Reflection Applied: Aspects

CPSC 501: Advanced Programming Techniques
Winter 2025

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

UNIVERSITY OF CALGARY

Intercession via aspects

Introduction

- What is AspectJ?
  - Aspect oriented programming (AOP) extension to Java

- What is an Aspect?
  - a particular part or feature of something.

UNIVERSITY OF
CALGARY

History

- Developed at Xerox PARC (Palo Alto RC)

- Launched in 1998

- PARC transferred AspectJ to an openly-developed eclipse.org project in December of 2002.

For more info: www.eclipse.org/aspectj

UNIVERSITY OF
CALGARY

Introduction

- What are goals of AOP?
  1. **Separation of concerns**
  2. **Modularity**
     - No more tangled code
     - Simplicity
     - Maintainability
     - Reusability
  3. **Aspects**
     - encapsulate behaviors that affect multiple classes (OO) into reusable modules.

UNIVERSITY OF CALGARY

I'm concerned?

UNIVERSITY OF CALGARY

Cross-Cutting Concern

- What is a **cross-cutting concern**?

  - Behavior that cuts across the typical divisions of responsibility**, such as logging or debugging**

  - A problem which a program tries to solve.

  - Aspects of a program that **do not relate to the core concerns** directly, but which **proper program execution nevertheless requires**.

UNIVERSITY OF CALGARY

Language: Dynamic VS Static crosscutting

- **Dynamic crosscutting**
  - define **additional behavior** to run at certain well-defined **points** in the **execution** of the program

- **Static crosscutting**
  - **modify the static structure** of a program (e.g., adding new methods, implementing new interfaces, modifying the class hierarchy)

We'll build around this

UNIVERSITY OF
CALGARY

Reference Object Structured for Following

Join In

UNIVERSITY OF CALGARY
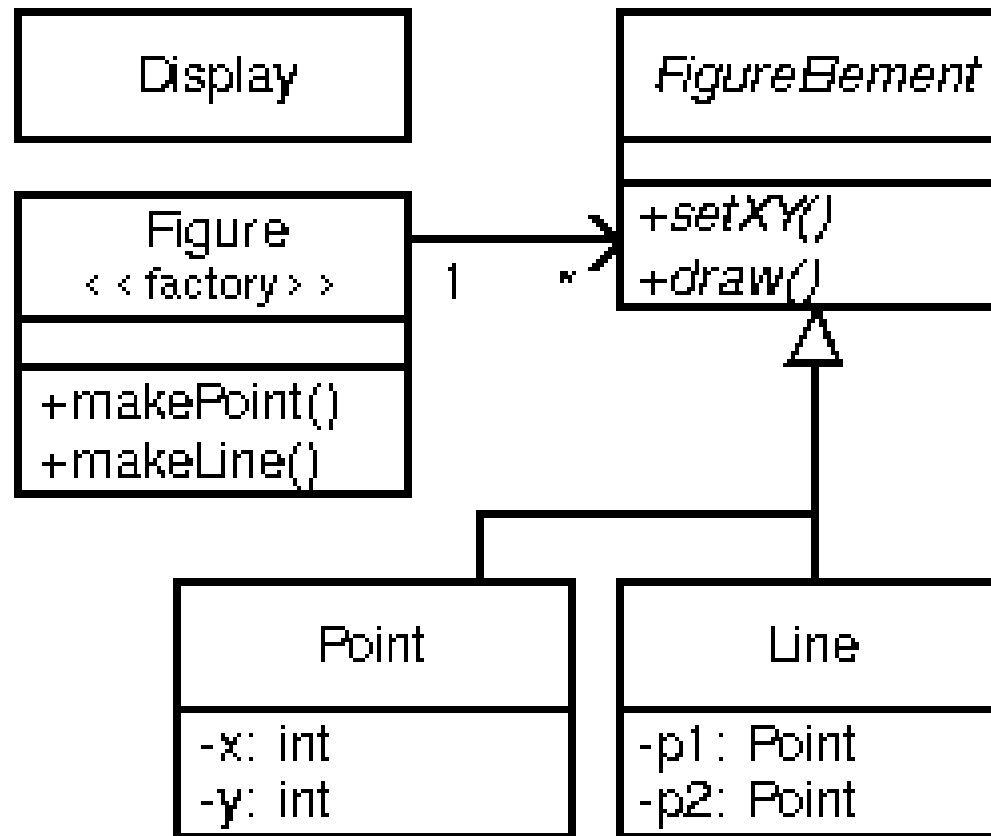
Language: Join Points

