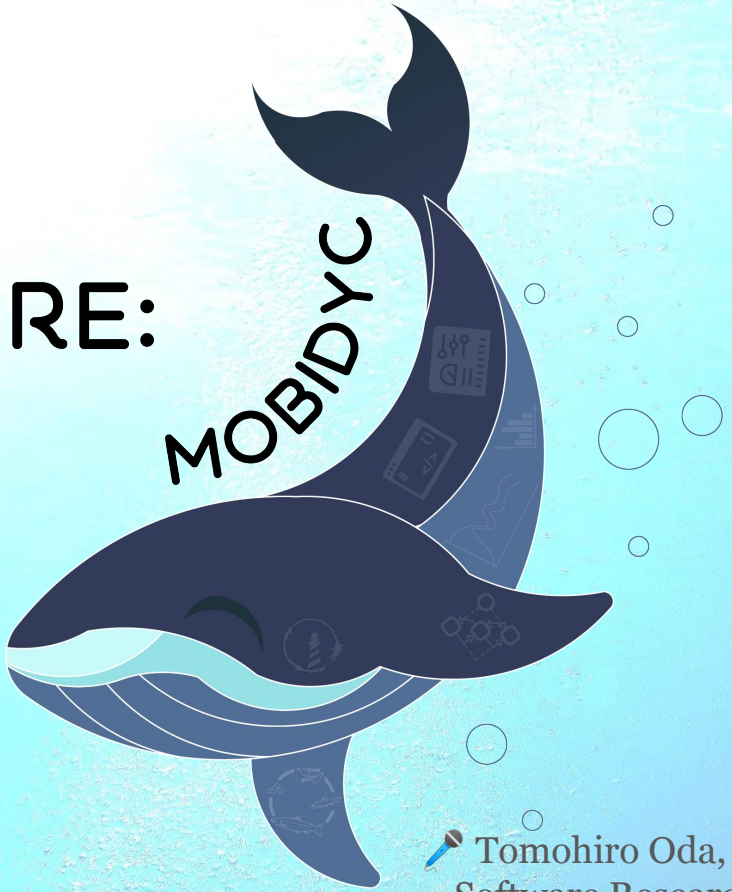
RE: MOBIDYC



🎤 Tomohiro Oda,
Software Research Associates, Inc.
@ESUG2022, Novi Sad, Serbia, Aug 26, 2022

re:mobidyc

the overview

Multi-Agent Simulation

stage species

agent definition

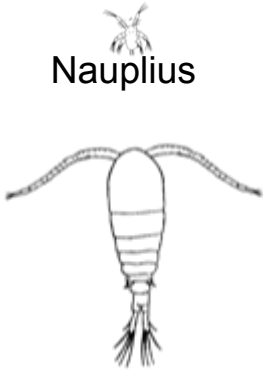
```
Nauplius is EAffinis with
        age [day].

Copepodite is EAffinis with
        age [day]
        heading [degree].
```

action definition

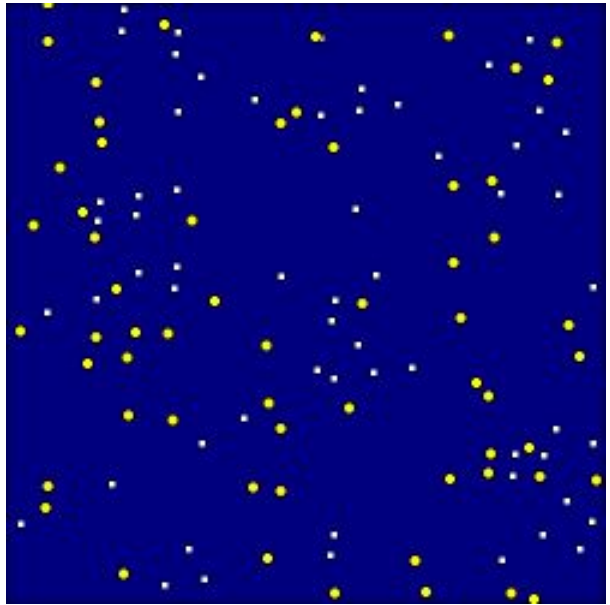
```
to random_walk is
        my d/dt x' = v * cos(theta)
        my d/dt y' = v * sin(theta)
where
        theta = uniform 0 [degree] to 360 [degree]
        v = normal 0 [m/s] sigma the speed.

to swim is
        when uniform 0 to 1 < 0.3
        my d/dt x' = the speed * cos(my heading)
        my d/dt y' = the speed * sin(my heading)
        the d/dt heading' = (normal 0 sigma 2) [degree/s].
```



Nauplius

Copepodite



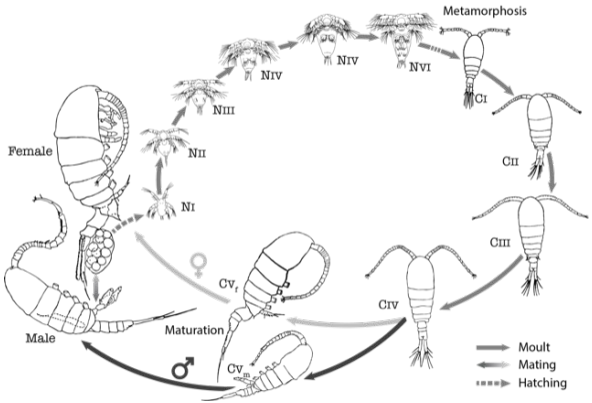
task definition

```
Nauplius random_walk
where
        the speed -> 0.3[mm/s].

Copepodite swim
where
        the heading -> my heading
        the speed -> 1[mm/s].
```

Expected applications

Ecology



Biology

Ecotoxicology

climate change

living marine resource management

aquaculture

Contributors

Individual Contributors (in alphabetical order)

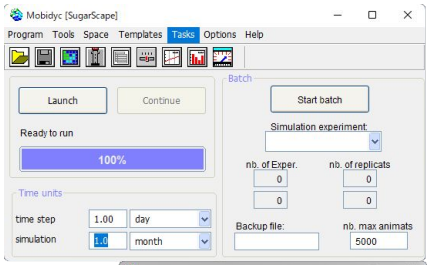
- Feddal Nordine, Université de Lille
- Gaël Dur, Shizuoka University
- Sami Souissi, Université de Lille
- Serge Stinckwich, UN University
- Stéphane Ducasse, Inria
- Tomohiro Oda, Software Research Associates, Inc.

