Lub
A language for Dynamic Context Oriented Programming

Steven Costiou
Mickaël Kerboeuf, Glenn Cavarlé, Alain Plantec
UMR CNRS 6285, Lab-STICC/MOCS
Université de Bretagne Occidentale

The drone fleet example

- The drones are flying in close formation
- One of the drones loses its guidance system
- Assumption: the fail-safe behaviour for that case has not been anticipated
- Possible solution: dynamically change the drone fleet's behaviour
  - *The faulty drone behaviour is changed to use the GPS of a mate drone*

1. **Problematic: Dynamic behavior adaptation**
2. Our proposition: Lub, a language for behavior adaptation through dynamic lookup instrumentation
3. Evaluation with Pharo: The drone example experiment

Dynamic behavior adaptation

- Autonomous systems need for dynamic behaviour adaptation in case of unexpected Events
- The system must not be lost
- The mission must not be cancelled

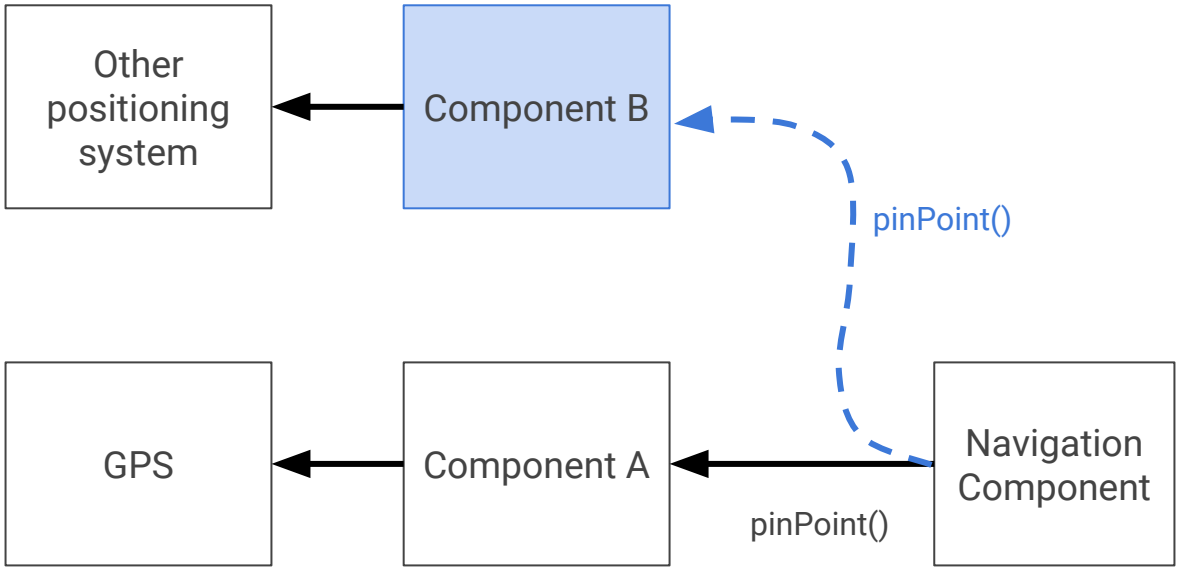
Dynamic behavior adaptation

- The runtime system must be able to dynamically updates its behavior
- One must be able to communicate with the system and safely push the new behavior

Existing approaches for dynamic behavior update ?

