



John Green

@johndashgreen



Follow

Password must contain a capital letter, a number, a plot, a protagonist with some character development, and a surprise ending.



Reply



Retweet



Favorited



More

5,069

RETWEETS

2,090

FAVORITES



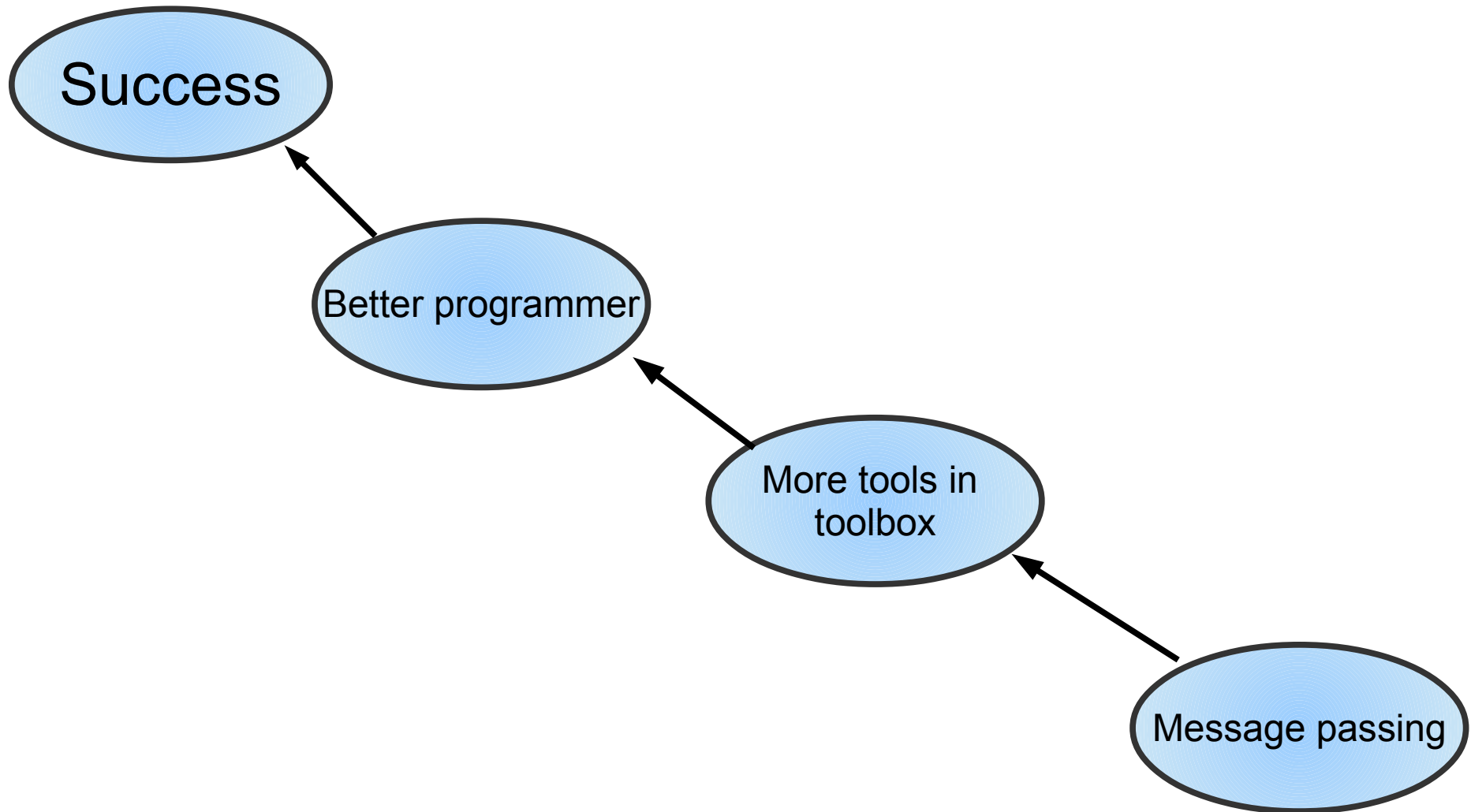
8:03 AM - 2 Apr 13

Message Passing

Peter Seale
@pseale
minimal beard
growth

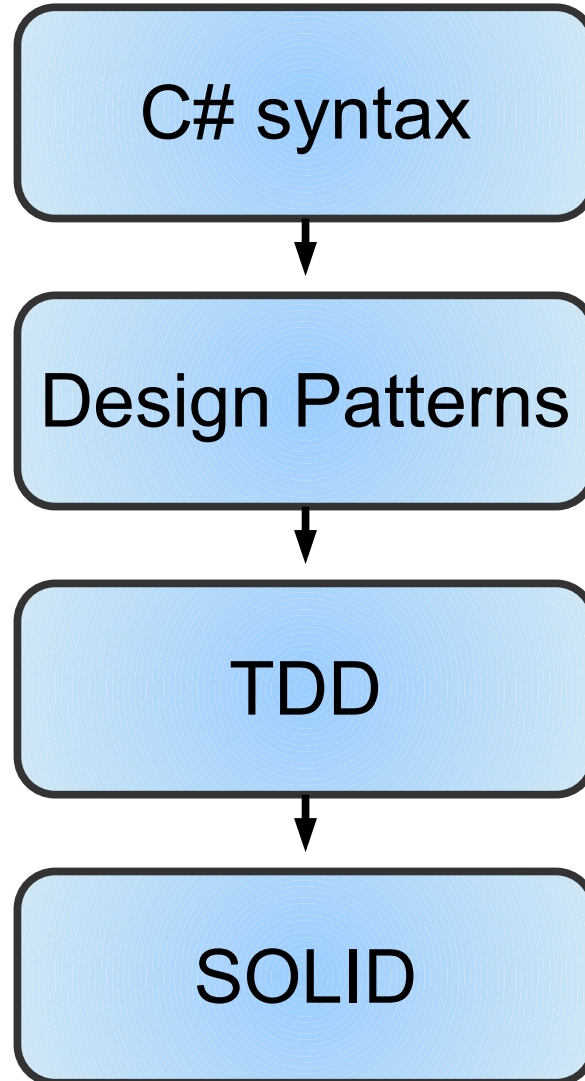


Success in life, through message passing

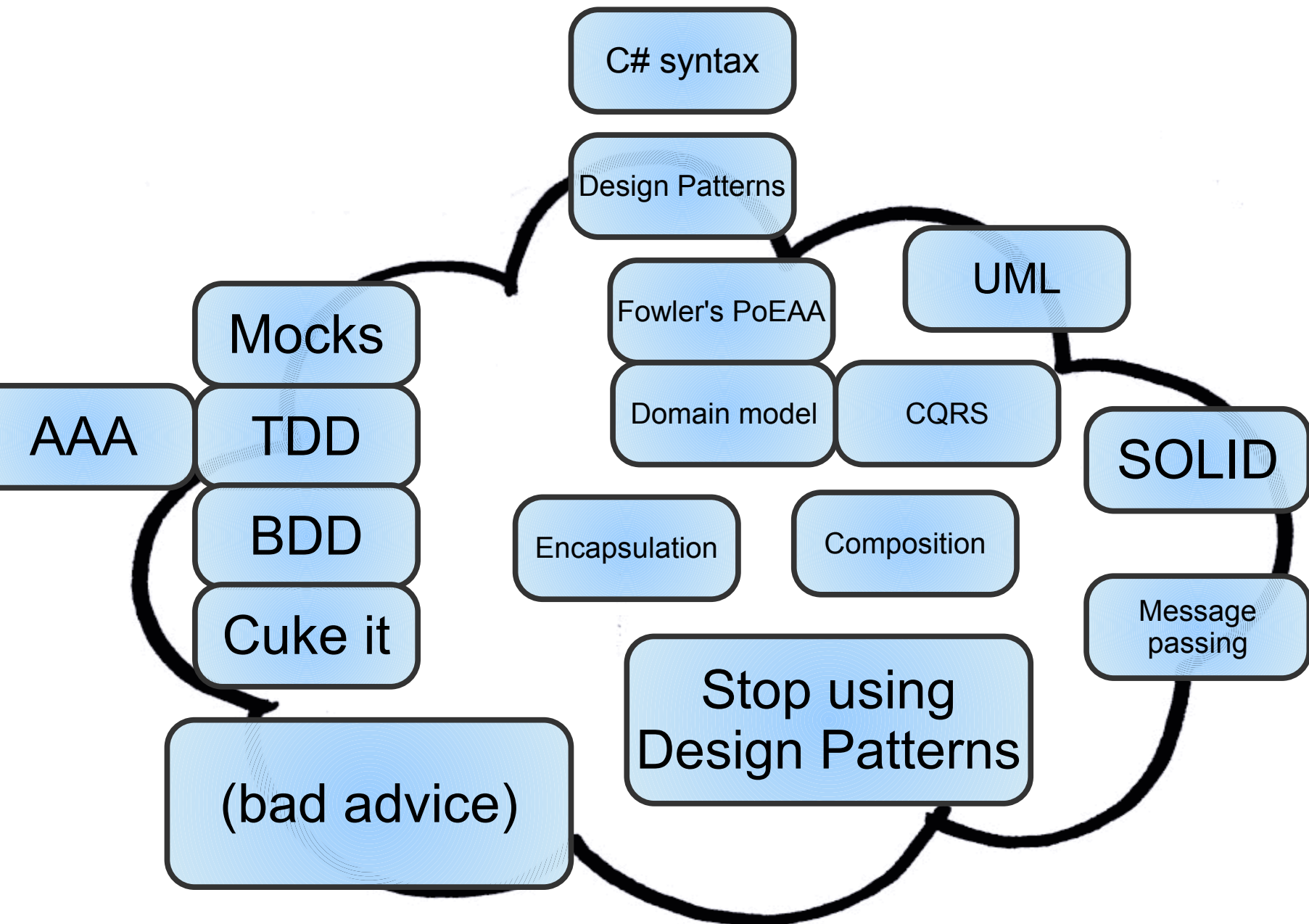


My goal is for you to have a
eureka moment

Learning OOP seems **straightforward**



...but learning OOP is messy



Alan Kay is the father of OOP, so
listen to him

Just a gentle reminder that I took some pains at the last OOPSLA to try to remind everyone that Smalltalk is not only NOT its syntax or the class library, it is not even about classes. I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.

The big idea is "messaging".

What is message
passing?

Wikipedia: “*Message passing is the paradigm of communication where messages are sent from a sender to one or more recipients*”

What makes message passing different from just calling methods?

What are the benefits?

What languages are known
for message passing (*/*
counted 4)?

