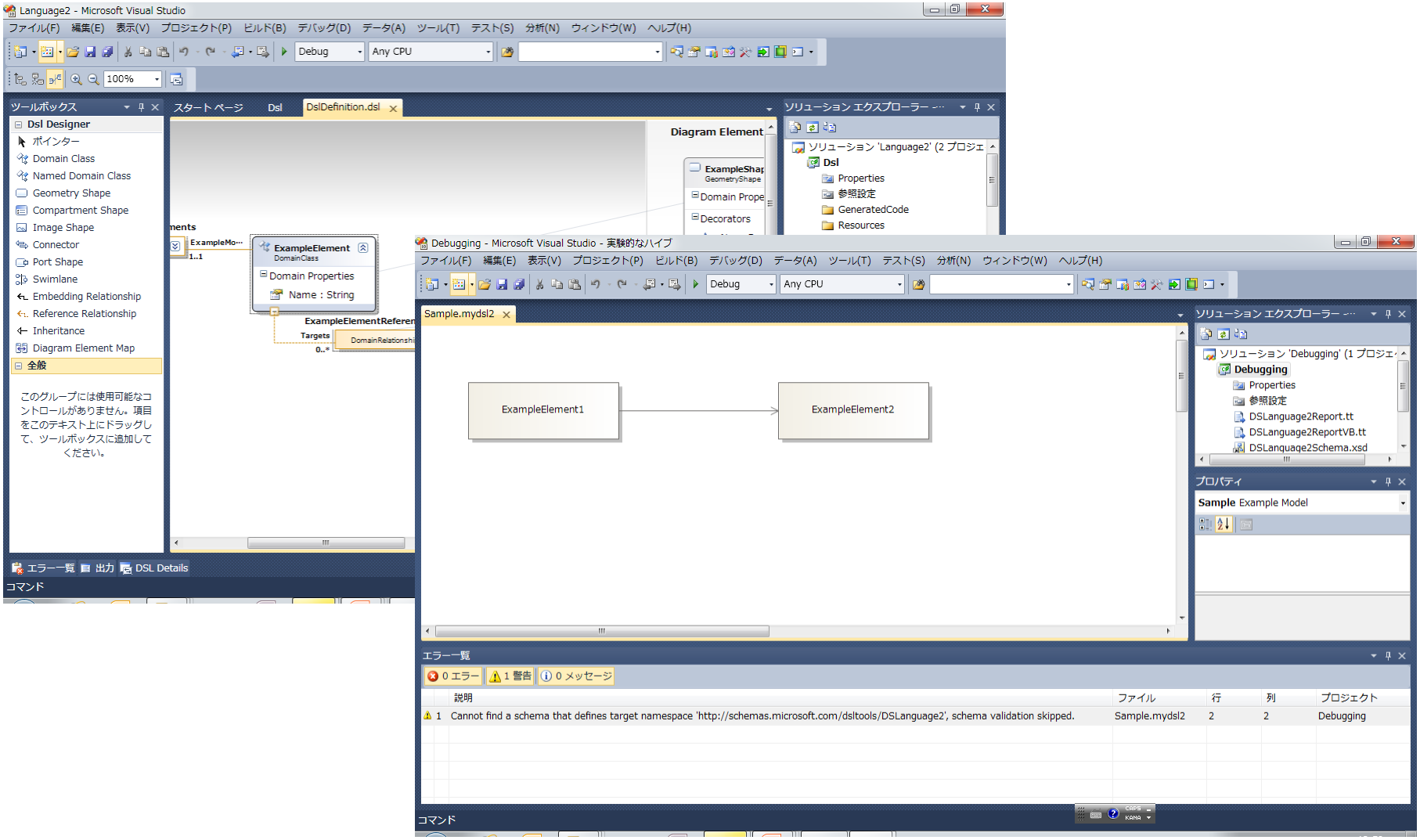
Microsoft

Add-in

Microsoft

DSL

DSL Toolkit


Language2 - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) データ(A) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
ツールボックス
Dsl Designer
ポインター
Domain Class
Named Domain Class
Geometry Shape
Compartment Shape
Image Shape
Connector
Port Shape
Swimlane
Embedding Relationship
Reference Relationship
Inheritance
Diagram Element Map
全般
このグループには使用可能なコントロールがありません。項目をこのテキスト上にドラッグして、ツールボックスに追加してください。
スタート ページ
Dsl
DslDefinition.dsl
Diagram Element
ExampleElement
DomainClass
Domain Properties
Name : String
ExampleElementReferen
Targets
0..*
ソリューション エクスプローラー
ソリューション 'Language2' (2 プロジェ
Dsl
Properties
参照設定
GeneratedCode
Resources
エラー一覧
出力
DSL Details
コマンド
Debugging - Microsoft Visual Studio - 実験的なハイブ
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) データ(A) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Sample.mydsl2
ExampleElement1
ExampleElement2
ソリューション エクスプローラー
ソリューション 'Debugging' (1 プロジェ
Debugging
Properties
参照設定
DSLanguage2Report.tt
DSLanguage2ReportVB.tt
DSLanguage2Schema.xsd
プロパティ
Sample Example Model
エラー一覧
0 エラー
1 警告
0 メッセージ
説明
ファイル
行
列
プロジェクト
1 Cannot find a schema that defines target namespace 'http://schemas.microsoft.com/dsltools/DSLanguage2', schema validation skipped.
Sample.mydsl2
2
2
Debugging
コマンド

T4 Template

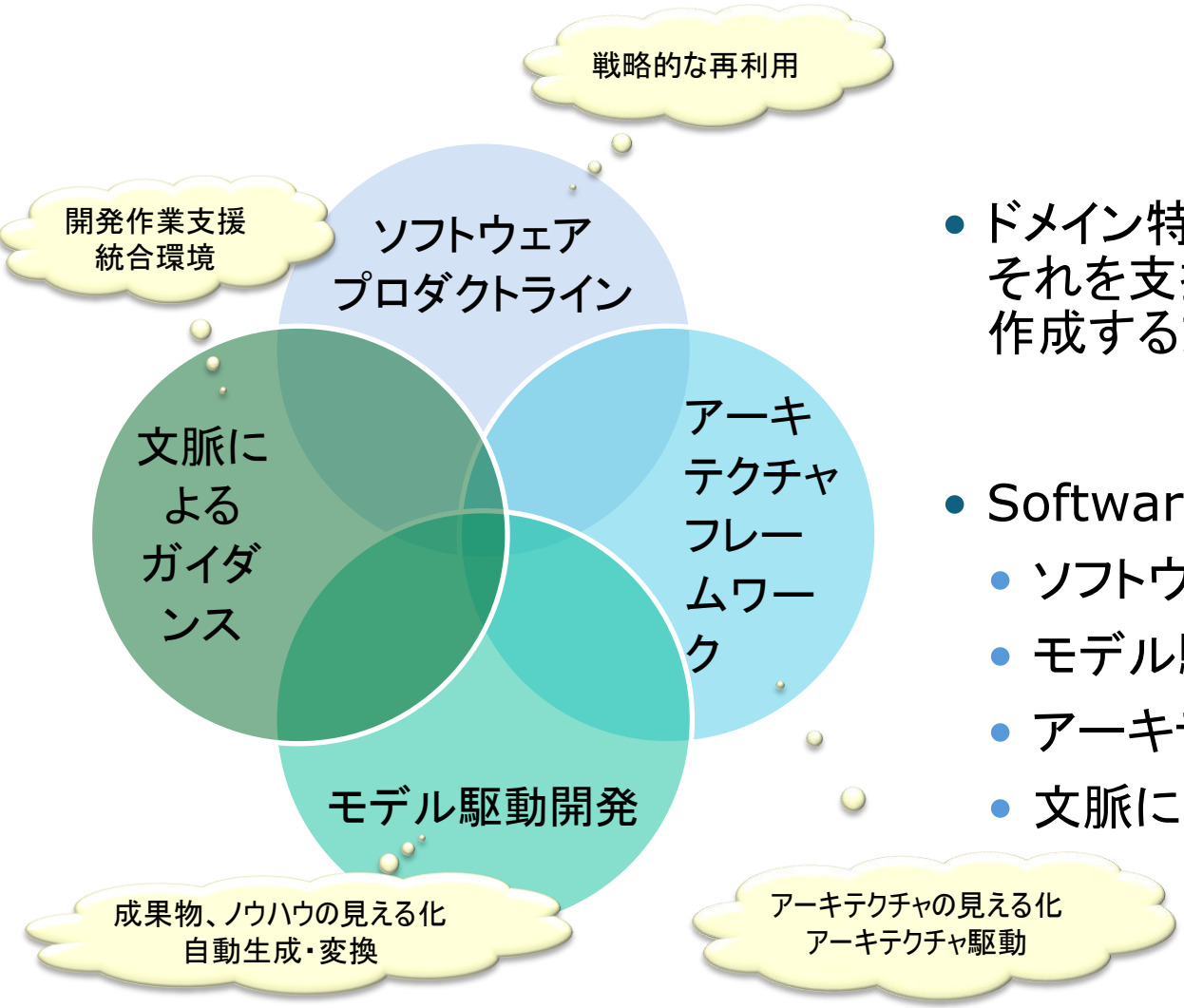
```
<#@ template inherits="Microsoft.VisualStudio.TextTemplating.VSHost.ModelingTextTransformation" debug="true"#>
<#@ output extension=".txt" #>
<#@ SimpleDataFlowDiagram processor="SimpleDataFlowDiagramDirectiveProcessor" requires="fileName='Test.sdfd'" #>
<#@ import namespace = "System.Collections.Generic" #>

Generated material. Generating code in C#.
<#
  // When you change the DSL Definition, some of the code below may not work.
  List<Process> processes = new List<Process>();

  foreach (NamedElement namedElement in this.DataFlowDiagram.Elements)
  {
    if (namedElement is Process) {
        processes.Add(namedElement as Process);
    }
#>
list :  <#= namedElement.Name #>
<#
  }

  List<Process> orderedProcesses = OrderProcessByDependency(processes);
  if (orderedProcesses.Count>0) {
#>

//-------------------------------------------------------
// Ordered Process External Declarations
//-------------------------------------------------------
<#
    foreach (Process process in orderedProcesses) {
#>
//extern void method<#= process.Name #>(CControlInfo* info);
<#
    }
#>
```

