STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Graphical Modeling using SysML/UML Diagrams

SYS-611: Simulation and Modeling

Paul T. Grogan, Ph.D.
Assistant Professor
School of Systems and Enterprises

Agenda

1. Introduction to SysML/UML Diagrams
2. Dice Fighters Exercise

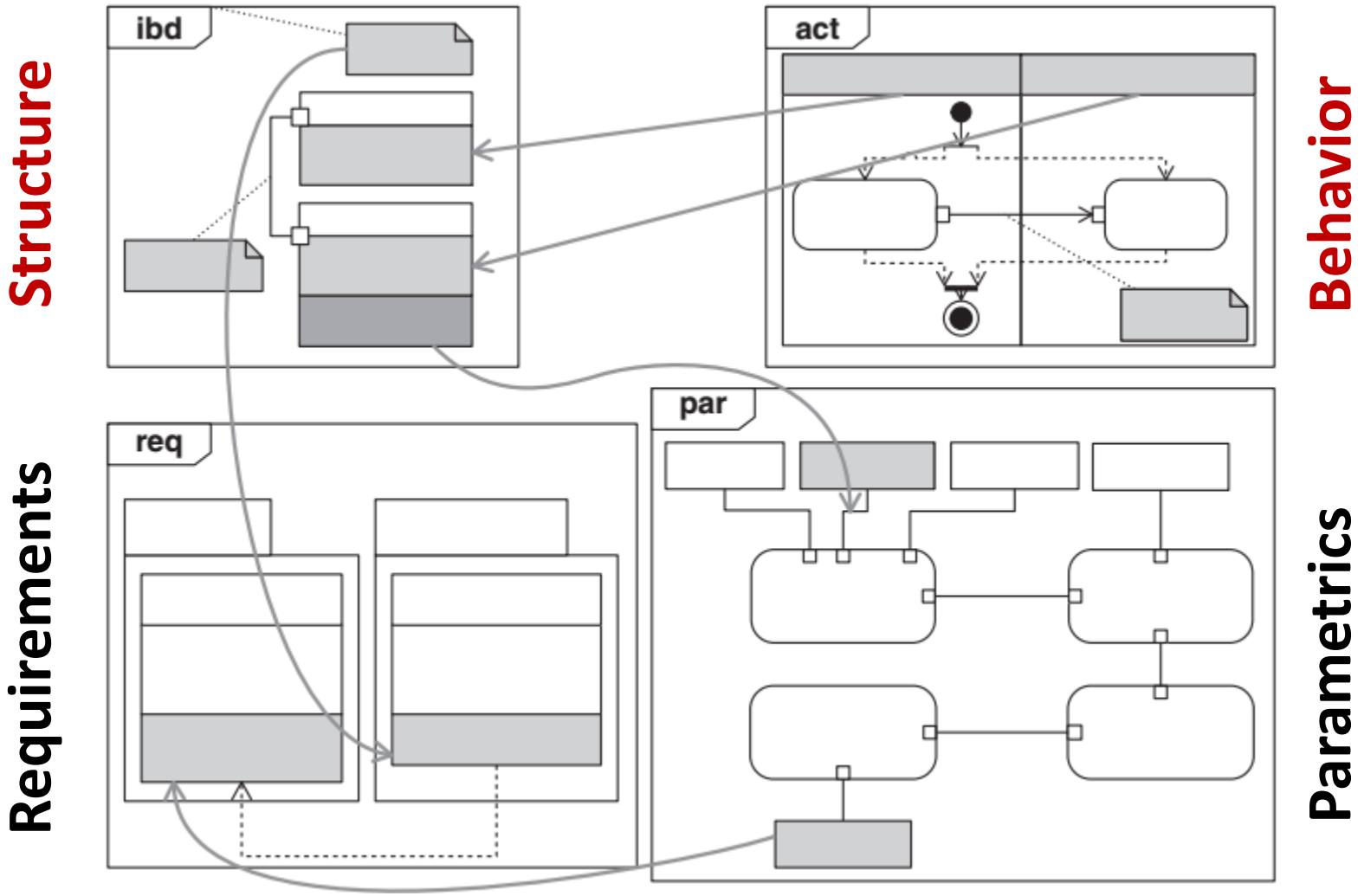
Optional: S. Friedenthal, A. Moore, and R. Steiner, “Getting Started with SysML,” Ch. 3 in *A Practical Guide to SysML*, 3rd Edition, 2014.