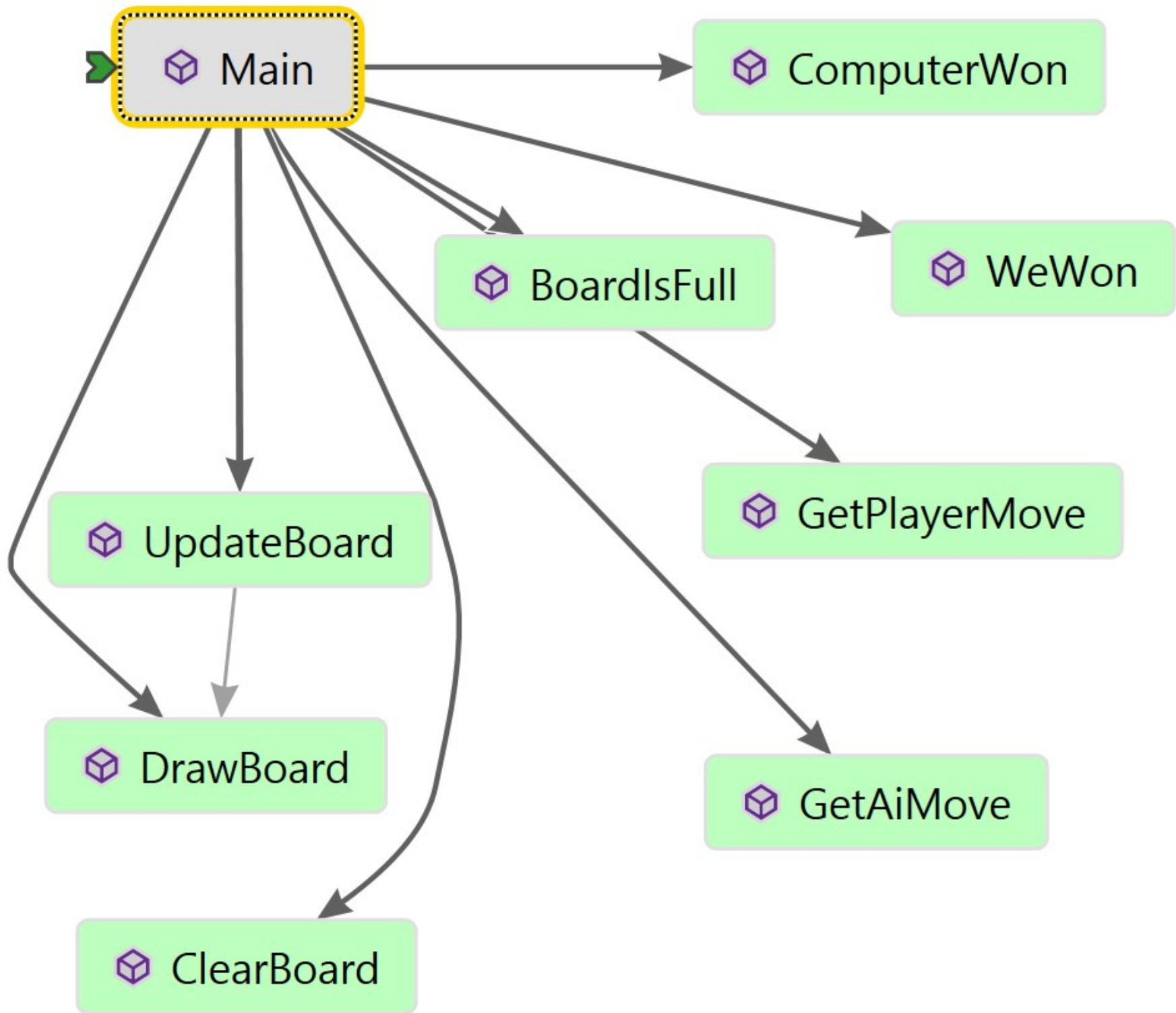
Where have you seen
message passing in .NET (*/*
counted 7)?

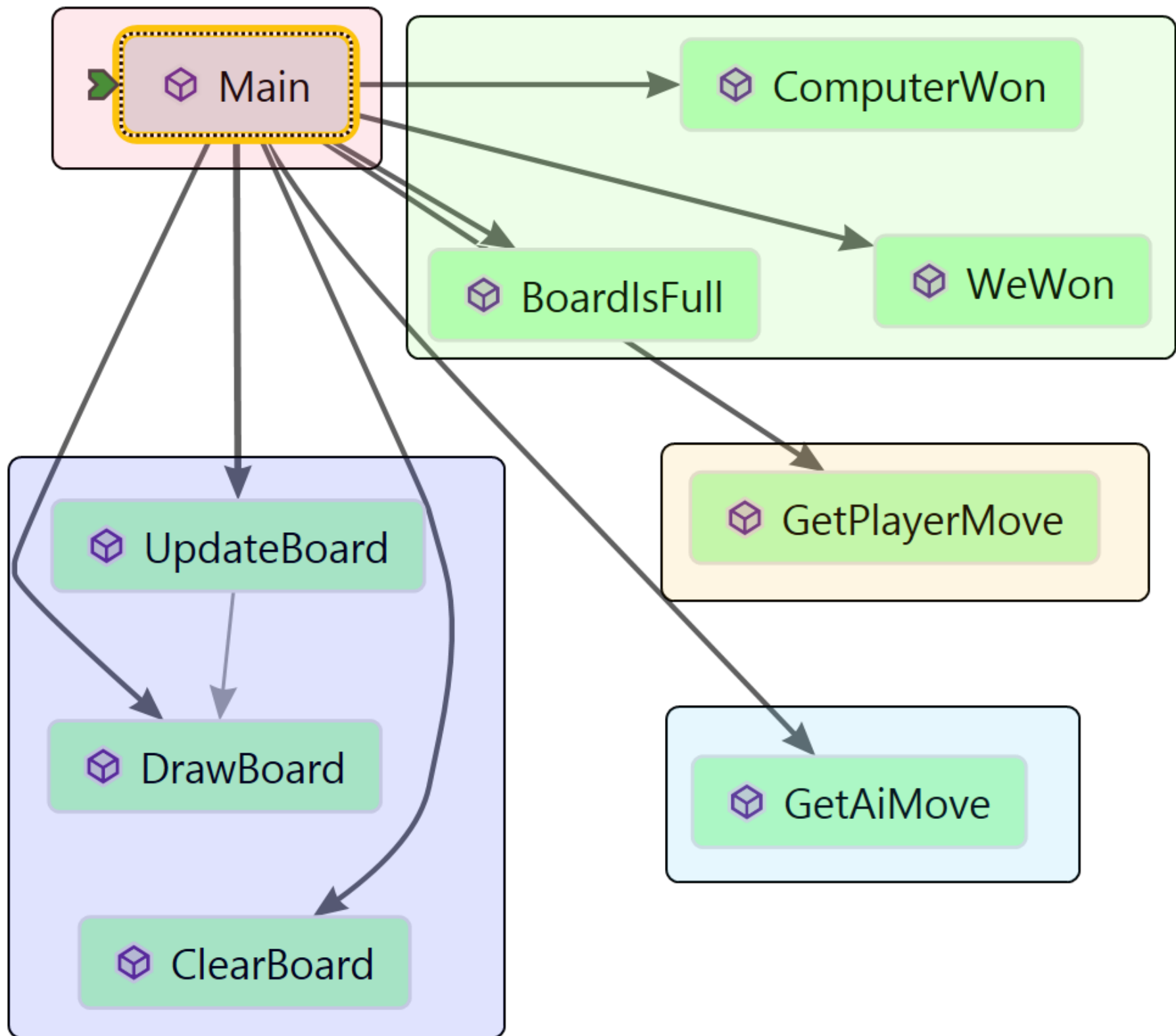
Tic Tac Toe demo
(pay attention)