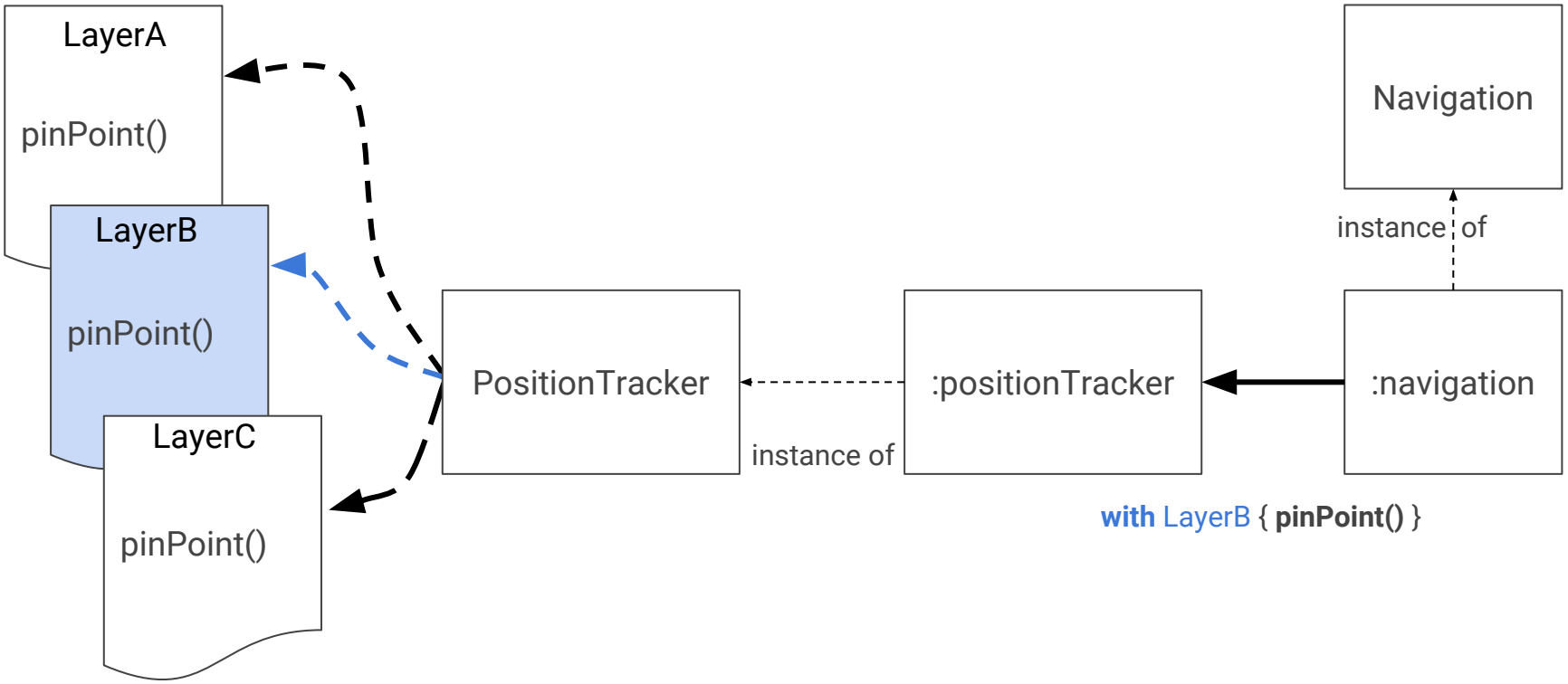
Component based solutions

- Adaptation through architectural reconfiguration



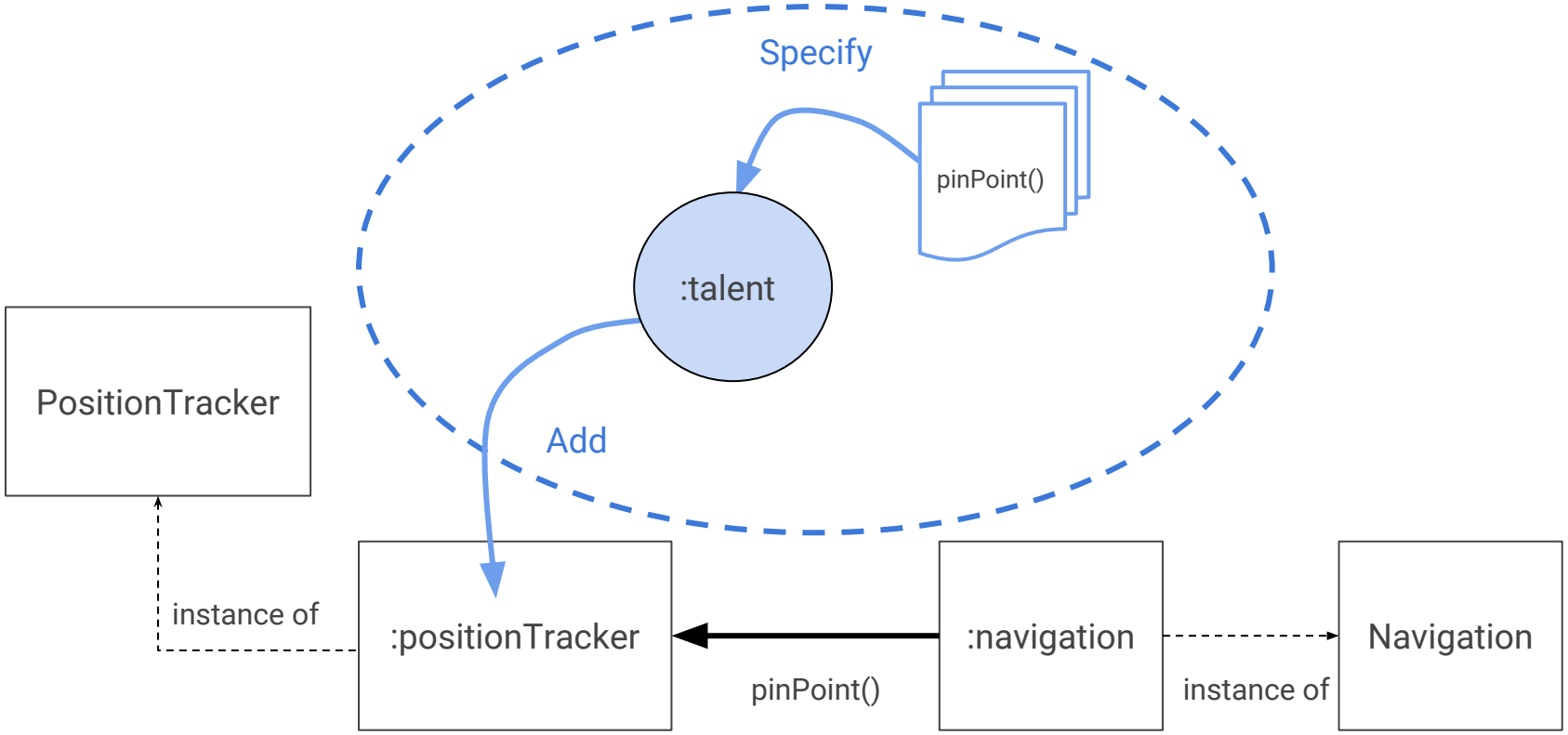
Context Oriented Programming (COP)

- Many COP languages
  - ContextL (Costanza and Hirschfeld 2005), ContextJ (Appeltauer et al. 2011), EventCj (Kamina et al. 2011), etc.
  - Combination “per-instance” + “no-scope-limitation” hard to find



Talents (Ressia 2014)

- Lookup delegated to a third party object
- Modifies the lookup semantics



Comparisons of these approaches

	Adaptation per-instance	Identity of the adapted object is preserved	Class / instance based
Component	**YES**	NO	**YES**
COP	POSSIBLE	**YES**	**YES**
Talents	**YES**	**YES**	NO

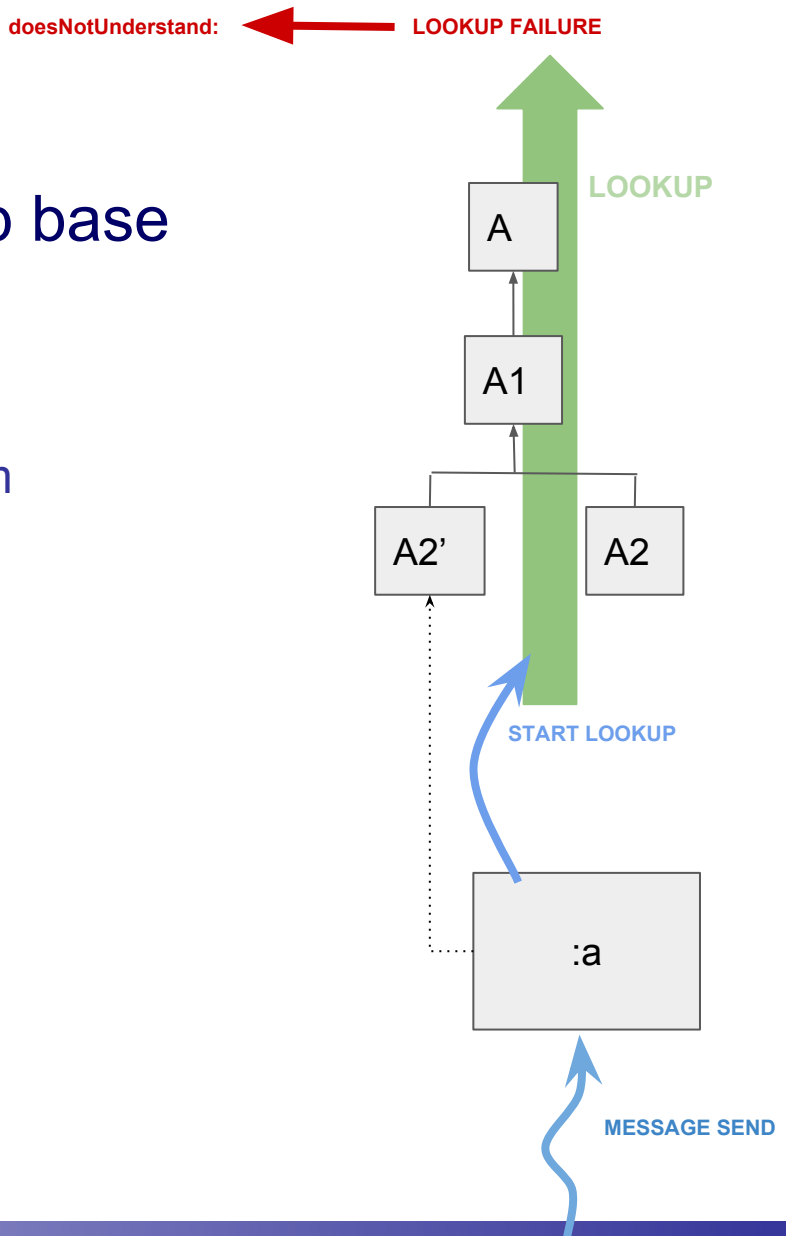
1. Problematic: Dynamic behavior adaptation
2. **Our proposition: Lub, a language for behavior adaptation through dynamic lookup instrumentation**
3. Evaluation with Pharo: The drone example experiment

Comparisons of these approaches

	Adaptation per-instance	Identity of the adapted object is preserved	Class / instance based
Component	**YES**	NO	**YES**
COP	POSSIBLE	**YES**	**YES**
Talents	**YES**	**YES**	NO
Lub	**YES**	**YES**	**YES**