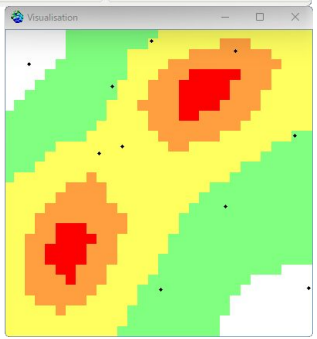
Organizational Supporters

DGtalAqualab
Shizuoka University,
Japan

Key Technology Group
Software Research Associates, Inc.,
Japan

History



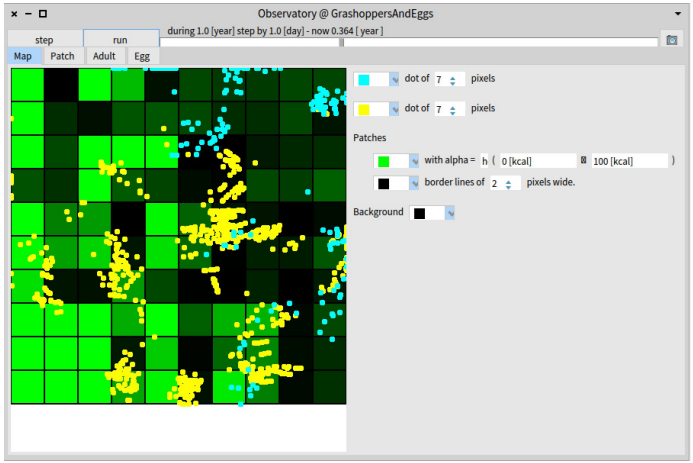


MoBIDyC

- on 32bit VisualWorks
- development started in 1996, gained its full form in 2002, and still in use

re:mobidyc

- on 64bit Pharo 10 + petitparser2
- open source at https://github.com/ReMobidyc/ReMobidyc/
- from the same design principles, we re-designed and added new technical elements.



Features inherited from the original MoBIDyC

- discrete-time individual-based simulation
  - Turtles + Cellular Automata
- modeling without programming skills
  - modeling with minimal coding
- 25 primitives from MoBIDyC.
  - move, eat, kill, die, stage, spawn, ...

In re:mobidyc, we added a few design principles

- reliable models
  - referential transparency: synchronous updates of variables
  - type checking: check models without testing
  - termination: no infinite loop
- reviewable results
  - traceability : record all attributes of all agents at all time steps
  - reproducibility : always the same result from the same model and the same initial conditions