Introduction to SysML/UML

Model-based Systems Engineering (MBSE)

- MBSE transitions from document- to model-based information storage
  - Information is human- and computer-readable
  - Maintain models for throughout analysis, specification, design, verification, and validation processes
- Models typically expressed in Systems Modeling Language (SysML) graphical language
  - Extension of Unified Modeling Language (UML)
  - See: Friedenthal, Moore, and Steiner, *A Practical Guide to SysML: The Systems Modeling Language*, 2014.

The Four Pillars of SysML



Friedenthal, Moore, and Steiner (2012)

Structure: Model State

The **model state** is the set of information required to recreate a snapshot in time

- Elementary state: minimal set of information
- Derived state: additional information computed from elementary state (store only for convenience)
- Experimental frame is critical to identify state!

Image courtesy Steve Jurvetson (Wikimedia)

Behavior: State Changes

State changes represent logical transitions between model states

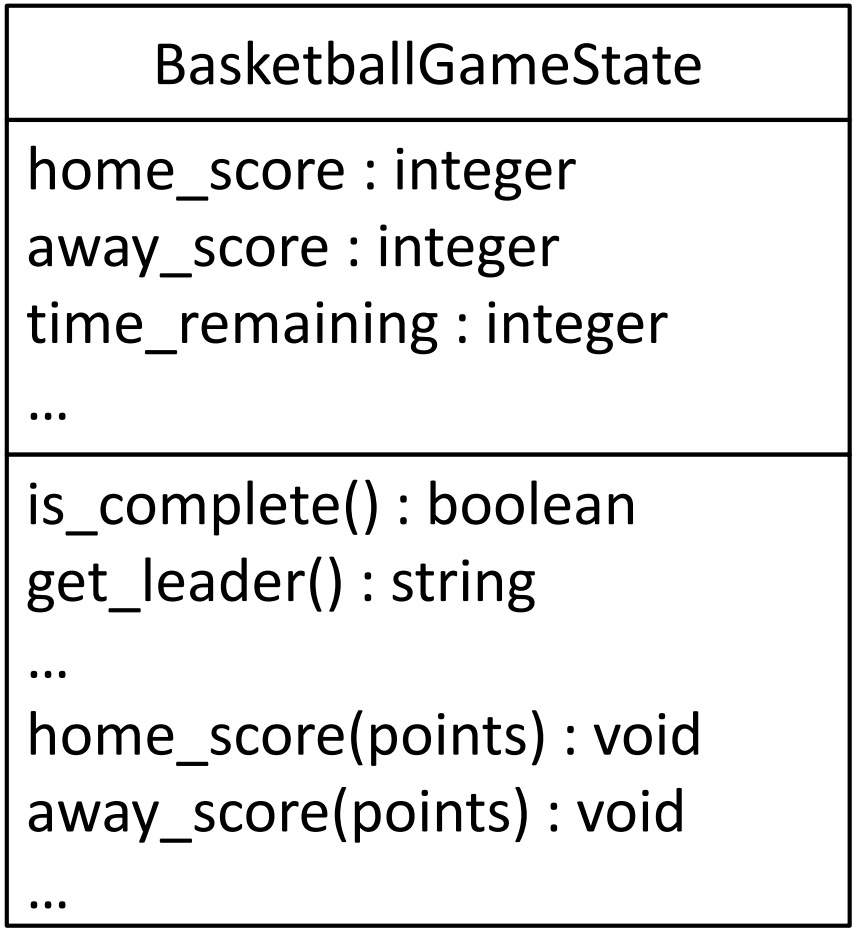
- Mathematically defined using “transition functions”
- Transform from: input state (with input arguments)
- Transform to: output state

Image courtesy Steve Jurvetson (Wikimedia)