Lookup base

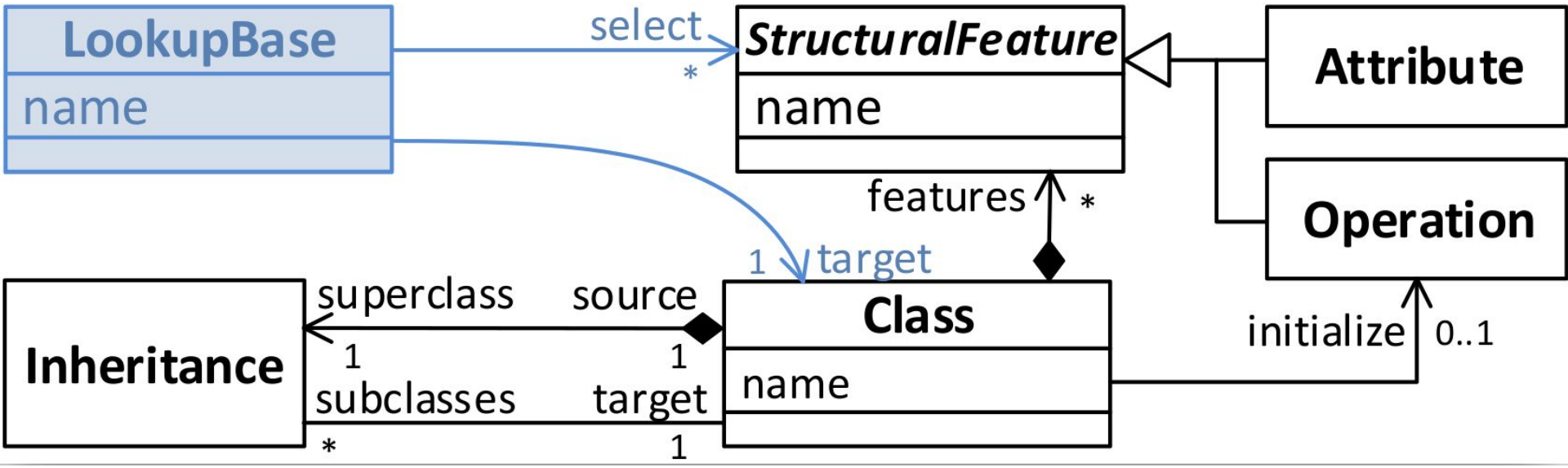
- Proposition: Changeable lookup base
- Lookup base
  - The class where the lookup starts from



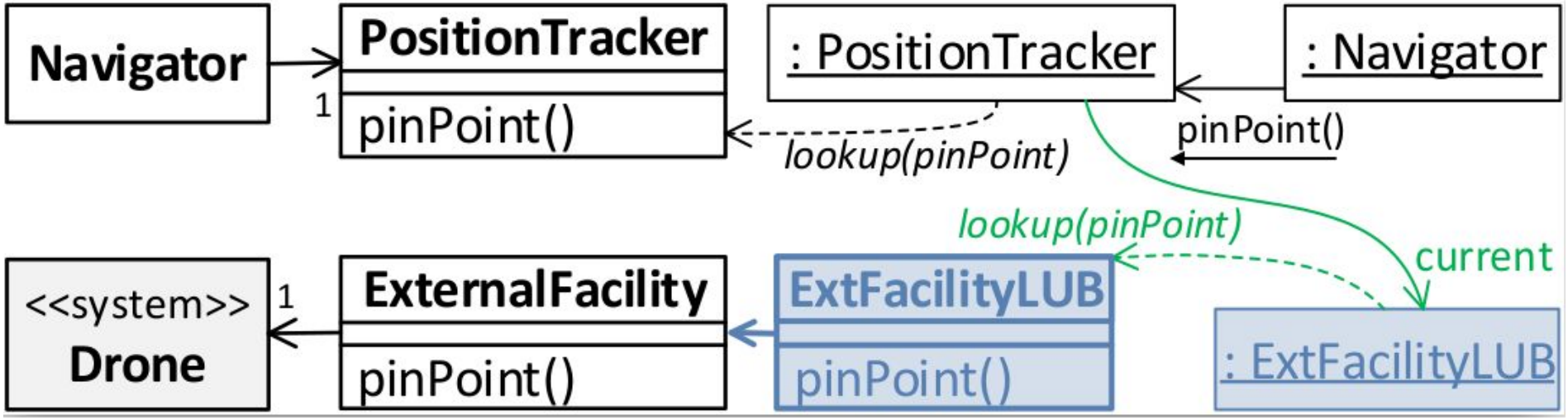
The Lub language

- Implements changeable lookup base
- OO language, Class/Instance based
- Instance based adaptation at runtime
- Preserved self reference: structural links and states are unchanged
- Two dedicated operators
  - to change the lookup base
  - to select methods impacted by the lookup base change