※ファイルの拡張子を“.tt”にすると自動的にT4テンプレートとして認識する

Microsoft

Software Factories

Software Factories

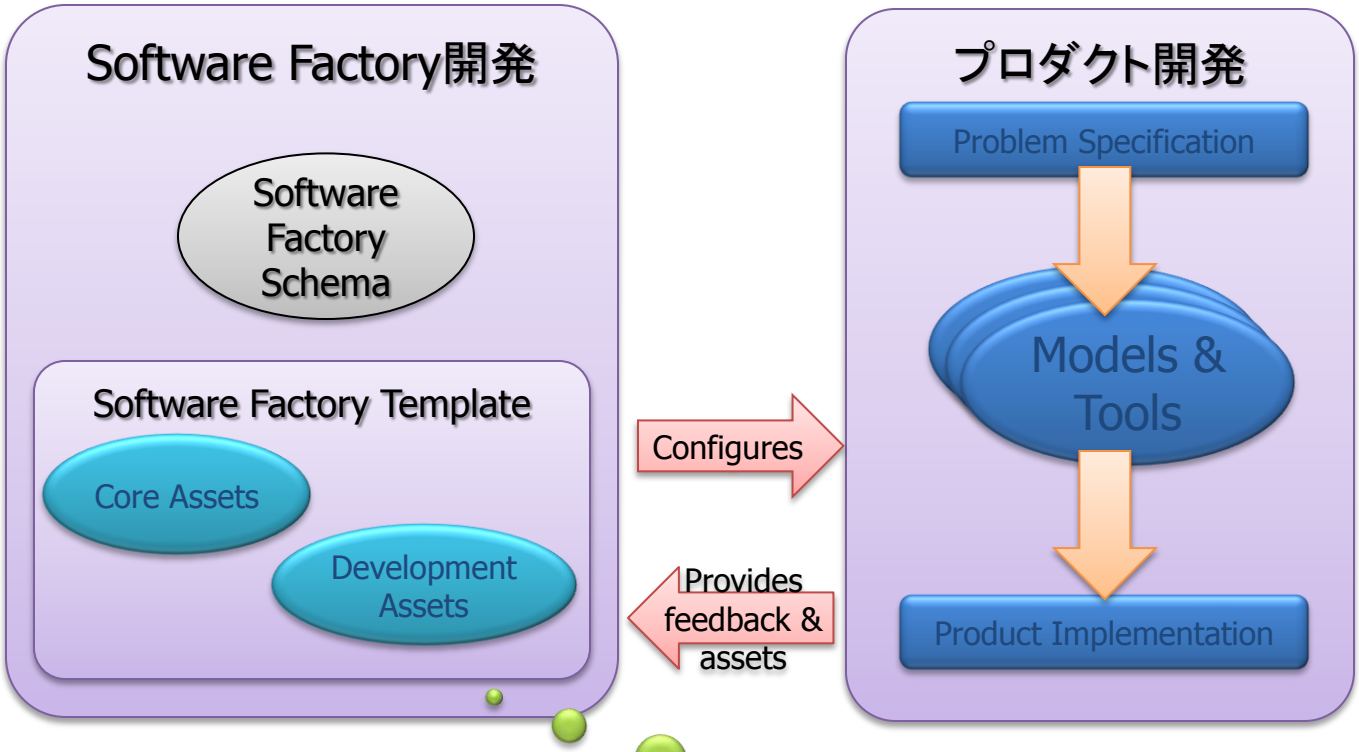


- ドメイン特化の開発プロセスと、それを支援するカスタム開発環境を作成する方法論

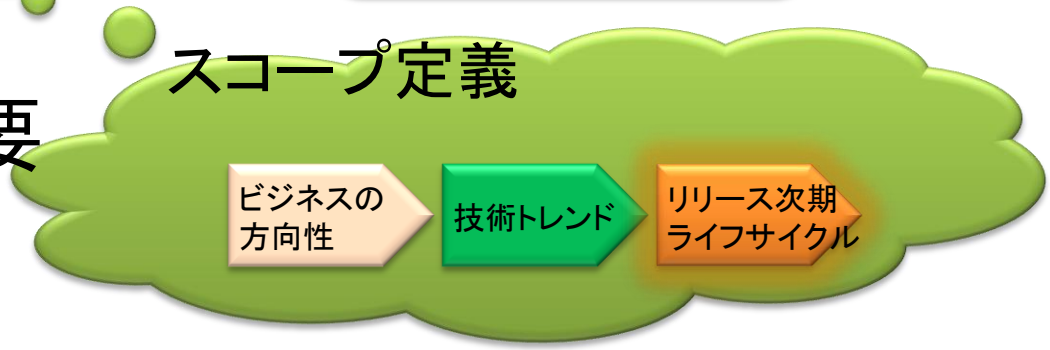
- Software Factoriesを支える4本柱
  - ソフトウェアプロダクトライン
  - モデル駆動型開発
  - アーキテクチャ フレームワーク
  - 文脈によるガイダンス

Windows Embedded

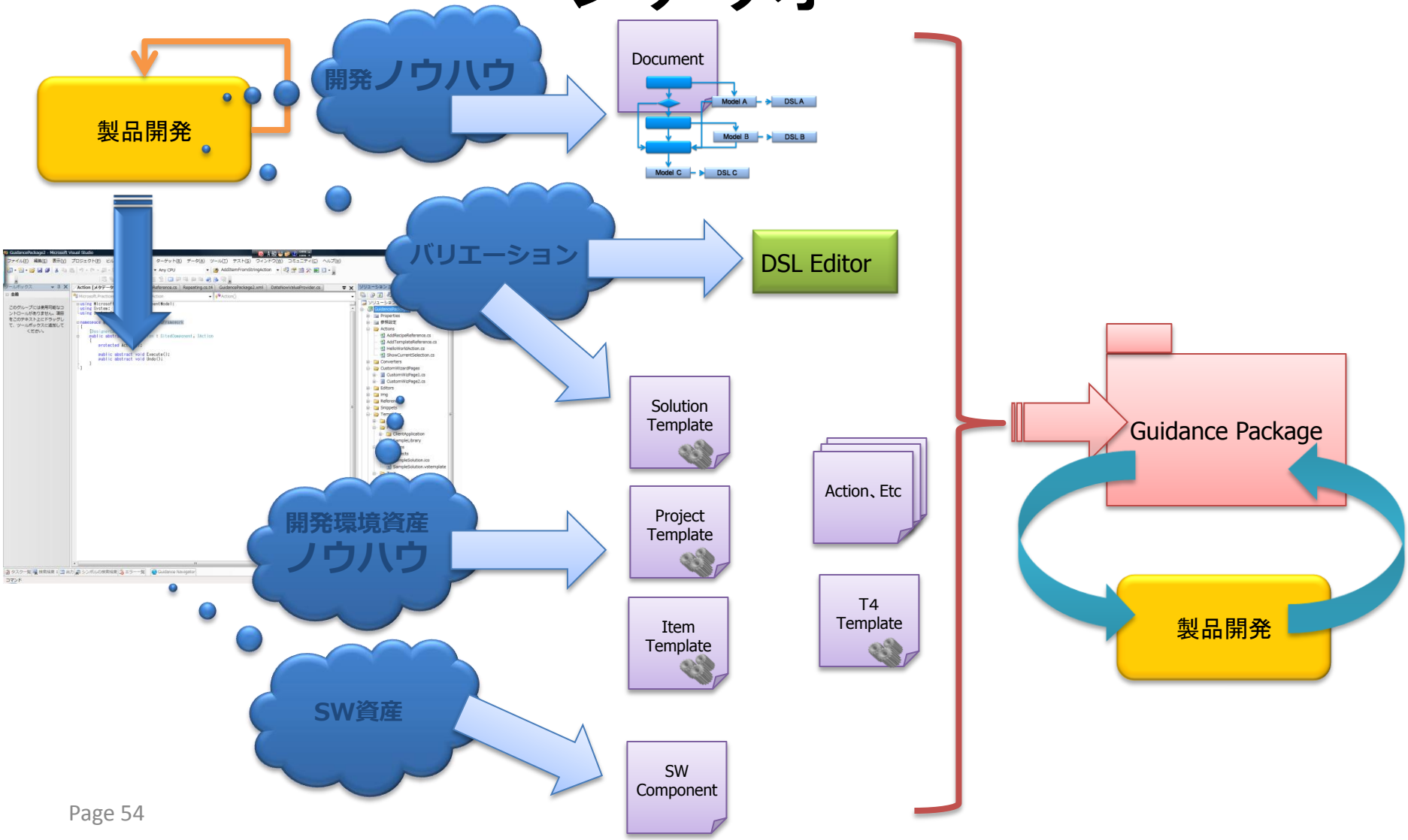
High level Software Factories development process



ビジネス戦略が重要



Visual StudioによるSoftware Factory化シナリオ

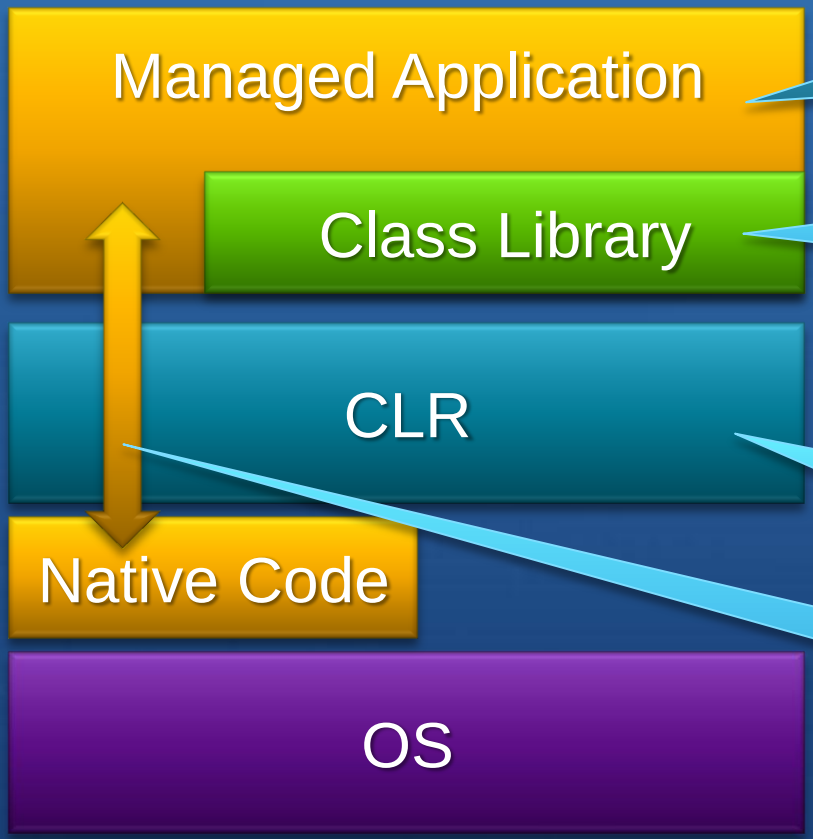


.NET Technology

- .NETの本質
  - モデル化の宝庫
  - アプリケーションレベルの一般化
  - 実装レベルの抽象化
  - 目的毎、役割ごとのView（プログラミング言語、エディタ）
  - WCF, Sensor& Location
- XAML
- LINQ
- Parallel & Concurrent

.NETとは

- Component Based
- Meta Data



目的に応じたプログラミング言語＋開発支援環境

システム開発に必要なソフトウェア部品

プラットフォーム特性に応じたプロファイル
Micro, Compact, Full

Managed/Native Interoperability

.NET Framework 3.0

Windows Presentation Foundation
Next generation user experience

Windows Communication Foundation
Service-oriented development

Windows Workflow Foundation
Business process modeling

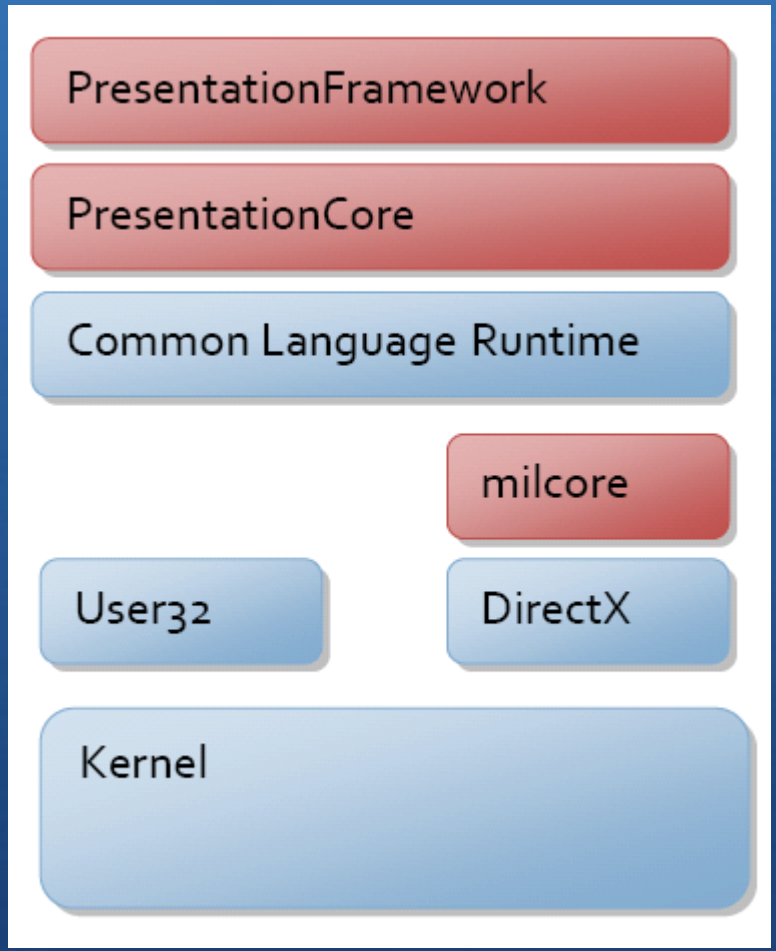
Windows CardSpace
Digital identity management