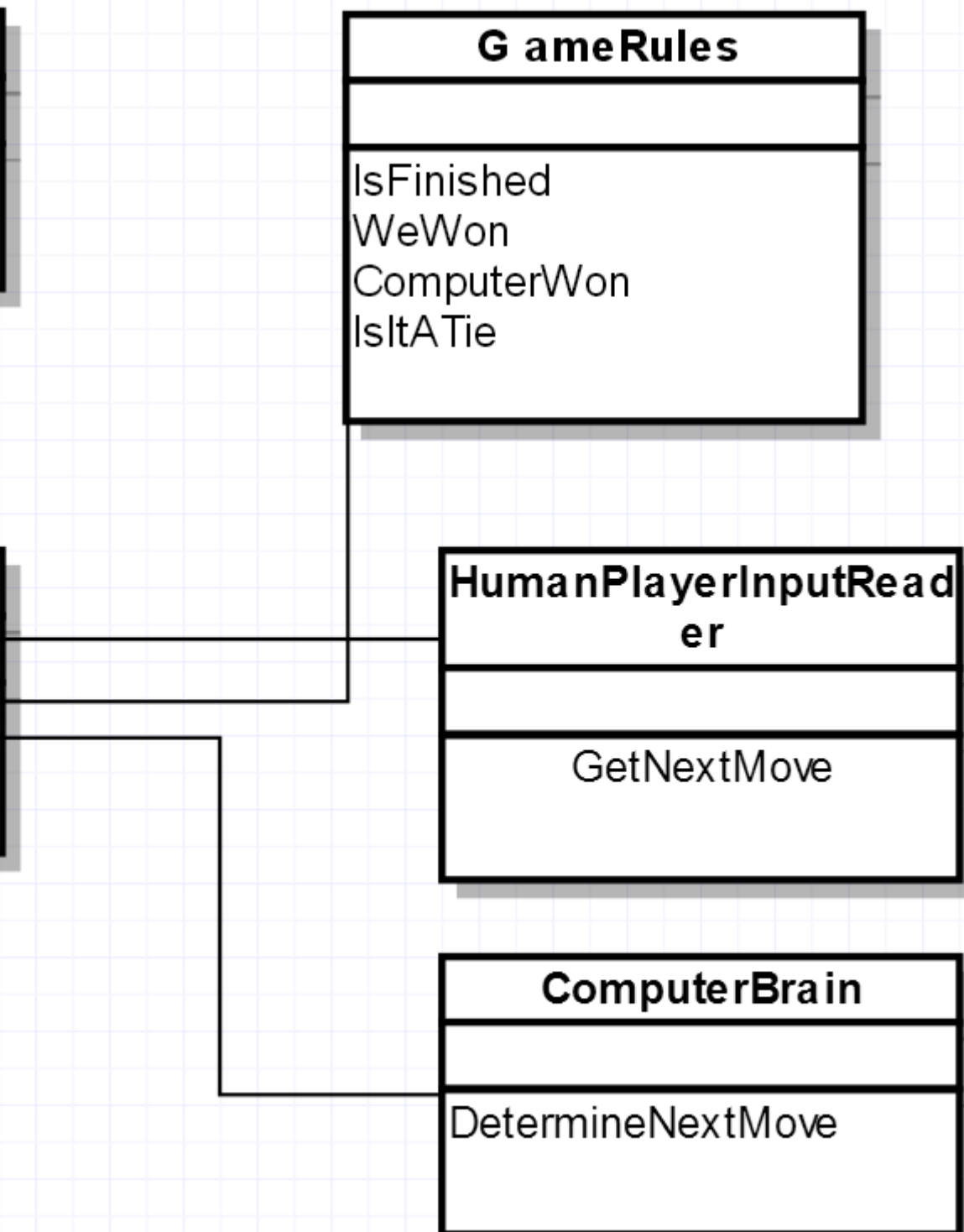
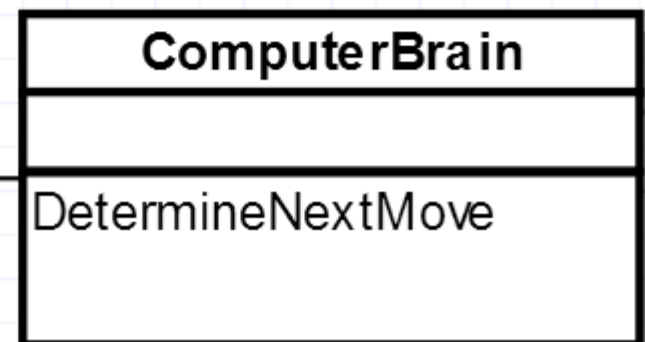
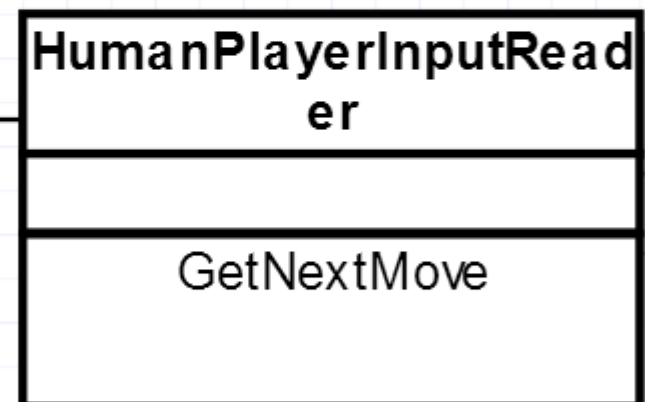
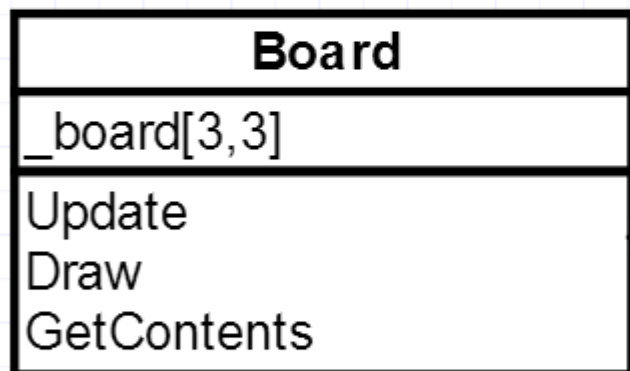
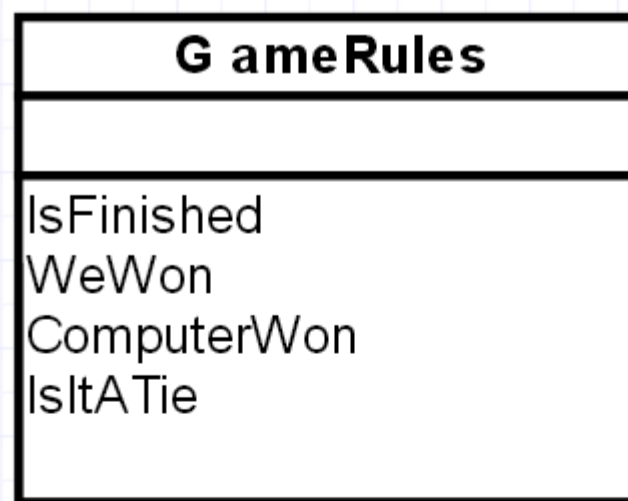
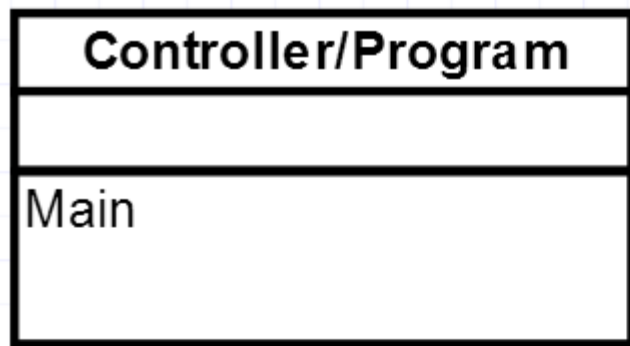
Rules:

