Edit Transactions

Dynamically Scoped Change Sets for Controlled Updates in Live Programming

Toni Mattis, Patrick Rein, Robert Hirschfeld
Software Architecture Group
HPI, University of Potsdam, Germany

‹**Programming**› 2017, Brussels

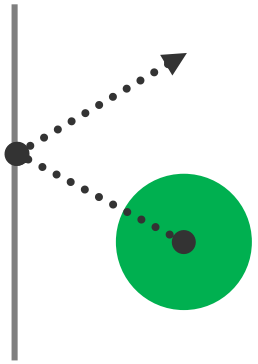
Live Programming

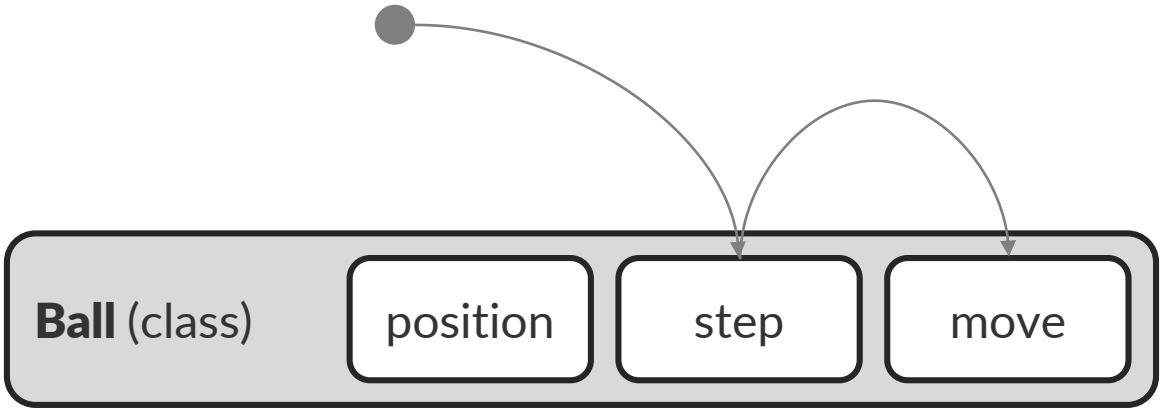
- Change & introspection of running programs
- Immediate & continuous feedback

("working definition" in the context of this talk)

Live Programming: Example

```
Ball >> step
  self reflect; move

Ball >> move
  self position: (self position + self speed)

Ball >> reflect "Check bounds and reflect"
```

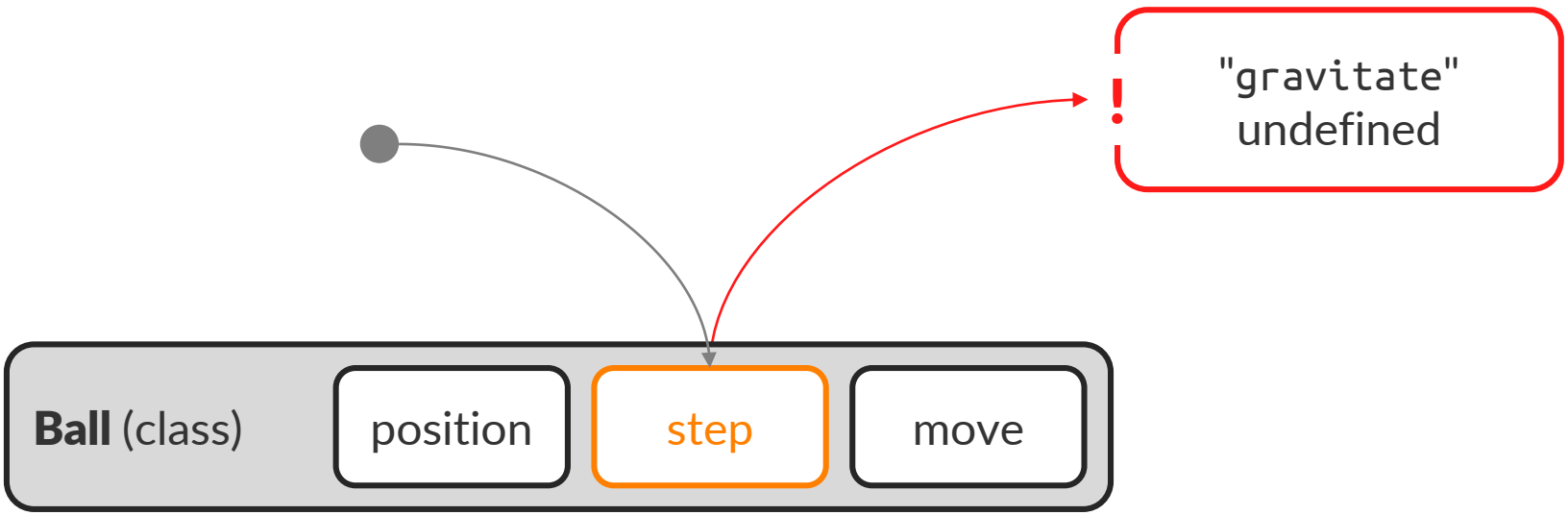




Live Programming: Example

```
Ball >> step
  self reflect; move; gravitate

Ball >> move
  self position: (self position + self speed)

Ball >> reflect "Check bounds and reflect"
```

insert call



Live Programming

- Change & introspection of running programs
- Immediate & continuous feedback
- Fragility

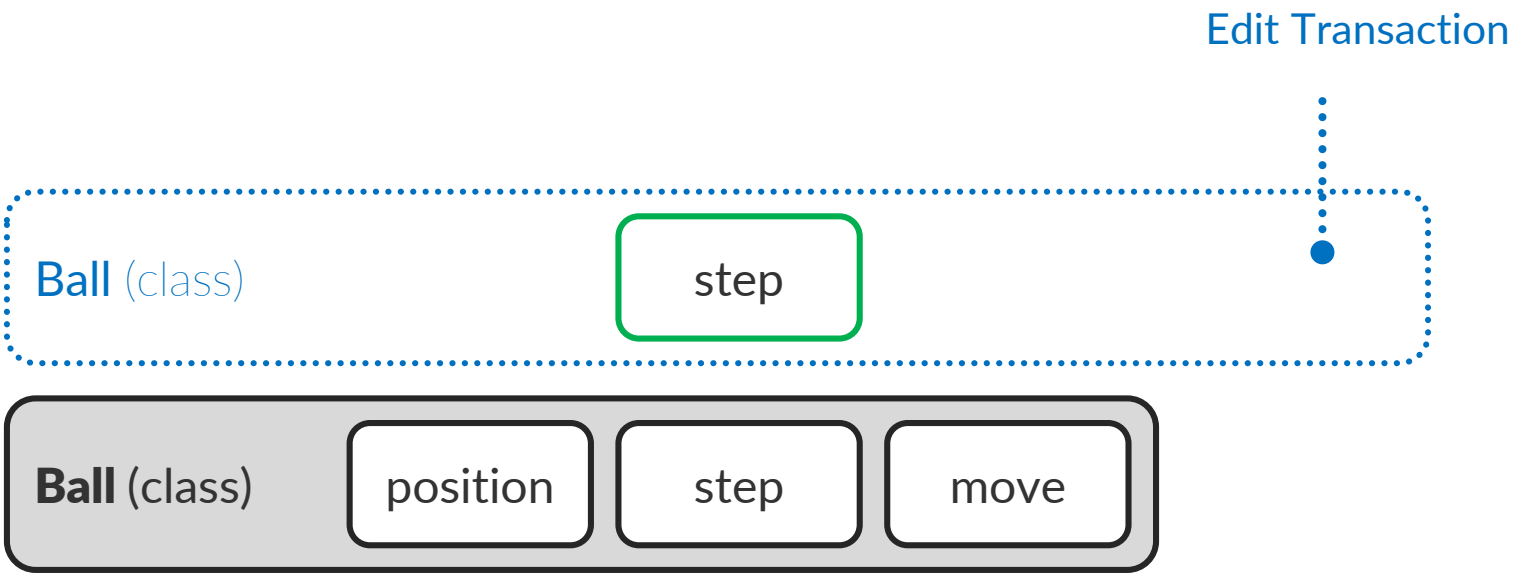
We propose "Edit Transactions"

- Collect a group of changes
- Activate and deactivate atomically

Edit Transactions: Collecting Changes

```
Ball >> step
  self reflect; move; gravitate

Ball >> move
  self position: (self position + self speed)

Ball >> reflect "Check bounds and reflect"
```

insert call

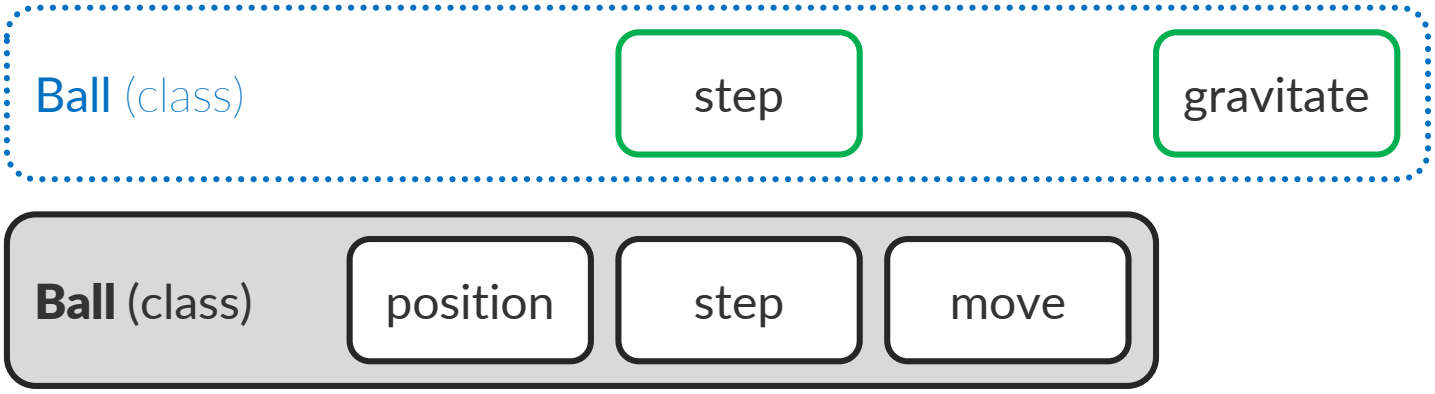


Edit Transactions: Collecting Changes

```
Ball >> step
  self reflect; move; gravitate
Ball >> gravitate [...]
Ball >> move
  self position: (self position + self speed)

Ball >> reflect "Check bounds and reflect"
```

insert call

implement method

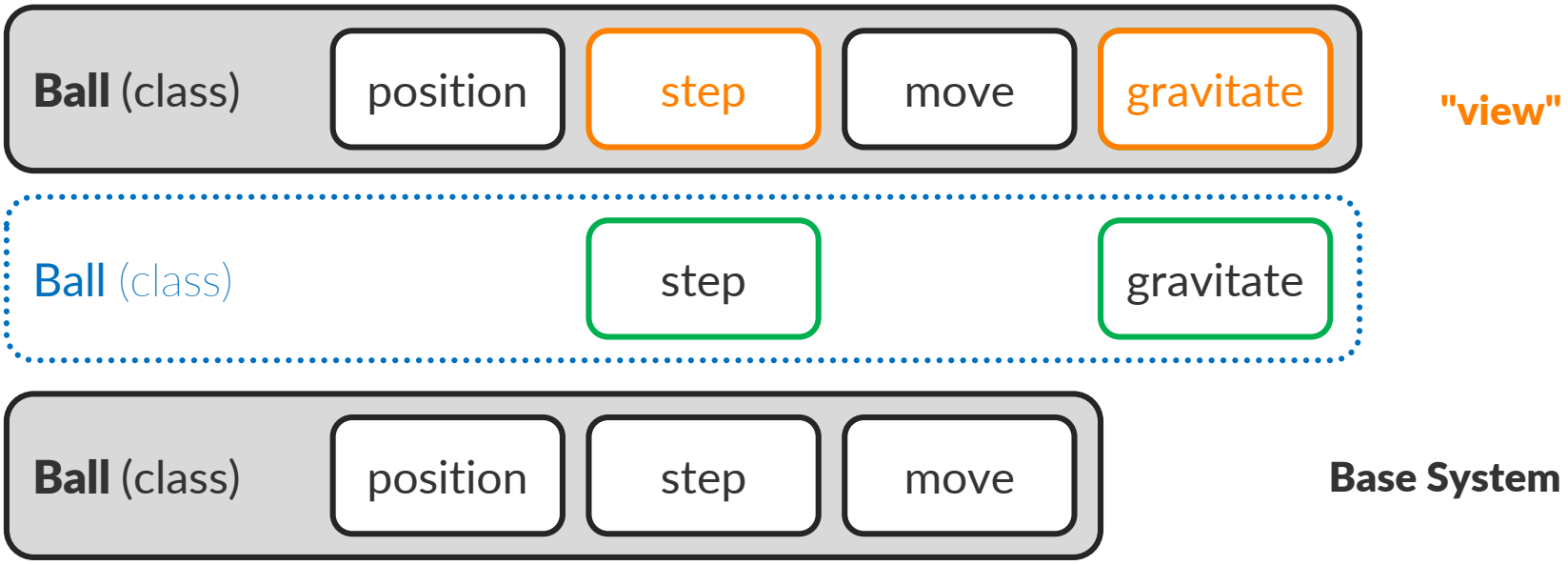


Edit Transactions: Activation

```
Ball >> step
  self reflect; move; gravitate
Ball >> gravitate [...]
Ball >> move
  self position: (self position + self speed)

Ball >> reflect "Check bounds and reflect"
```

