



# Small Potatoes: Microsimulation with Vivarium

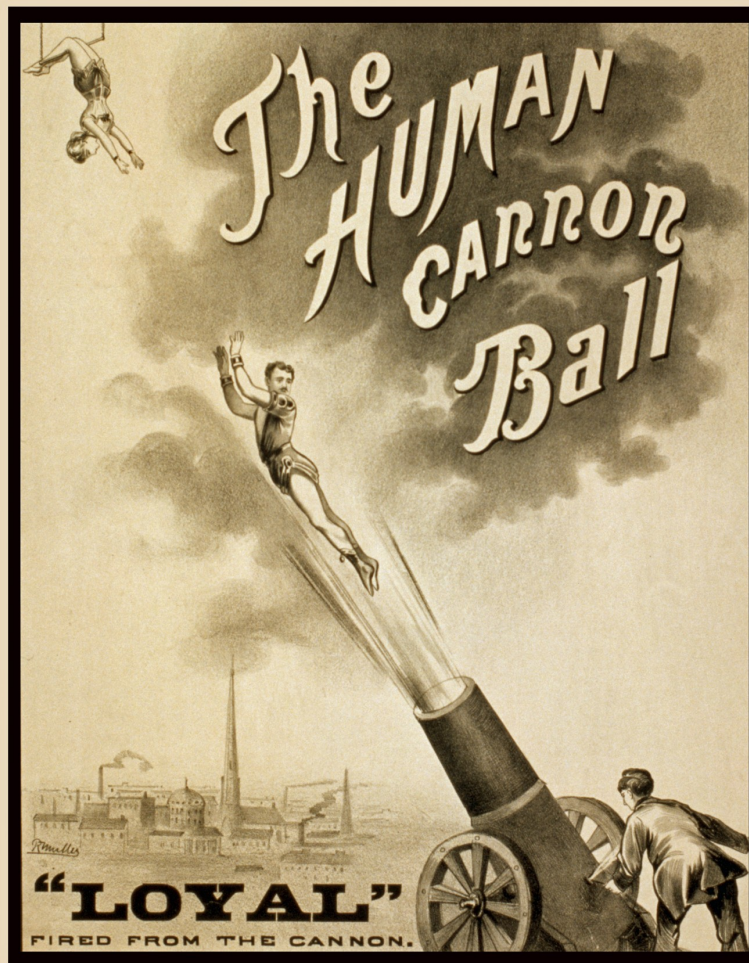
---

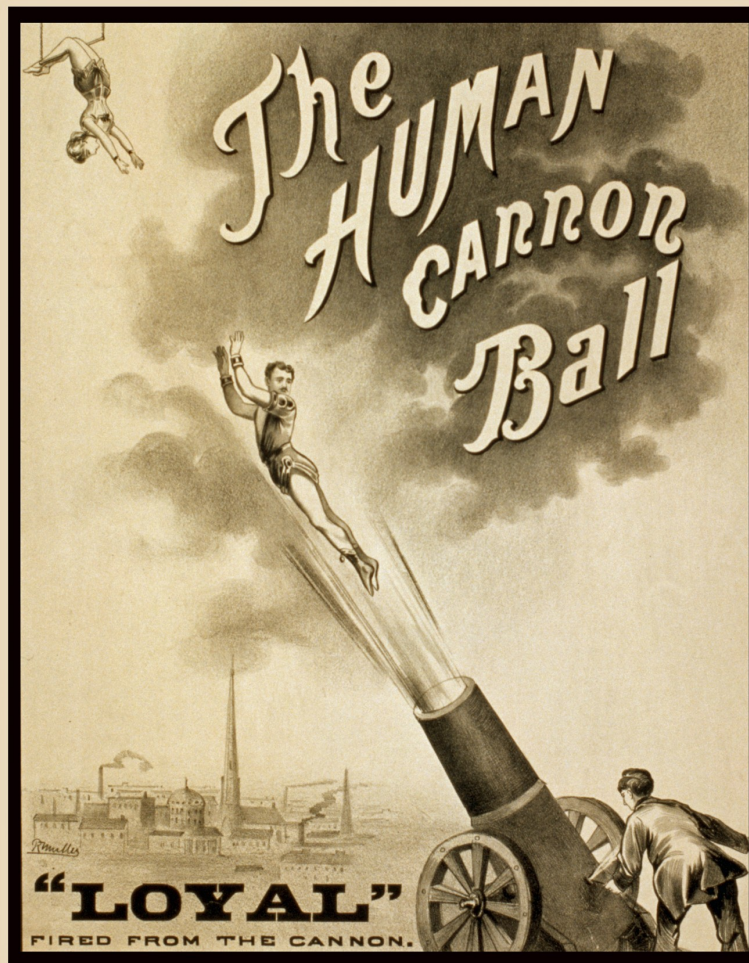
Alec Deason

Why simulation?

[https://github.com/alec-deason/small\\_potatoes](https://github.com/alec-deason/small_potatoes)

<https://github.com/ihmeuw/vivarium>