- 3x3 grid #
- Player goes first as 'X'
- Computer plays as 'O'
- Player wins if they get 3 in a row.
- A tie occurs when the board fills up.


```
14 DrawBoard(board);
15 while (!finished)
16 {
17     var move = GetPlayerMove();
18     UpdateBoard(board, move[0], move[1], 'X');
19     if (WeWon(board))
20     {
21         Console.WriteLine("You win!"); finished = true; continue;
22     }
23
24     if (BoardIsFull(board))
25     {
26         Console.WriteLine("It's a draw!"); finished = true; continue;
27     }
28
29     int[] aiMove = GetAiMove(board);
30     UpdateBoard(board, aiMove[0], aiMove[1], 'O');
31
32     if (ComputerWon(board))
33     {
34         Console.WriteLine("You lose!"); finished = true; continue;
35     }
36 }
```







Program

Class


Methods

 Main








Board

Class

Fields

 _board

Methods

 Board
 Draw
 DrawRow
 GetBoardDataF...
 IsFull
 PlayerWon
 Update









Game

Class

Fields

 _board

Methods

 DidComputerW...
 DidWeWin
 DrawInitialBoard
 Finished
 Game
 IsItATie
 PlayerWon
 Update

PlayerMoveRea...

Class

Methods

 Read

ComputerBrain

Class

Fields

 _board

Methods

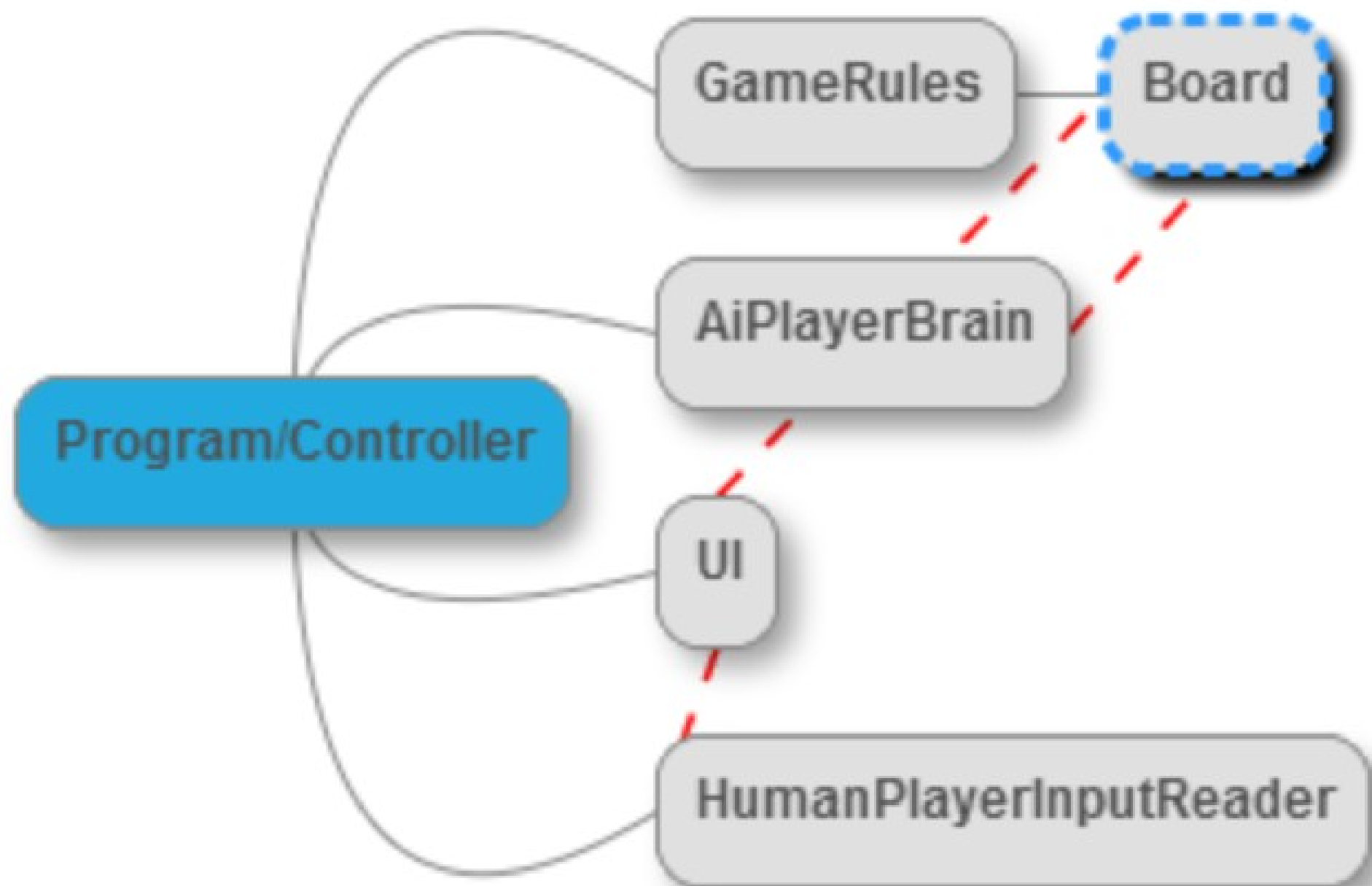
 ComputerBrain
 DetermineNext...


```
14 DrawBoard(board);
15 while (!finished)
16 {
17     var move = GetPlayerMove();
18     UpdateBoard(board, move[0], move[1], 'X');
19     if (WeWon(board))
20     {
21         Console.WriteLine("You win!"); finished = true; continue;
22     }
23
24     if (BoardIsFull(board))
25     {
26         Console.WriteLine("It's a draw!"); finished = true; continue;
27     }
28
29     int[] aiMove = GetAiMove(board);
30     UpdateBoard(board, aiMove[0], aiMove[1], 'O');
31
32     if (ComputerWon(board))
33     {
34         Console.WriteLine("You lose!"); finished = true; continue;
35     }
36 }
```

