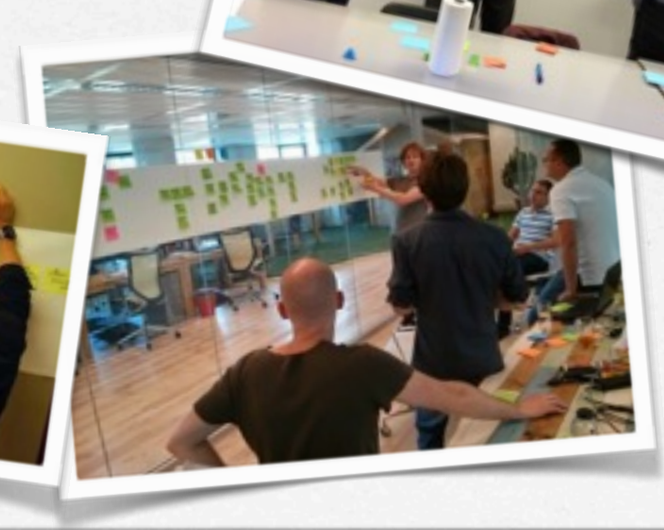
Akka.NET

Sudoko – einmal ganz anders

"Schwarm-Intelligenz"

- Beispiel Event Storming
- Kollaboration
- aus Chaos wird Ordnung
- Jeder leistet seinen Anteil



Sudoku "schwarmgelöst"

- Angaben werden nacheinander bestückt
- Zelle kennt ihren Inhalt und teilt ihn mit 81 Aktoren
- Zeile, Spalte und Block zählt Ziffern 27 Aktoren
- Summe: 108 Aktoren

			8				4	5
4	6							1
			9					8
	7			3		5		
		9		5		7		6
				2		3		
	5				2		6	
	4				6	8	2	
6	3				4			

Sudoku erzeugen

- ☐ Printer:
  Monitor-Ausgabe
- ☐ Row/Col/Block:
  Statistik
- ☐ Cell:
  sichtbare Zelle
- ☑ ohne Verbindungen

```
private static void CreateSudokuActors(ActorSystem system)
{
    var printer = system.ActorOf<Printer>();

    for (int row = 0; row < 9; row++)
    {
        system.ActorOf(
            Props.Create<SudokuRow>(printer, row),
            String.Format("r-{0}", row)
        );

        for (int col = 0; col < 9; col++)
        {
            if (row == 0)
                system.ActorOf(
                    Props.Create<SudokuCol>(printer, col),
                    String.Format("c-{0}", col)
                );

            system.ActorOf(
                Props.Create<SudokuCell>(printer, row, col),
                String.Format("{0}-{1}", row, col)
            );
        }
    }

    for (int block = 0; block < 9; block++)
        system.ActorOf(
            Props.Create<SudokuBlock>(printer, block),
            String.Format("b-{0}", block)
        );
}
```

Menge beherrschen

- 108 Aktoren
- Jeder kennt jeden (11556 Verbindungen)
- Daher: Publish – Subscribe

Publish Beispiel

- Wenn etwas passiert ist
  z.B. Zelle wurde gelöst
- Allen anderen mitteilen

```
private void SolveCell(SetDigit setDigit)
{
    var digit = setDigit.Digit;

    if (!IsSolved(digit))
    {
        possibleDigits.ToList()
            .ForEach(d => Publish(new StrikeDigit(row, col, d)));

        possibleDigits.Clear();
        possibleDigits.Add(digit);

        Draw();
    }
}
```

Subscribe Beispiel

- Subscribe sorgt für "Empfang"

```
var eventStream = Context.System.EventStream;

eventStream.Subscribe(Self, typeof(SetDigit));
eventStream.Subscribe(Self, typeof(StrikeDigit));

eventStream.Subscribe(Self, typeof(FindRowDigit));
eventStream.Subscribe(Self, typeof(FindColDigit));
eventStream.Subscribe(Self, typeof(FindBlockDigit));
```

- Receive<> legt Verhalten fest

```
Receive<SetDigit>(RestrictPossibilities, IsRowOrColOrBlock);
Receive<SetDigit>(SolveCell, IsMyCell);

Receive<FindRowDigit>(FindDigitHandler, f => f.Row == row);
Receive<FindColDigit>(FindDigitHandler, f => f.Col == col);
Receive<FindBlockDigit>(FindDigitHandler, f => f.Block == block);
```

beim Lösen

SudokuSolver — mono-sgen — 80×50

```
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|7| |9| |4|   | |3| |8| |5|   | |2| |1| |6|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|1| |3| |5|   | |9| |6| |2|   | |7| |8| |4|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|2| |6| |8|   | |1| |4| |7|   | |5| |3| |9|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
-------------------------------------------
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|8| |7| |3|   | |6| |9| |4|   | |1| |5| |2|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|6| |1| |2|   | |5| |7| |8|   | |4| |9| |3|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|4| |5| |9|   | |2| |1| |3|   | |8| |6| |7|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
-------------------------------------------
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|5| |4| |1|   | |7| |3| |6|   | |9| |2| |8|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|9| |8| |6|   | |4| |2| |1|   | |3| |7| |5|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
              |               |
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
|3| |2| |7|   | |8| |5| |9|   | |6| |4| |1|
+-+ +-+ +-+   | +-+ +-+ +-+   | +-+ +-+ +-+
```

```
b-0: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-1: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-2: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-3: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-4: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-5: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-6: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-7: 1, 1, 1, 1, 1, 1, 1, 1, 1
b-8: 1, 1, 1, 1, 1, 1, 1, 1, 1

c-0: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-1: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-2: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-3: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-4: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-5: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-6: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-7: 1, 1, 1, 1, 1, 1, 1, 1, 1
c-8: 1, 1, 1, 1, 1, 1, 1, 1, 1

r-0: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-1: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-2: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-3: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-4: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-5: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-6: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-7: 1, 1, 1, 1, 1, 1, 1, 1, 1
r-8: 1, 1, 1, 1, 1, 1, 1, 1, 1
```

Houston! Problems!

- Basisklasse „SudokuActor“

```
protected void PrintLine(int row, int col, string line)
{
    // NEVER do this in production. NEVER!
    lock (Context.System)
    {
        Console.SetCursorPosition(col, row);
        Console.WriteLine(line);
    }
}
```

- lock hat bei Aktoren nichts verloren!
- Aufgabe: eigenen Aktor einsetzen