insert call

implement method

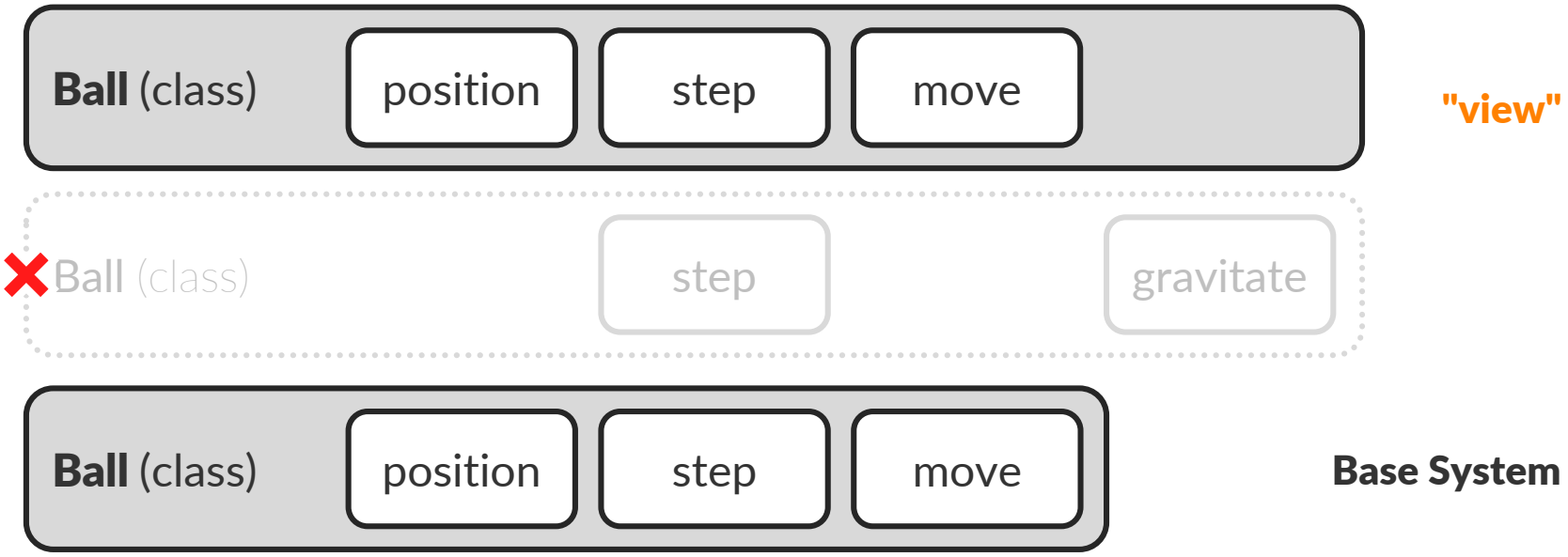


Edit Transactions: Undo

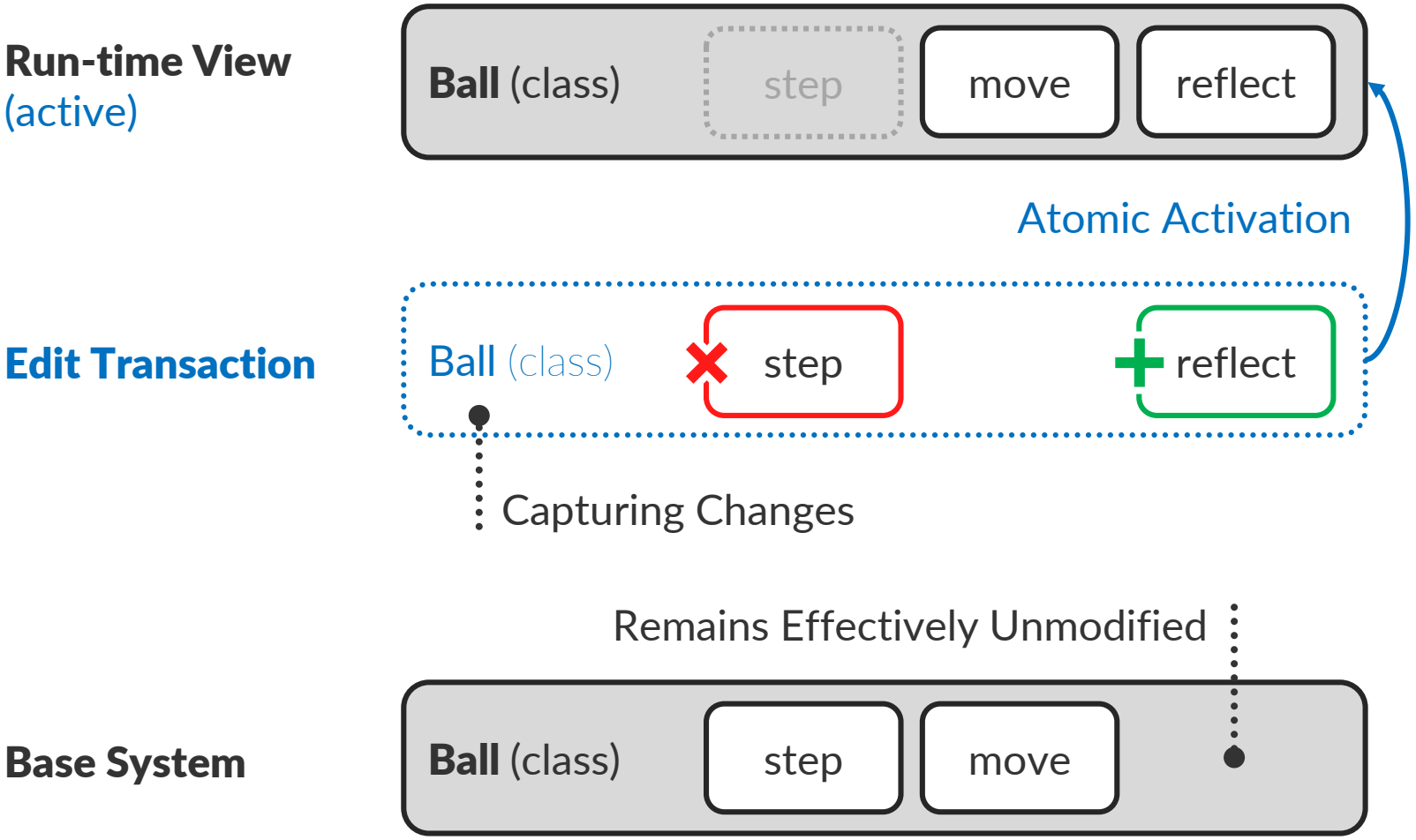
```
Ball >> step
  self reflect; move; gravitate
Ball >> gravitate [...]
Ball >> move
  self position: (self position + self speed)

Ball >> reflect "Check bounds and reflect"
```

insert call

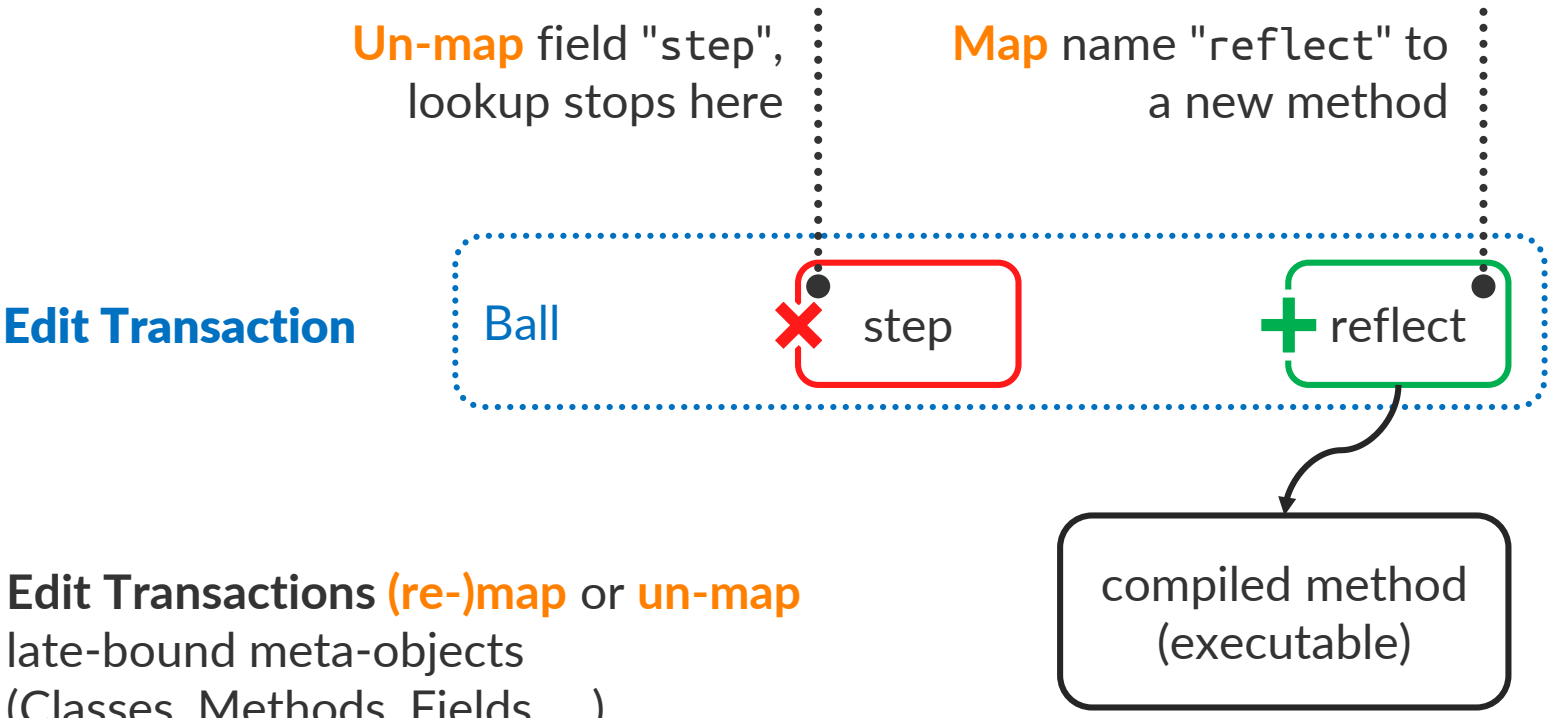
implement method



Edit Transactions: Operations



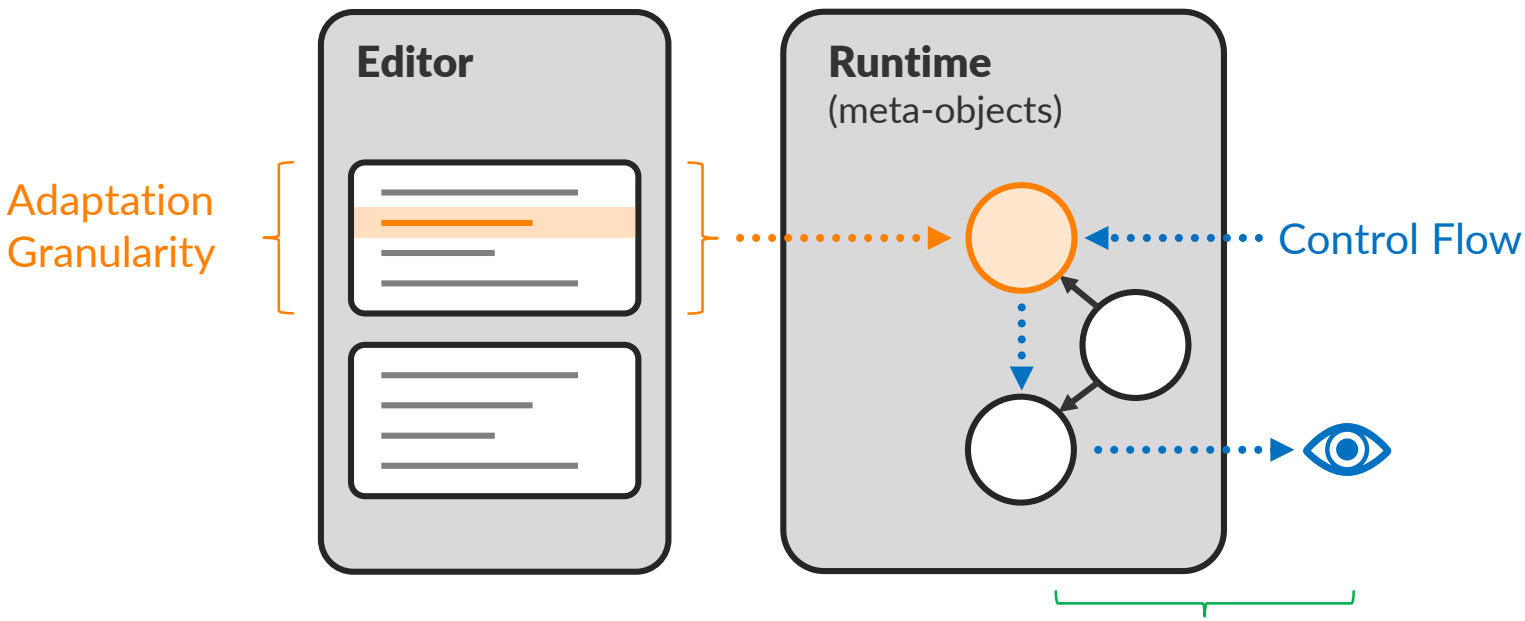
Edit Transactions as Dispatchers



Edit Transactions (re-)map or un-map late-bound meta-objects (Classes, Methods, Fields, ...).