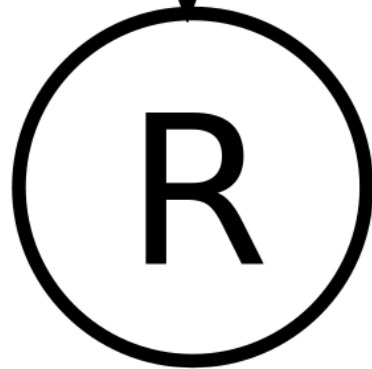
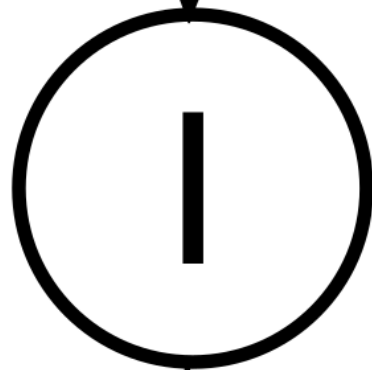
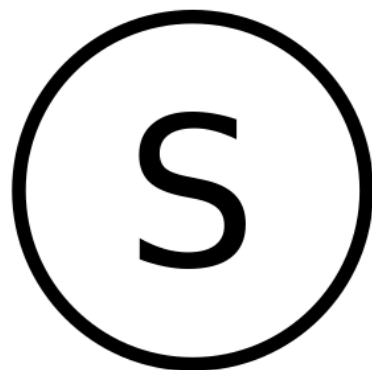


$$d = \frac{v^2 \sin(2\theta)}{g}$$

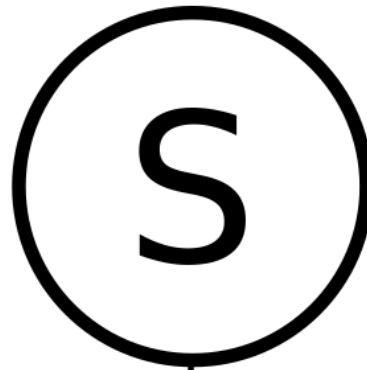




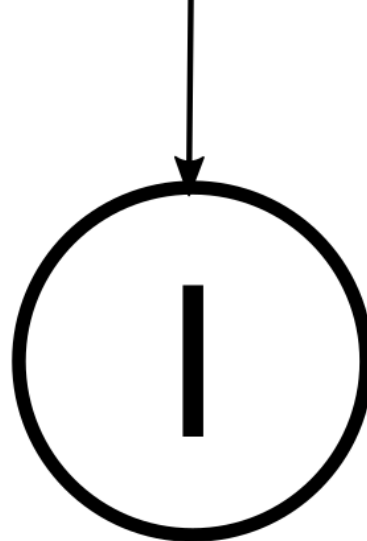
But, why microsimulation?