Model State Diagrams

State diagrams illustrate the content and structure of model state information

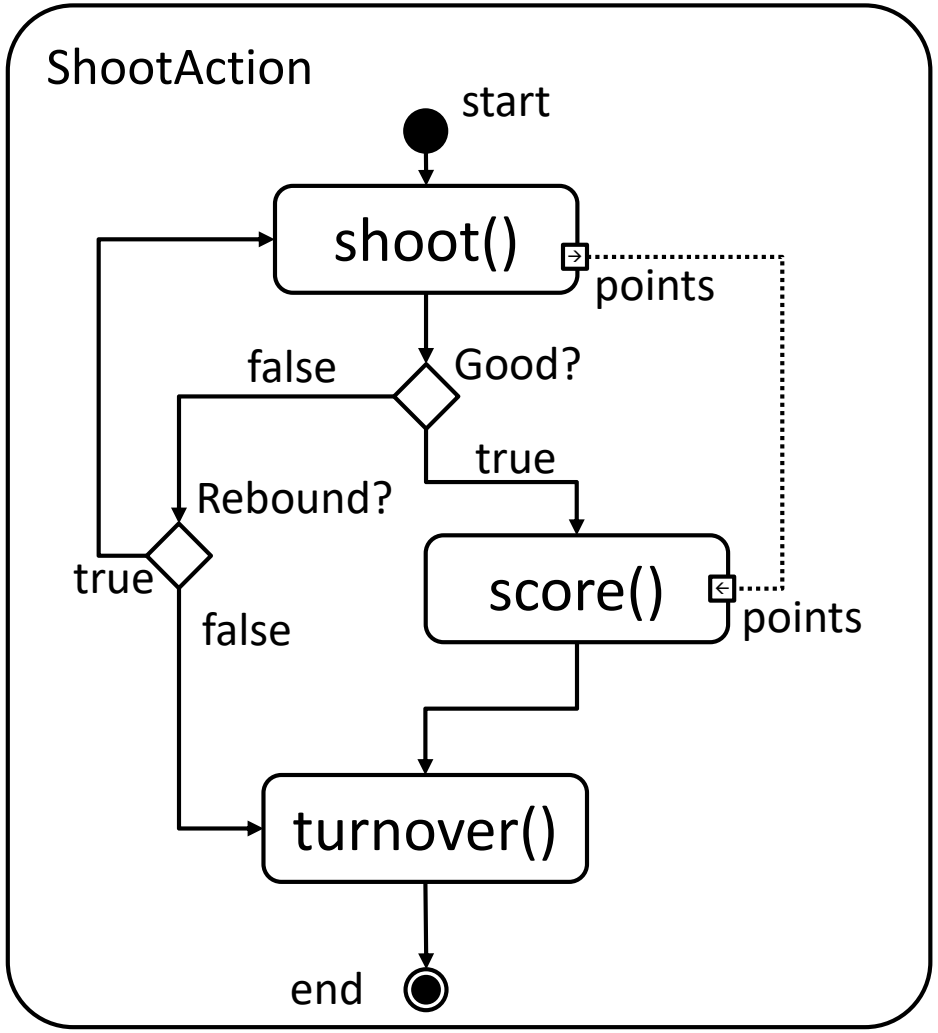
- SysML block definition diagram (BDD)
- UML class diagram
- Diagram elements:
  - Elementary state
  - Derived state (function)
  - State transition function



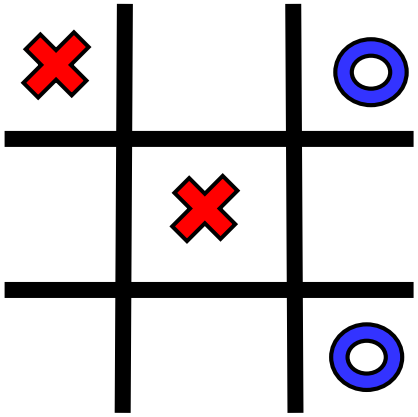
State Change Diagrams

State change diagrams illustrate how to execute state transitions over time

- SysML/UML activity diagram (processes)
  - "Flow chart" perspective
  - Best at showing logical flow
- SysML/UML sequence diagram (interactions)
  - "Swim lane" perspective
  - Best at showing interactions



Example: Tic-Tac-Toe



Assume X always goes first.

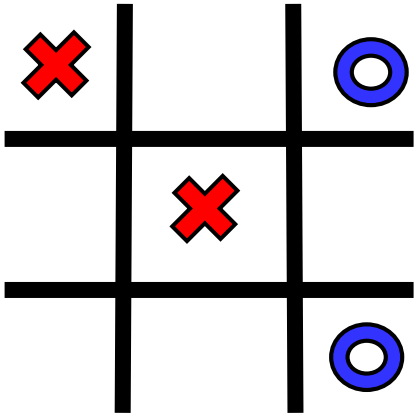
Model state

- What information is required to recreate a snapshot in time?
- What other derived state may be useful in Tic-Tac-Toe?