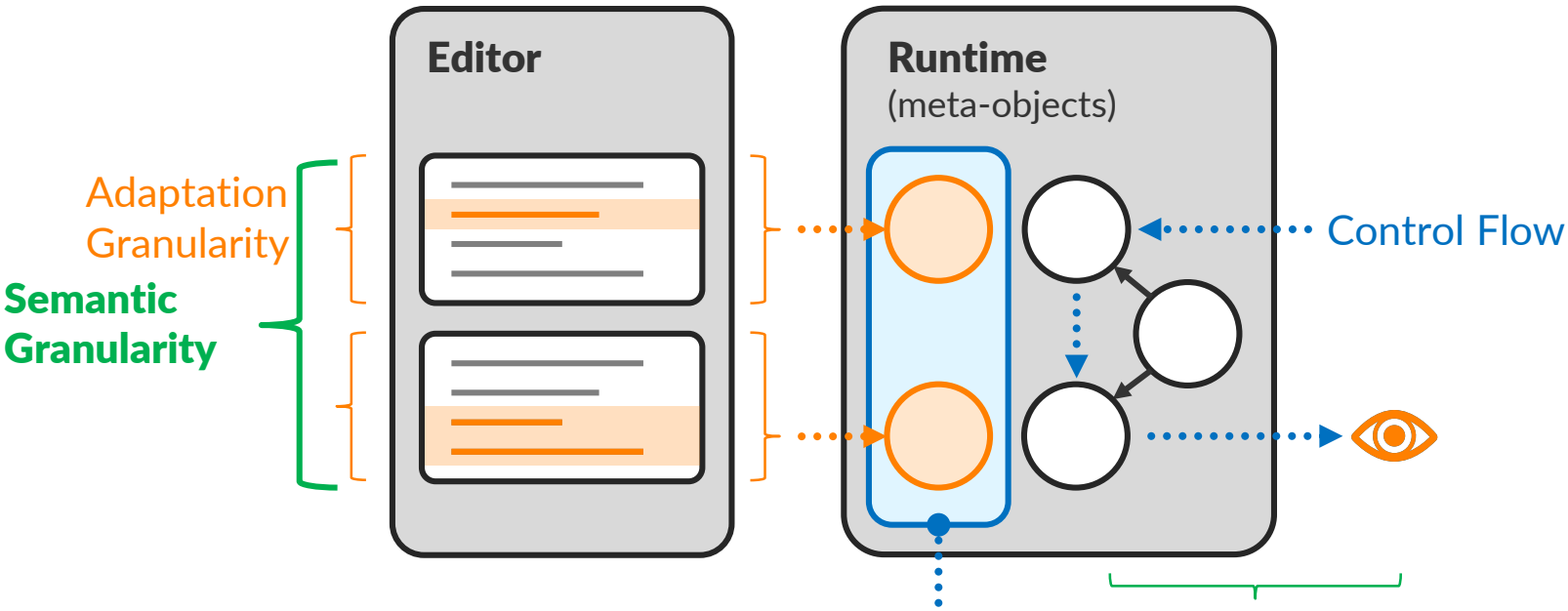
Changes stored as executable meta-objects.

Adaptation:
Translation of an edit interaction into executable representation



Emergence:
Changes to executable representation taking effect

Adaptation:
Translation of an edit interaction into executable representation

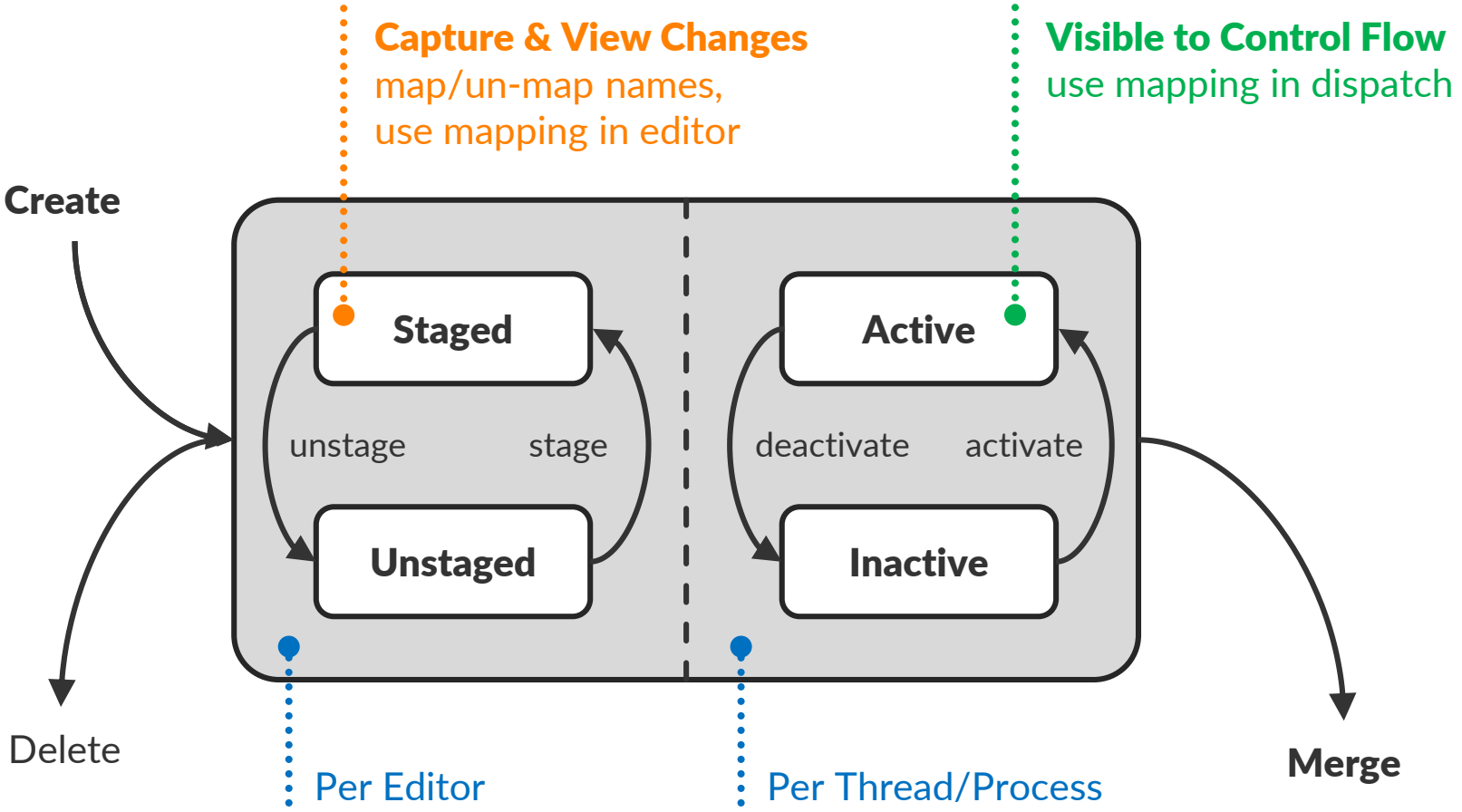


Emergence:
Changes to executable representation taking effect

Edit Transactions

- decouple adaptation from emergence
- support semantic change granularity

Edit Transaction Lifecycle



Edit Transaction Scoping

Staged only

Review code (e.g. imported from version control), check if it compiles, run static analyses …

Control-flow Activation

```
[ block ] withTransactions: et.
```

Run unit tests (auto-tester), experiment with change

Thread/Global activation

```
et activateFor: aProcess.
et activateGlobally.
```

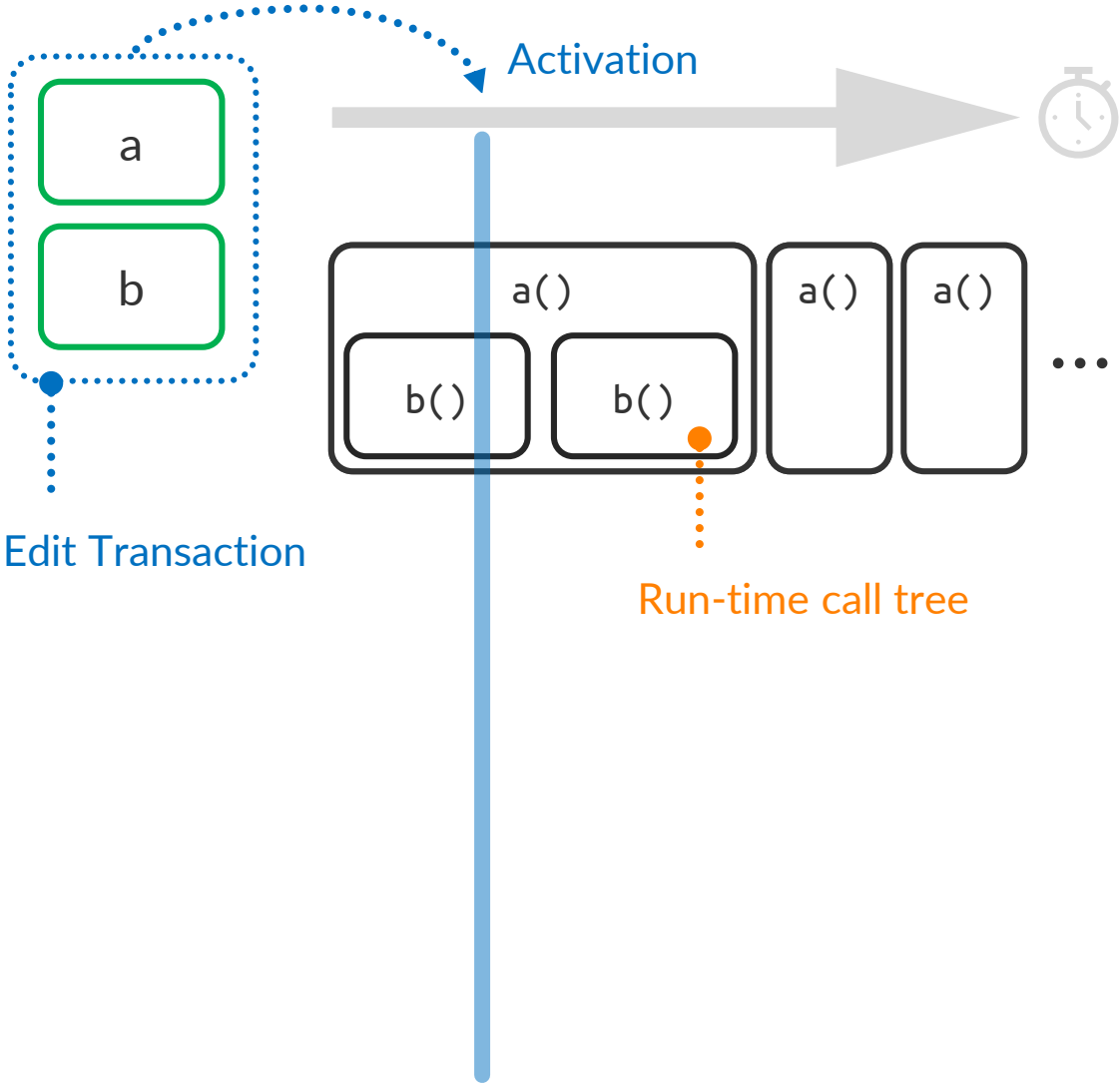
Change considered "stable", asynchronous

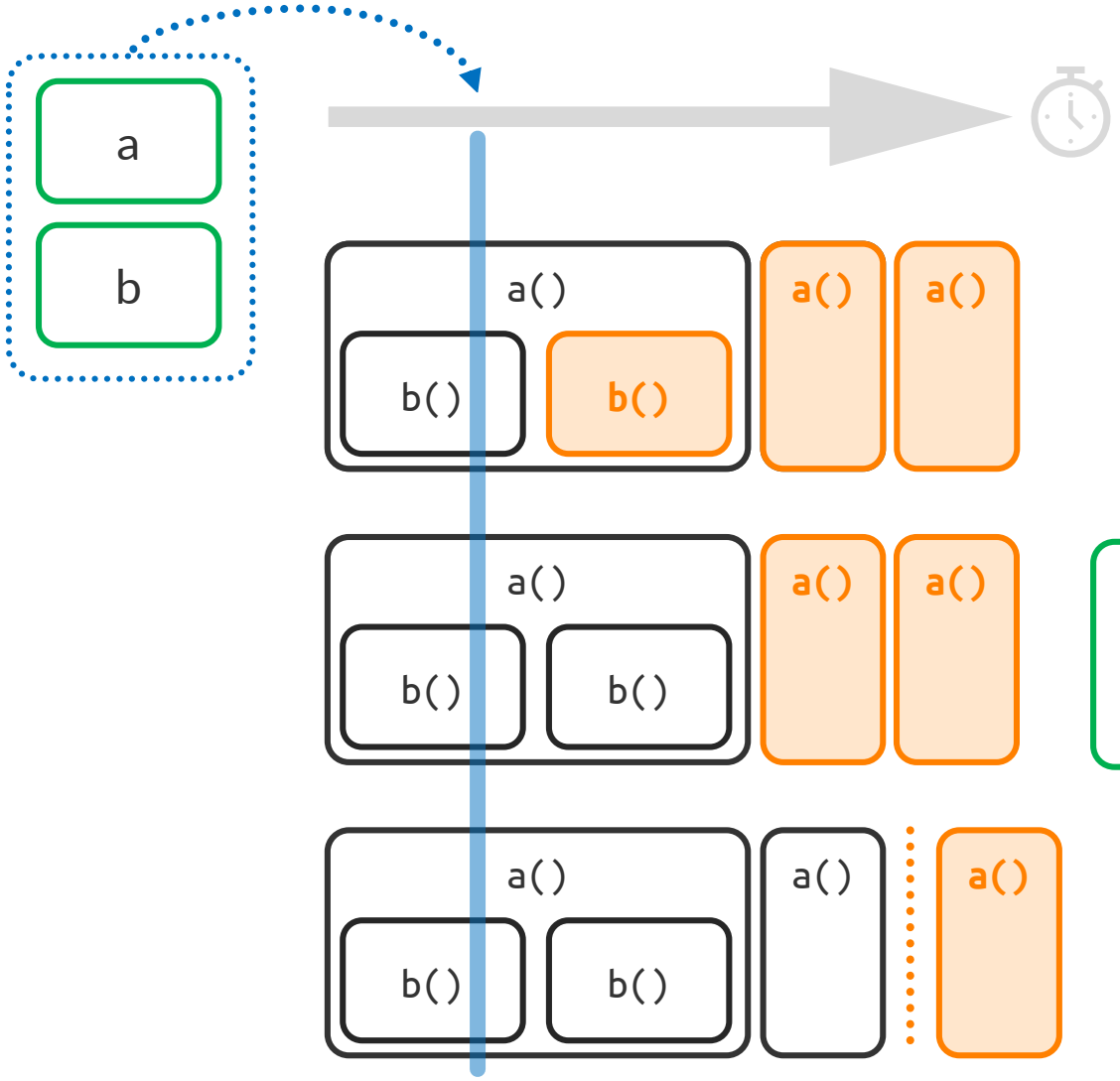
Types of Changes (Class-based OOP)