Windows XP Embeddedで利用可能

WPF：Windows Presentation Foundation

- 次世代プレゼンテーションシステム
  - 魅力的な外観のユーザーエクスペリエンスを持つWiodnwsクライアントアプリケーションを作成するための基盤技術
- 特徴
  - 最新のグラフィックスハードウェアを利用
  - XAMLによる宣言的プログラミング
  - プレゼンテーションと制御動作の分離
  - リッチなコントロール部品

WPFのアーキテクチャ



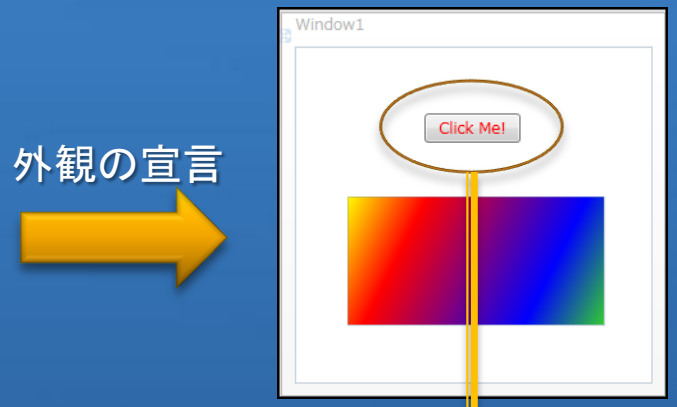
- UI要素の統合
  - System.Windows.UIElement
- バインディング
- イベントルーティング
- パネルのレイアウト
- ２D描画機能
- ３D描画機能
- メディア機能

XAMLの主な特徴

XAML

```xaml
<Window x:Class="WpfApplication2.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300">
    <Window.Resources>
        <TransformGroup...>
        <Style TargetType="Button">
            <Setter Property="Foreground" Value="Red"/>
        </Style>
        <Storyboard x:Key="myStoryboardRectangleOpacity">
            <DoubleAnimation
                Storyboard.TargetName="myRectangle"
                Storyboard.TargetProperty="Opacity"
                From="1.0" To="0.0" Duration="0:0:5"
                AutoReverse="True" RepeatBehavior="Forever"
                ></DoubleAnimation>
        </Storyboard>
    </Window.Resources>
    <Grid>
        <Button
            Height="23" Margin="101,52,102,0"
            Name="button1"
            VerticalAlignment="Top" OpacityMask="Cyan"
            Click="button1_Click"
            >Click Me!</Button>
        <Rectangle x:Name="myRectangle" Margin="41,117,37,45"
                   RenderTransform="{StaticResource MyTransformGroupResource}">
            <Rectangle.Fill>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                    <GradientStop Color="Yellow" Offset="0.0" />
                    <GradientStop Color="Red" Offset="0.25" />
                    <GradientStop Color="Blue" Offset="0.75" />
                    <GradientStop Color="LimeGreen" Offset="1.0" />
                </LinearGradientBrush>
            </Rectangle.Fill>
            <Rectangle.Triggers>
                <EventTrigger RoutedEvent="Rectangle.MouseEnter">
                    <BeginStoryboard>
                        <Storyboard x:Name="rotateSB">
                            <DoubleAnimation
                                Storyboard.TargetName="myRectangle"
                                Storyboard.TargetProperty="RenderTransform.Children[1].Angle"
                                From="0" To="360" Duration="0:0:10"
                                AutoReverse="True" RepeatBehavior="Forever"></DoubleAnimation>
                        </Storyboard>
                    </BeginStoryboard>
                </EventTrigger>
            </Rectangle.Triggers>
        </Rectangle>
    </Grid>
</Window>
```

外観の宣言



クリック時のハンドラ

C#

```csharp
/// <summary>
/// Window1.xaml の相互作用ロジック
/// </summary>
public partial class Window1 : Window
{
    private Storyboard _storyboardRectangleOpacity;
    private bool _animation = false;
    public Window1()
    {
        InitializeComponent();
        _storyboardRectangleOpacity =
            (Storyboard)this.Resources["myStoryboardRectangleOpacity"];
    }

    private void button1_Click(object sender, RoutedEventArgs e)
    {
        if (_animation)
        {
            _animation = false;
            _storyboardRectangleOpacity.Stop(myRectangle);
        }
        else
        {
            _animation = true;
            _storyboardRectangleOpacity.Begin(myRectangle,true);
        }
    }
```

開発環境

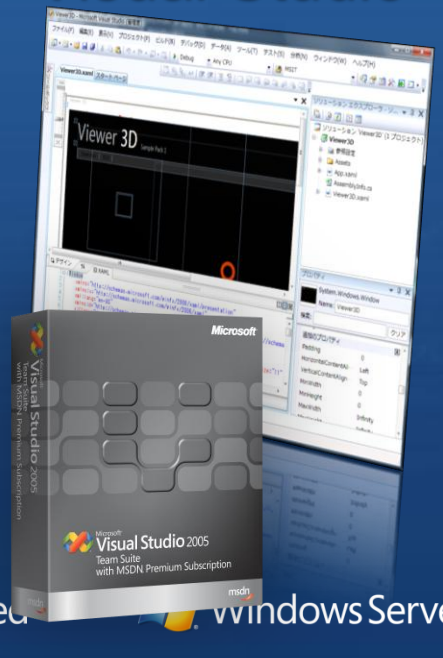
- ExpressionとVisual Studioによる分離開発

デザイナー

Expression

開発者

Visual Studio



通信の3要素→WCF

Address

- どこへ？
- エンドポイント：=ベースアドレス＋相対アドレス

Binding

- トランスポート
- エンコード

Contract

- クライアントへのサービスのインターフェース情報
  - Service Contract、Operation Contract
  - Data Contract、Error Contract、Message Contract

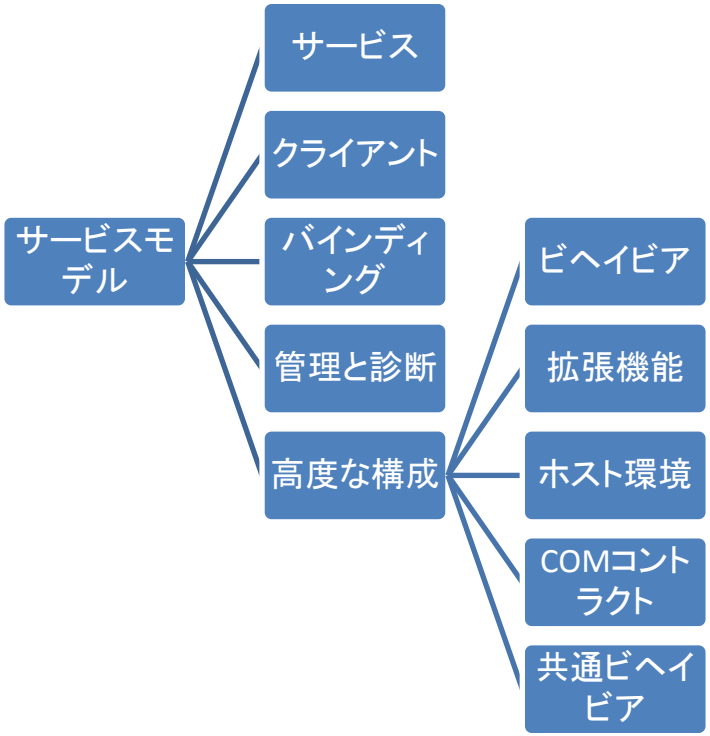
Binding

Binding	構成	Security	Session	Transaction	2 Way
BasicHttpBinding	Basic Profile	N/A	N/A	N/A	N/A
WSHttpBinding	WS-*	Message	Option	○	—
WSDualHttpBinding	WS-*	Message	○	○	○
WSFederationHttpBinding	WS-Federation	Message	○	○	N/A
NetTCPBinding	.NET	Transport	Option	○	○
NetNamedPipeBinding	.NET	Transport	○	○	○
NetMsmqBinding	.NET	Transport	×	○	×
NetPeerTcpBinding	Peer	Transport	×	×	○
MsmqIntegrationBinding	MSMQ	Transport	○	○	×

※「これからはじめるWCFプログラミング」から抜粋
※他にも多数あり。.NET Frameworkが3.0、3.5、3.5SP1、4.0とバージョンアップするたびに増えている