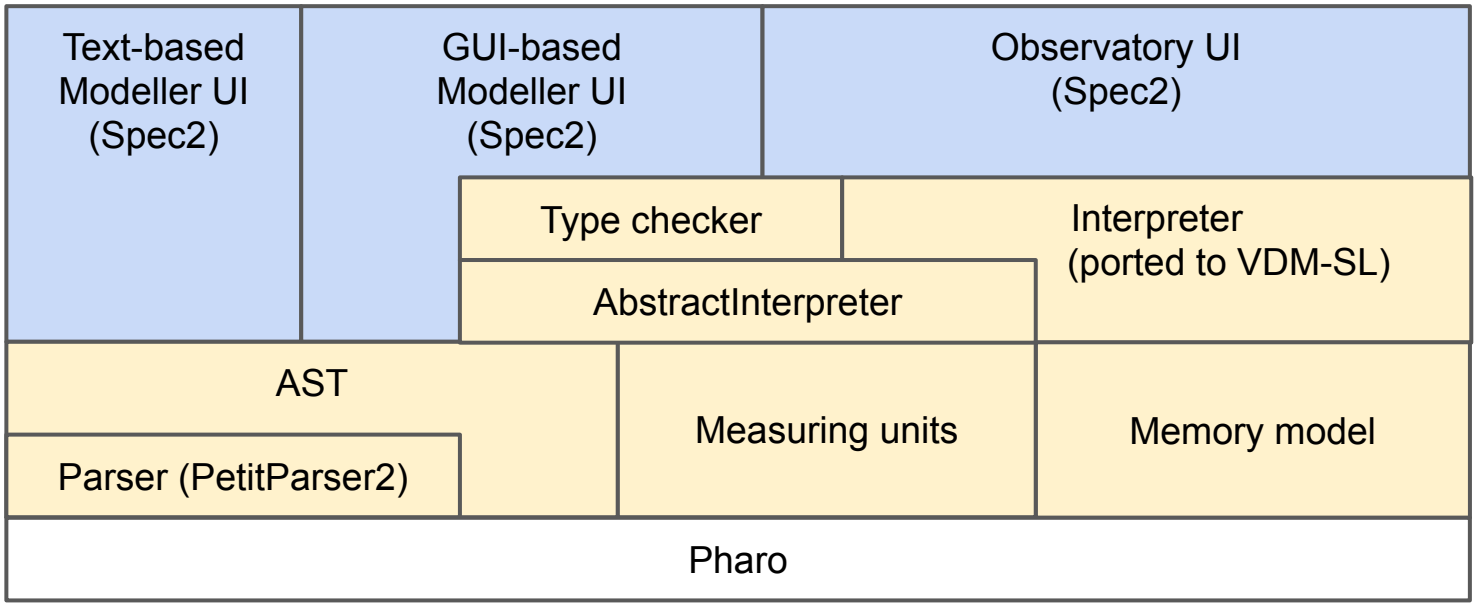
constructing DSL on Pharo

overview of re:mobidyc

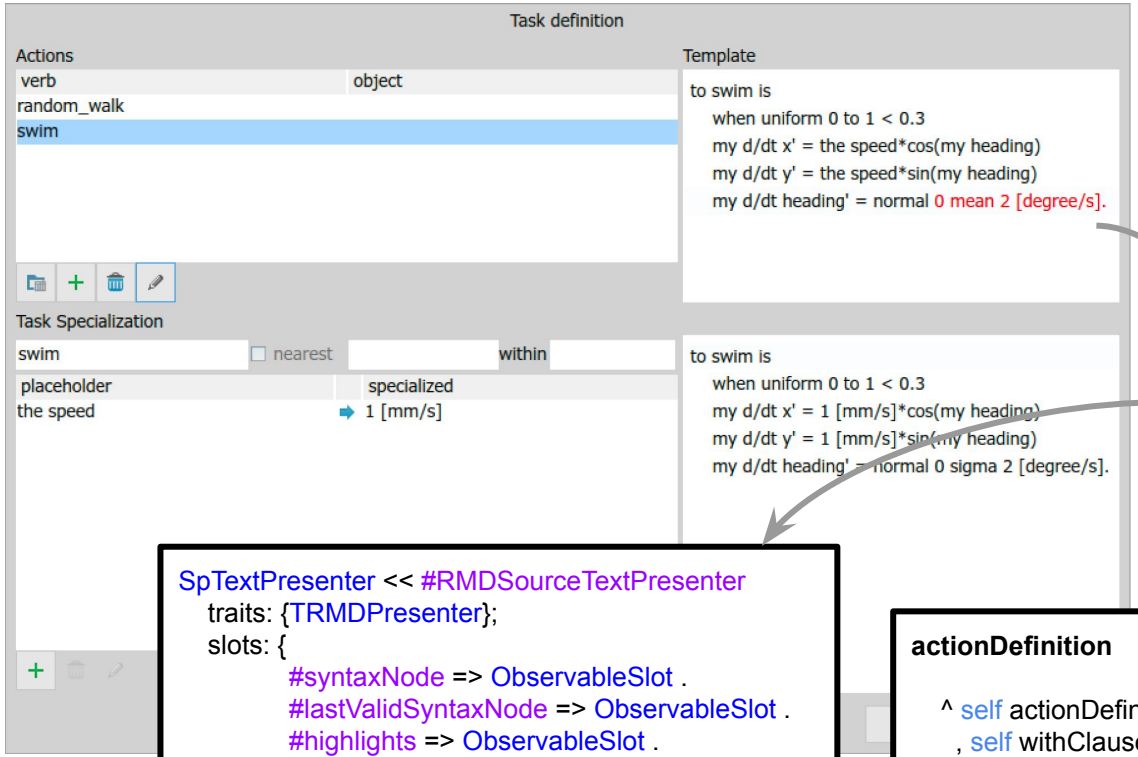
- Original modeling language
  - domain specific modeling language
  - syntax definition
  - formal semantics ... implement in Pharo first, and then specify in formal language (VDM-SL)
  - memory model

- Pharo
  - cross platform
  - high productivity
  - Spec2 and PetitParser2

Implementation Layers



Spec2 + PetitParser2 → easy construction of GUI for DSL




```
(self newSourceWith: RMDGrammar new actionDefinition)
    ...
    highlights: Array new;
    whenLastValidSyntaxNodeChangedDo: [ :syntaxNode |  ];
    yourself.
```

```
SpTextPresenter << #RMDSourceTextPresenter
    traits: {TRMDPresenter};
    slots: {
            #syntaxNode => ObservableSlot .
            #lastValidSyntaxNode => ObservableSlot .
            #highlights => ObservableSlot .
            #highlightColor => ObservableSlot .
            #parser };
    tag: 'Widgets';
    package: 'ReMobidyc-Spec2'
```

```
actionDefinition

    ^ self actionDefinitionHeaderLine , self whenClause optional
      , self withClause optional
      , self actionDefinitionAttributePart optional
      , self actionDefinitionUtilityPart optional
      , '.' asPParser trimBlanks , #newline asPParser trimBlanks optional
      ==> [ :array | ... ]
```

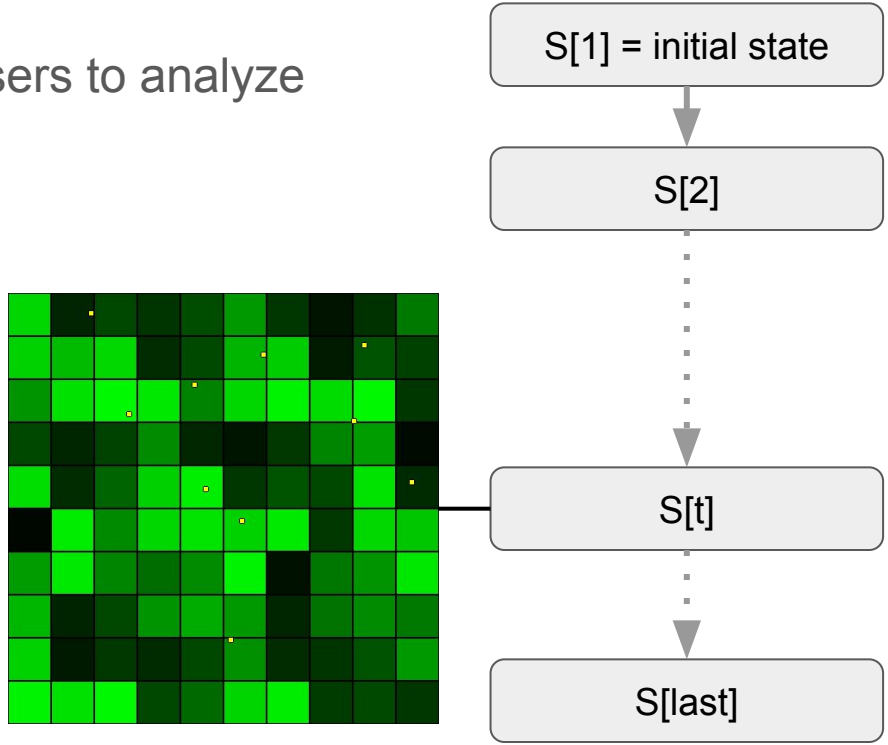
need for original memory model: the time-course matters