The Lub metamodel

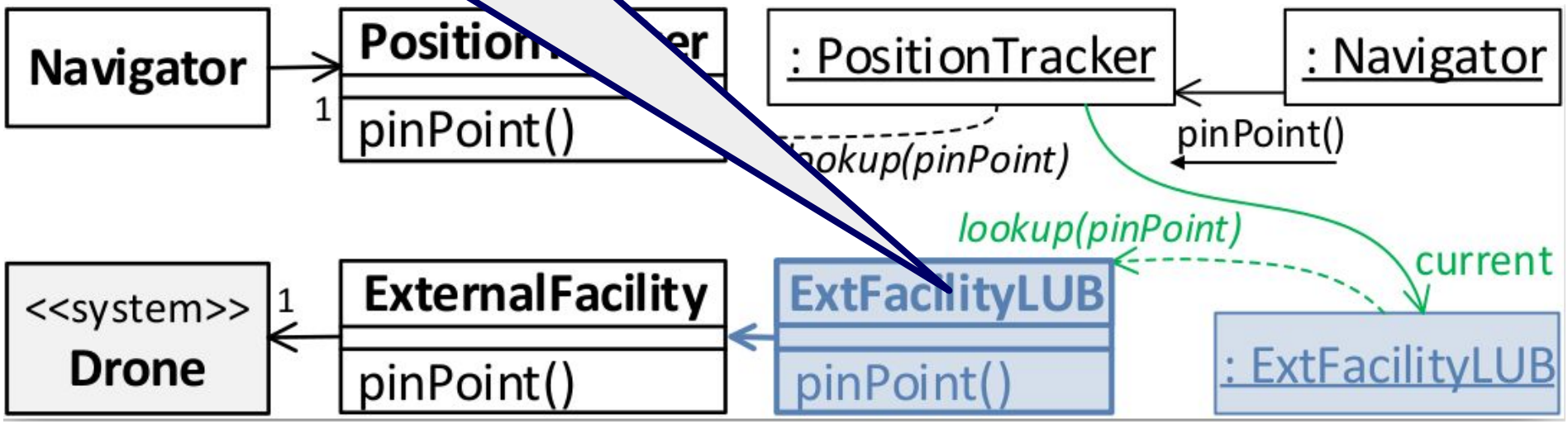


Lub adaptation example



Lub adaptation example

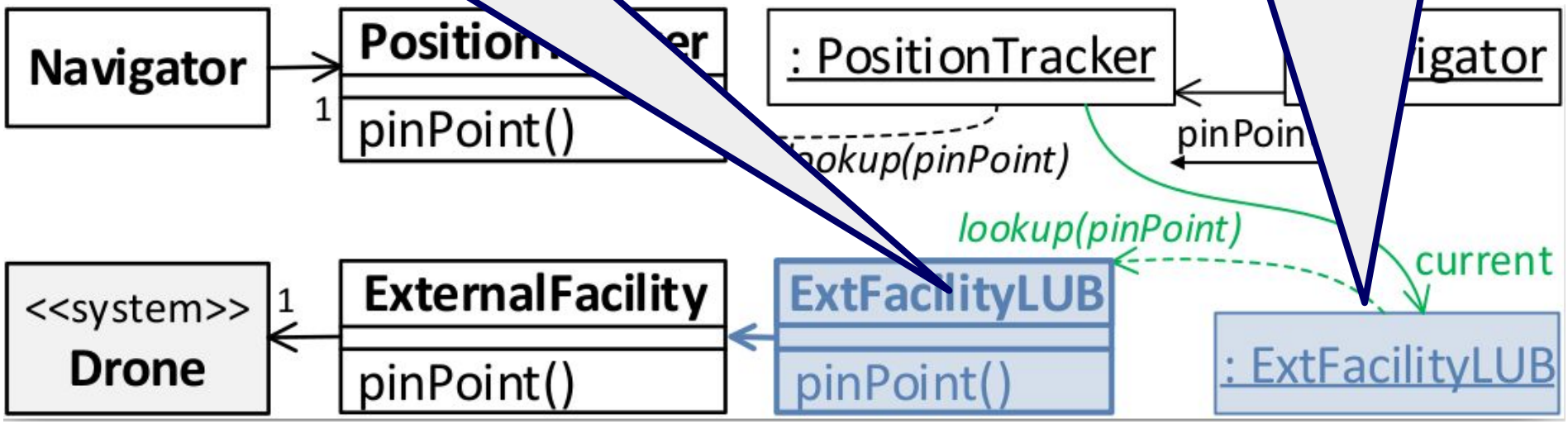
Definition of a LookupBase (LUB): references a class where to perform the lookup.



Lub adaptation example

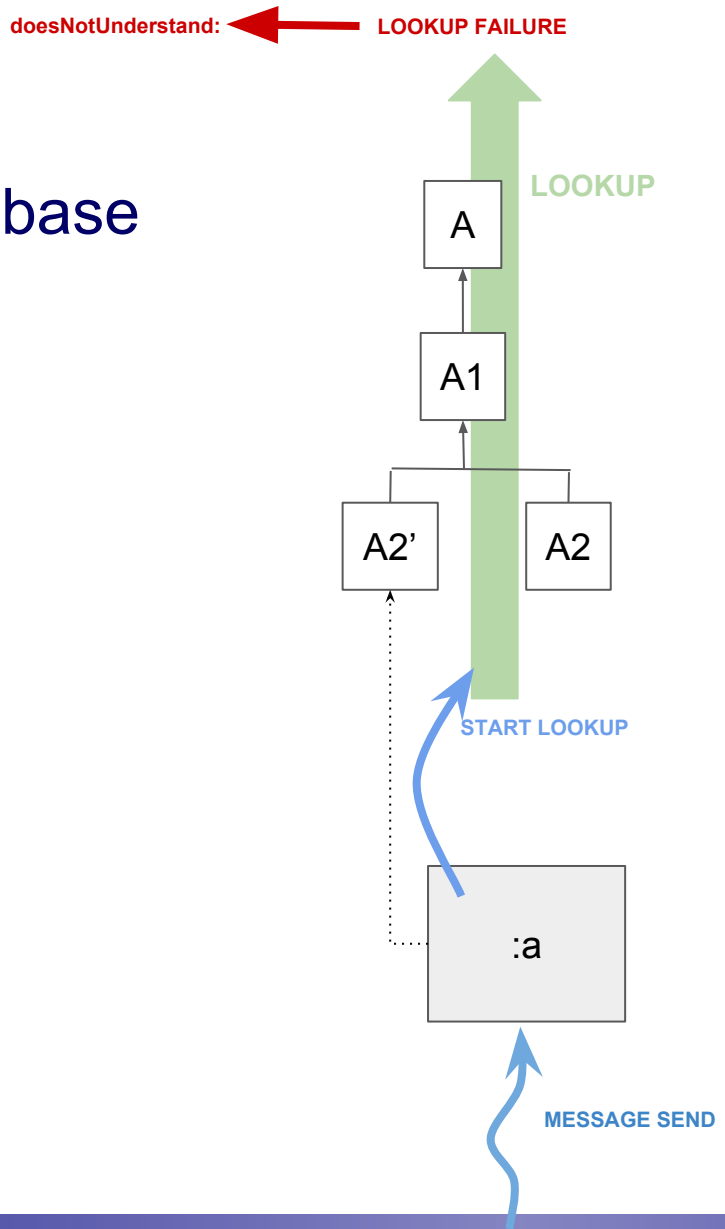
Definition of a LookupBase (LUB): references a class where to perform the lookup.

Instance of the LookupBase, set as the current one of the object to adapt.



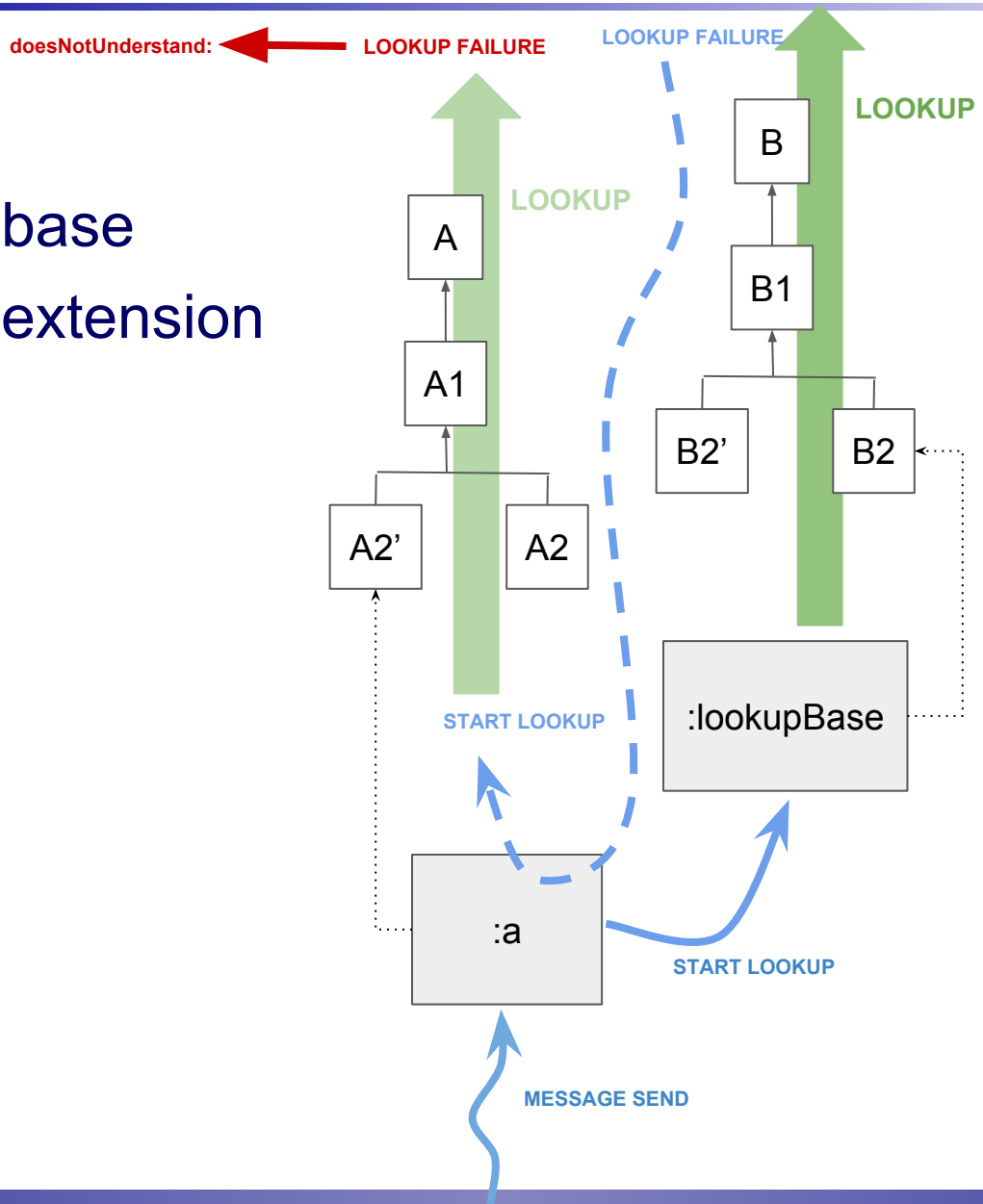
Lookup mechanics extension

- Lookup base



Lookup mechanics extension

- Lookup base
- Lookup extension



A Lub specification

```
class PeerPositionTracker {

        attributes { }
        operations {
                pinPoint: drone
                        "Computes the drone's position using the GPS of a mate drone"

                printTracker
                        ^'Adapted Tracker'
        }
}

LookupBase PeerTrackerLookupBase {
        class := PeerPositionTracker.
}
```