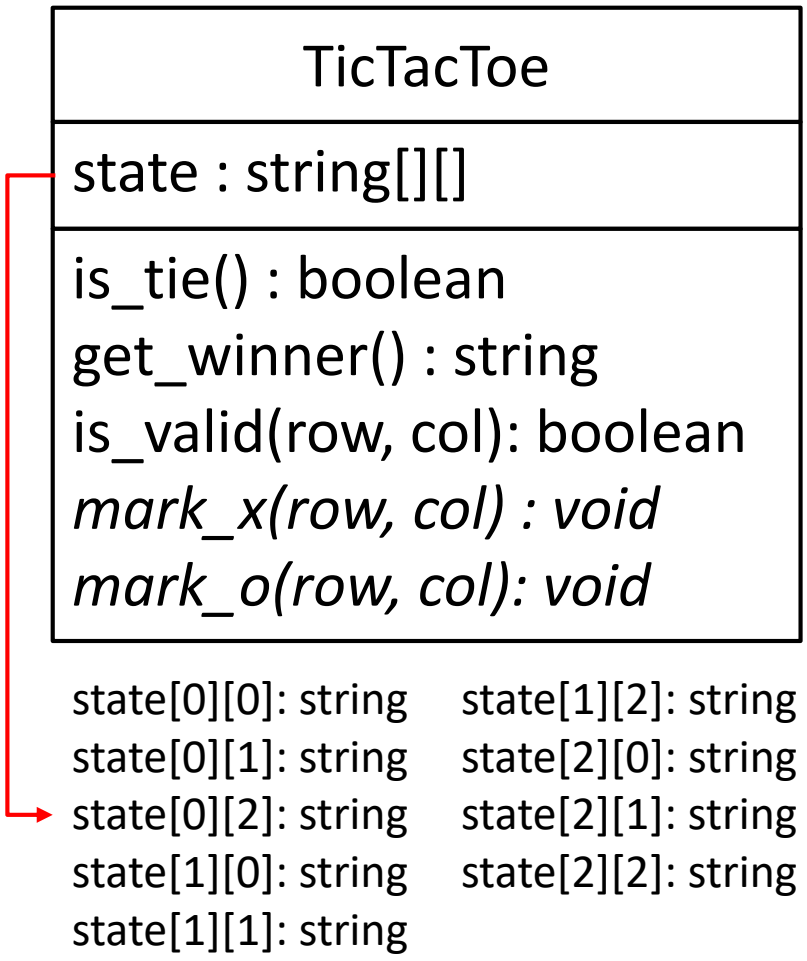
State changes

- What actions change the model state?

