LESSON 3 · 15 OCTOBER 2026

Applied Systems Sciences 1

Models, agent-based models and LLMs

How could we find out what saves the lake?

David Maier
Department of Environmental Systems Sciences
University of Graz

Yesterday half of our lakes collapsed

Why do fish stocks collapse, even when no fisher wants them to?

One or two sentences. Use the system, the feedback and the emergence.

In class, you answer here and the answers are grouped into themes.

Today we do half of the modelling cycle

1 Formulate the question

Would talking save the lake?

→

2 Assemble hypotheses

Fishers adapt to the lake and to what they hear.

→

3 Choose the model structure

Fishers, a lake and rules for catch and regrowth.

↓

4 Implement the model

A rule for every fisher, in code or in words.

←

5 Analyse the model

Many lakes, not one: how often does it survive?

←

6 Communicate the model

Describe it so that others can judge it.

↑

The first version is always wrong, and that's fine.

Today: steps 1 to 3 for the lake, and a look at 4 and 5 with LLM fishers. From session 5 you do them with your own model.

Railsback & Grimm (2019)

How could we find out what saves the lake?

Question to structure 1 2 3 4 5 6

Think it through

Quick and free.

But we can't follow several loops in our heads.

Causal loop diagram

Finds the loops, like yesterday's diagram.

But it doesn't show what happens over time.

System dynamics

Equations run the loops over time and test rules.

But every fisher is the average fisher.

Experiment, like our game

Real people adapt and talk.

But few rounds, few lakes, only the rules we tried.

Statistics on data

Finds patterns in what happened.

But it can't test a rule nobody has tried.

Network analysis

Shows who talks to whom.

But it shows structure, not how the lake changes.

Which one is right depends on the question.

Same lake, two questions

Question to structure 1 2 3 4 5 6

In pairs, 4 minutes.

A How many fish can the lake give for ever?

The largest catch per round that the lake can sustain.

B Would talking save the lake?

Whether fishers who can agree on rules catch less and earn more.

> For each question: name one thing you would **include** and one you would **leave out**.

An equation is enough

Only the stock and the total catch. Every fisher is the average fisher.

```
left = stock - catch
stock = min(2 * left, capacity)
```

The fishers have to be in the model

Own needs, who talks to whom, what each sees, how each adapts.

```
for fisher in fishers:
    fisher.fish(lake)
```

Why an agent-based model for our lake?

Question to structure 1 2 3 4 5 6

What would saving the lake need? Which tools are go-to methods for it?

	Over time	Fishers differ	Fishers adapt	Fisher ↔ lake	Test a rule	Uncertainty
System dynamics	●				●	
Experiment, like our game	●		●			
Statistics on data		●				●
Network analysis						
Agent-based model	●	●	●	●	●	●

It can change the answer

For rabies in foxes, equations said to vaccinate 70%. An agent-based model found much less is enough, and field studies agree.

It has a price

Every detail is a choice. Too many details make a model hard to analyse and reproduce.

It builds on the others

Fieldwork, theory and equations feed into it.

Biggs et al. (2022) · Railsback & Grimm (2019)

What makes a model agent-based?

Agents are **unique** and **autonomous**, and usually **interact locally** with each other and their environment.

Unique

Fishers differ: debts, boats, trust.

Yesterday: every boat had its own place.

Autonomous

Each decides their own catch, for their own goals.

Yesterday: you chose your own catch.

Local interaction

They deal with neighbours, not with everyone.

Yesterday: a boat saw two spots around it, a starling sees its few neighbours.

Adaptive

They change their catch with the lake and the others.

Yesterday: a boat moved to the most fish.

Few agent-based models have all four: yesterday's starlings needed only three local rules.

Railsback & Grimm (2019)

Quick check

Question to structure 1 2 3 4 5 6

Which of these questions about a fishery needs an agent-based model?

1. How many fish the lake can sustain
2. Whether fishers who trust each other form a cooperative
3. How big last year's total catch was
4. How fast a single fish grows

What if the fishers were LLMs?

A fisher is a rule for choosing a catch. So far it was code, or it was you.

A fisher made of code

```
catch = 3 if lake.stock > 12 else 1
```

We write the rule, and it does the same every time.

A fisher made of language

> *"You are Fisher A. The lake holds at most 24 fish. Every round you choose how many to catch, 0 to 5. [...] Your goal: catch as many fish as you can. How many do you catch?"*

We write the rule as text. The model reads it every round to choose the catch, and may choose differently each time.

We play yesterday's lake with four of them.

Piatti et al. (2024), GovSim · Park et al. (2023), generative agents

Four LLM fishers, no talking

Four LLM fishers share the lake and cannot talk. Does it last 8 rounds?

1. Yes, the lake survives
2. No, it is empty before the end

Four LLM fishers, the same rules

In class, the lake is played live by four fishers that are a language model. You see what they say to each other and why they choose what they choose.

One lake is a story, many lakes are a result

2 of 20 lakes survived

No talking

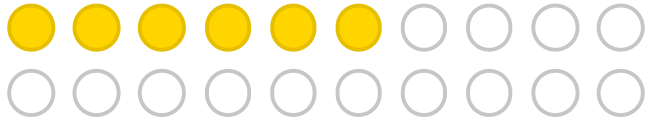
The fishers stay silent.