A Lub specification

Definition of an adaptation: dynamic adding of the **PeerPositionTracker class**

```
class PeerPositionTracker {

        attributes { }
        operations {
                pinPoint: drone
                        "Computes the drone's position using the GPS of a mate drone"

                printTracker
                        ^'Adapted Tracker'
        }
}

LookupBase PeerTrackerLookupBase {
        class := PeerPositionTracker.
}
```

A Lub specification

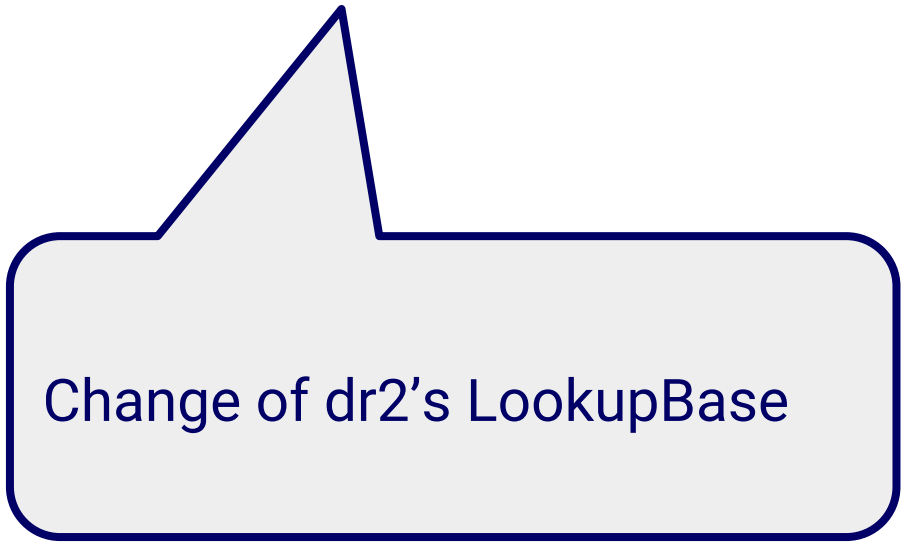
```
class PeerPositionTracker {

        attributes { }
        operations {
                pinPoint: drone
                        "Computes the drone's position using the GPS of a mate drone"

                printTracker
                        ^'Adapted Tracker'
        }
}

LookupBase PeerTrackerLookupBase {
        class := PeerPositionTracker.
}
```

Definition of an adaptation: dynamic adding of the **PeerPositionTracker class**

Definition of a **Lookup Base**: references the PeerPositionTracker class

A Lub specification

tracker := (simulation agentNamed: ‘dr2’) positionTracker.
tracker lookupBase: PeerTrackerLookupBase

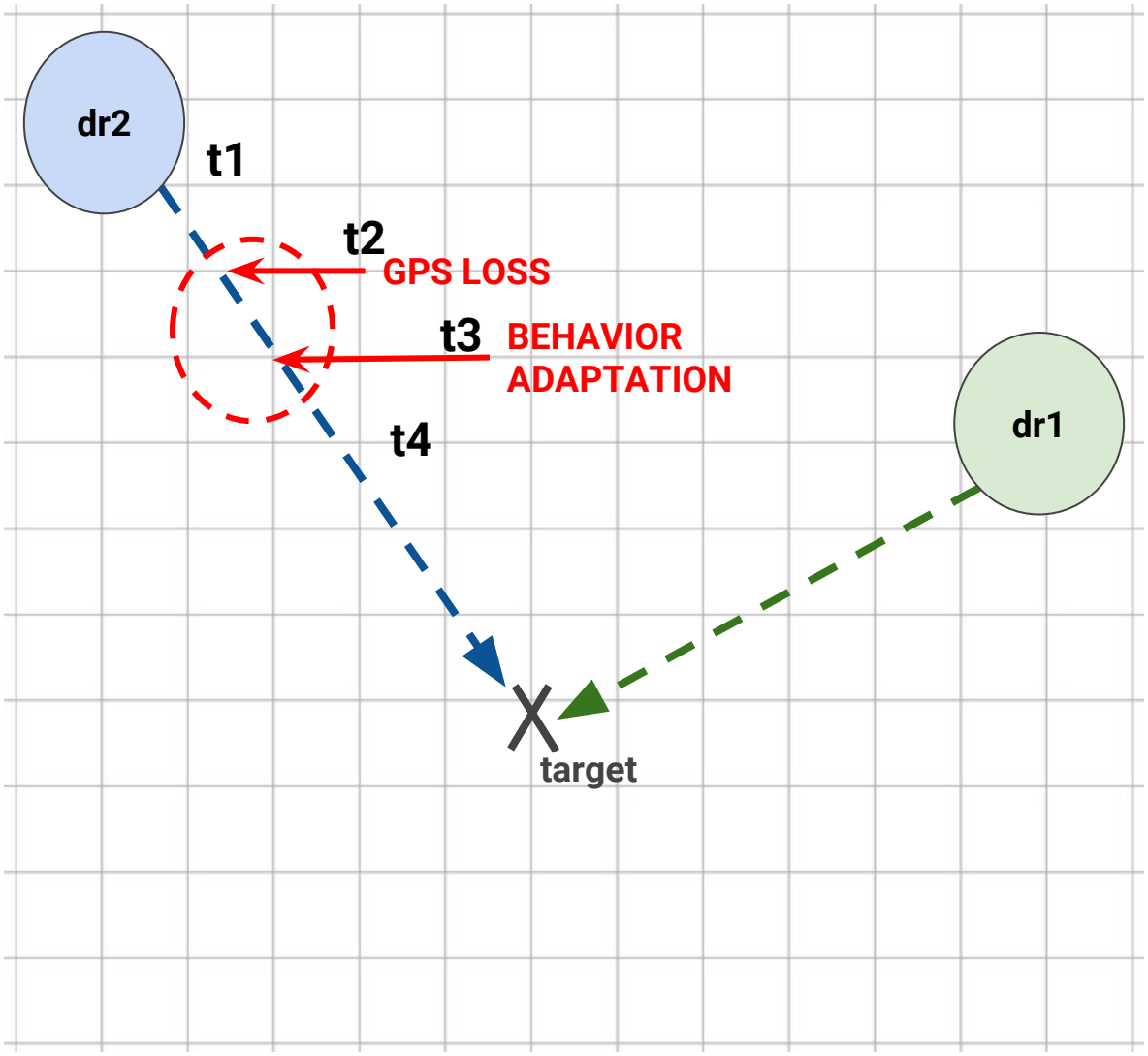


1. Problematic: Dynamic behavior adaptation

2. Our proposition: Lub, a language for behavior adaptation through dynamic lookup instrumentation

3. **Evaluation with Pharo: The drone example experiment**

Drone fleet example simulation



Simulation log after adaptation

t = 2

Accessing Tracker 1 [dr1 : GPSMobileDrone] this is dr1 at **(90@39)**

Accessing Tracker 2 [dr2 : GPSMobileDrone] **No pinpoint device available.**

Tracker 2 updating lookup base with : PeerTrackerLookupBase

t = 3

Accessing Tracker 1 [dr1 : GPSMobileDrone] this is dr1 at **(89@40)**

Accessing Adapted tracker (dr2 requesting dr1 position: Accessing Tracker 1)