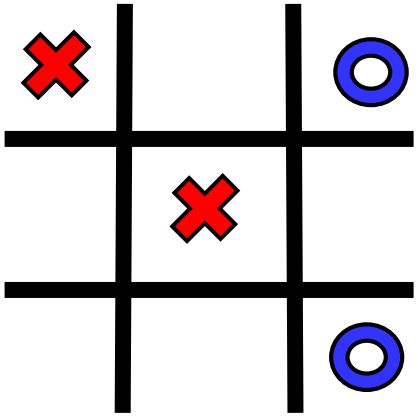
Tic-Tac-Toe State Diagram



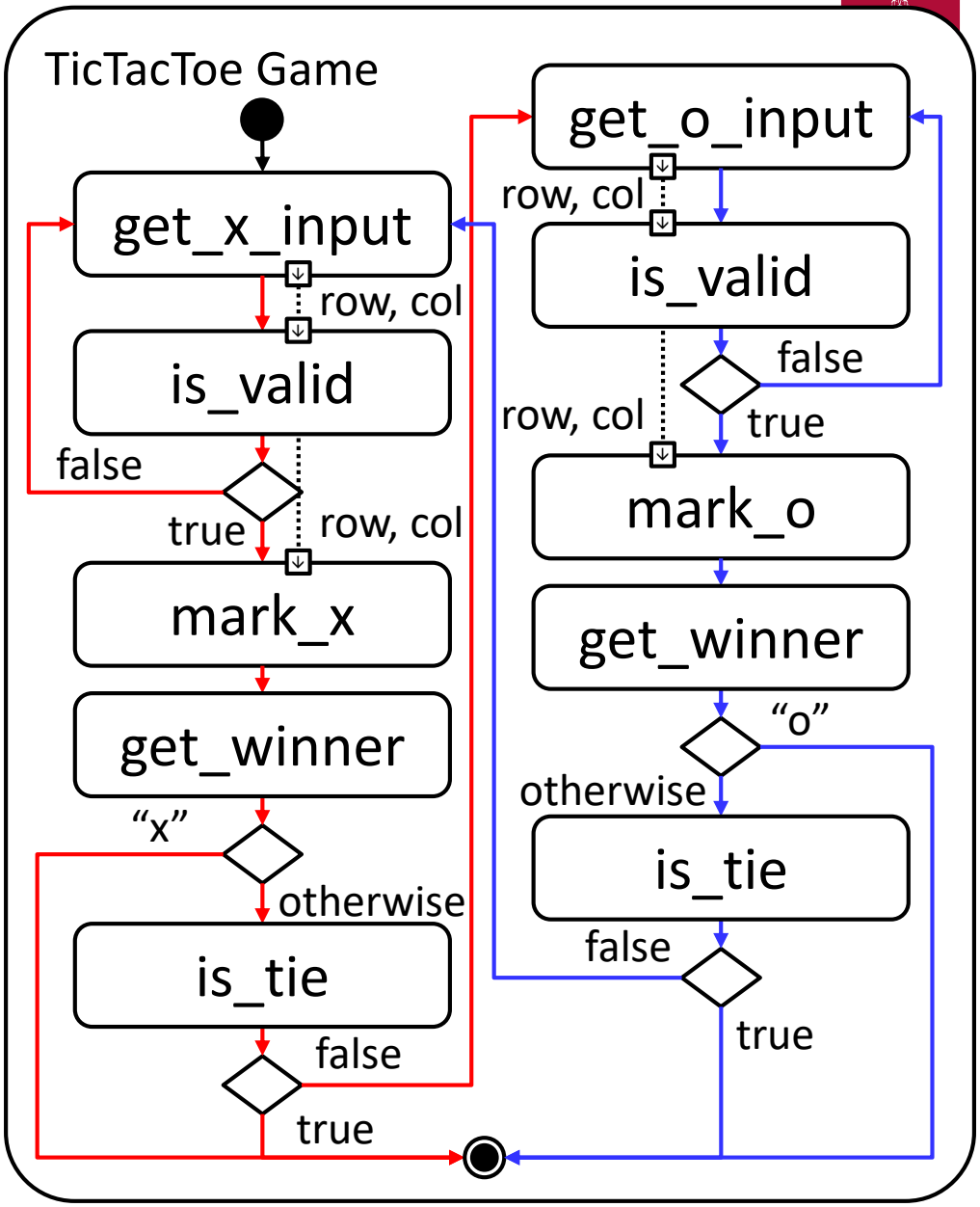
Assume X always goes first.