Meta-object	Operation	
Class	add	
	remove	
	update hierarchy	
	rename	● remove + add
Method/ Function	add	
	remove	
	update implementation	● add (overwrite)
	rename	
Field (Instance Variable)	add	
	remove	
	rename	

Challenges:

1. Identify safe point to activate a composite change
2. Ensure safe composition of multiple edit transactions
3. Late-bound, scoped state (implementation-specific)
4. Intercepting message dispatch (implementation-specific)


Activation
a
b
a()
b()
b()
a()
a()
Edit Transaction
Run-time call tree



Method atomicity
(mixed-version call stack)

default

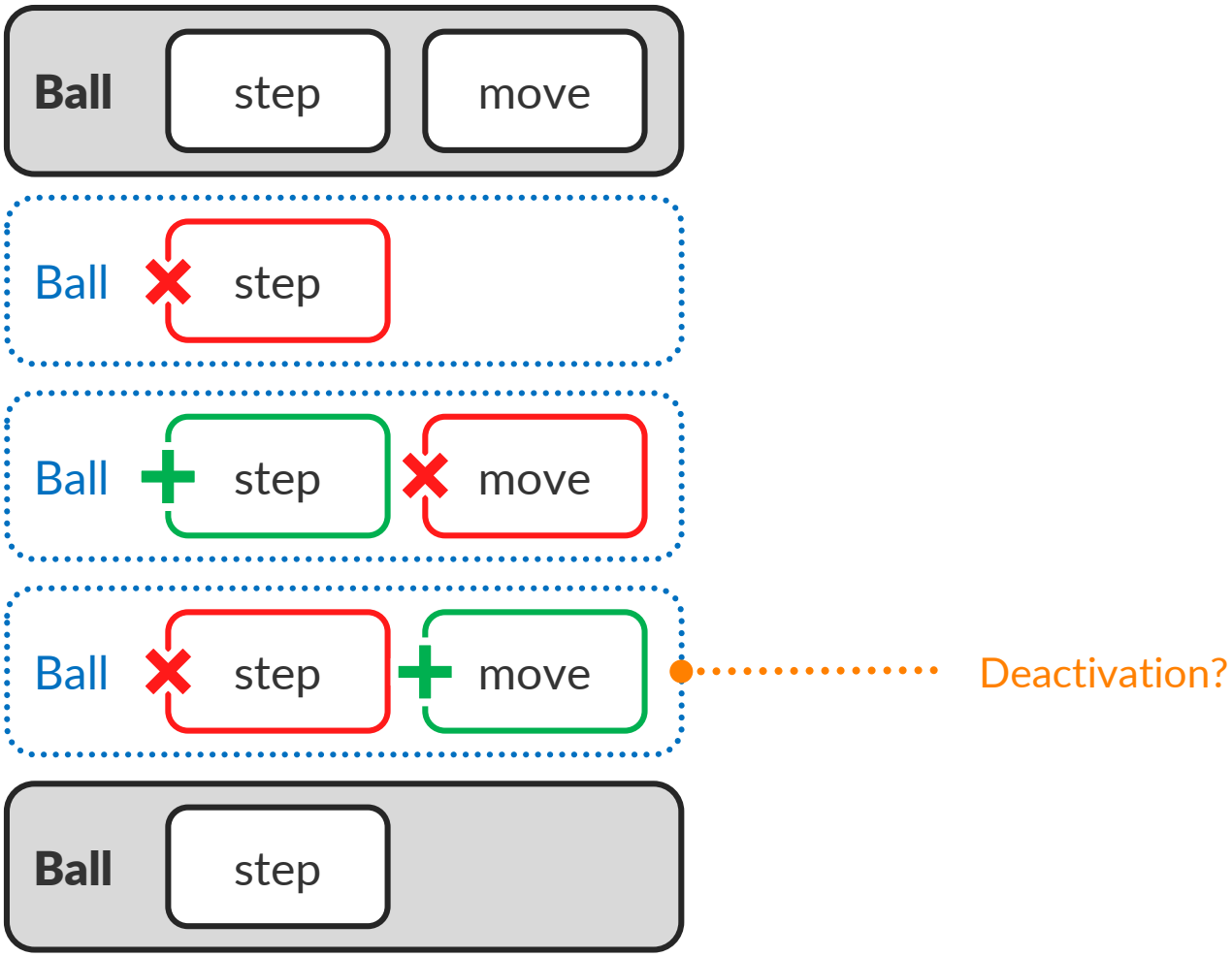
Re-entrant atomicity
(single-version call stack)

Explicit boundaries
`activate` call,
`atomic do: [ … ]`

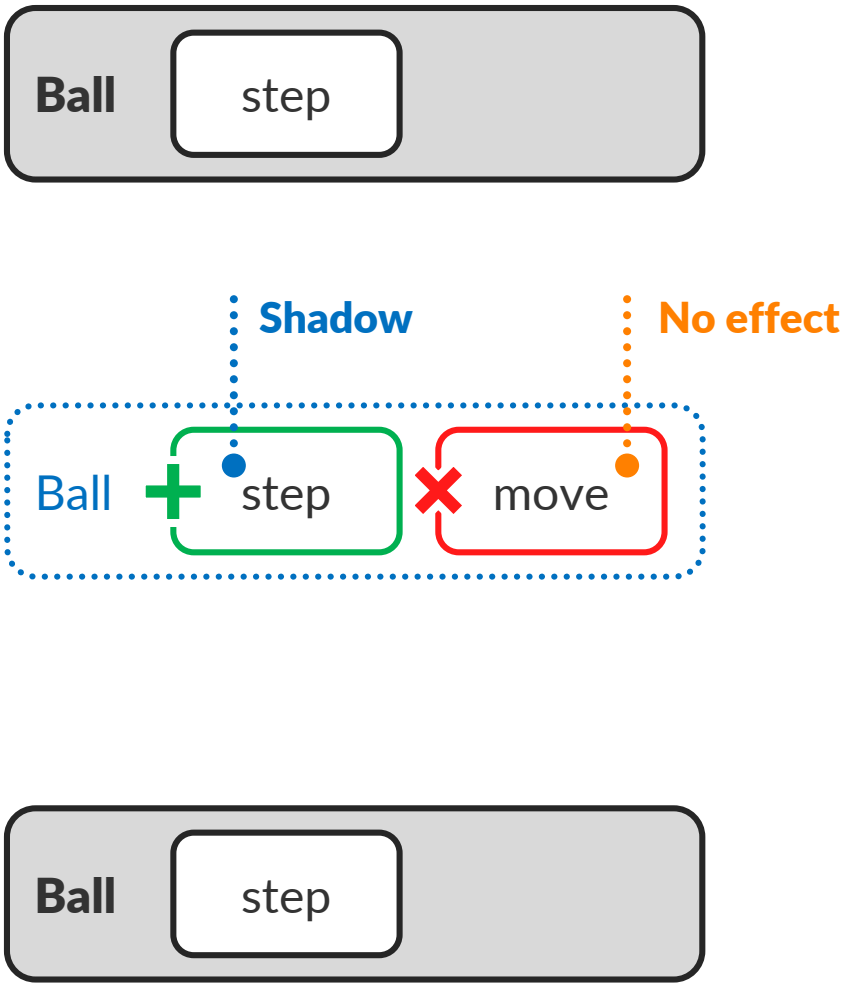
Challenges:

1. Identify safe point to activate a composite change
2. Ensure safe composition of multiple edit transactions
3. Late-bound, scoped state (implementation-specific)
4. Intercepting message dispatch (implementation-specific)

Composition



Composition: Conflicting Activation

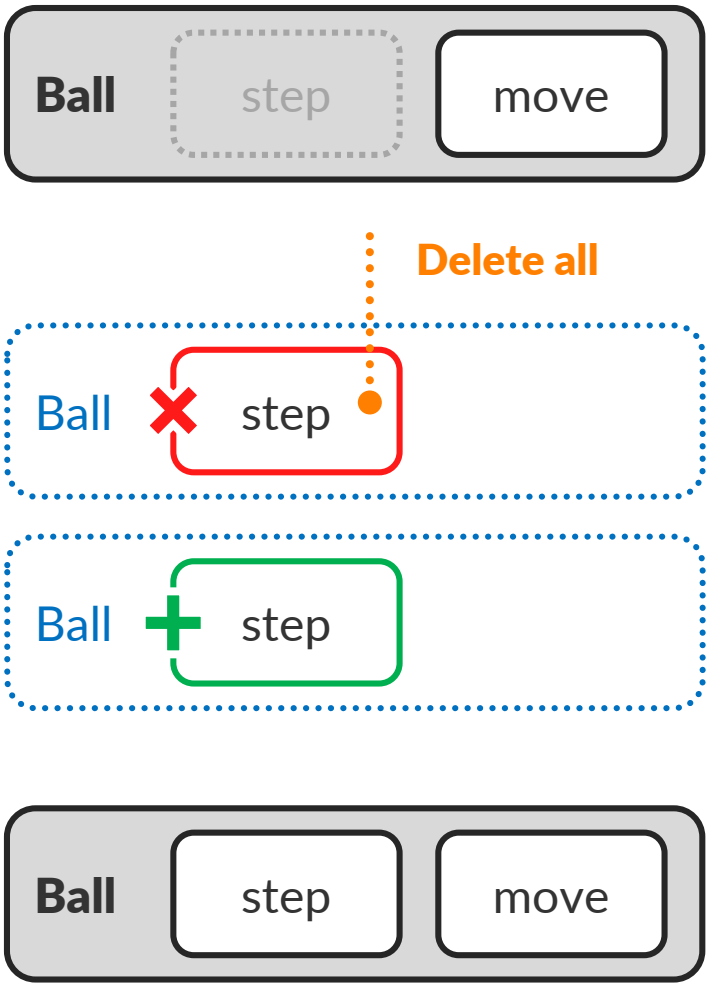


Minimize friction during composition:

Adding existing meta-objects shadows the underlying one (even if inherited)

Removing non-existing meta-objects should not have an effect (also does not block inherited ones)