[dr2 : GPSMobileDrone] this is dr2 at **(175@81)**

t = 4

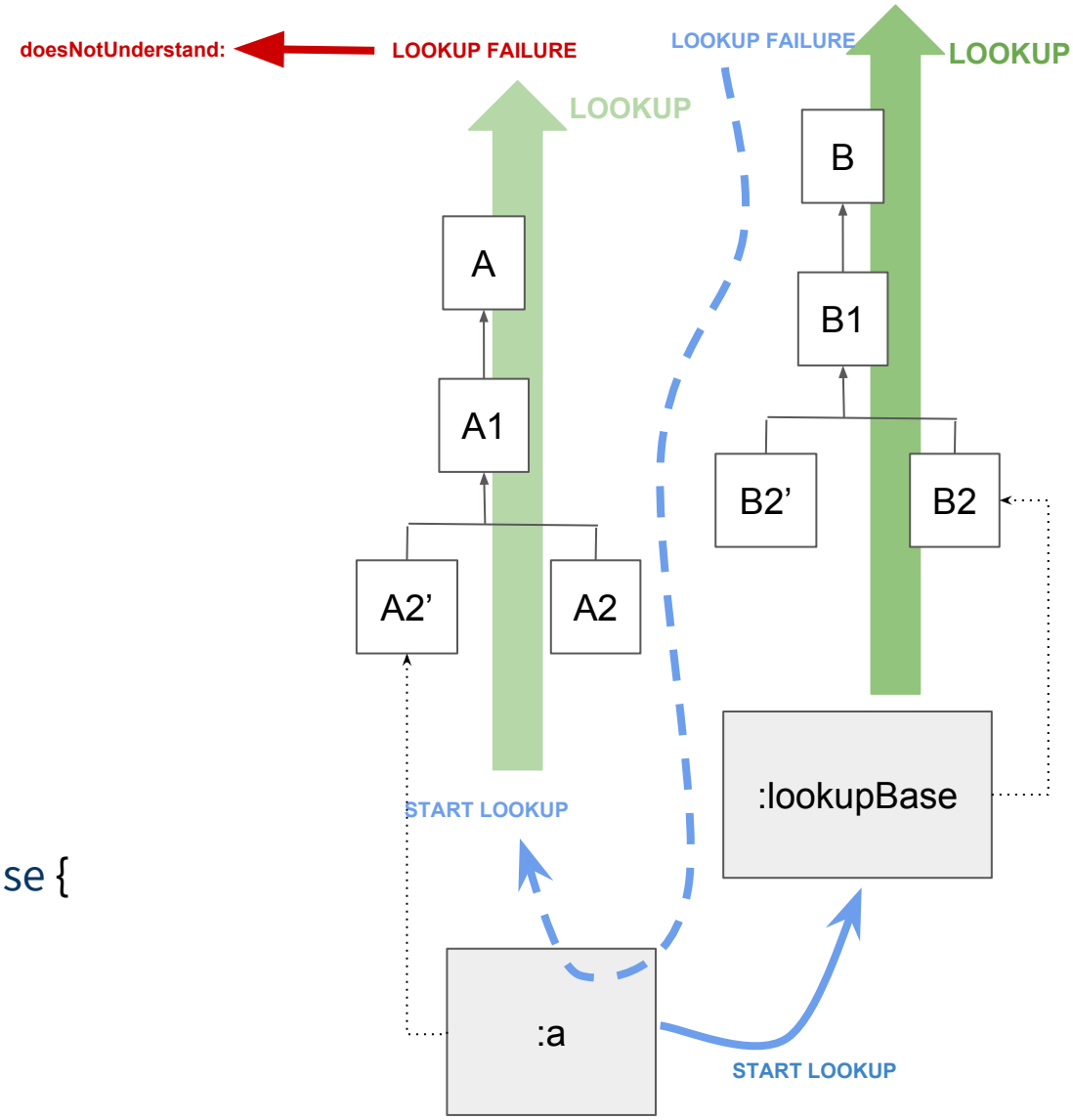
Accessing Tracker 1 [dr1 : GPSMobileDrone] this is dr1 at **(88@41)**

Accessing Adapted tracker (dr2 requesting dr1 position: Accessing Tracker 1)

[dr2 : GPSMobileDrone] this is dr2 at **(174@84)**

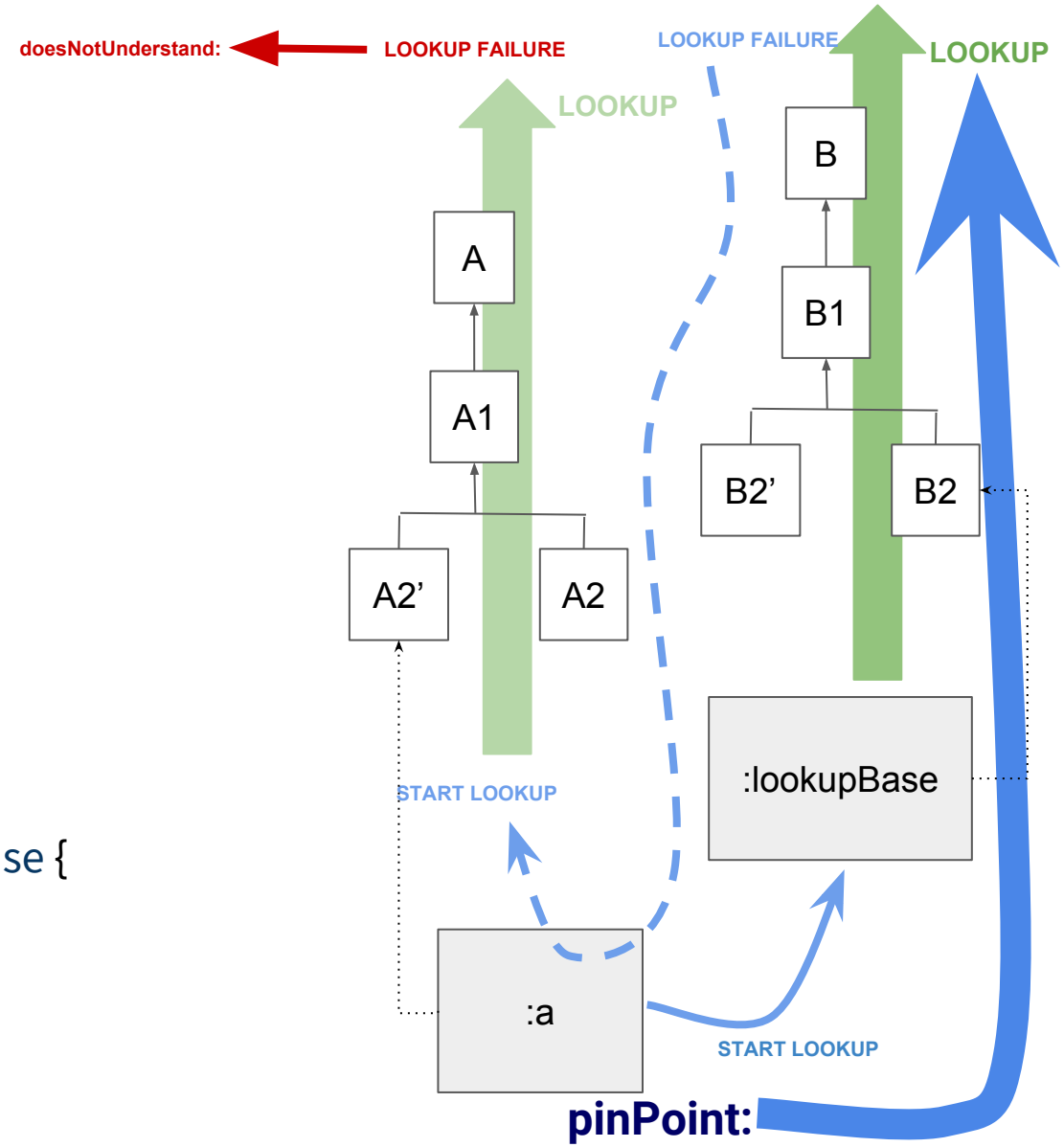
Operator selection

```
class PeerPositionTracker {

    attributes { }
    operations {
        pinPoint: drone
            "computations"

        printTracker
            ^'Adapted Tracker'
    }
}

LookupBase PeerTrackerLookupBase {

    class := PeerPositionTracker.
}
```