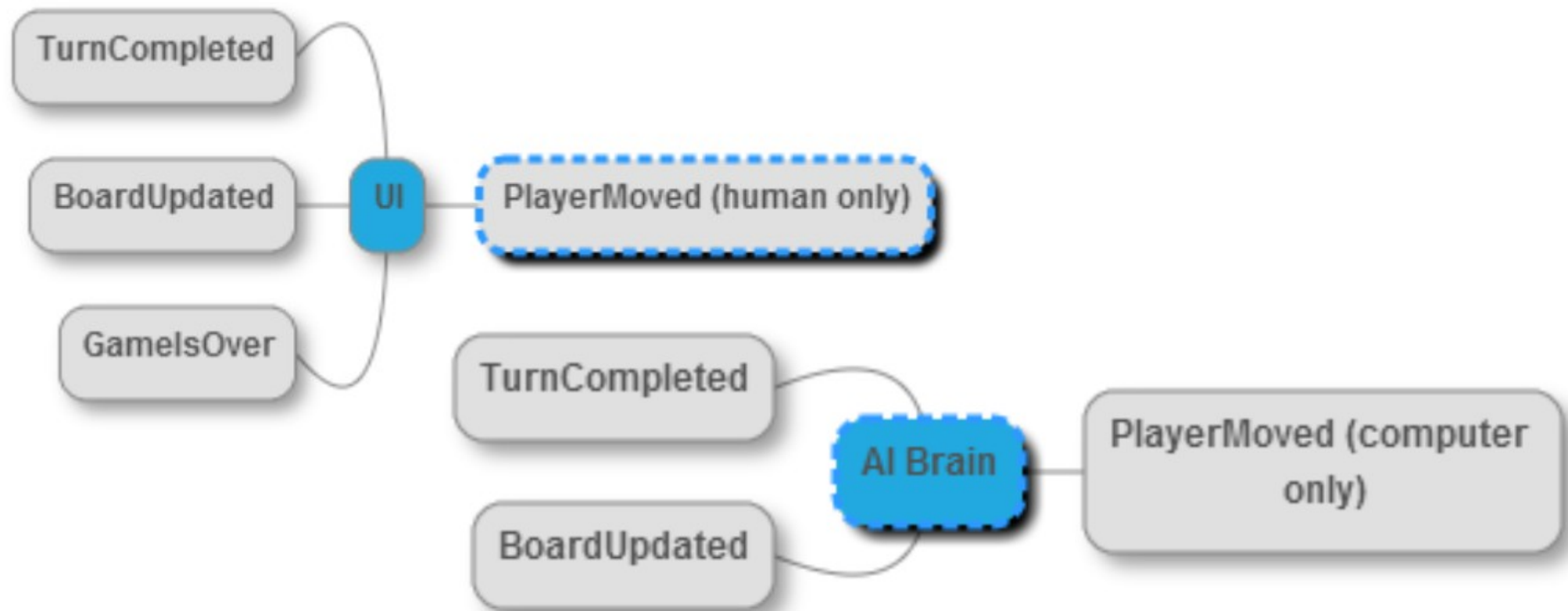
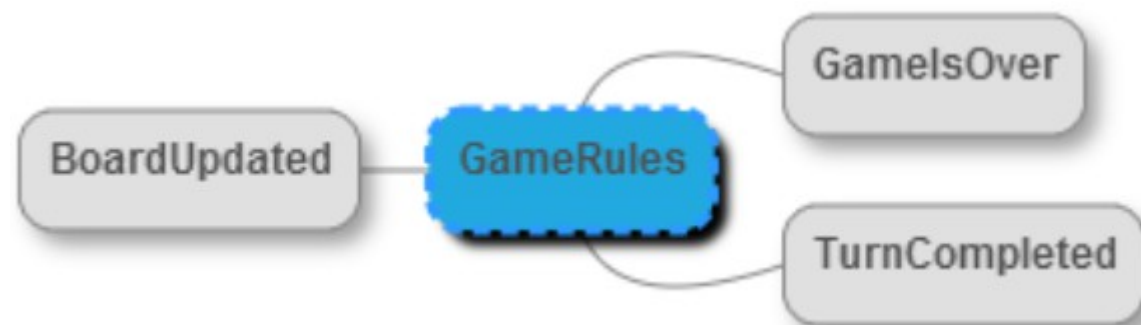
```
13 var board = new Board();
14 var game = new Game(board);
15 var playerMoveReader = new PlayerMoveReader();
16 var computerBrain = new ComputerBrain(board);
17 game.DrawInitialBoard();
18 while (!game.Finished())
19 {
20     var move = playerMoveReader.Read();
21     game.Update(move);
22     if (game.DidWeWin())
23     {
24         Console.WriteLine("You win!"); continue;
25     }
26
27     if (game.IsItATie())
28     {
29         Console.WriteLine("It's a draw!"); continue;
30     }
31
32     game.Update(computerBrain.DetermineNextMove());
33
34     if (game.DidComputerWin())
35     {
36         Console.WriteLine("You lose!"); continue;
37     }
38 }
```

```
14 DrawBoard(board);
15 while (!finished)
16 {
17     var move = GetPlayerMove();
18     UpdateBoard(board, move[0], move[1]);
19     if (WeWon(board))
20     {
21         Console.WriteLine("You win!");
22     }
23
24     if (BoardIsFull(board))
25     {
26         Console.WriteLine("It's a draw!");
27     }
28
29     int[] aiMove = GetAiMove(board);
30     UpdateBoard(board, aiMove[0], aiMove[1]);
31
32     if (ComputerWon(board))
33     {
34         Console.WriteLine("You lose!");
35     }
36 }
```

```
13 var board = new Board();
14 var game = new Game(board);
15 var playerMoveReader = new PlayerMoveReader();
16 var computerBrain = new ComputerBrain(board);
17 game.DrawInitialBoard();
18 while (!game.Finished())
19 {
20     var move = playerMoveReader.Read();
21     game.Update(move);
22     if (game.DidWeWin())
23     {
24         Console.WriteLine("You win!"); continue;
25     }
26
27     if (game.IsItATie())
28     {
29         Console.WriteLine("It's a draw!"); continue;
30     }
31
32     game.Update(computerBrain.DetermineNextMove());
33
34     if (game.DidComputerWin())
35     {
36         Console.WriteLine("You lose!"); continue;
37     }
38 }
```