Composition: Least Surprise

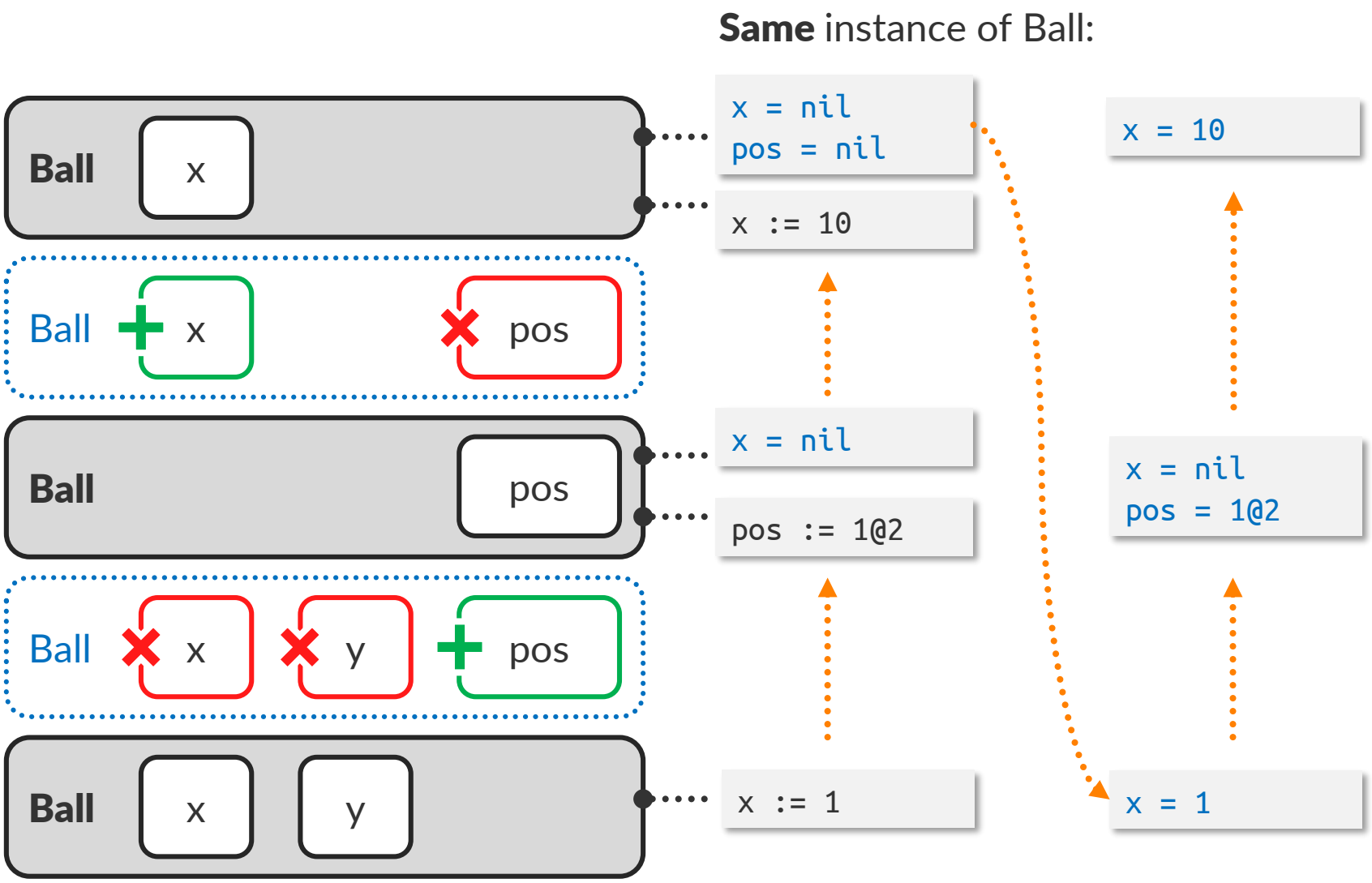


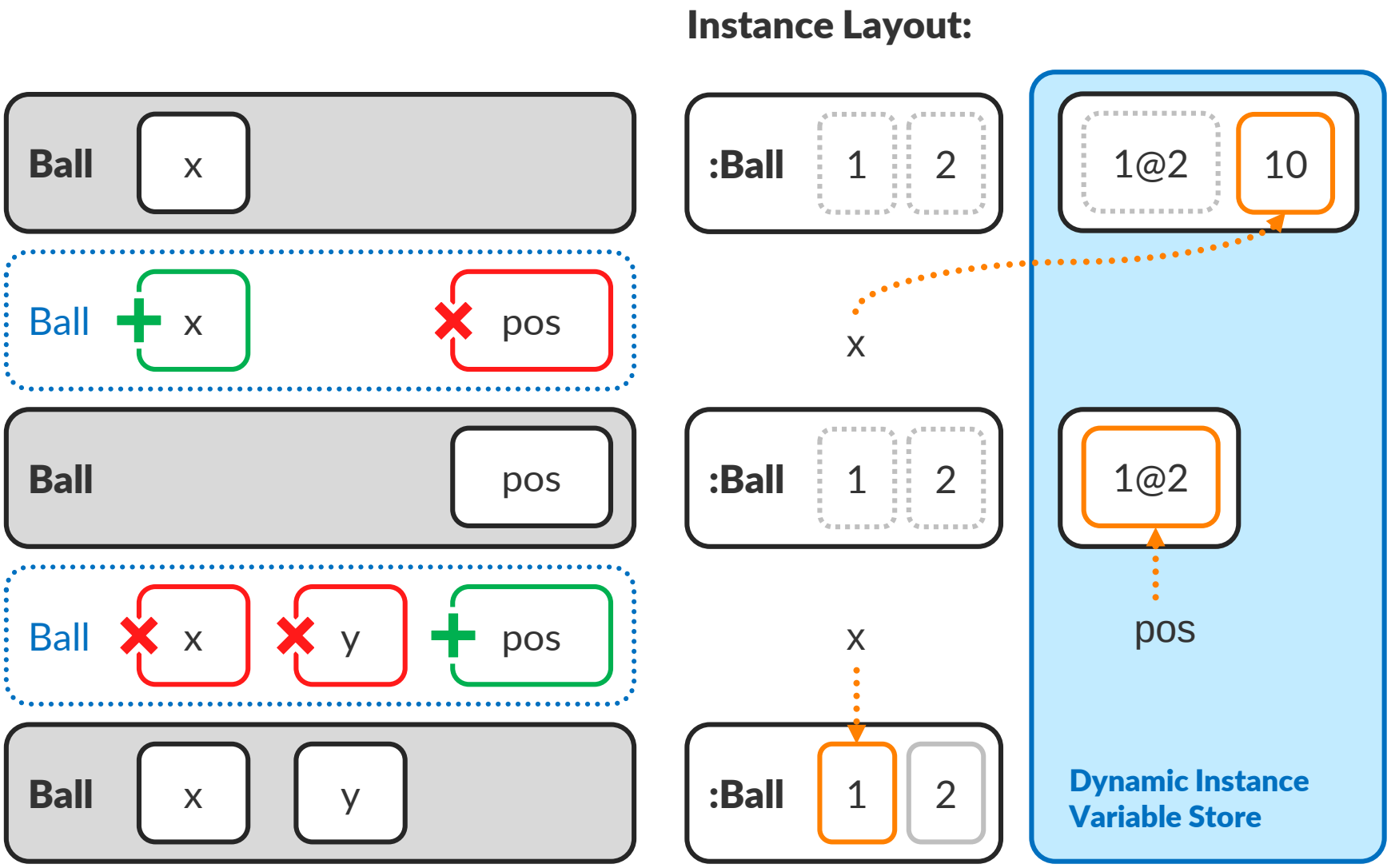
Prioritize most recent intent:

Removing doubly added meta-objects blocks all of them and prevents old methods or field data from becoming readable again.

Challenges:

1. Identify safe point to activate a composite change
2. Ensure safe composition of multiple edit transactions
3. Late-bound, scoped state (implementation-specific)
4. Intercepting message dispatch (implementation-specific)


Same instance of Ball:
Ball
x
Ball
x
pos
Ball
pos
Ball
x
y
pos
Ball
x
y
x = nil
pos = nil
x := 10
x = nil
pos := 1@2
x := 1
x = 10
x = nil
pos = 1@2
x = 1


Instance Layout:
Ball
x
Ball
x
pos
Ball
pos
Ball
x
y
pos
Ball
x
y
:Ball
1
2
:Ball
1
2
:Ball
1
2
1@2
10
x
1@2
pos
x
Dynamic Instance Variable Store

Concurrent Implementation

L-Value:

```
x := expr
```

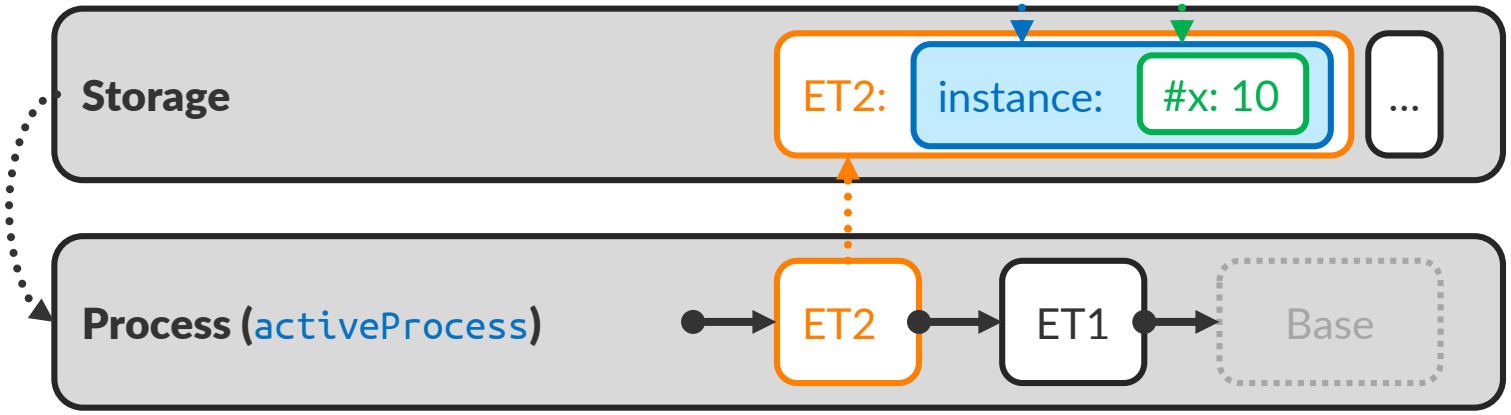
```
Storage for: self at: #x put: (expr)
```

Compile-time Transform

R-Value:

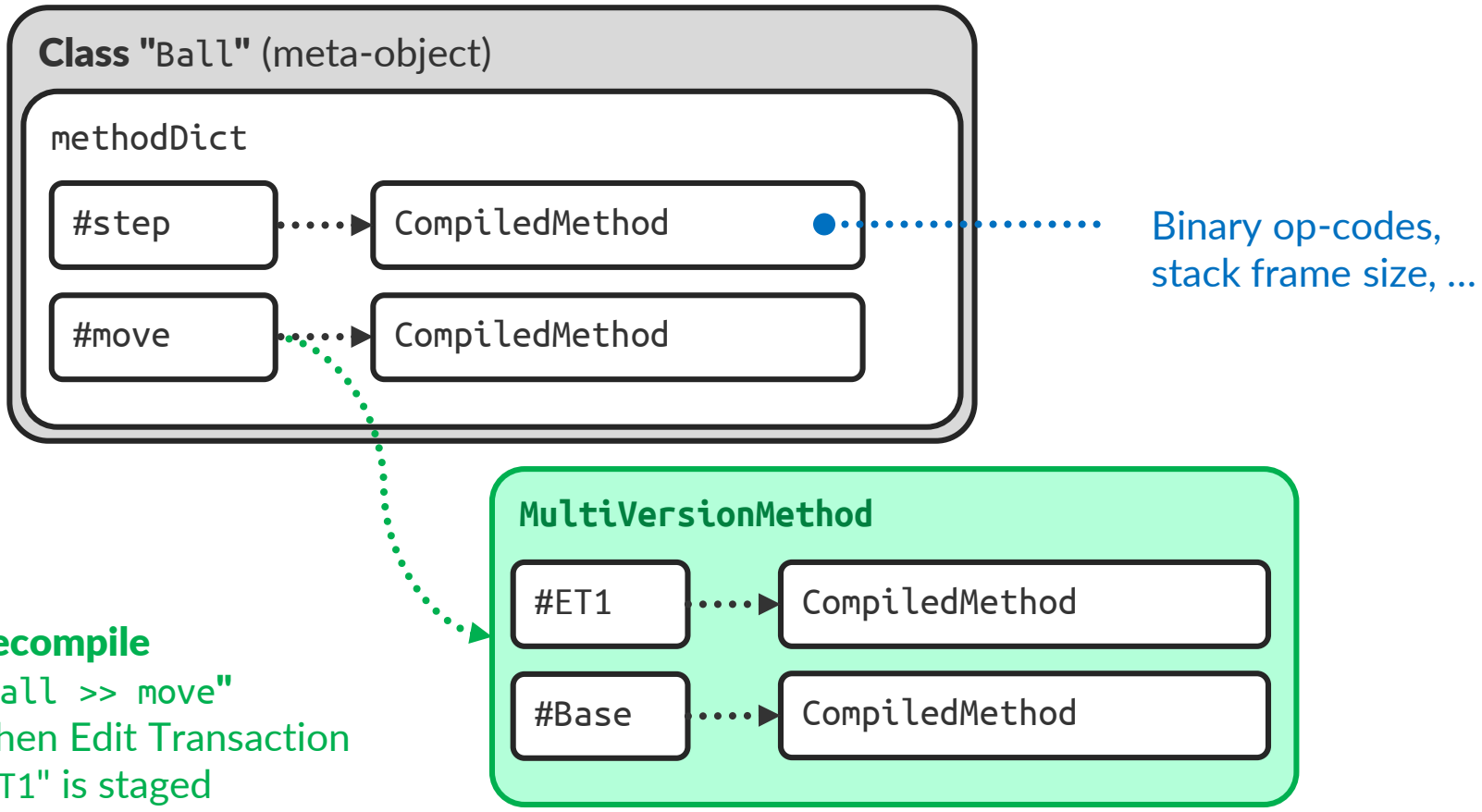
```
x
```

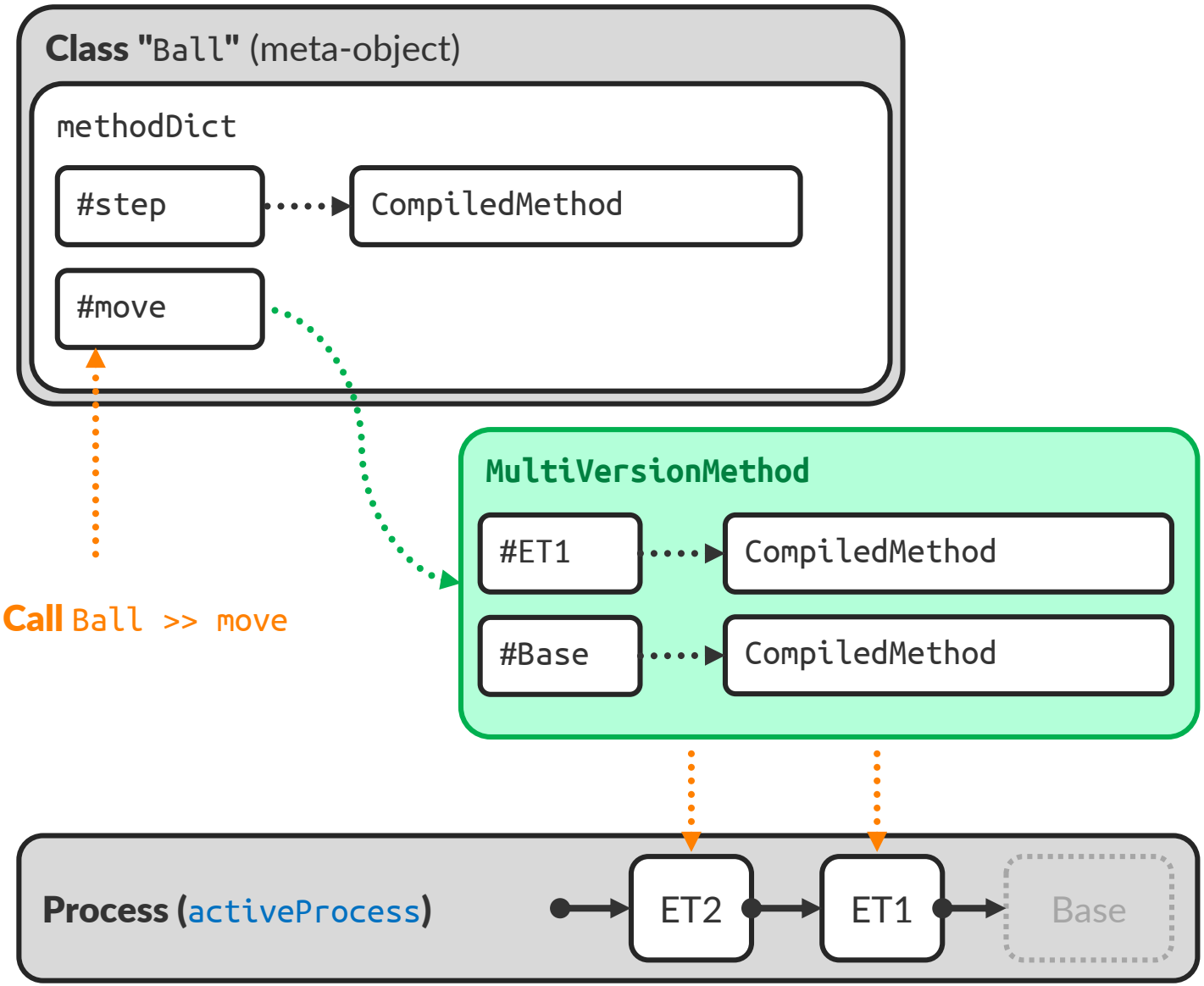
```
Storage for: self at: #x
```



Challenges:

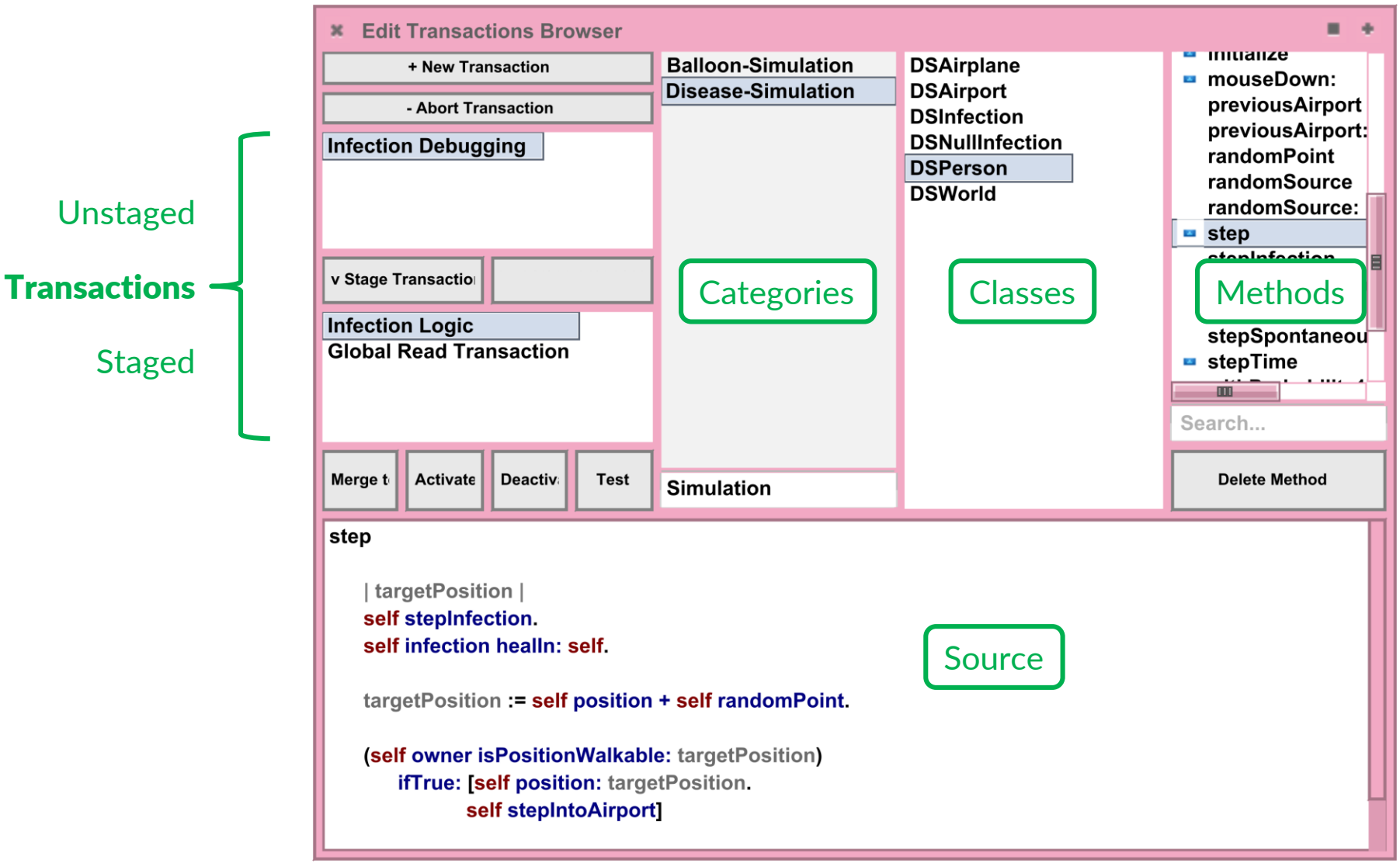
1. Identify safe point to activate a composite change
2. Ensure safe composition of multiple edit transactions
3. Late-bound, scoped state (implementation-specific)
4. Intercepting message dispatch (implementation-specific)

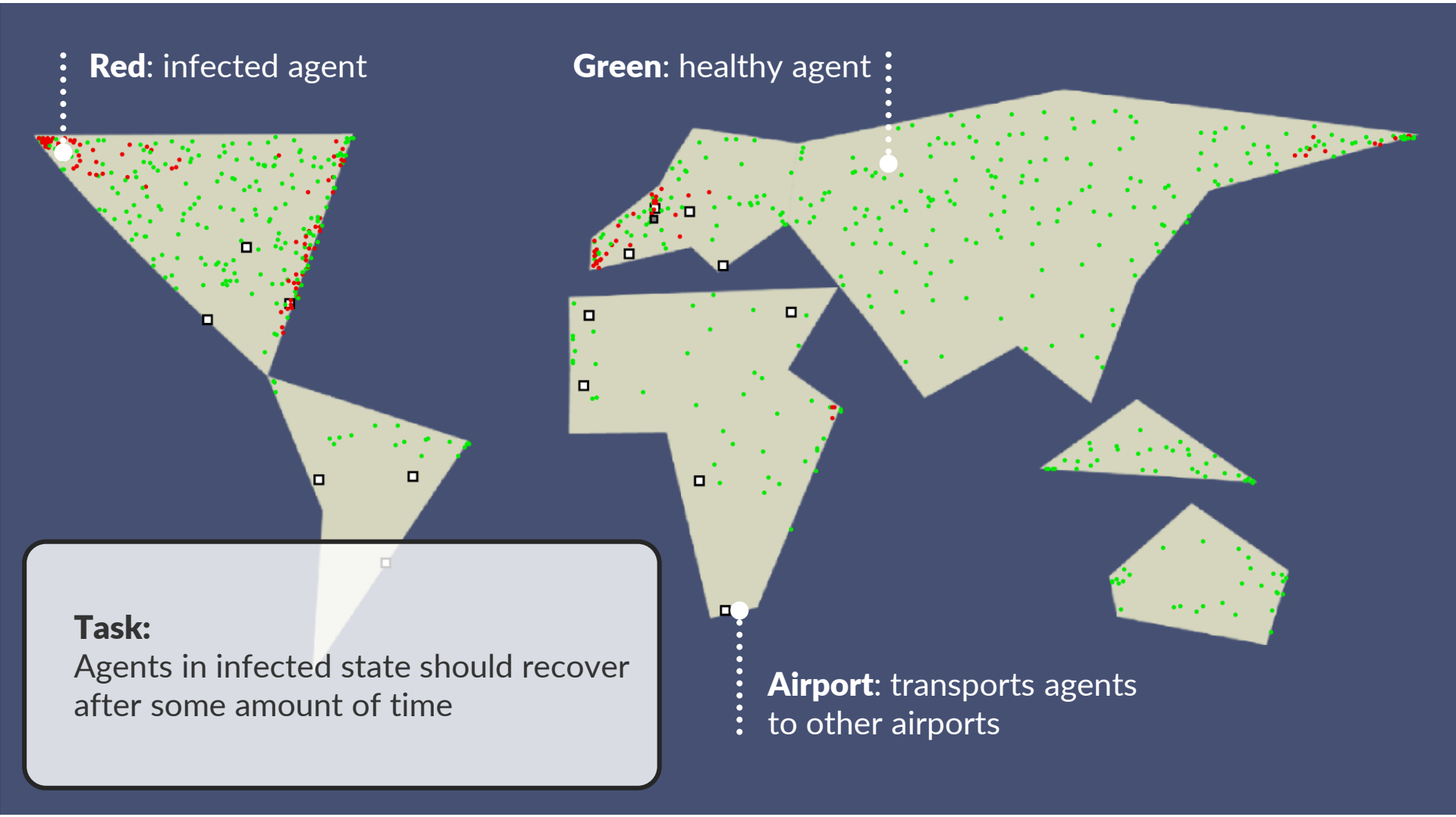

Class "Ball" (meta-object)
methodDict
#step
CompiledMethod
#move
CompiledMethod
Binary op-codes,
stack frame size, ...
MultiVersionMethod
#ET1
CompiledMethod
#Base
CompiledMethod
Recompile
"Ball >> move"
when Edit Transaction
"ET1" is staged


Class "Ball" (meta-object)
methodDict
#step
CompiledMethod
#move
MultiVersionMethod
#ET1
CompiledMethod
#Base
CompiledMethod
Call Ball >> move
Process (activeProcess)
ET2
ET1
Base

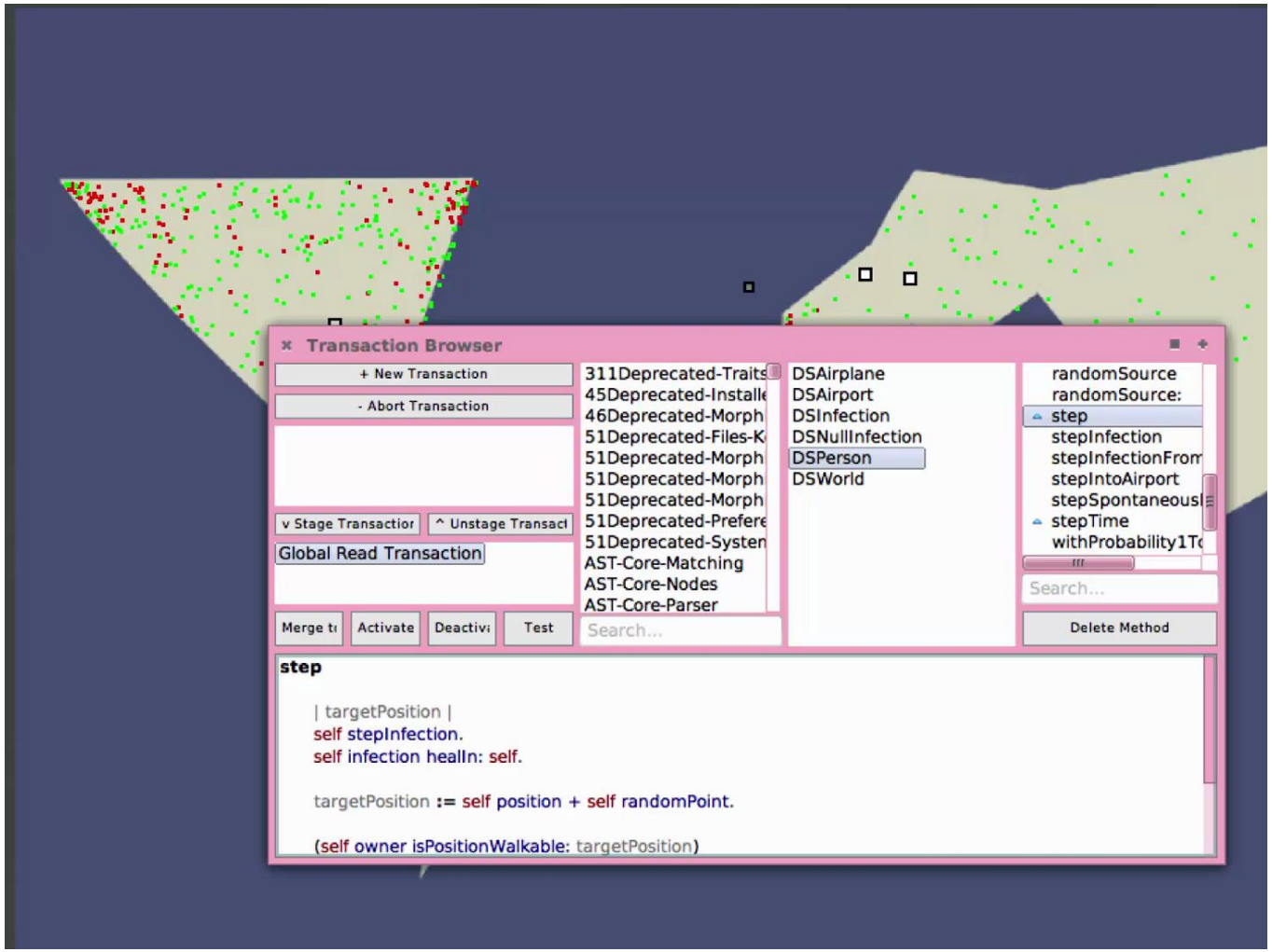
Case Study:

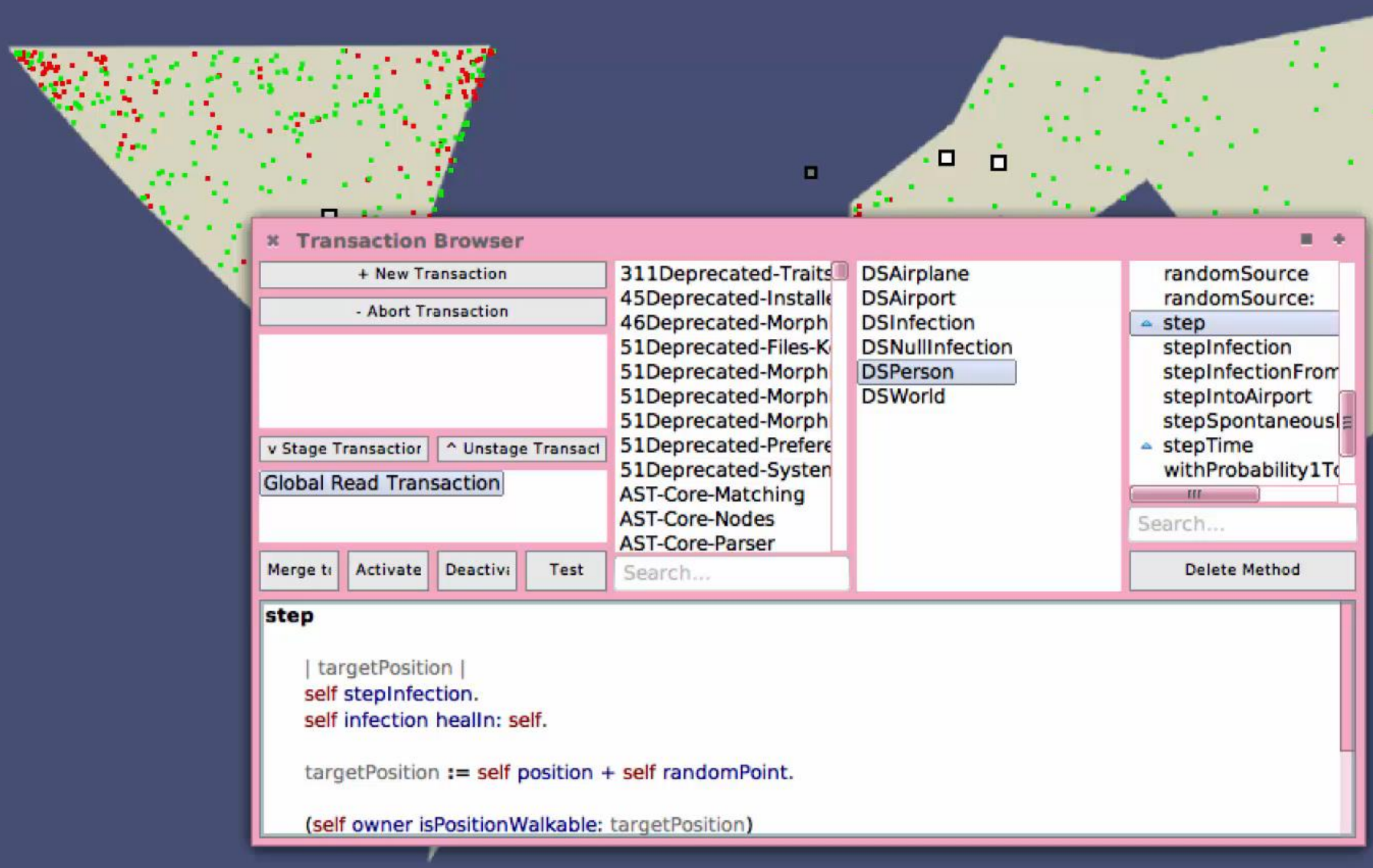
1. Tooling: Transaction-aware editor in Vivide/Smalltalk
2. Setup/Task:
   Adding features to a multi-agent simulation
3. Observed changes in programming workflow


Unstaged
Transactions
Staged
Edit Transactions Browser
+ New Transaction
- Abort Transaction
Infection Debugging
v Stage Transactio
Infection Logic
Global Read Transaction
Merge t
Activate
Deactiv
Test
Balloon-Simulation
Disease-Simulation
Simulation
Categories
DSAirplane
DSAirport
DSInfection
DSNullInfection
DSPerson
DSWorld
Classes
mouseDown:
previousAirport
previousAirport:
randomPoint
randomSource
randomSource:
step
stepSpontaneou
stepTime
Methods
Search...
Delete Method
step
| targetPosition |
self stepInfection.
self infection healIn: self.
targetPosition := self position + self randomPoint.
(self owner isPositionWalkable: targetPosition)
ifTrue: [self position: targetPosition.
self stepIntoAirport]
Source

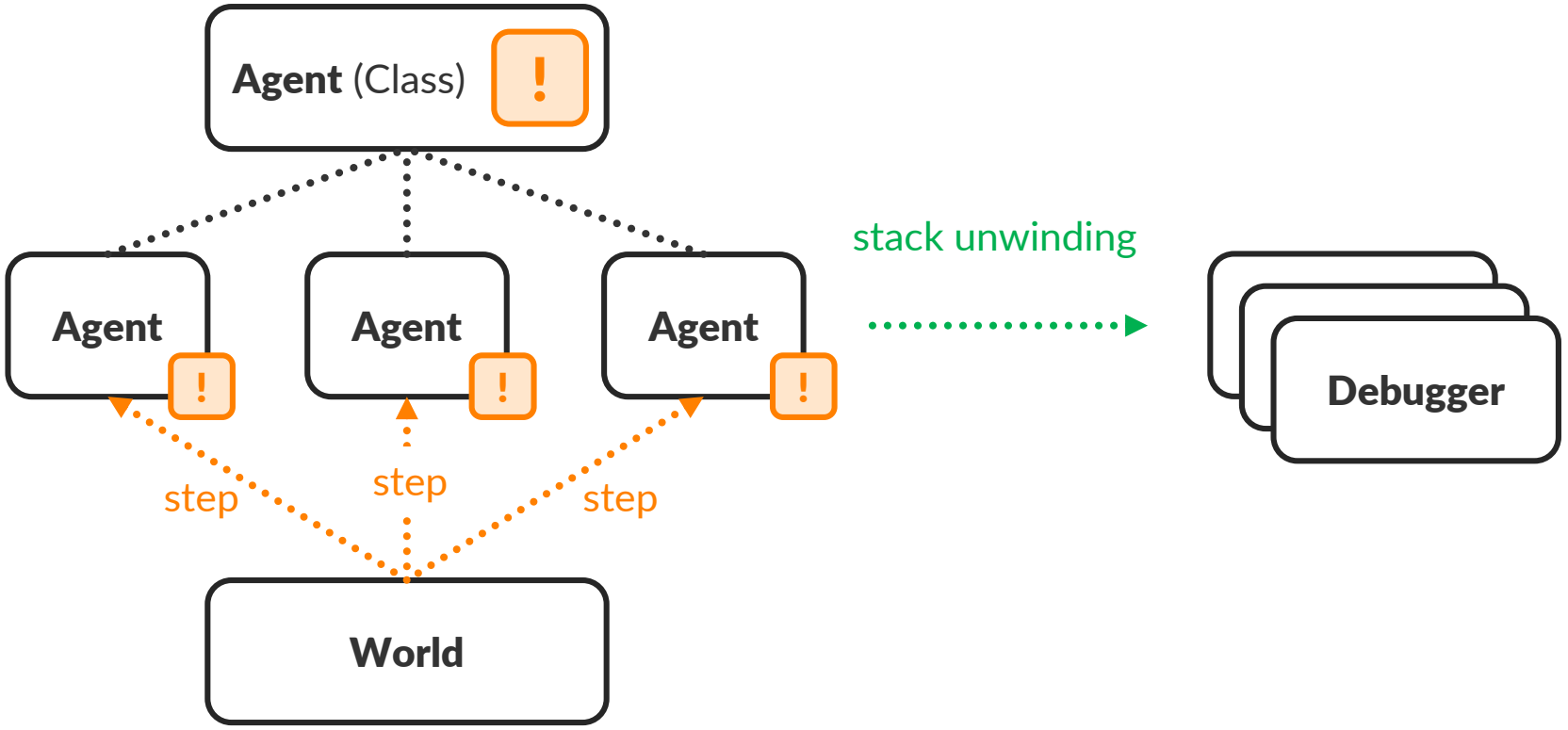

Red: infected agent
Green: healthy agent
Task:
Agents in infected state should recover after some amount of time
Airport: transports agents to other airports