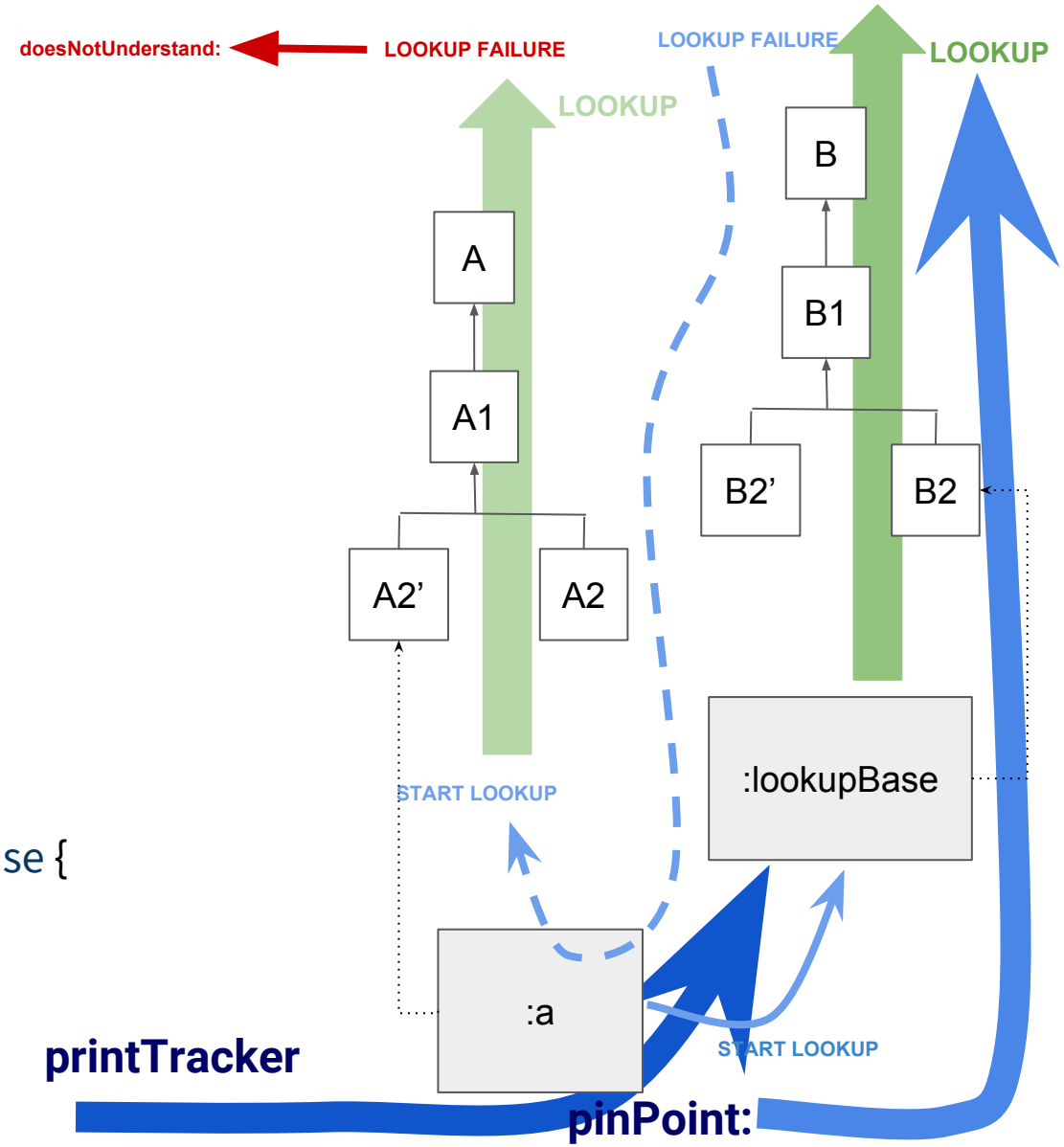
Operator selection

```
class PeerPositionTracker {

    attributes { }
    operations {
        pinPoint: drone
            "computations"

        printTracker
            ^'Adapted Tracker'
    }
}

LookupBase PeerTrackerLookupBase {

    class := PeerPositionTracker.
    with { pinPoint: }
}
```



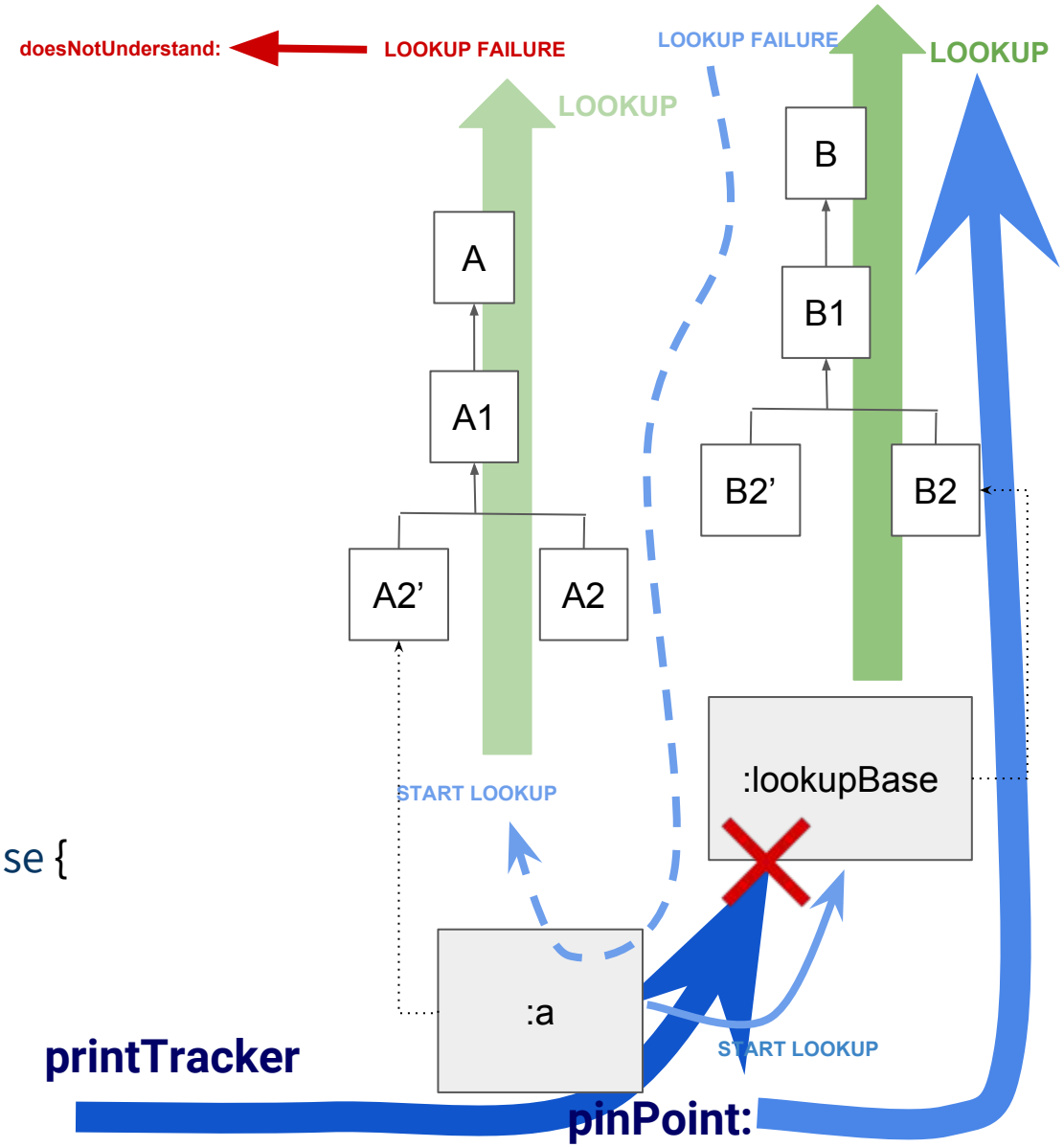
Operator selection

```
class PeerPositionTracker {

    attributes { }
    operations {
        pinPoint: drone
            "computations"

        printTracker
            ^'Adapted Tracker'
    }
}

LookupBase PeerTrackerLookupBase {

    class := PeerPositionTracker.
    with { pinPoint: }
}
```