The objective of re:mobidyc is to enable users to analyze

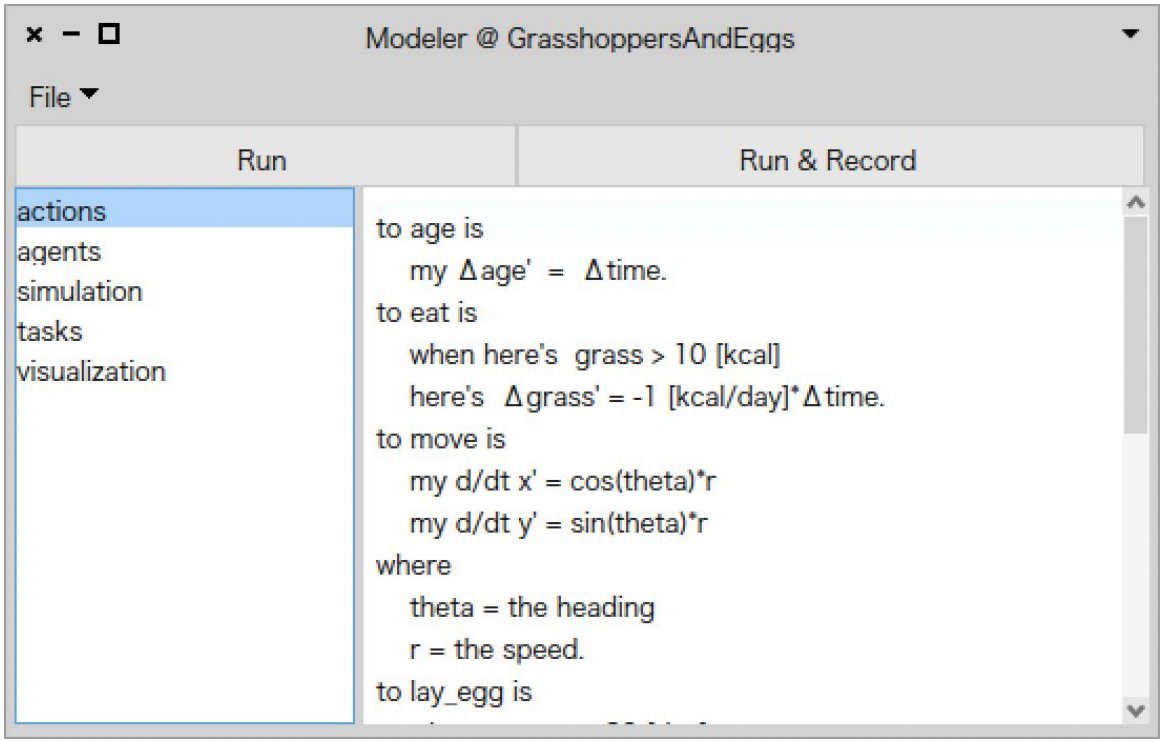
- what happens
- why it happens
- how it happens

We need a memory model with

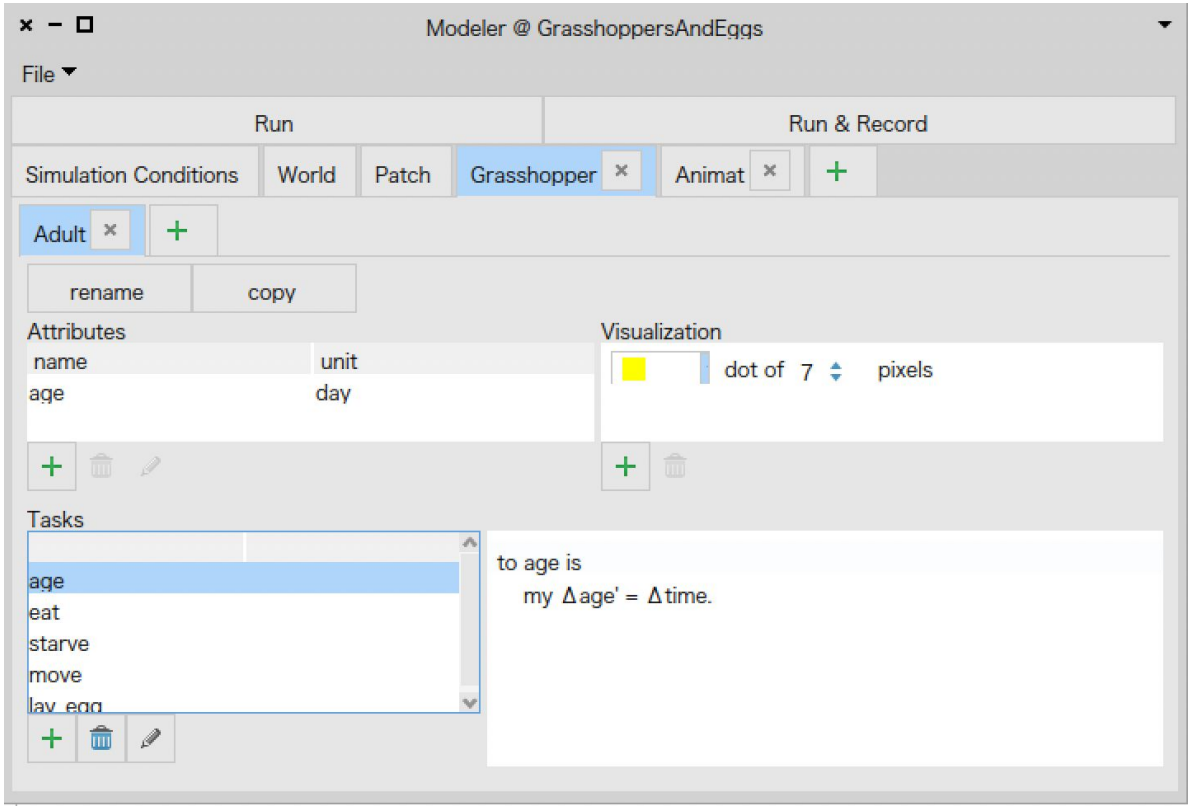
- lightweight snapshot
  - to trace cause and effect
- synchronous update
  - to isolate effects of each action
  - to eliminate intermediate state