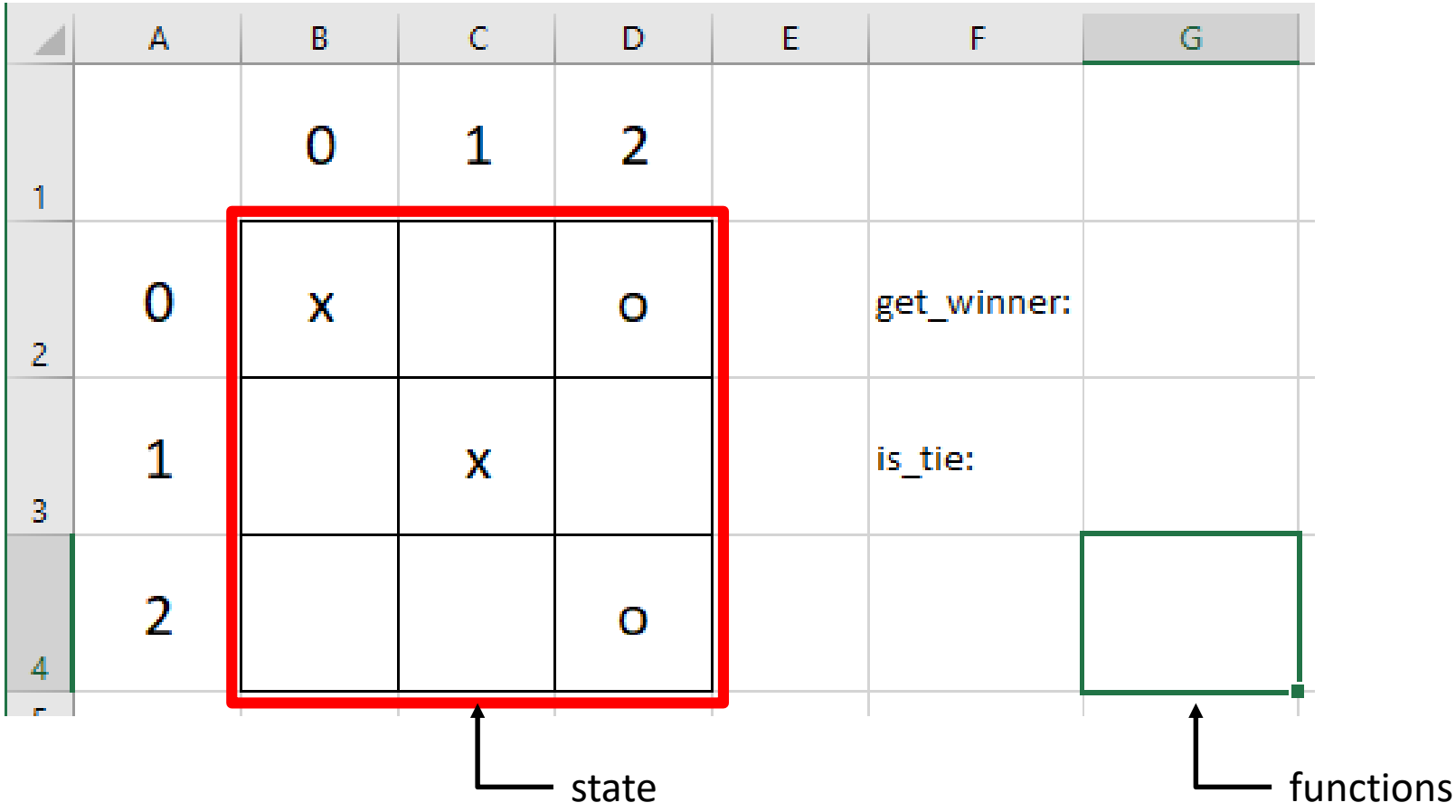
Tic-Tac-Toe Activity Diagram



Assume X always goes first.



Tic-Tac-Toe Excel Sheet



Tic-Tac-Toe Python Script

```
import pandas as pd

state = [
    [" "," "," "],
    [" "," "," "],
    [" "," "," "]
]

def is_valid(row, col):
    return state[row][col] == " "

def mark_x(row, col):
    if is_valid(row, col):
        state[row][col] = "x"

def mark_o(row, col):
    if is_valid(row, col):
        state[row][col] = "o"
```

```
def show_grid():
    print pd.DataFrame(state)

mark_x(state, 1, 1)
show_grid(state)
mark_o(state, 0, 2)
show_grid(state)
mark_x(state, 0, 0)
show_grid(state)
mark_o(state, 2, 2)
show_grid(state)
```

Dice Fighters Activity

Intro. to Combat Modeling

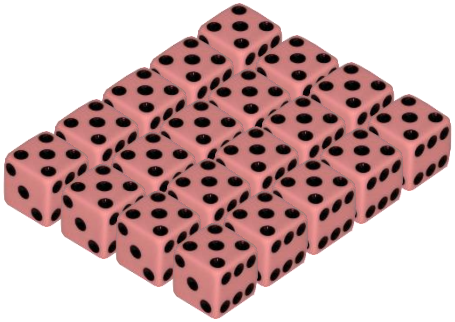
- Combat modeling simulates competing forces
- Long history of applications dating to 1800s

1:8000 Kriegsspiel (von Reisswitz, 1824) Photo by Roman März in P. von Hilgers (2000).

Dice Fighters Exercise

Red Team:

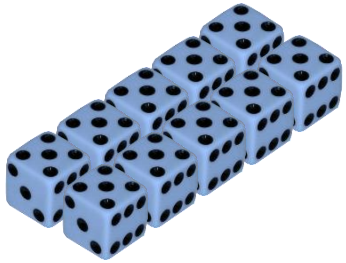
- 2x fighting force size
- Simple weapons



- Roll 6 to hit target

Blue Team:

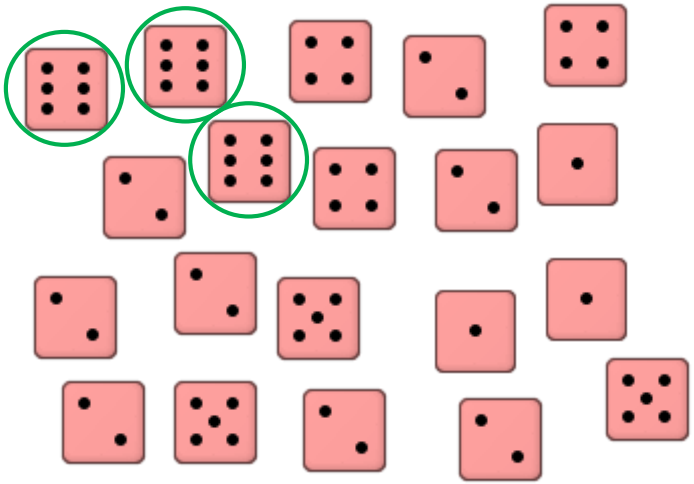
- Small fighting force
- 3x effective weapons