Now for the Full message
passing Monty – we
communicate ONLY with
messages



PlayerMoved

Board

BoardUpdated

```
5 public class Board
6 {
7     readonly int[,] _board =
8     public Board()
9     {
10         MessageBus.Subscribe<PlayerMovedMessage>(UpdateBoard);
11     }
12
13     private void UpdateBoard(PlayerMovedMessage message)
14     {
15         //do work
16         _board[message.Row, message.Col] = message.Mark;
17
18         //publish message
19         var boardUpdatedMessage = new BoardUpdatedMessage {Board = CloneBoard(), IsFull = IsFull()};
20         MessageBus.Publish(boardUpdatedMessage);
21     }
22
23     private bool IsFull()...
24
25     private int[,] CloneBoard()...
26 }
27
28 public class BoardUpdatedMessage
29 {
30     public int[,] Board { get; set; }
31     public bool IsFull { get; set; }
32 }
33
34 public class PlayerMovedMessage...
```

Demo: Back to the OOP
solution, and introducing the
“Board was updated”
message.

```
9 var board = new Board();
10 var boardUi = new BoardUi();
11 var game = new Game(board);
12 var playerMoveReader = new PlayerMoveReader();
13 var computerBrain = new ComputerBrain();
14 //Demo warning: this may be a bad practice because
15 //and this may cause memory leaks
16 board.Updated += boardUi.Draw;
17 board.Updated += computerBrain.StoreBoardUpdate;
18 boardUi.DrawInitialBoard();
19 while (!game.Finished())
20 {
21     var move = playerMoveReader.Read();
22     game.Update(move);
23     if (game.DidWeWin())
24     {
25         Console.WriteLine("You win!"); continue;
26     }
27
28     if (game.IsItATie())
29     {
30         Console.WriteLine("It's a draw!"); continue;
31     }
32
33     game.Update(computerBrain.DetermineNextMove());
34
35     if (game.DidComputerWin())
36     {
37         Console.WriteLine("You lose!"); continue;
38     }
39 }
```

```

13 var board = new Board();
14 var game = new Game(board);
15 var playerMoveReader = new PlayerMoveReader();
16 var computerBrain = new ComputerBrain();
17 game.DrawInitialBoard();
18 while (!game.Finished())
19 {
20     var move = playerMoveReader.Read();
21     game.Update(move);
22     if (game.DidWeWin())
23     {
24         Console.WriteLine("You win!");
25     }
26
27     if (game.IsItATie())
28     {
29         Console.WriteLine("It's a draw!");
30     }
31
32     game.Update(computerBrain.DetermineNextMove());
33
34     if (game.DidComputerWin())
35     {
36         Console.WriteLine("You lose!");
37     }
38 }

```

```

9 var board = new Board();
10 var boardUi = new BoardUi();
11 var game = new Game(board);
12 var playerMoveReader = new PlayerMoveReader();
13 var computerBrain = new ComputerBrain();
14 //Demo warning: this may be a bad practice because
15 //and this may cause memory leaks
16 board.Updated += boardUi.Draw;
17 board.Updated += computerBrain.StoreBoardUpdate;
18 boardUi.DrawInitialBoard();
19 while (!game.Finished())
20 {
21     var move = playerMoveReader.Read();
22     game.Update(move);
23     if (game.DidWeWin())
24     {
25         Console.WriteLine("You win!"); continue;
26     }
27
28     if (game.IsItATie())
29     {
30         Console.WriteLine("It's a draw!"); continue;
31     }
32
33     game.Update(computerBrain.DetermineNextMove());
34
35     if (game.DidComputerWin())
36     {
37         Console.WriteLine("You lose!"); continue;
38     }
39 }

```

```

1 namespace TicTacToeObjects
2 {
3     public class Game
4     {
5         private readonly Board _board;
6
7         public Game(Board board) ...
11
12         public void DrawInitialBoard() ...
16
17         public bool Finished() ...
27
28         public bool DidWeWin() ...
32
33         public bool DidComputerWin() ...
37
38         private bool PlayerWon(char mark) ...
42
43         public bool IsItATie() ...
47
48         public void Update(Move move)
49         {
50             _board.Update(move);
51             _board.Draw();
52         }
53     }
54 }

```

```

1 namespace TicTacToeMessages
2 {
3     public class Game
4     {
5         private readonly Board _board;
6
7         public Game(Board board) ...
11
12         public bool Finished() ...
22
23         public bool DidWeWin() ...
27
28         public bool DidComputerWin() ...
32
33         private bool PlayerWon(char mark) ...
37
38         public bool IsItATie() ...
42
43         public void Update(Move move)
44         {
45             _board.Update(move);
46         }
47     }
48 }