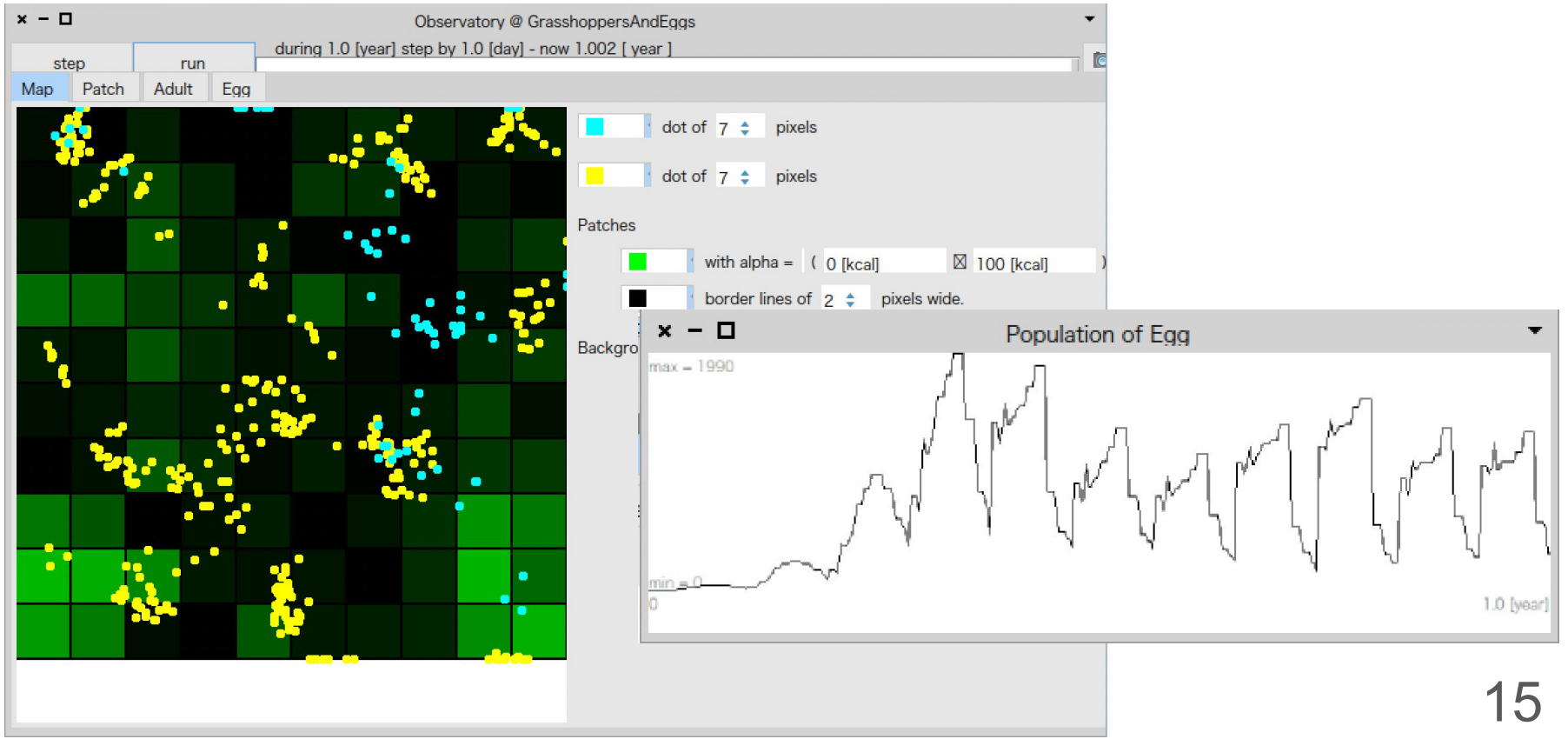
UI : Text-based Modeler



UI : GUI-based Modeler

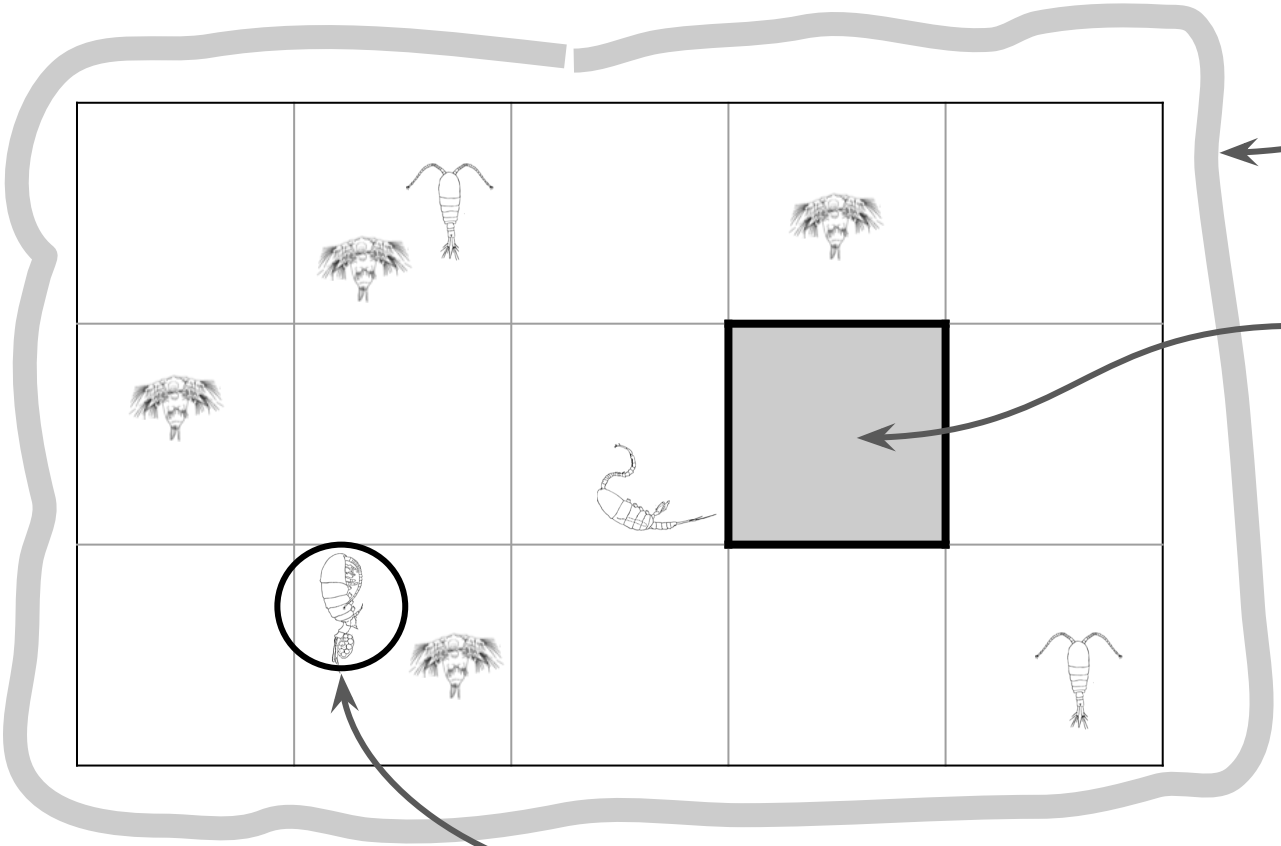


UI : Observatory



modeling with agents

three kinds of agents



World: global environment

Patch: local environment

Animat: individual body

Agents

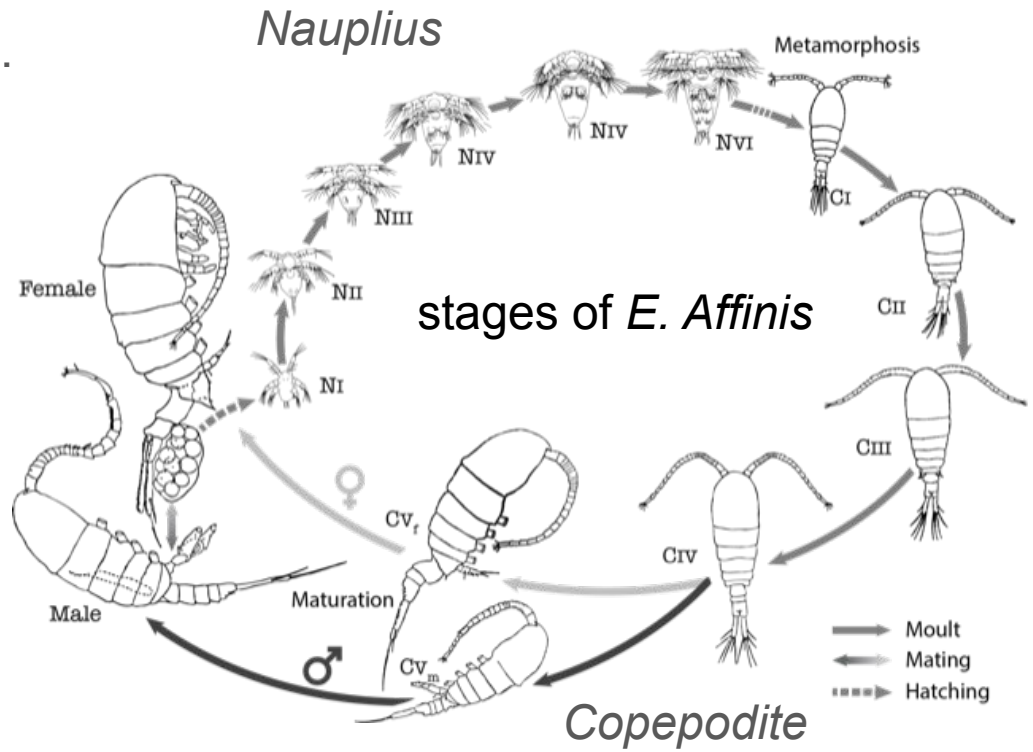
- An agent has a set of attributes.
- e.g.

stage species

```
Nauplius is EAffinis
with
    age [day].

Copepodite is EAffinis
with
    age [day]
    heading [degree].
```

attribute



Interactions

Actions (verb)

- A verb updates attributes of interacting agents.
- A verb is either *vi* or *vt*.
- e.g.

verb

updates on attributes

```
to random_walk is
    my d/dt x' = v * cos(theta)
    my d/dt y' = v * sin(theta)
where
    theta = uniform 0 [degree] to
360 [degree]
    v = normal 0 sigma the speed.
```