- **Join Points:** well-defined points in the execution of a program

  - Method call, Method execution
  - Constructor call, Constructor execution
  - Static initializer execution
  - Object pre-initialization, Object initialization
  - Field reference, Field set
  - Handler execution
  - Advice execution

UNIVERSITY OF CALGARY

Language: Join Points

Method-execution

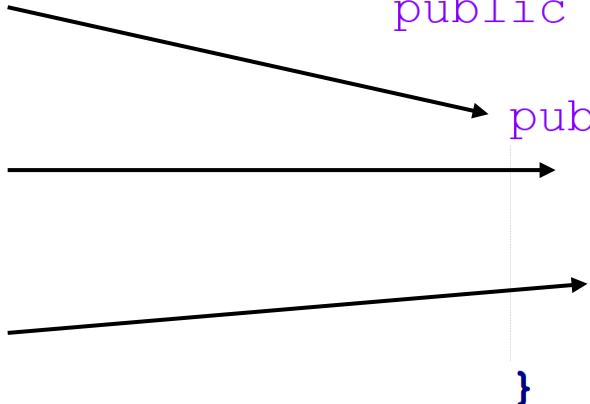"Test.main(..)"

Constructor-call

"Point(..)"

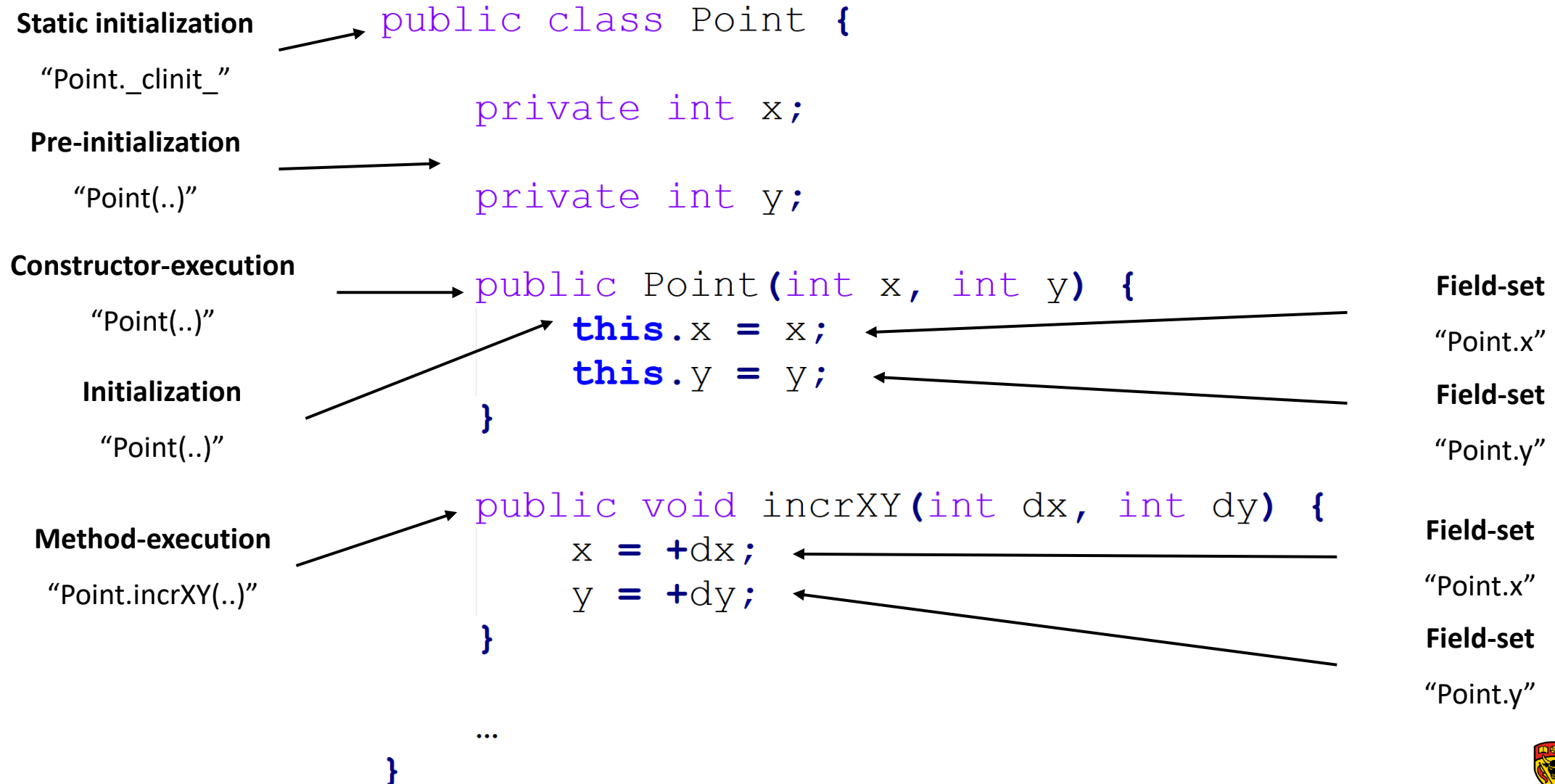**Method-call**

"Point.incrXY(..)"

```java
public class Test {

    public static void main(String[] args) {
        Point pt1 = new Point(0, 0);

        pt1.incrXY(3, 6);

    }

}
```