```

```

1 using System;|
2
3 namespace TicTacToeObjects
4 {
5     public class Board
6     {
7         private readonly char[,] _board = new ch
8
9         public Board()...
16
17         public void Draw()...
25
26         private static void DrawRow(char[,] boar
32
33         public bool PlayerWon(char mark)...
49
50         public void Update(Move move)...
54
55         public bool IsFull()...
62
63         public char[,] GetBoardDataForAi()...
72     }
73 }

```

```

1 using System;|
2
3 namespace TicTacToeMessages
4 {
5     public class BoardUi
6     {
7         public void Draw(char[,] board)...
16
17         private static void DrawRow(char[,] b
23
24         public void DrawInitialBoard()...
32     }
33 }

```

What felt awkward about
that?

Was anything better?

Still awkward? People use
IoC containers to wire up
events/messages

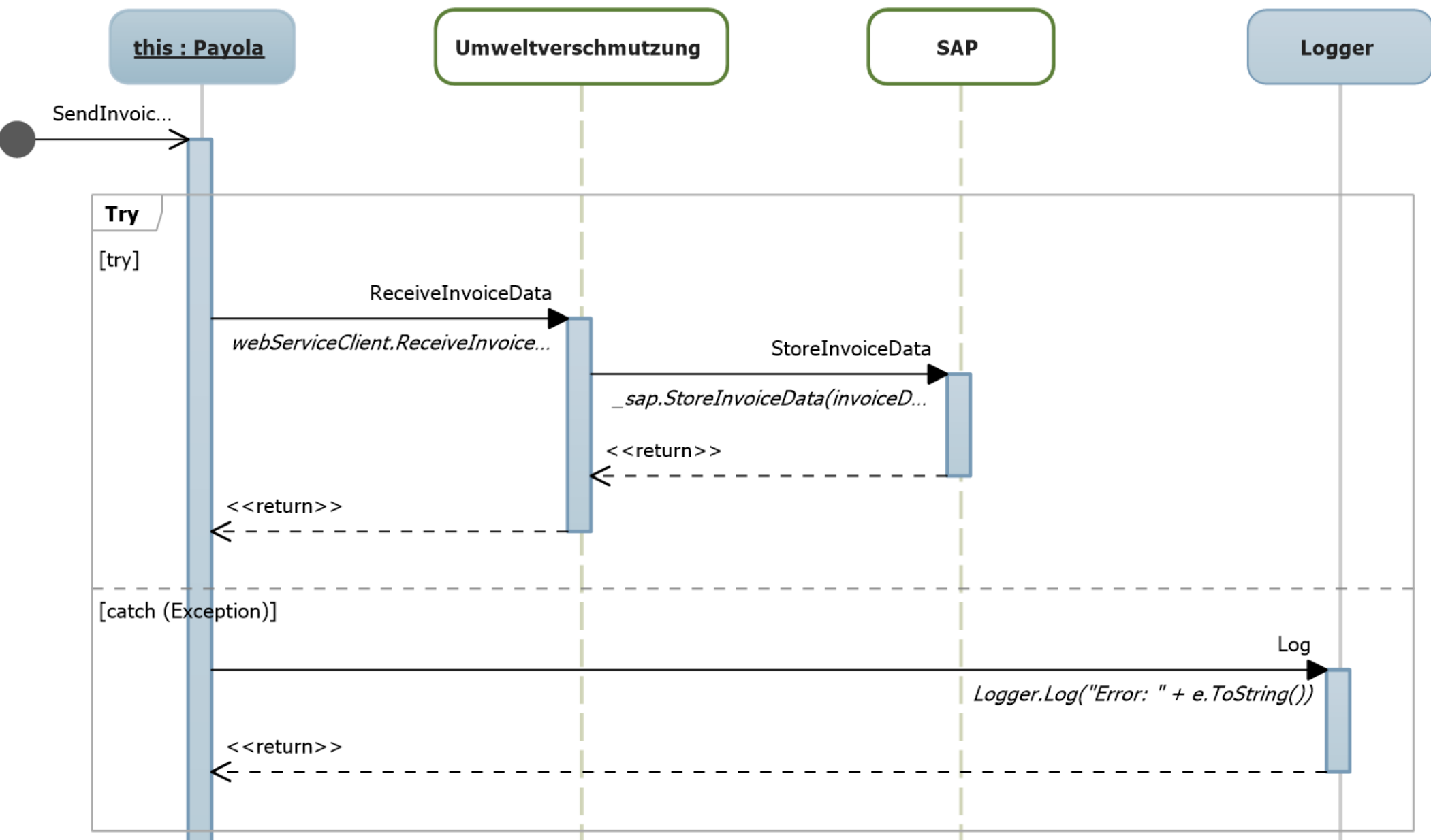
Moment of reflection

What makes message
passing different from just
calling methods?

What are the benefits?

Durable messaging (or whatever it's called when you use MSMQ or another *MQ)

DEMO: Payola (your .NET app) sends invoice data to Umweltverschmutzung (SAP)



Reflection: Could we have solved those problems without durable messages?

Reflection: How does durable messaging/guaranteed delivery help? What are the new problems if you switch?

Any **QUESTIONS** before I
conclude?

Takeaway: if you're doing distributed computing and are not using durable messages, either start using durable messages or suffer

Takeaway: Don't use MSMQ.
Or at least “raw” MSMQ.

Takeaway: BizTalk is
expensive and hokey. Look
into MassTransit and
NServiceBus.

Takeaway: Reduce # of collaborators, encourage simplicity by sending messages.

Takeaway: Create and use
your own C# events.

Takeaway: When C# events
aren't enough, look into
“Domain Events”

Takeaway: If you're doing WPF, look into the Event Aggregator pattern and use it (sometimes).

Takeaway: If you're having problems with threading issues or need to scale, look into durable messages and using background workers

Takeaway: If you've tried “classical DDD” from a few years back and don't like all the overhead, read Ayende's blog on architecture and how he builds a *simple* architecture replacing service methods with messages.

Alternately, CQRS