local definitions

Tasks (sentence)

- A task defines interaction among agents.
- A task is either S+V or S+V+O
- The interpreter performs a task for every instance of the subject agent at every time step.
- e.g.

```
Nauplius random_walk
where
    the speed -> 0.3[mm/s].
```

```
delta_x += v * cos(theta) * timestep;
```

Interactions

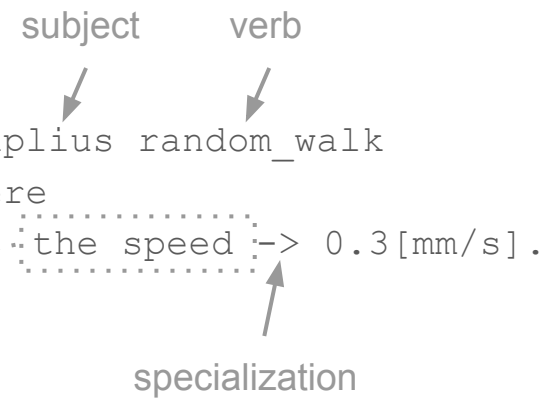
Actions (verb)

- A verb updates attributes of interacting agents.
- A verb is either *vi* or *vt*.
- e.g.

```
to random_walk is
    my d/dt x' = v * cos(theta)
    my d/dt y' = v * sin(theta)
where
    theta = uniform 0 [degree] to
360 [degree]
    v = normal 0 sigma the speed.
```

Tasks (sentence)

- A task defines interaction among agents.
- A task is either S+V or S+V+O
- The interpreter performs a task for every instance of the subject agent at every time step.
- e.g.