- Roll 4|5|6 to hit target

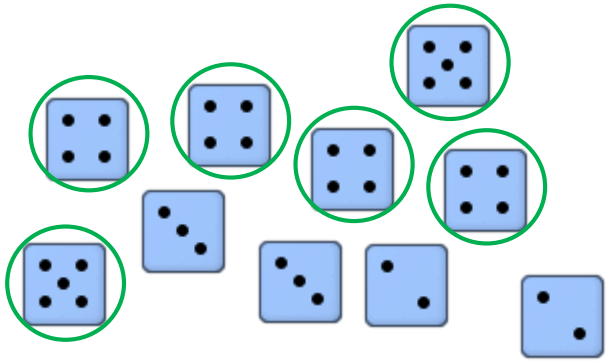
Dice Fighters: Example

- Roll 20 red dice



- 3 hits (≥6) – take away 3 blue dice for next round

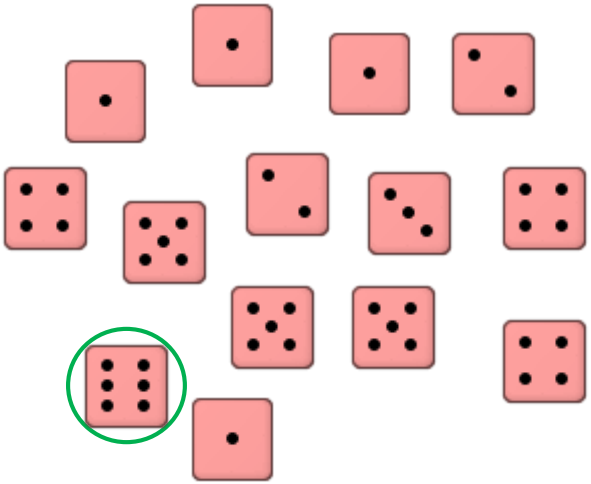
- Roll 10 blue dice



- 6 hits (≥4) – take away 6 red dice for next round

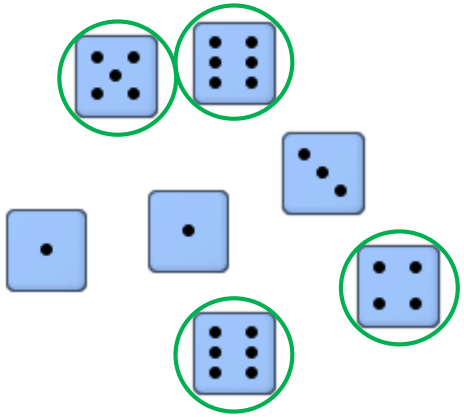
Dice Fighters: Example

- Roll $20 - 6 = 14$ red dice



- 1 hits (≥6) – take away 1 blue dice for next round

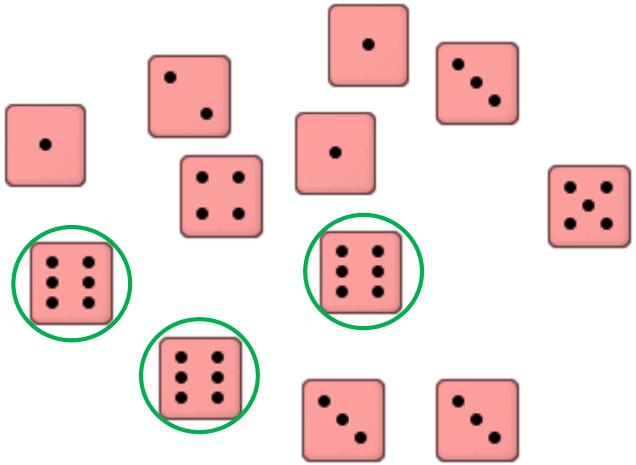
- Roll $10 - 3 = 7$ blue dice



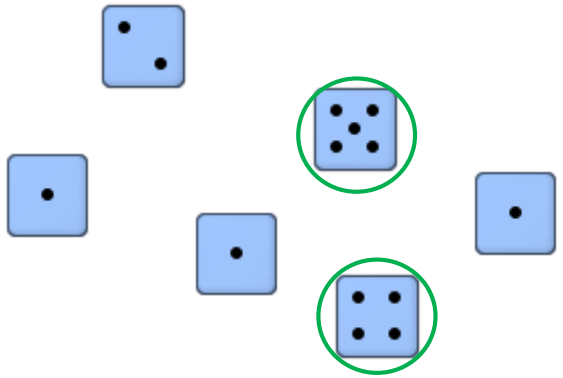
- 4 hits (≥4) – take away 4 red dice for next round