Service Configuration

- App.configで設定
  - XMLファイルの直接編集
  - SvcConfigEditor.exeの利用

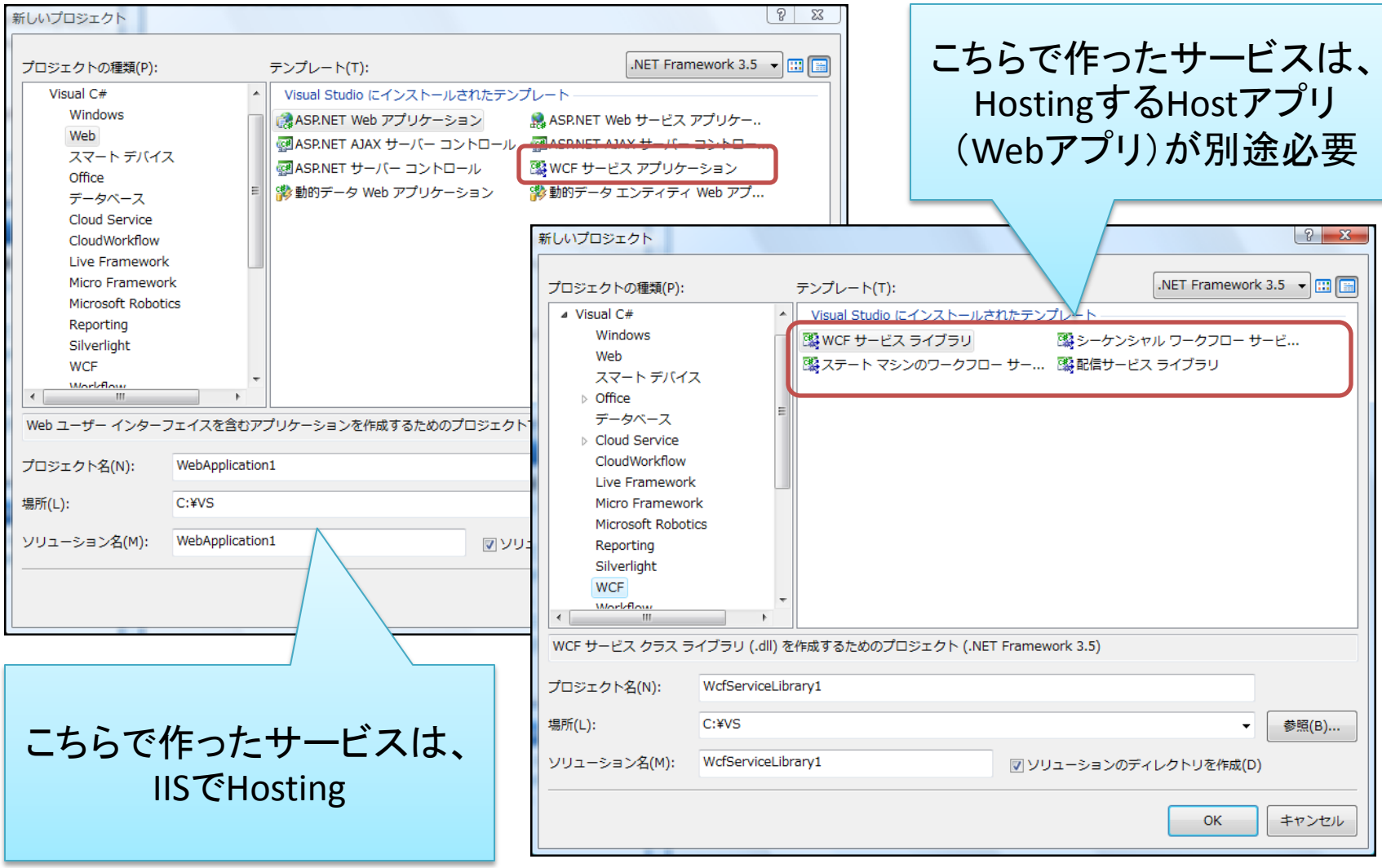


WCF Hosting

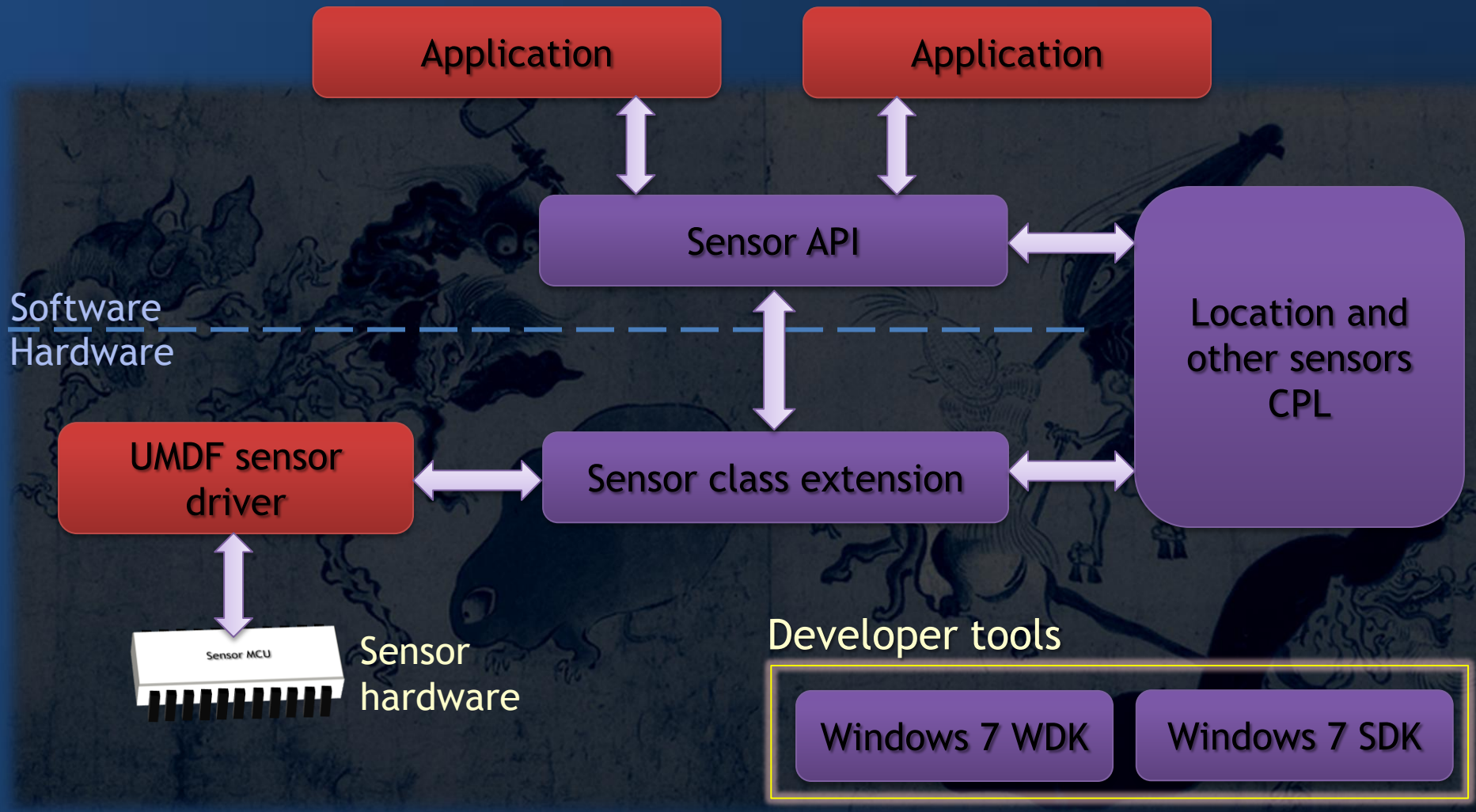
	Managed Windows App	Managed Windows Service	IIS5・1/6.0	WAS（IIS7.0）
使用可能なプロトコル	全て	全て	HTTP	HTTP、TCP、Named-Pipe、MSMQ
UI	あり	なし	なし	なし
実行環境	ログオン環境	バックグラウンド	バックグラウンド	バックグラウンド
起動/停止	ログオンユーザー	サービスコントロールMgr	IIS	IIS（WAS）
自己ホスト	可能	可能	IIS依存	IIS依存
Windows XP	○	○	○	○
Windows Vista	○	○		○
Windows Server	2003、2008	2003、2008	2003（6.0）	2008