Language: Join Points

Static initialization

"Point._clinit_"

Pre-initialization

"Point(..)"

Constructor-execution

"Point(..)"

Initialization

"Point(..)"

Method-execution

"Point.incrXY(..)"

Field-set

"Point.x"

Field-set

"Point.y"

Field-set

"Point.x"

Field-set

"Point.y"

```java
public class Point {

    private int x;

    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void incrXY(int dx, int dy) {
        x = +dx;
        y = +dy;
    }

    ...
}
```

UNIVERSITY OF CALGARY

Cut in

UNIVERSITY OF CALGARY

Language: Pointcuts

- A set of join point, plus, optionally, some of the values in the execution context of those join points.

- Can be composed using boolean operators || , &&

- Matched at runtime

UNIVERSITY OF CALGARY

Language
Pointcut examples

Matches if the join point is a method call with this signature.

```
call(public void Point.setX(int))
```

Matches if the join point is a method call to any kind of FigureElement.

```
call(public void FigureElement.incrXY(int,int))
```

Matches any call to setX OR setY

```
call(public void Point.setX(int)) || call(public void Point.setY(int))
```

UNIVERSITY OF
CALGARY

Language
Pointcut examples

```
pointcut move():
    call(void FigureElement.setXY(int,int)) ||
    call(void Point.setX(int))              ||
    call(void Point.setY(int))              ||
    call(void Line.setP1(Point))            ||
    call(void Line.setP2(Point));
```

- There is a cross-cutting concern here relating to moving

- We can capture these in our own user defined pointcut

UNIVERSITY OF CALGARY

When to cut in?

UNIVERSITY OF
CALGARY

Language: Advice

- Method-like mechanism used to declare that certain code should execute at each of the join points in the pointcut.

- Advice:

    - before
    - around
    - after
        - after
        - after returning
        - after throwing

UNIVERSITY OF
CALGARY

Language: Advice

```
before(): move() {
    System.out.println("about to move");
}


after() returning: move() {
    System.out.println("just successfully moved");
}
```

UNIVERSITY OF
CALGARY

Language: Exposing context

We can also interact with parameters of pointcut

```
after(FigureElement fe, int x, int y) returning:
        ...SomePointcut... {
    System.out.println(fe + " moved to (" + x + ", " + y + ")");
}
```

Filling in an applicable pointcut

```
after(FigureElement fe, int x, int y) returning:
        call(void FigureElement.setXY(int, int))
        && target(fe)
        && args(x, y) {
    System.out.println(fe + " moved to (" + x + ", " + y + ")");
}
```

UNIVERSITY OF
CALGARY

All together now

UNIVERSITY OF CALGARY

Language: Aspects

- Mix everything we've seen up to now and put it one or more modular units called Aspects.

- Looks a lot like a class!

- Can contain pointcuts, advice declarations, methods, variables ….

- Single instances (default behavior)

UNIVERSITY OF CALGARY

The methods we weave

UNIVERSITY OF
CALGARY

Implementation

- Aspect weaving: makes sure that applicable advice runs at the appropriate join points.

- In AspectJ, almost all the weaving is done at compile-time to expose errors and avoid runtime overhead.

UNIVERSITY OF
CALGARY

Developmental Aspects

- What are some places Aspects can assist developmental processes

- Exist in along-side but apart from existing coding

- Tracing, profiling/logging, pre-post conditions, …
  - Enabling tracing as an 'weaved' in process that doesn't exist in production
  - Although many sophisticated profiling tools are available, and these can gather a variety of information and display the results in useful ways, you may sometimes want to profile or log some very specific behavior.
  - "Design by Contract" style where explicit pre-conditions test that callers of a method call it properly and explicit post-conditions test that methods properly do the work they are supposed to.