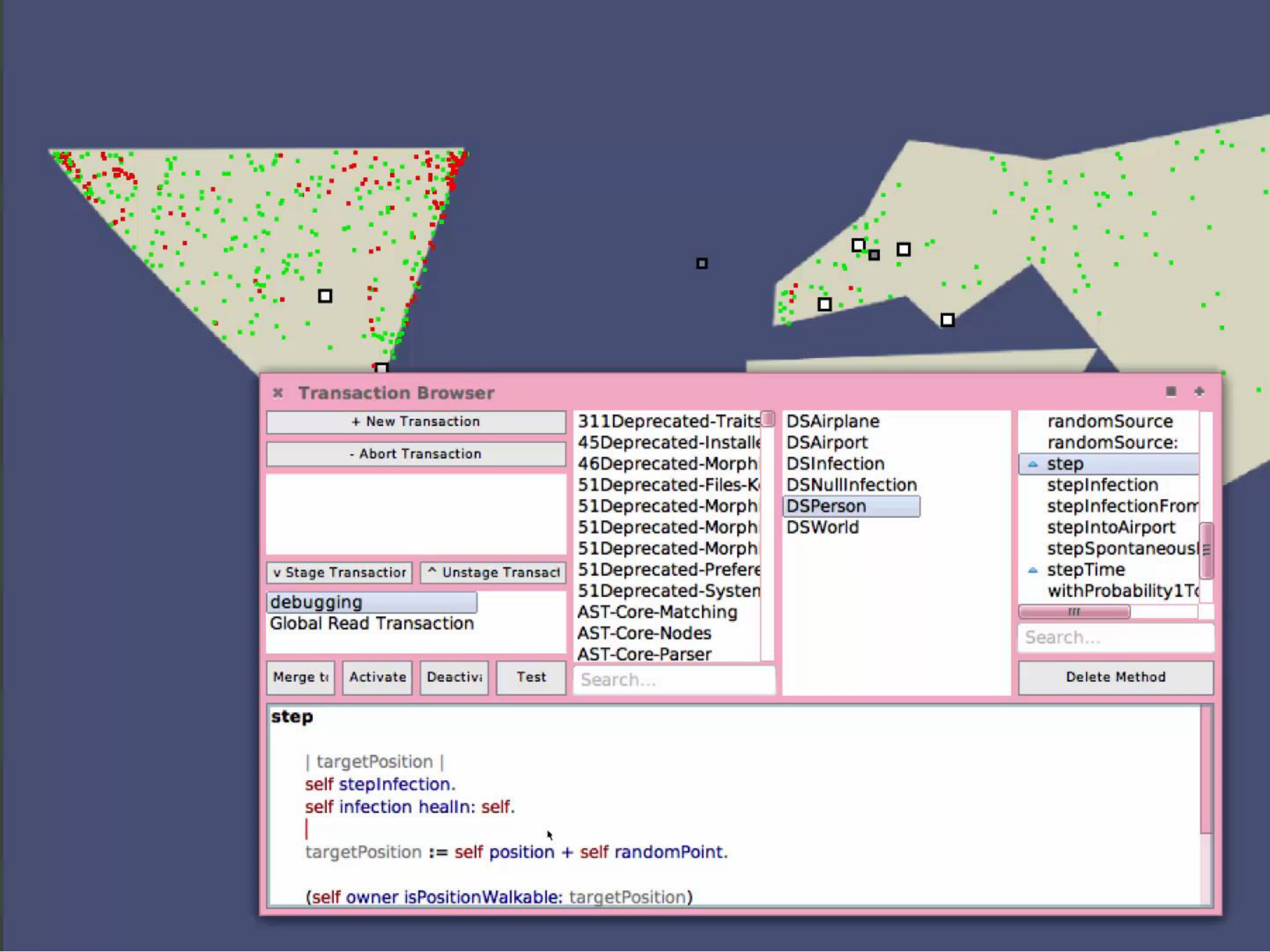
Observation 1: Debuggers



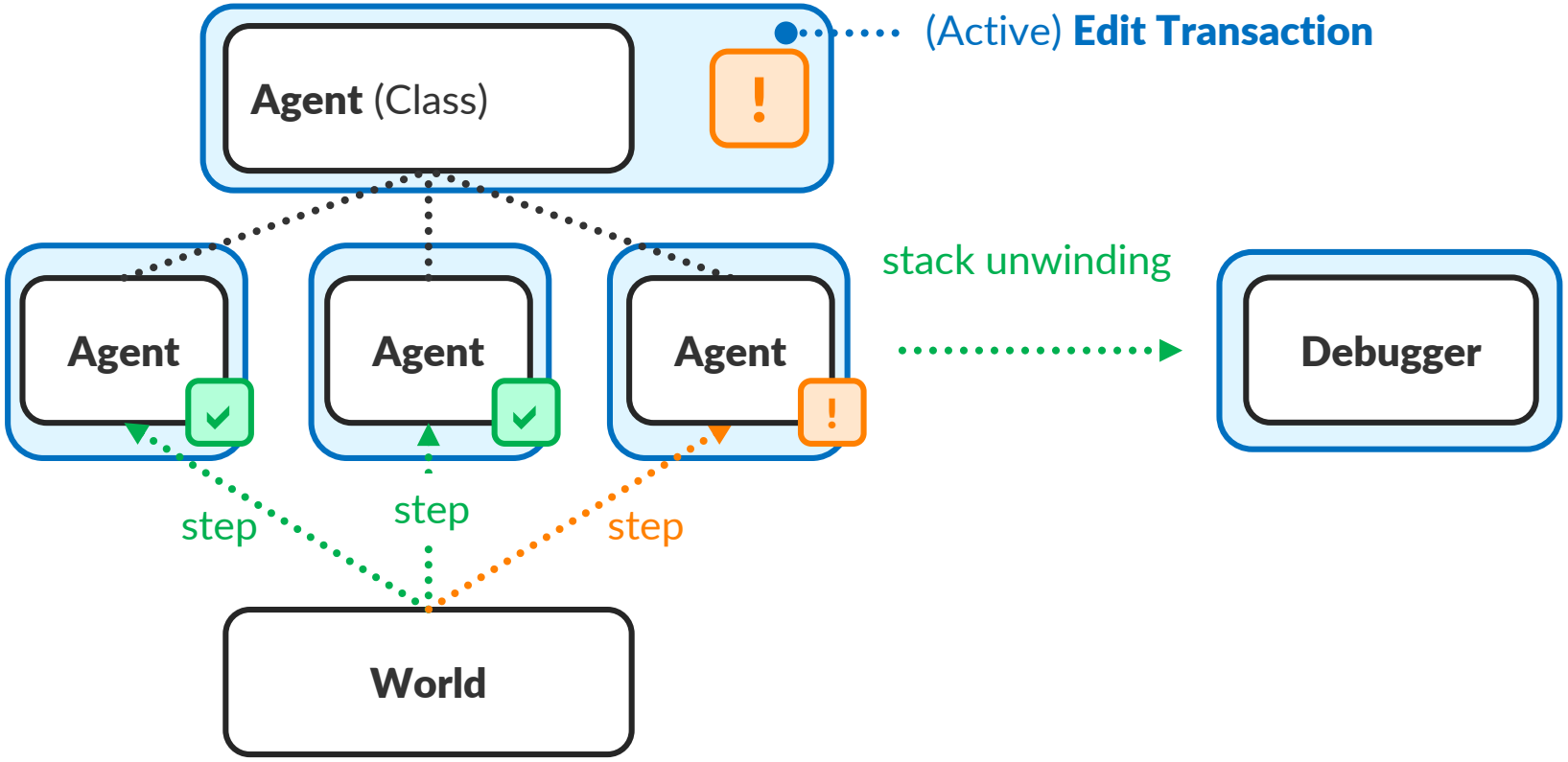

Transaction Browser
+ New Transaction
- Abort Transaction
v Stage Transactior
^ Unstage Transacl
Global Read Transaction
Merge t
Activate
Deactiv
Test
311Deprecated-Traits
45Deprecated-Installe
46Deprecated-Morph
51Deprecated-Files-K
51Deprecated-Morph
51Deprecated-Morph
51Deprecated-Morph
51Deprecated-Prefere
51Deprecated-Systen
AST-Core-Matching
AST-Core-Nodes
AST-Core-Parser
Search...
DSAirplane
DSAirport
DSInfection
DSNullInfection
DSPerson
DSWorld
randomSource
randomSource:
step
stepInfection
stepInfectionFrom
stepIntoAirport
stepSpontaneousl
stepTime
withProbability1T
Search...
Delete Method
step
| targetPosition |
self stepInfection.
self infection healIn: self.
targetPosition := self position + self randomPoint.
(self owner isPositionWalkable: targetPosition)

Debugger catching an Edit Transaction




Transaction Browser
+ New Transaction
- Abort Transaction
v Stage Transactior
^ Unstage Transacl
debugging
Global Read Transaction
Merge t
Activate
Deactiv
Test
311Deprecated-Traits
45Deprecated-Installe
46Deprecated-Morph
51Deprecated-Files-K
51Deprecated-Morph
51Deprecated-Morph
51Deprecated-Morph
51Deprecated-Prefere
51Deprecated-Systen
AST-Core-Matching
AST-Core-Nodes
AST-Core-Parser
Search...
DSAirplane
DSAirport
DSInfection
DSNullInfection
DSPerson
DSWorld
randomSource
randomSource:
step
stepInfection
stepInfectionFrom
stepIntoAirport
stepSpontaneousl
stepTime
withProbability1To
Search...
Delete Method
step
| targetPosition |
self stepInfection.
self infection healIn: self.
targetPosition := self position + self randomPoint.
(self owner isPositionWalkable: targetPosition)

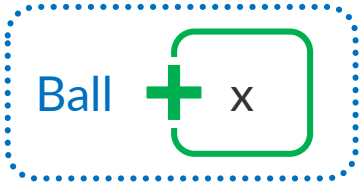
Debugger catching an Edit Transaction



Observation 2: Lazy Initialization

Problem: Adding a field will leave it **unitialized** for all instances.

Solution:




```
Ball >> x
   x ifNil: [x := 0].
   ^ x
```

Observation 3: Less Editor Windows

Normal "composite" workflow:

- Edit Method 1
- Discover dependency and switch to other window
- Edit Method 2
- Save both in correct order

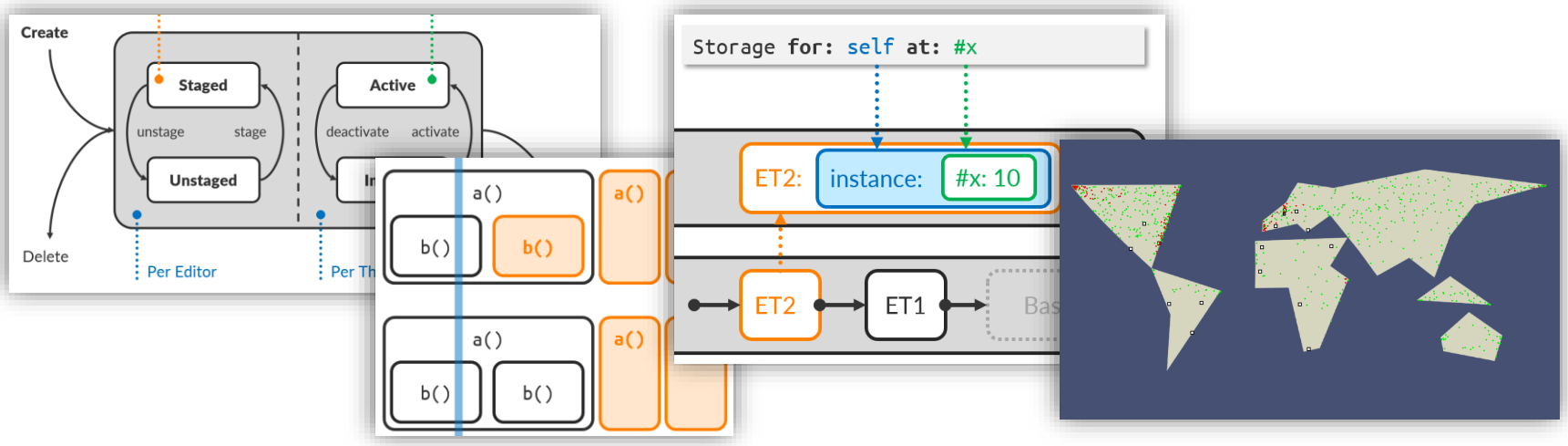
With Edit Transactions:

- Stage new Edit Transaction
- Edit Method 1
- Edit Method 2
- Activate

Conclusion

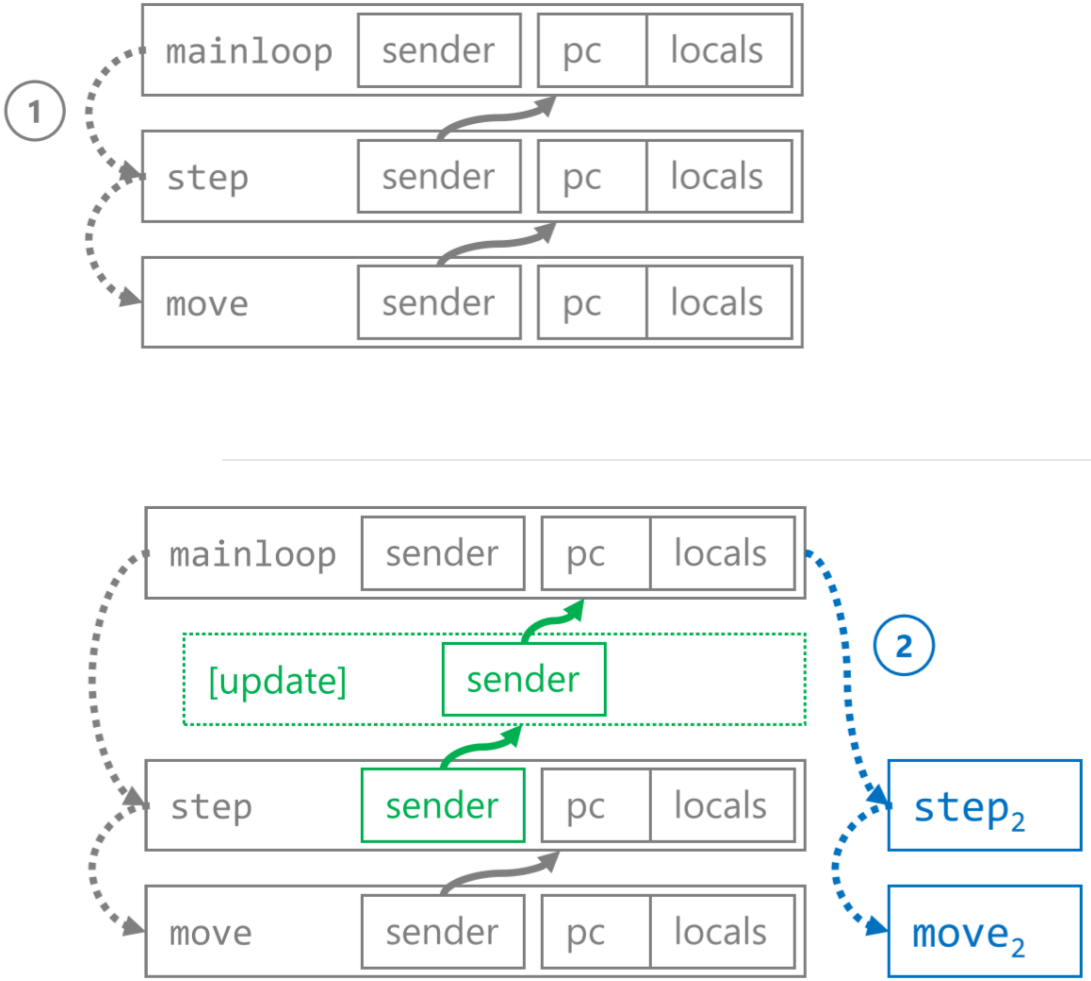
Edit Transactions

- decouple adaptation from emergence in live programming
- allow semantic granularity of emergence without sacrificing live adaptation and test feedback
- trade immediacy for correct granularity of feedback



Additional Slides

Return-frame Insertion



Call and State Access Overhead

	Results in ms (Standard Deviation)			Slow-down	
No of Transactions	Without	Call	State	Call	State
0	14 (6.26)	579 (5.21)	515 (9.85)	41.36	36.79
1	15 (0.00)	610 (2.74)	5,854 (11.75)	40.67	390.27
2	16 (0.42)	654 (1.49)	11,789 (41.95)	40.88	736.81
3	17 (0.00)	683 (1.23)	18,328 (21.32)	40.18	1078.12
4	17 (0.53)	696 (1.52)	21,989 (42.98)	40.94	1293.47
5	18 (0.42)	717 (0.85)	27,653 (51.31)	39.83	1536.28
6	20 (0.00)	737 (1.51)	33,179 (19.39)	36.85	1658.95
7	21 (0.00)	839 (0.82)	38,830 (36.77)	39.95	1849.05
8	21 (0.00)	795 (1.08)	44,368 (64.10)	37.86	2112.76
9	22 (0.00)	822 (3.20)	50,339 (69.76)	37.36	2288.17