Visual Studio 2008による開発

- サーバー側

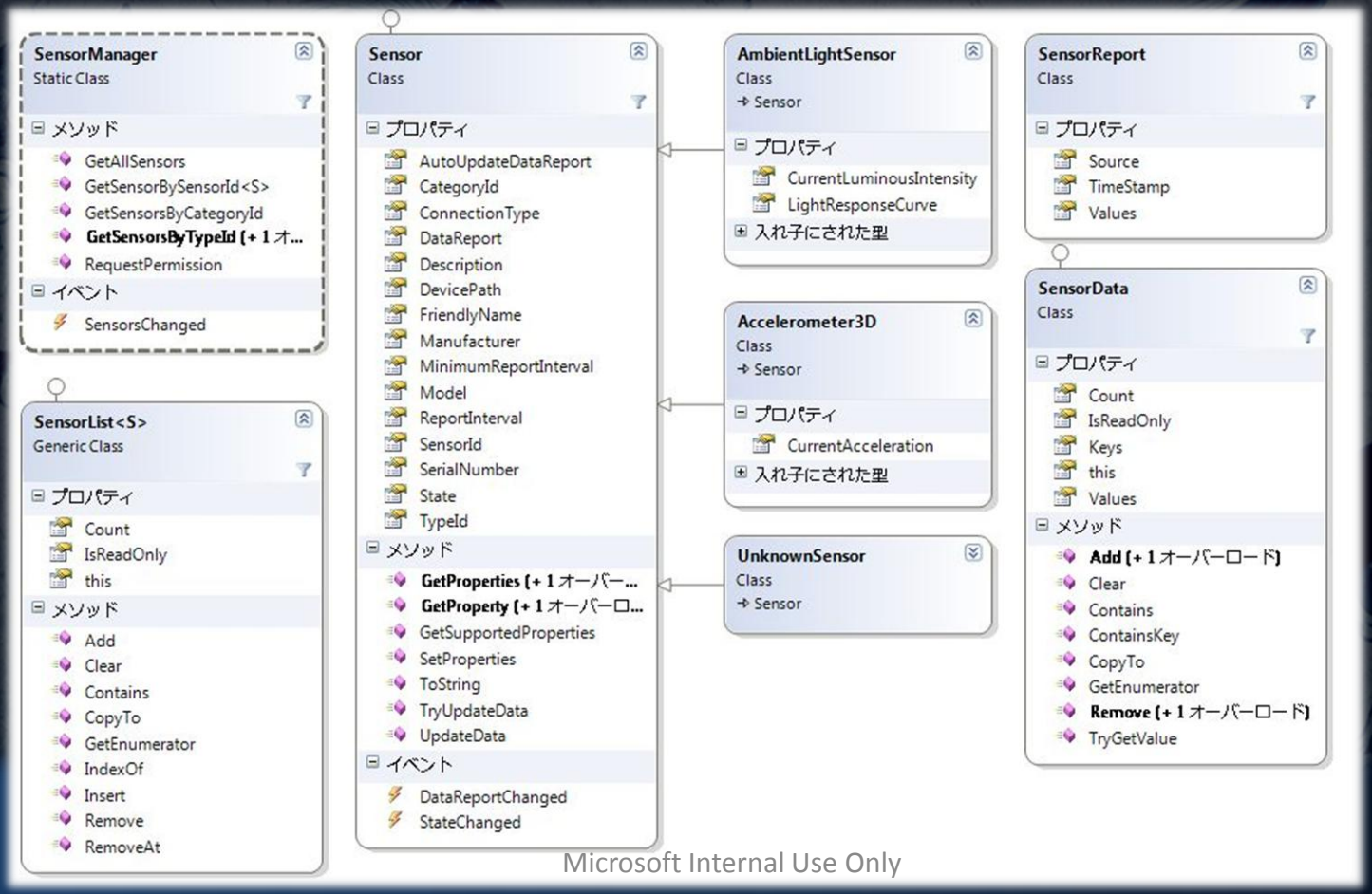


Sensor Components



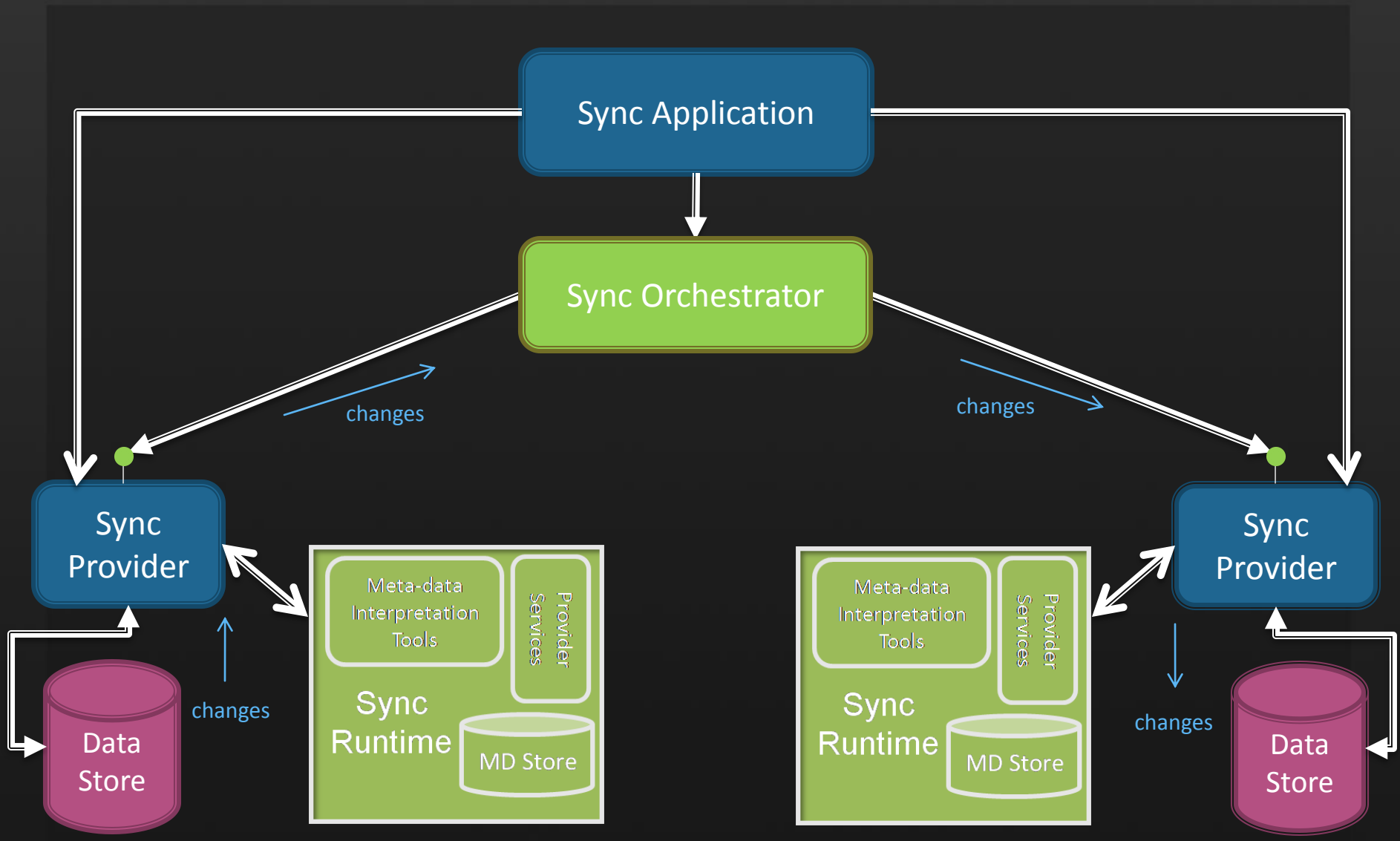
Managed Library

- Windows API Code Packで提供

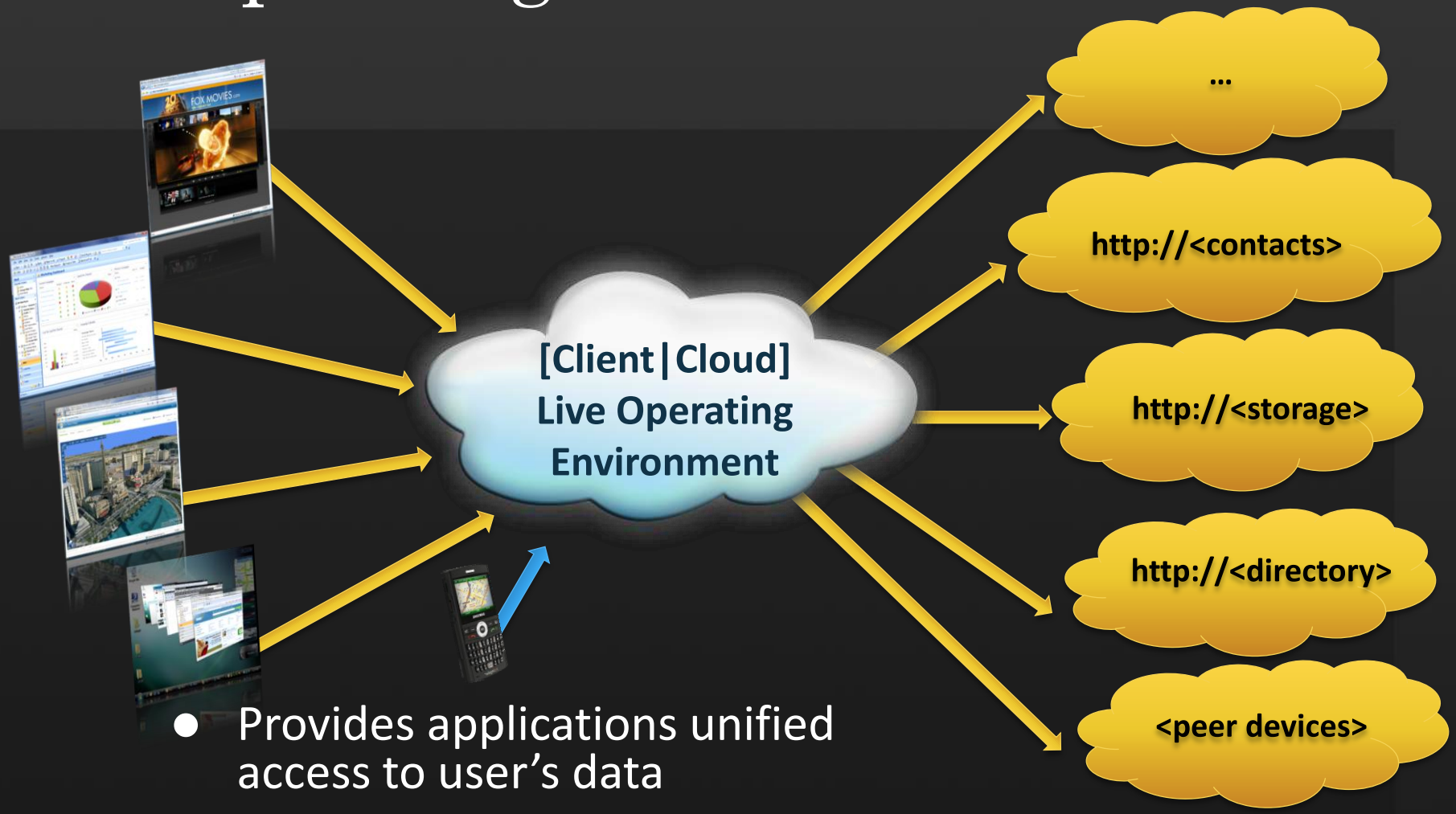


Using Microsoft Sync Framework

How synchronization happens

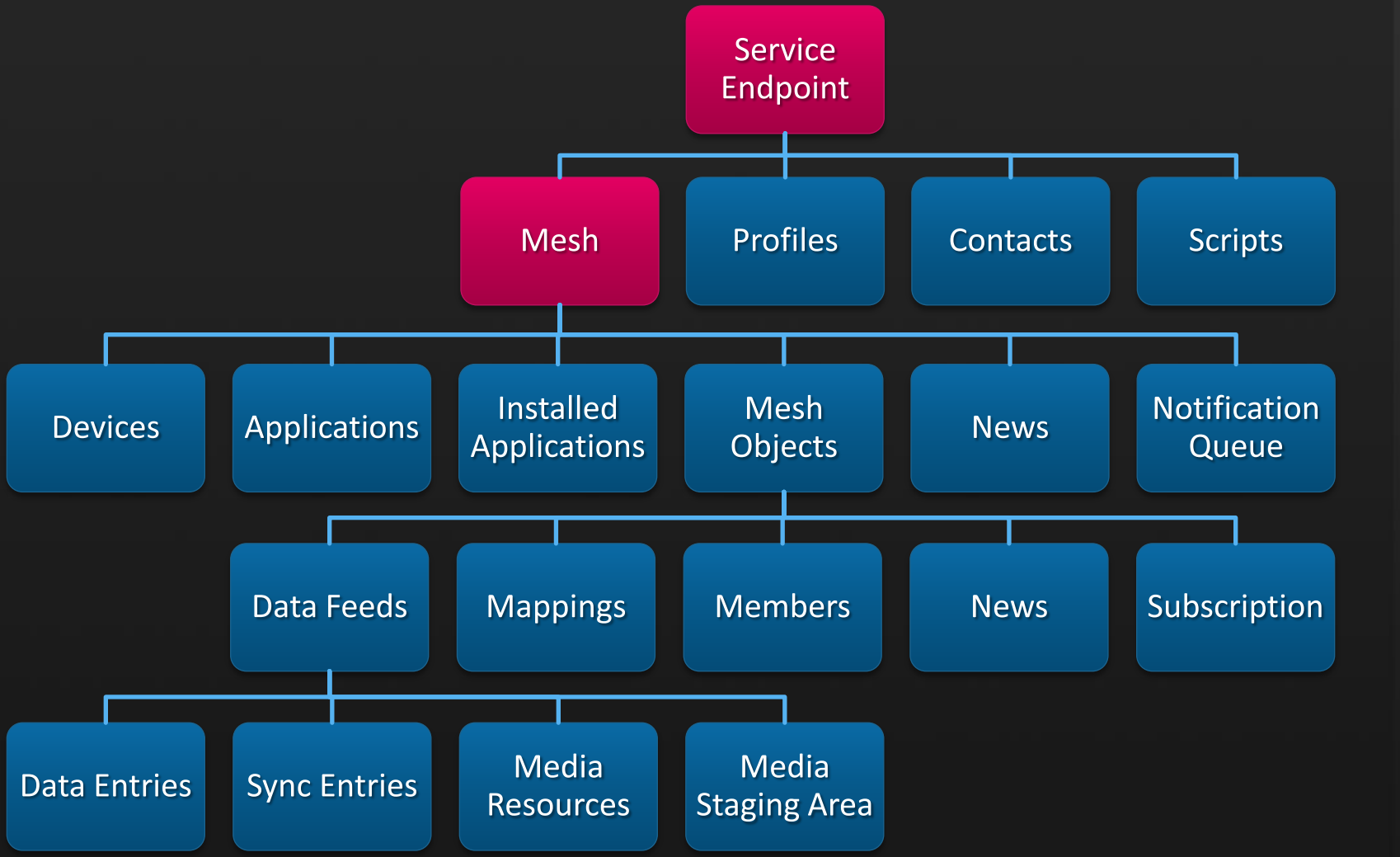


Live Operating Environment

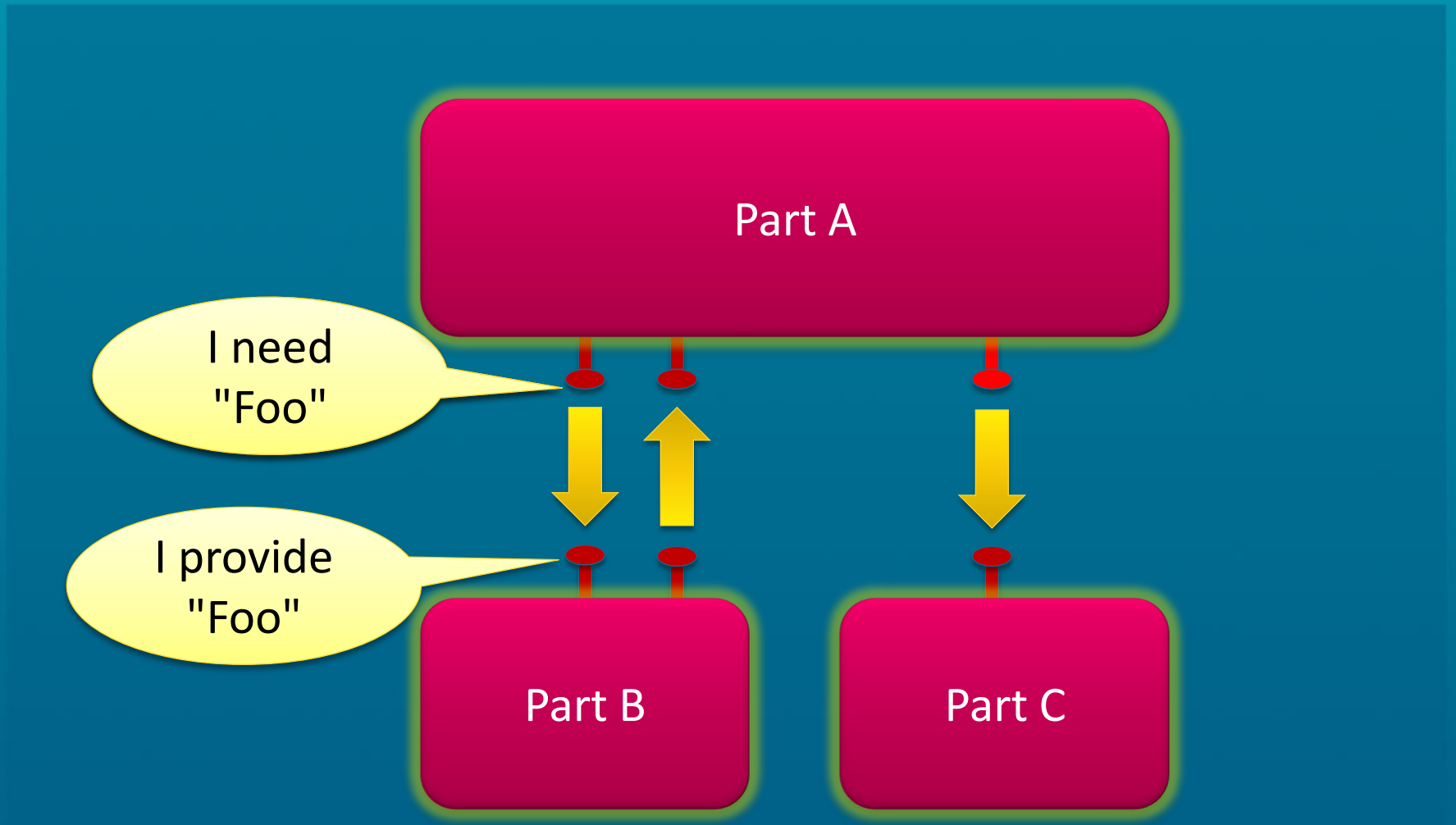


- Provides applications unified access to user's data
- Is everywhere