Production Aspects

- What are some places Aspects can assist production code

- Expected to be enabled and in operation

- Change monitoring, Context passing, Consistent Behaviour
  - Ex. maintain a dirty bit associated with object having moved since last display occurred
  - Ex. Bypassing chain of messages by triggers being able to move outside regular class diagram
  - Ex. Making all methods log errors to common output location

Optional: Complex Example from Research

UNIVERSITY OF CALGARY

My Experience: Advised Multi-Agent System

Location
Station
Depot
↔ Connection
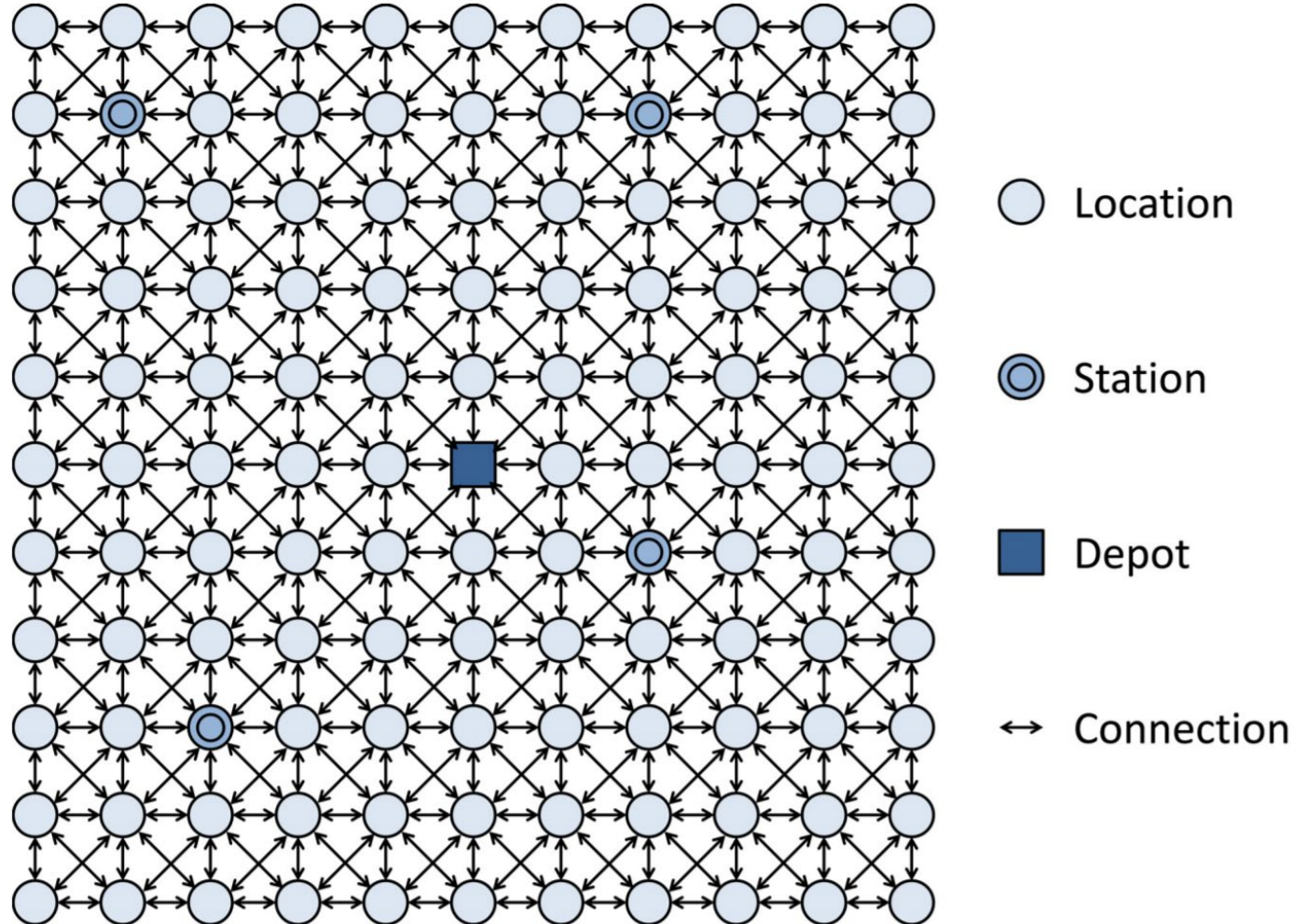
My Experience : Advised Multi-Agent System
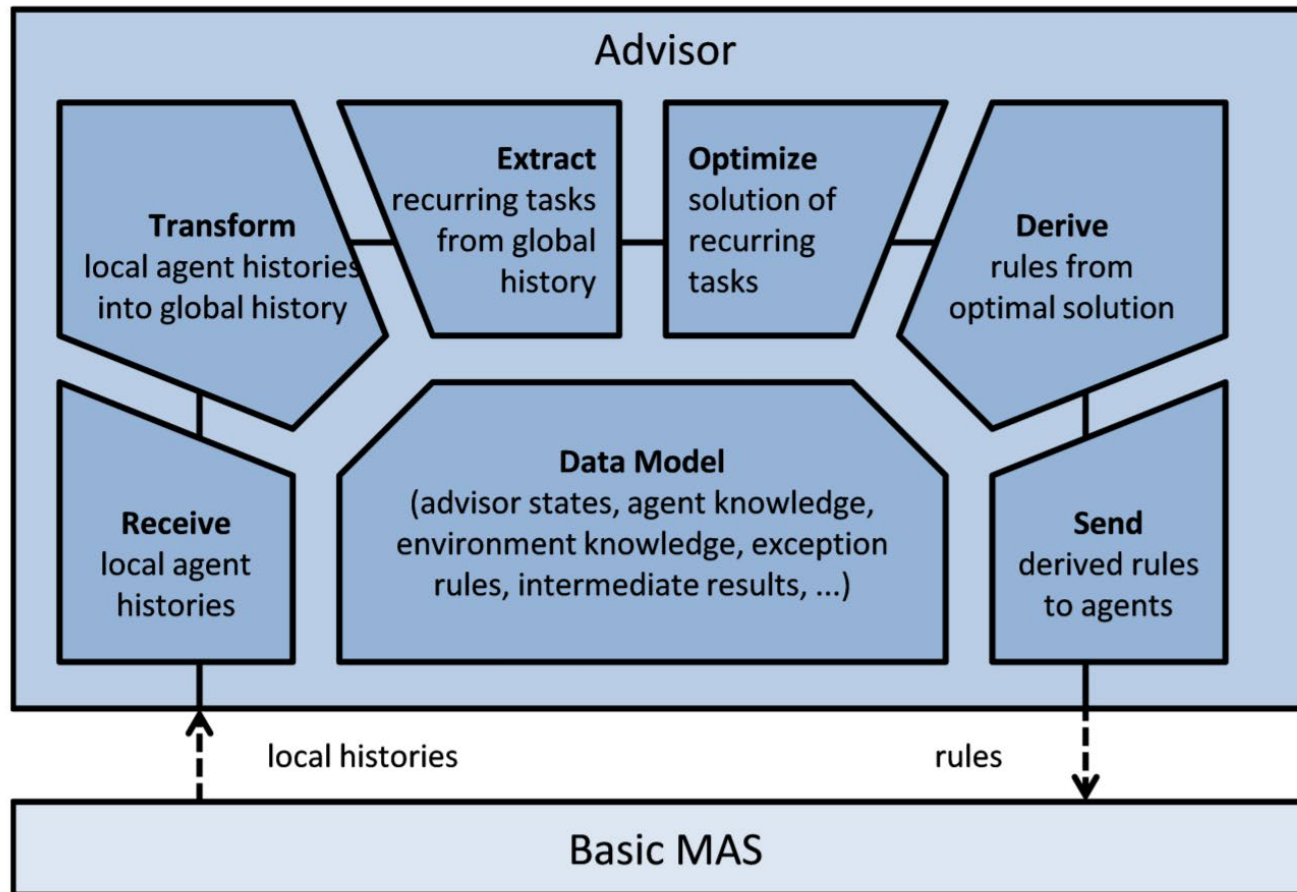
My Experience: Advised Multi-Agent System

- Advisor monitors each agent when actions happened (collecting histories)
- From histories environment reconstructed, as well as agent behaviour
- Agent behaviour compared to optimal (ish)
- Rules to attempt to make agents act like optimal
- Rules added to agent

UNIVERSITY OF
CALGARY

My Experience: Advised Multi-Agent System

- Aspects:
  - Advisor Aspect that hooks onto Agents when actions occur and records them (methods are called)
  - Also is able to notice when simulation runs have finished and do its number crunching to extract info, optimize, derive rules, and communicate them
  - Aspect around each agent to store advisor communicated rules and inter-cede in methods to change their behaviour decisions based on rules

UNIVERSITY OF
CALGARY

My Experience: Advised Multi-Agent System

1. The MAS designer never had to change his code

2. The distributed aspect concerns related to the advisor were all centralized into very few classes, despite their interaction with code base being distributed

3. Could be flagged on and off at runtime

- The negative was a negligible runtime cost of hooking in aspects (the optimization AI step was much longer)

- Code always had to be run with additional configuration setup than basic Java code

UNIVERSITY OF CALGARY

Onward to … optimization.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF
CALGARY