6 of 20 lakes survived

Talking from round 4

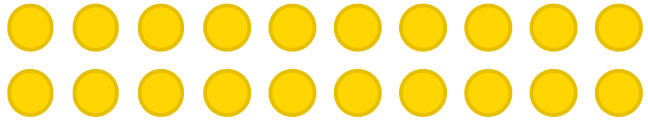
The setup of our class.



20 of 20 lakes survived

Talking from round 1

From the very first round.



Our class, with talking from round 4: about half of the lakes collapsed.

Our own runs before class: one model, one prompt, 20 lakes per setup. Not a research result.

Would you trust these fishers?

An LLM can play a fisher, as here, or read what people say, as it sorted your answers today. Either way it is a model like any other, so the same four questions apply.

Purpose

What question do these fishers answer, and is a language model the right tool for it?

Assumptions

The prompt gave rules and a goal. Trust, fairness and what a neighbour will do come from the texts the model learned from.

Validation

Ours mostly kept to their plan in the last round. Do real fishers? What would we compare with?

Reproducibility

The same prompts gave different lakes. Would a second model, or someone else, get our numbers?

The same four questions apply to our lake equation and to every model in this course.

Build the lake yourself

Python with uv

Runs the lake model. uv installs Python and packages.

VS Code

Where you write the model, with git and Copilot.

git

Saves every version of your model.

GitHub

Shares the model with your team. You hand in here.

Copilot

AI help in VS Code, free with GitHub Education.

Now: setup lab. Step 4, implementing, starts in session 5, so every laptop has to be ready.

Python and uv

We use **Python 3.12+**, installed and managed by **uv**.

Install uv

Windows (PowerShell):

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

macOS and Linux:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

The two commands you need

```
uv add package-name    # install a package
uv run script.py       # run a script
```

Check

`uv --version` in a terminal.

VS Code and git

Visual Studio Code

Download: code.visualstudio.com

Windows: the User Installer (64-bit). macOS: the Universal build.

Then install the **Python extension** by Microsoft. VS Code suggests it when you open a .py file.

git

Windows: git-scm.com, the default settings are fine.

macOS: already installed, or installed on first use.

Check: `git --version` in a terminal.

GitHub and Copilot

1

GitHub account

The one you log in with on this website. No account yet? Sign up at **github.com**.

2

GitHub Education

Add your university e-mail to your account, then apply for the Student Developer Pack. Approval can take a few days.

3

Copilot in VS Code

Sign in to VS Code with your GitHub account. Free for verified students, with monthly limits.

Setup lab

Work on your own laptop. Ask when something doesn't work.

uv

Installed. This prints a version:

```
uv --version
```

VS Code

Open a folder, add hello.py with a print line, run it in the terminal:

```
uv run hello.py
```

git

Installed. This prints a version:

```
git --version
```

GitHub

You can log in, and the Education application is sent.

Done? Help a neighbour, or change your Python file and run it again.

How far is your setup?

How far does your setup work?

1. Nothing yet
2. uv works
3. uv and VS Code work
4. uv, VS Code and git work
5. Everything works, and the Education application is sent

Before lesson 4: variables and if

A variable is a label for a value

Variables

```
x = 5
y = x
x = x + 1
print(x, y)    # 6 5
```

`y = x` gives the same value a second label. `x = x + 1` moves the label `x` to a new value. `y` keeps the old one.

if and else

```
speed = 3
if speed > 5:
    print("fast")
else:
    print("slow")    # slow
```

The block under `if` runs when the condition is true, otherwise the `else` block.

Next week: Thursday 22 October, 08:30 to 09:45, SR 35.K1. Short Python challenges on these two slides open the lesson, and a right answer earns participation points. Prepare at home.

Before lesson 4: lists and loops

A list holds several values in order, a loop visits them one by one

A list

```
catches = [3, 1, 2]
print(catches[0])      # 3, counting starts at 0
print(len(catches))    # 3, how many items
catches.append(4)
print(catches)         # [3, 1, 2, 4]
```

Square brackets, items in order. `append` adds one at the end.

A loop

```
catches = [3, 1, 2]
for catch in catches:
    print(catch)    # 3, 1, 2
```

In each pass, `catch` is a variable for the next item. The loop body works with that variable.

Log in with GitHub to take part. More examples: the Python cheatsheet on the course website.

Try Python yourself

The code of the two slides before. Change it, press Run. Nothing is saved.

```python
x = 5
y = x
x = x + 1
print(x, y)

speed = 3
if speed > 5:
    print("fast")
else:
    print("slow")

```

▶ Run Python ready

```python
catches = [3, 1, 2]
print(catches[0])
print(len(catches))
catches.append(4)
print(catches)

for catch in catches:
    print(catch)

```

▶ Run Python ready

Sources for this lesson

Railsback, S. F. & Grimm, V. (2019). *Agent-Based and Individual-Based Modeling: A Practical Introduction.* 2nd ed.

Biggs, R. et al. (eds., 2022). *The Routledge Handbook of Research Methods for Social-Ecological Systems.*

Piatti, G. et al. (2024). Cooperate or collapse: emergence of sustainable cooperation in a society of LLM agents.

Park, J. S. et al. (2023). Generative agents: interactive simulacra of human behavior.

Also: Jackson (2024) · Hardin (1968) · Ostrom (1990) · Forrester (1994)