  - On local device (Client LOE)
  - In the cloud (Cloud LOE)
  - All incarnations share an identical architecture

Resource Model

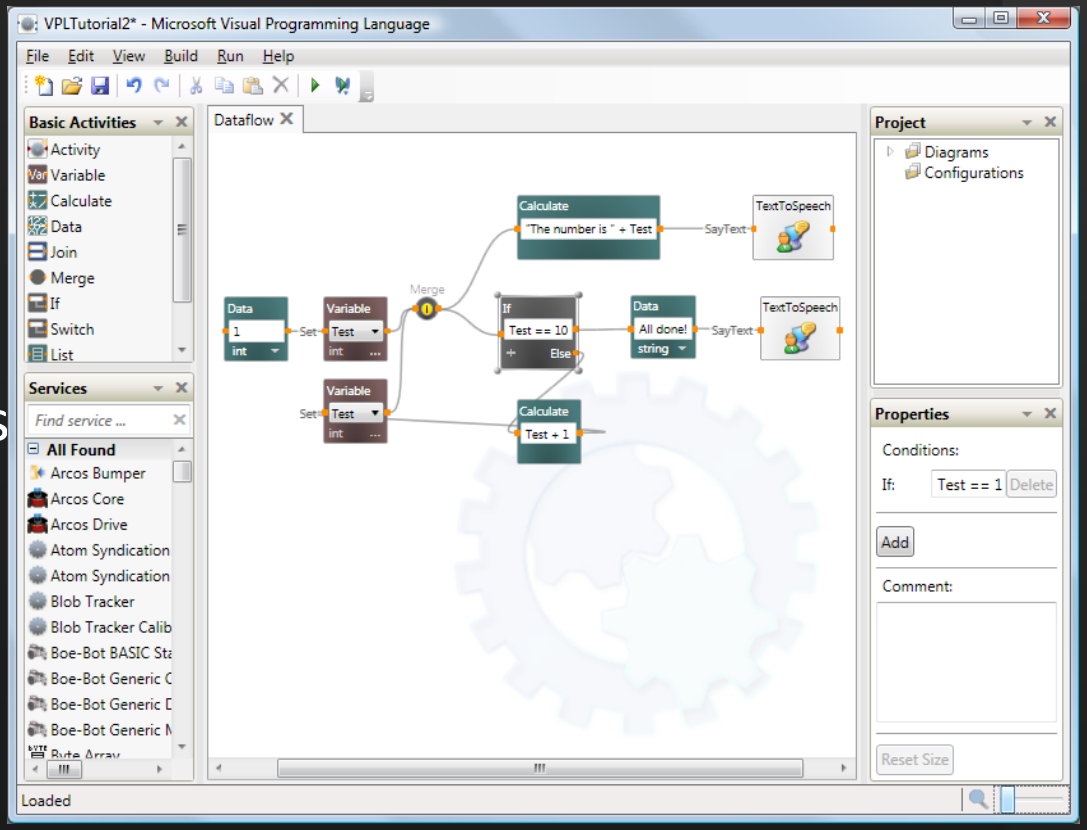


MEF: Managed Extensibility Framework

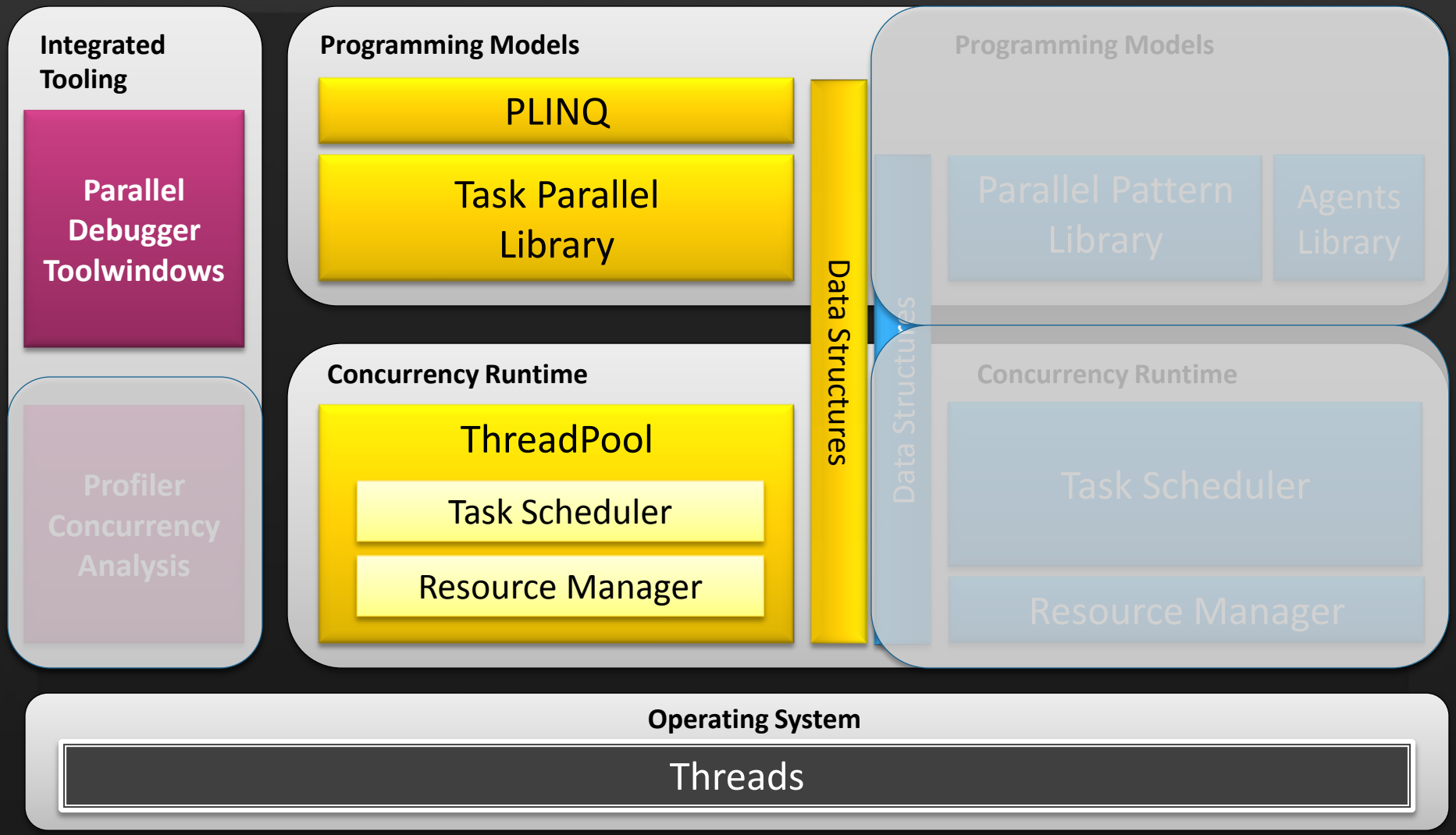


Visual Programming Language

- Dataflow-based development
  - Simple drag and drop composition
  - Services as blocks
  - Connections = messages
- Novice to expert
- Extensible
- Optional C# codegen