Dice Fighters: Example

- Roll $14 - 4 = 12$ red dice



- Roll $7 - 1 = 6$ blue dice



- 3 hits (≥6) – take away 3 blue dice for next round
- 2 hits (≥4) – take away 2 red dice for next round

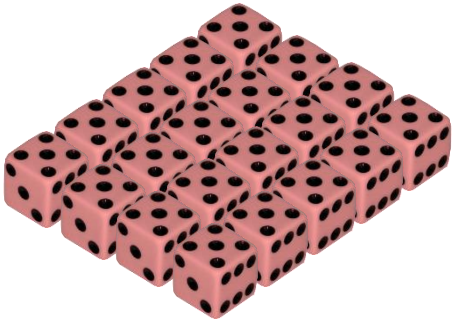
Repeat until one or both teams has no dice remaining!

Which team has a stronger force? goo.gl/jGTDBH

Dice Fighters Exercise

Red Team:

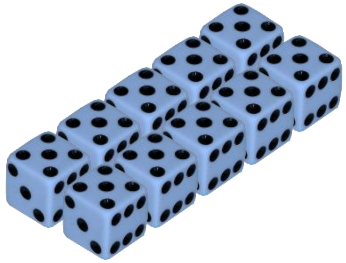
- 2x fighting force size
- Simple weapons



- Roll 6 to hit target

Blue Team:

- Small fighting force
- 3x effective weapons



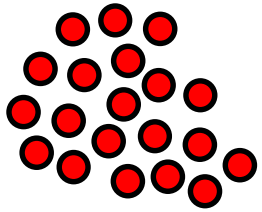
- Roll 4|5|6 to hit target

Remote students: use **www.random.org/dice/**

Submit results: **goo.gl/ApnTPh**

Analytical Model

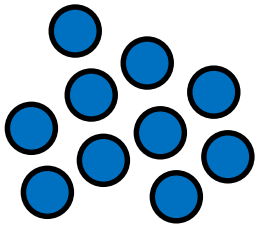
Red Team (A)



$A_0 = 20$

$\alpha = 1/6$

Blue Team (B)



$B_0 = 10$

$\beta = 3/6$

- Time-rate change of red team:

$$\frac{dA}{dt} = -\beta \cdot B$$

- Time-rate change of blue team:

$$\frac{dB}{dt} = -\alpha \cdot A$$

- Solve initial value problem:

$$\frac{dA}{dB} = \frac{\beta}{\alpha} \cdot \frac{B}{A} \Rightarrow \alpha \cdot A \cdot dA = \beta \cdot B \cdot dB$$

$$\alpha \cdot (A_0^2 - A^2) = \beta \cdot (B_0^2 - B^2)$$

Analytical Solution

- Lanchester's Square Law (1916):

$$\alpha \cdot (A_0^2 - A^2) = \beta \cdot (B_0^2 - B^2)$$

- Find required blue team "hit chance" for parity:

$$\beta = \alpha \cdot \frac{A_0^2 - A^2}{B_0^2 - B^2}$$

$$= \frac{1}{6} \cdot \frac{20^2 - 0^2}{10^2 - 0^2}$$

$$= \frac{1}{6} \cdot \frac{400}{100} = \frac{4}{6}$$

Classifying Dice Fighters

- Dice Fighters Exercise
  - System model > mathematical > simulation
  - Dynamic > discrete time
  - Stochastic
- Lanchester's Square Law
  - System model > mathematical > analytical
  - Static
  - Deterministic

Dice Fighters State Diagram

Model state

- What information is required to recreate a snapshot in time?
- What other derived state may be useful?

State changes

- What actions change the model state?

DiceFighters
round_number : int blue_size : int blue_chance_hit : float red_size : int red_chance_hit : float
is_complete() : boolean generate_blue_hits() : int generate_red_hits() : int blue_suffer_losses(int) : void red_suffer_losses(int) : void next_round() : void

Dice Fighters Activity Diagram

DiceFighters
round_number : int blue_size : int blue_chance_hit : float red_size : int red_chance_hit : float
is_complete() : boolean generate_blue_hits() : int generate_red_hits() : int blue_suffer_losses(int) : void red_suffer_losses(int) : void next_round() : void