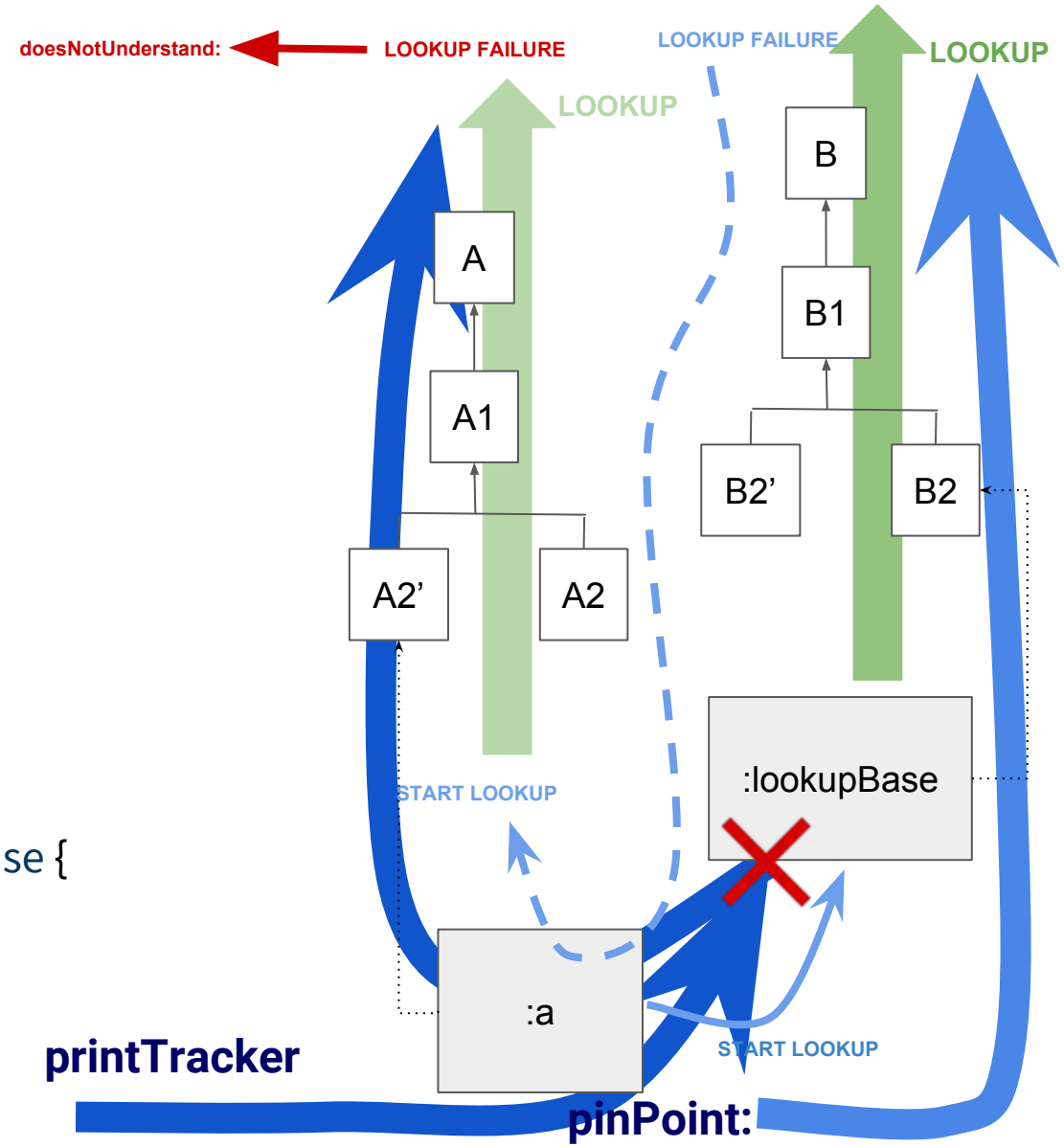
Operator selection

```
class PeerPositionTracker {

    attributes { }
    operations {
        pinPoint: drone
            "computations"

        printTracker
            ^'Adapted Tracker'
    }
}

LookupBase PeerTrackerLookupBase {

    class := PeerPositionTracker.
    with { pinPoint: }
}
```



Operator selection

```
class PeerPositionTracker {

    attributes { }
    operations {
        pinPoint: drone
            "computations"

        printTracker
            ^'Adapted Tracker'
    }
}

LookupBase PeerTrackerLookupBase {

    class := PeerPositionTracker.
    with { pinPoint: }
}
```



Simulation log after *pinPoint*: adaptation

t = 2

Accessing Tracker 1 [dr1 : GPSMobileDrone] this is dr1 at (90@39)

Accessing Tracker 2 [dr2 : GPSMobileDrone] No pinpoint device available.

Tracker 2 updating lookup base with : PeerTrackerLookupBase

t = 3

Accessing Tracker 1 [dr1 : GPSMobileDrone] this is dr1 at (89@40)

Accessing Tracker 2 (dr2 requesting dr1 position: Accessing Tracker 1)

[dr2 : GPSMobileDrone] this is dr2 at (175@81)

t = 4

Accessing Tracker 1 [dr1 : GPSMobileDrone] this is dr1 at (88@41)

Accessing Tracker 2 (dr2 requesting dr1 position: Accessing Tracker 1)

[dr2 : GPSMobileDrone] this is dr2 at (174@84)

Perspectives

- Vérification/validation
- Validation on a physical device
  - Reproducing the GPS drone example adaptation
- Investigate the communication problem
  - How to communicate and to push new behaviors ?
  - Nickolaos Papoulias thesis (2013): “remote debugging and reflection in resource constrained devices”

Lub
A language for Dynamic Context Oriented Programming

Steven Costiou
Mickaël Kerboeuf, Glenn Cavarlé, Alain Plantec
UMR CNRS 6285, Lab-STICC/MOCS
Université de Bretagne Occidentale

Lub runtime model