Visual Studio 2010
Tools / Programming Models / Runtimes

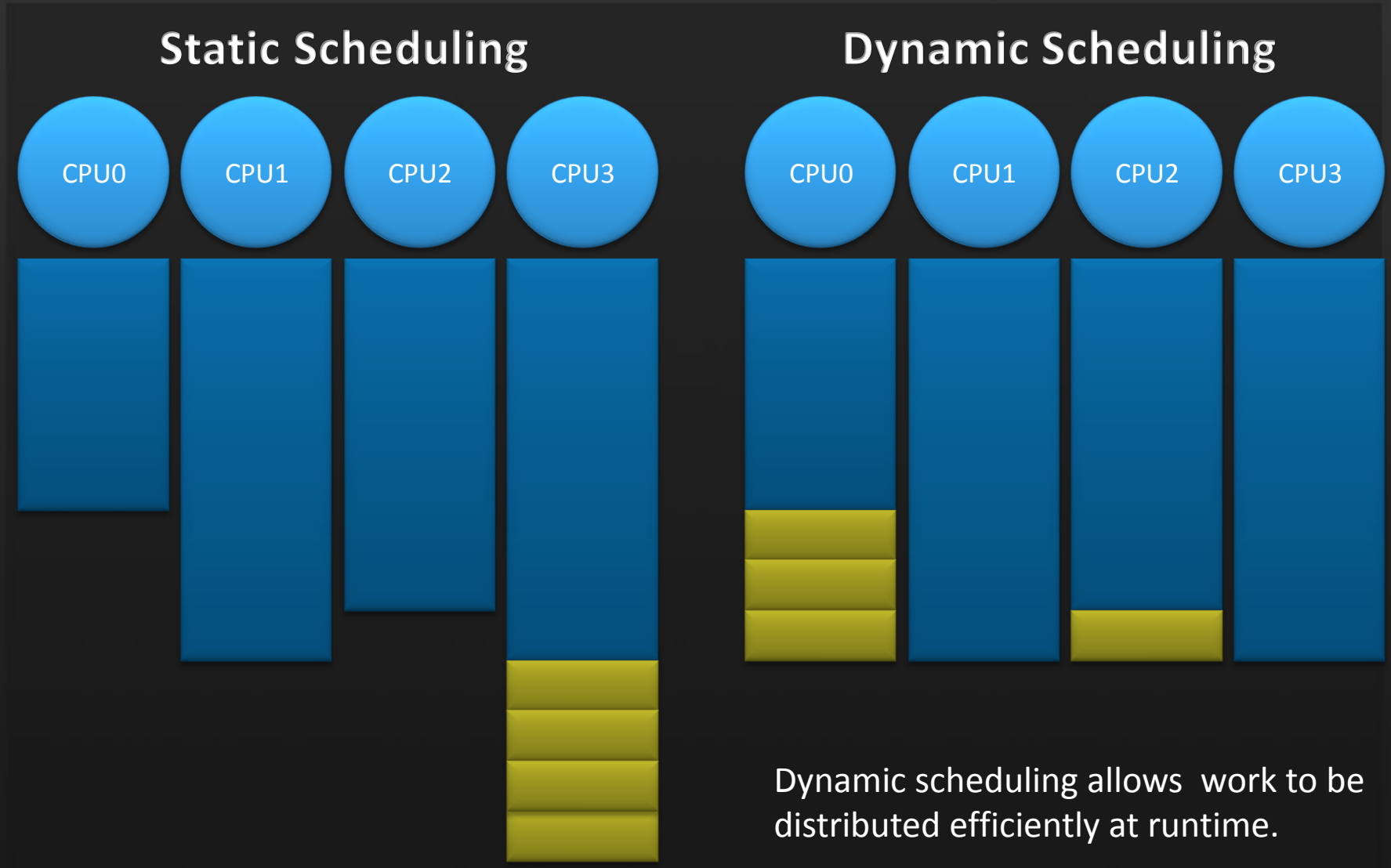


Manual Parallel Solution

```
void MatrixMult(
  int size, doubl                ble** m2, double** result)
 int N = size;
 int P = 2 * NUM
 int Chunk = N / P;
 HANDLE hEvent = Crea
 long counter = P;
 for (int c = 0; c < P; c++) {
  std::thread t ([&,c] {
    for (int i = c * Chunk;
       i < (c + 1 == P ? N
     for (int j = 0; j < s
      result[i][j] =
      for (int k
       result[i][j                m2[k][j];
      }
     }
```

- Static partitioning
- tion Knowledge
- Error prone
- Lots of boilerplate
- Tricks
- f thread reuse
- synchronization

Dynamic Load-Balancing



Dynamic scheduling allows work to be distributed efficiently at runtime.

Debugger Support For Tasks

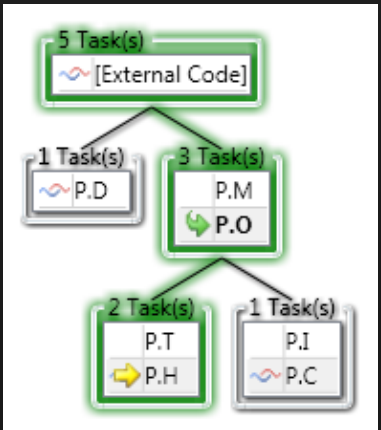
Support both managed and native

1. Parallel Tasks

Tasks

	Id.	Status	Location	Parent Task	Thread
	1242864	Waiting	Concurrency::StructuredTaskPool::Wait		5328
	1963792	Waiting	Concurrency::StructuredTaskPool::Wait	1242864	5328
	1961324	Waiting	Concurrency::StructuredTaskPool::Wait	1963792	5328
	1958136	Waiting	Concurrency::StructuredTaskPool::Wait	1961324	5328
	1954948	Scheduled	0x004110be CChore::Payload(void *)	1958136	
	1955128	Running	State::Remove	1958136	5328
	1243044	Waiting	Concurrency::StructuredTaskPool::Wait		540
	1701972	Waiting	Concurrency::StructuredTaskPool::Wait	1243044	540
	1698964	Waiting	Concurrency::StructuredTaskPool::Wait	1701972	540
	1695776	Scheduled	0x004110be CChore::Payload(void *)	1698964	
	1695956	Waiting	Concurrency::StructuredTaskPool::Wait	1698964	540
	1692108	Scheduled	0x004110be CChore::Payload(void *)	1695956	
	1692288	Running	State::Solve	1695956	540

2. Parallel Stacks



Also supports traditional threading

Combinable Objects
Computing Sums

- int ***sum*** = 0;
- for (vector<int>::iterator it = myVec.begin(); it != myVec.end(); ++it) {
- int element = *it;
- SomeFineGrainComputation(element);
- ***sum*** += element;
- }

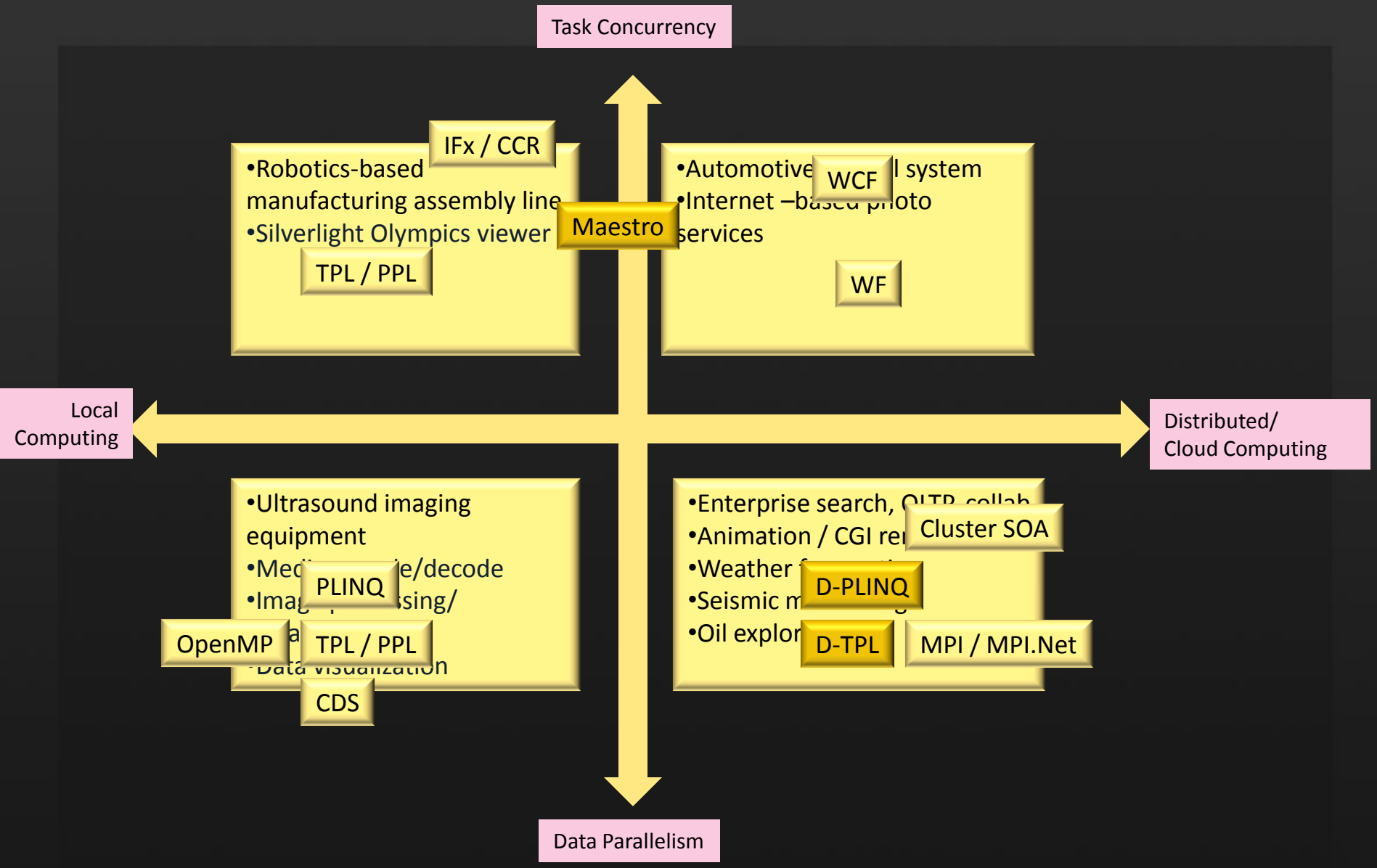
Declarative Data Parallelism

Parallel LINQ-to-Objects (PLINQ)

- Enables LINQ devs to leverage multiple cores
- Fully supports all .NET standard query operators
- Minimal impact to existing LINQ model

```
var q = from p in people.AsParallel()
        where p.Name == queryInfo.Name &&
              p.State == queryInfo.State &&
              p.Year >= yearStart &&
              p.Year <= yearEnd
        orderby p.Year ascending
        select p;
```

MSFT Parallel Computing Technologies



Microsoft®

離

"Oslo" とは何か?

THE PLATFORM FOR MODEL-DRIVEN APPLICATIONS

"M"

モデルと DSL を作成するための言語

"Quadrant"

モデルや DSL と対話するためのツール

Repository

モデルを配置・共有するためのデータベース

なぜOsloが作られたのか?