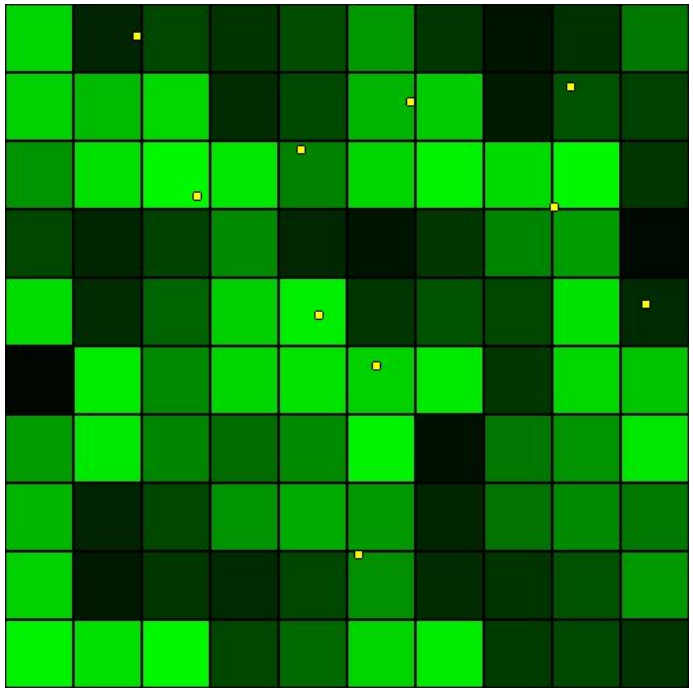


major features of the re:mobidyc modeling language

- no messaging
- no inheritance
- no `become:` primitive
- no while-loop
- no recursion
- non-turing complete
- no string, no bool, float is the only first-class object

but has

- static typing
- type casting
- templates
- time-series memory with synchronous update
- termination because no loop!
- instance creation trace



to model interactions among the global environment, local environment patches, and animats.

Types

in a language where the only FCO is float

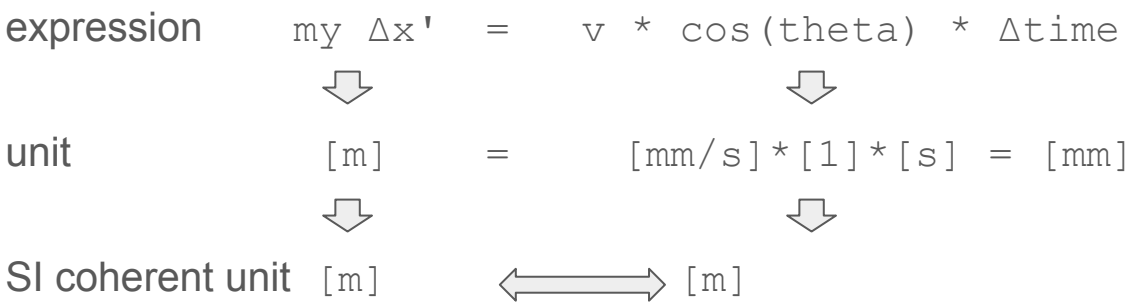
static typing with measuring units

Animat definitions

```
Nauplius is EAffinis with
    x [m]
    y [m]
    heading [degree].
```

The system converts values into SI coherent units so that all computation will be in coherent SI. Also `[rad]`,`[℃]`, `[℉]` are handled as SI base units.

Type checking on attribute definitions



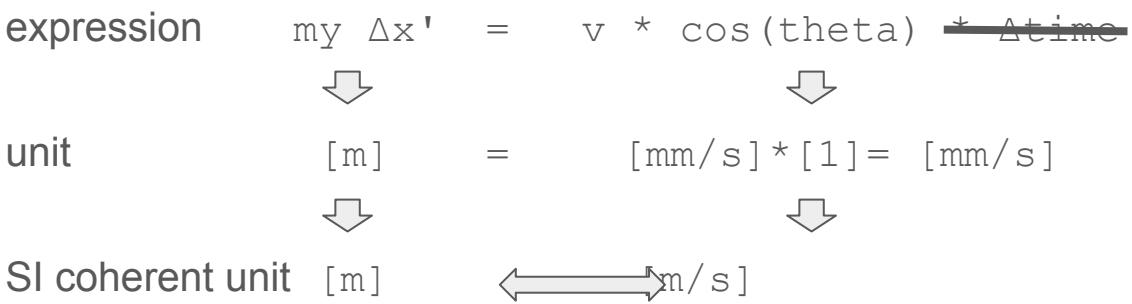
the both types agree

detecting type error

Animat definitions

```
Nauplius is EAffinis with
    x [m]
    y [m]
    heading [degree].
```

Type checking on attribute definitions

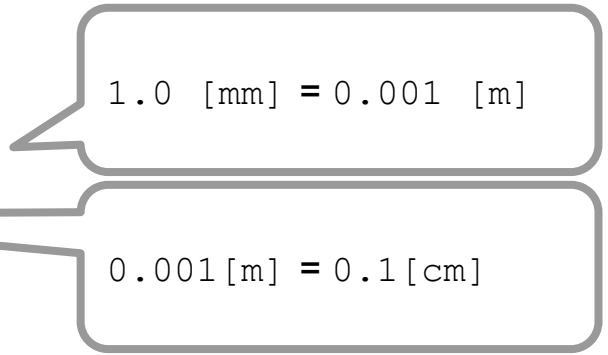


type casting with measuring units

- **de-unit casting**

```
(cm) x
    where x = 1.0 [mm]
```

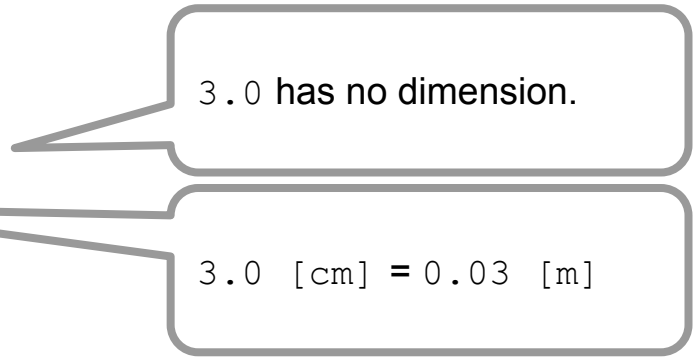
= 0.1



- **en-unit casting**

```
x [cm]
    where x = 3.0
```

= 0.03 [m]