TRANSPARENCY

アプリケーションを十分に把握可能に

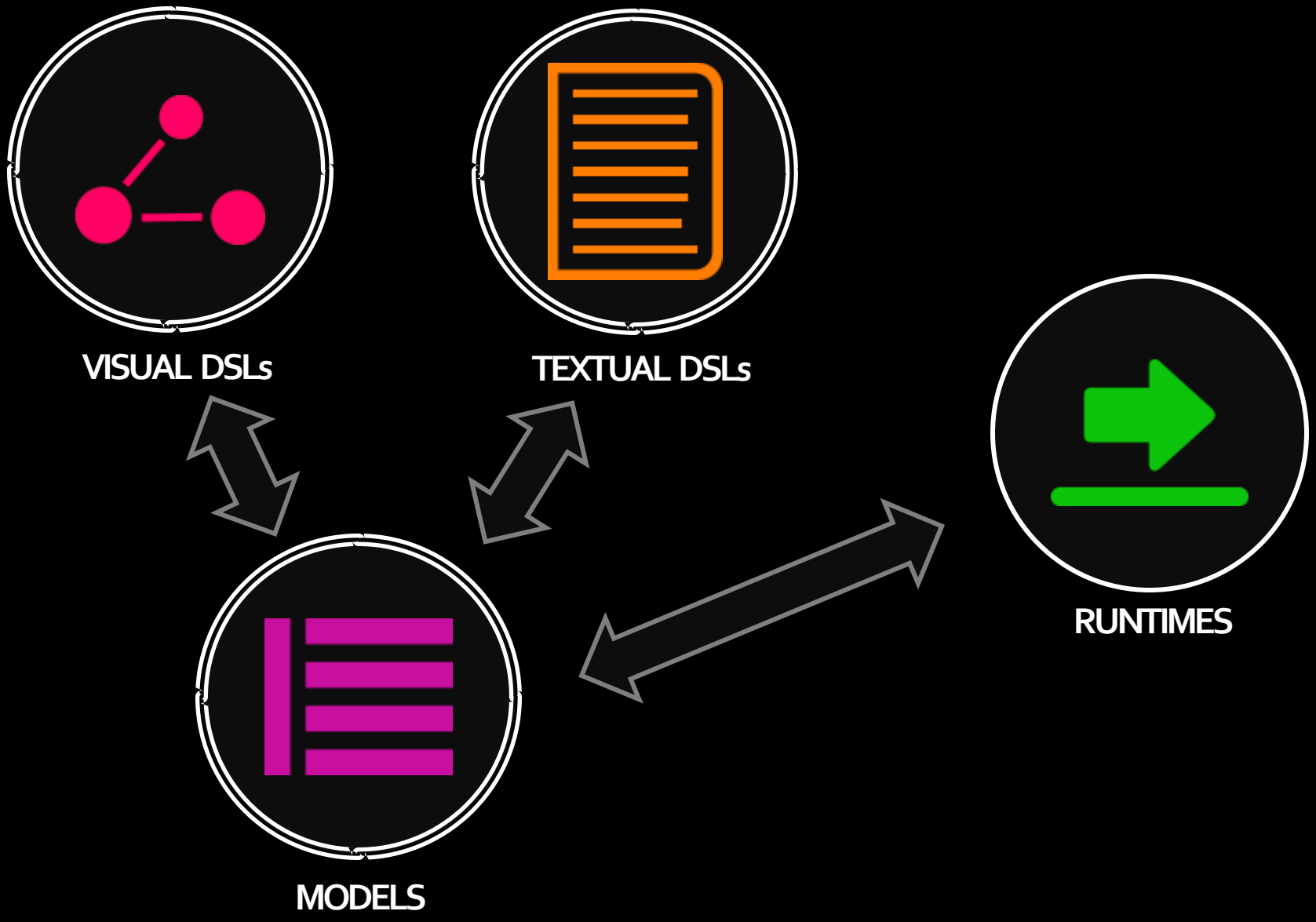
FLEXIBLITY

アプリケーションの変更を迅速に実現

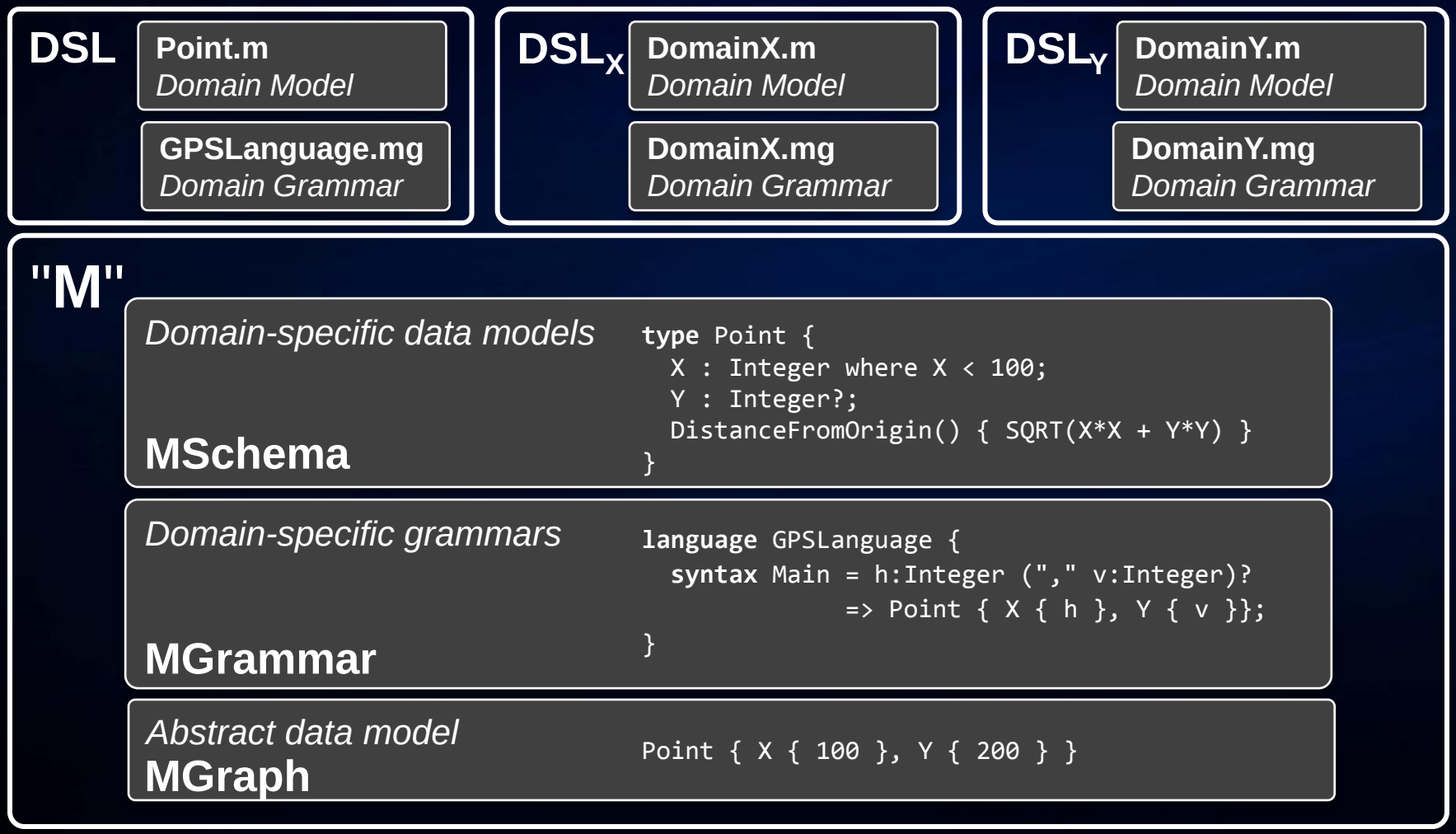
PRODUCTIVITY

より本質に注力し、セレモニーを少なく

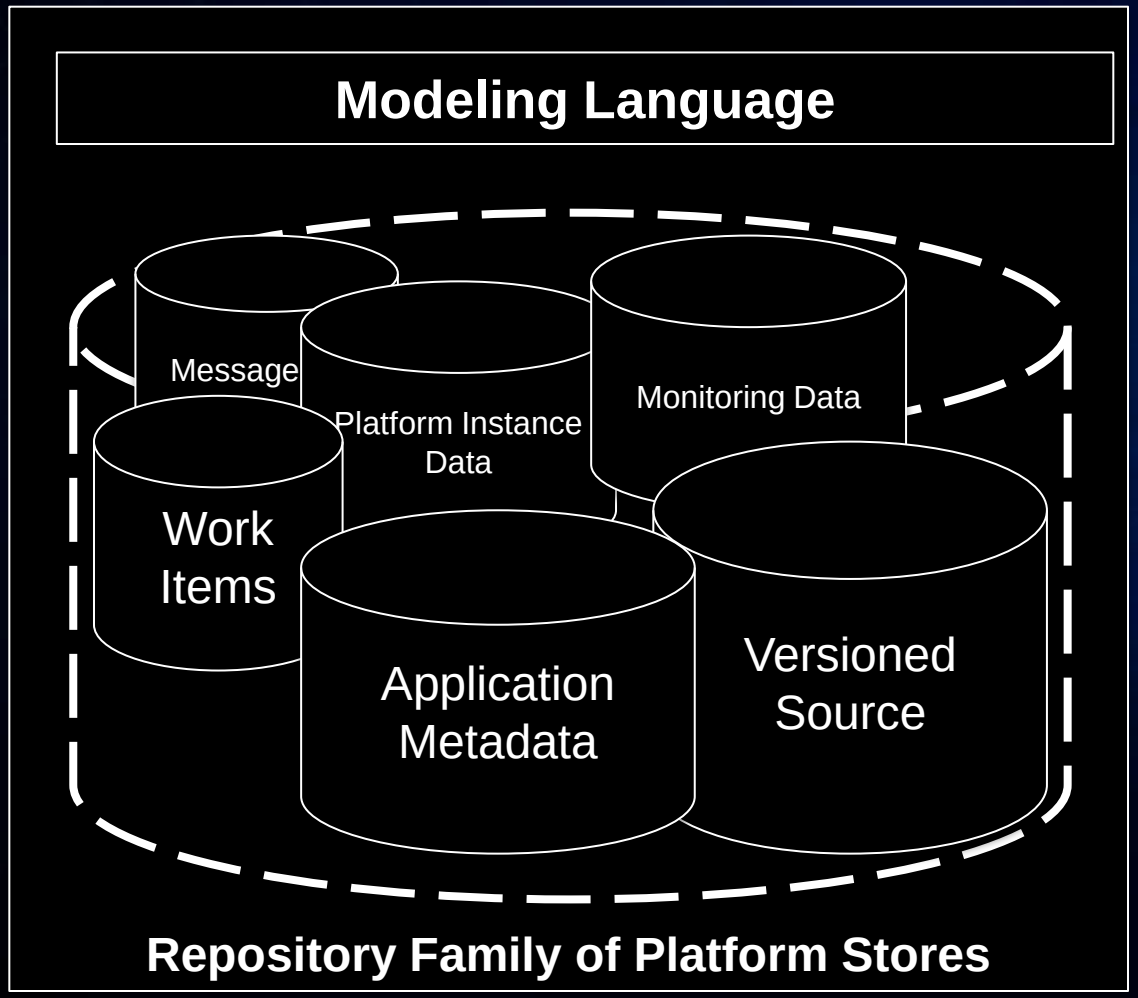
"Oslo" キーコンセプト



The “M” Language



A Repository Database



- モデルの保持と共有向けの最適化
  - 拡張性
  - allows for query, linkage, impact assessment across models
  - 共通タスクサポート バージョン管理、アクセス制御、…
  - 拡張可能なメタデータ
- end-to-end system / lifecycle modelsを網羅
  - out of the box models
  - 設計時, 実行時
- 'Natural' なSQL Server database
  - database ecosystemのテコ入れ: tools, reporting, BI, etc

"Quadrant" Architecture

SERVICES

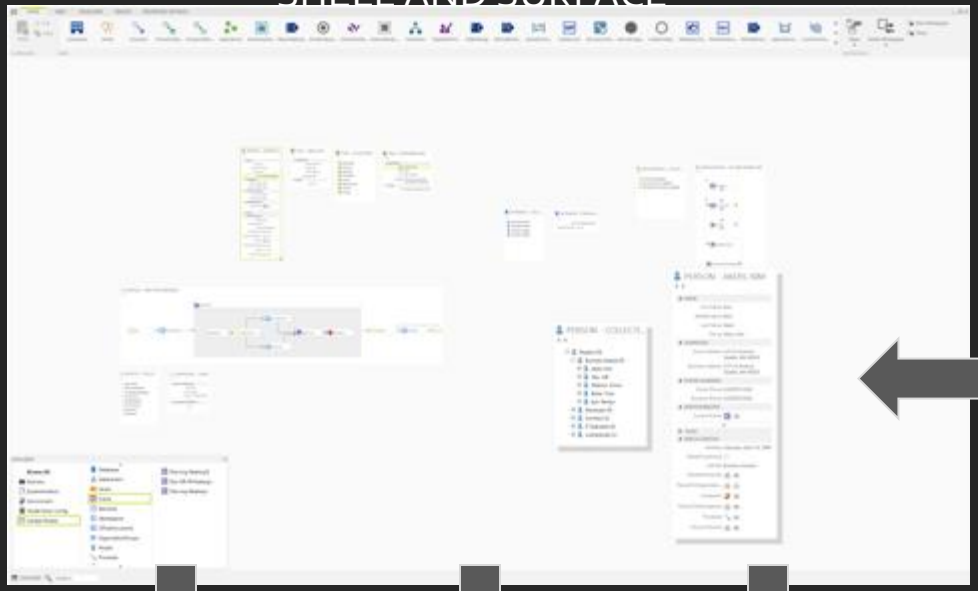
Core Services

Undo/Redo
Commands
Drag/Drop
Selection
Activation
Error Handling

General Services

Search
Validation
Annotations
Relationship
Highlighting

SHELL AND SURFACE



COMPOSITION ENGINE

Nesting
Sizing
Layout
Snapping

DATAFLOW ENGINE

Caching,
Virtualization,
Change tracking, and
Notification

Target Data

Configuration

View State

REPOSITORY

Microsoft®

Your potential. Our passion.™