time-series memory with synchronous updates

re:mobidyc = computing big sequences of numbers

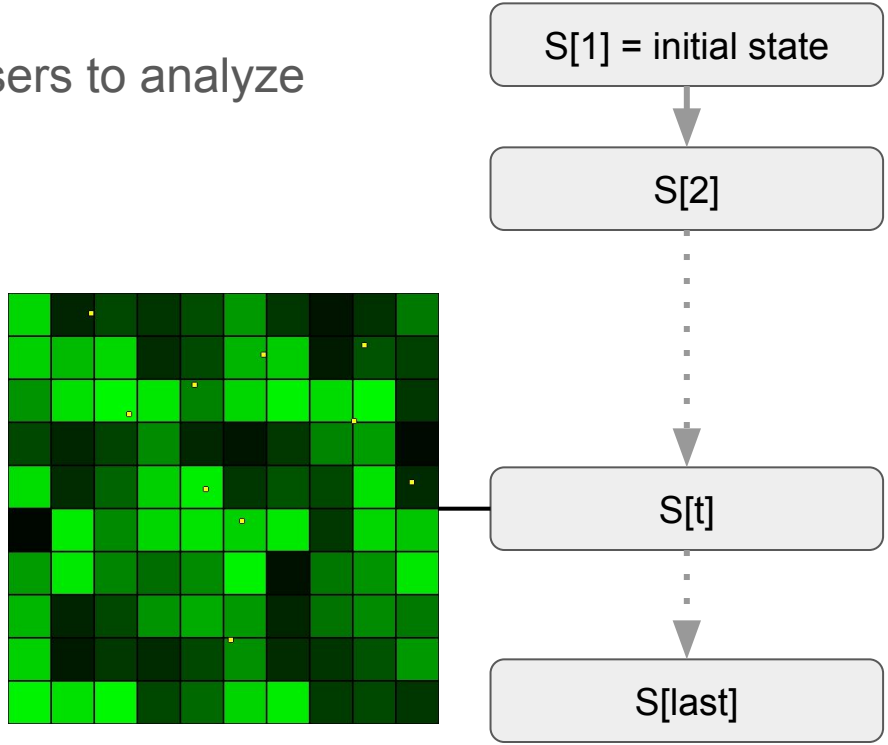
The time-course matters

The objective of re:mobidyc is to enable users to analyze

- what happens
- why it happens
- how it happens

We need a memory model with

- lightweight snapshot
  - to trace cause and effect
- synchronous update
  - to isolate effects of each action
  - to eliminate intermediate state



three cells per address

```
a = my x + cos(my heading)
memory at: 1  memory at: 3
```

agent		1			4			
attribute		x	y	heading	x	y	heading	
address		1	2	3	4	5	6	7
value	read	12.5	-5.2	1.23	-4.5	34.1	3.0	
nextValue	write	12.5	-5.2	-0.2	-4.5	34.1	-0.2	
nextDelta	write	-2.4	0.0	0.0	-2.4	0.0	0.0	

```
my x' = -2.4[m]
memory nextAt: 1 put: -2.4
```

```
my Δheading' = -0.2[rad]
memory nextDeltaAt:3 add:
```

synchronous update

time 354

address		1	2	3	4	5	6	7
value	read	12.5	-5.2	-0.2	-4.5	34.1	3.0	
nextValue	write	12.5	-5.2	-0.2	-4.5	34.1	-0.2	
nextDelta	write	-2.4	0.0	0.0	-2.4	0.0	0.0	

time 355

address		1	2	3	4	5	6	7
value	read	10.1	-5.2	-0.2	-6.9	34.1	-0.2	
nextValue	write	10.1	-5.2	-0.2	-6.9	34.1	-0.2	
nextDelta	write	0.0	0.0	0.0	0.0	0.0	0.0	

time-series memory

time 355

address		1	2	3	4	5	6	7
value	read	10.1	-5.2	-0.2	-6.9	34.1	-0.2	
nextValue	write	10.1	-5.2	-0.2	-6.9	34.1	-0.2	
nextDelta	write	0.0	0.0	0.0	0.0	0.0	0.0	

3 dictionaries:
values, nextValues, nextDelta

backend storage (on-memory, file system, ...)

address \ time	1	2	3	4	5	6	...
...	...	...	...	...	...	...	...
350	22.1	-5.2	-4.2	5.1	34.1	-1.5	...
351	19.7	-5.2	-4.2	2.7	34.1	2.3	...
352	17.3	-5.2	3.2	0.3	34.1	5.1	...
353	14.9	-5.2	-5.2	-2.1	34.1	-5.2	...
354	12.5	-5.2	1.23	-4.5	34.1	3.0	...
355	10.1	-5.2	-0.2	-6.9	34.1	-0.2	...
356							
...							

Summary

moving forward

Summary

- multi-agent simulator
  - re:realization of Object Orientation as a modeling language
    - de-centralized autonomous entities interacting each other.
  - open source at https://github.com/ReMobidyc/ReMobidyc/
  - built on Pharo 10
  - for scientific research of biology, ecology and ecotoxicology
- Future work
  - complete 25 primitives from MoBIDyC
  - formal specification
  - memory models with DBMS backends (RDB, GemStone/S, and so on)
  - debugger
  - git interface
  - computation server
  - data analysis environment
  - visualization
  - verification

Thank you!