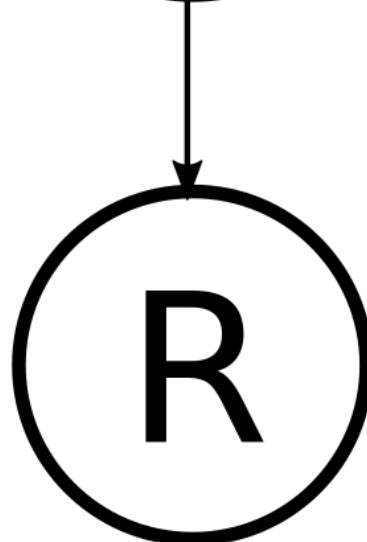




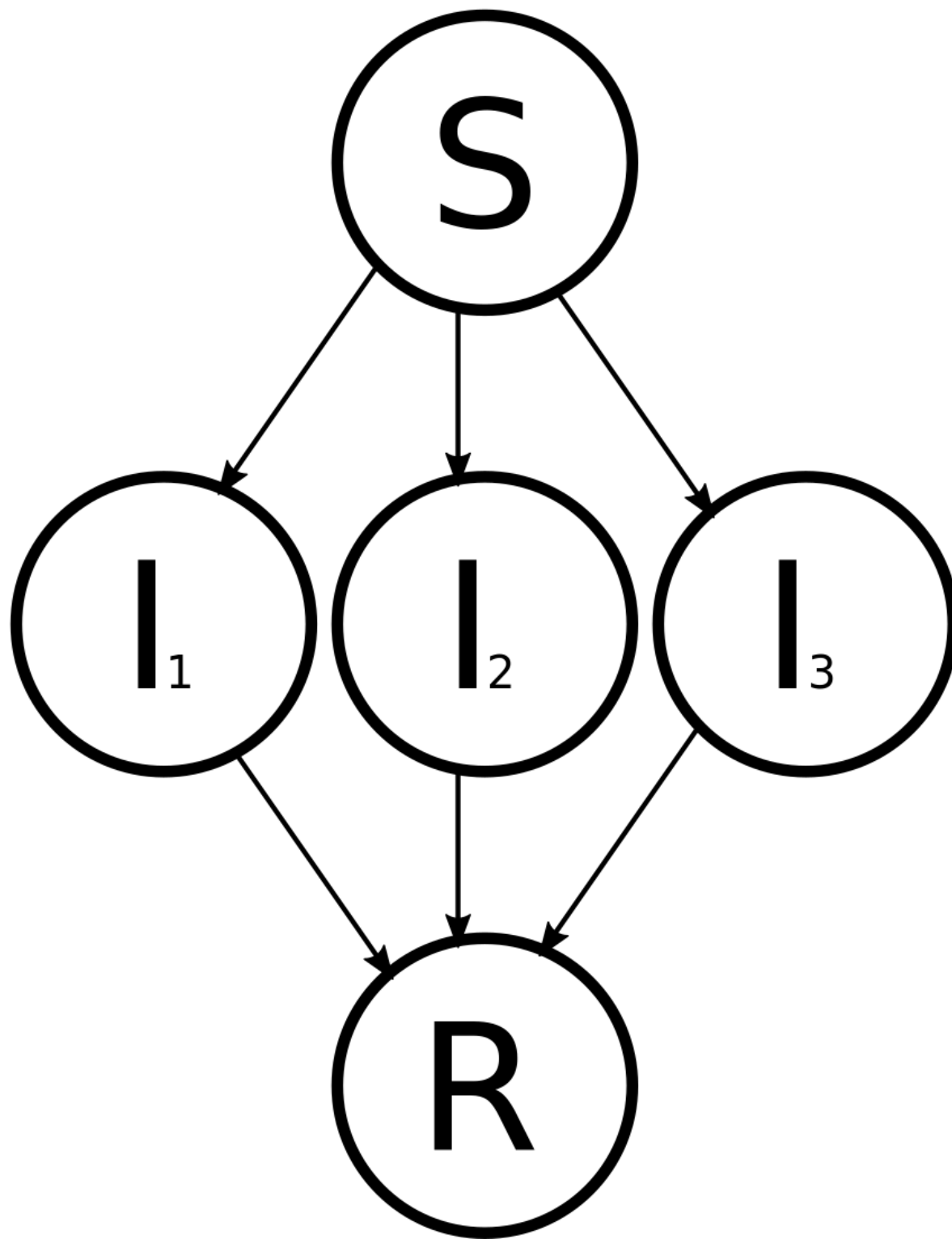
$$\frac{dS}{dt} = -\frac{\beta IS}{N}$$

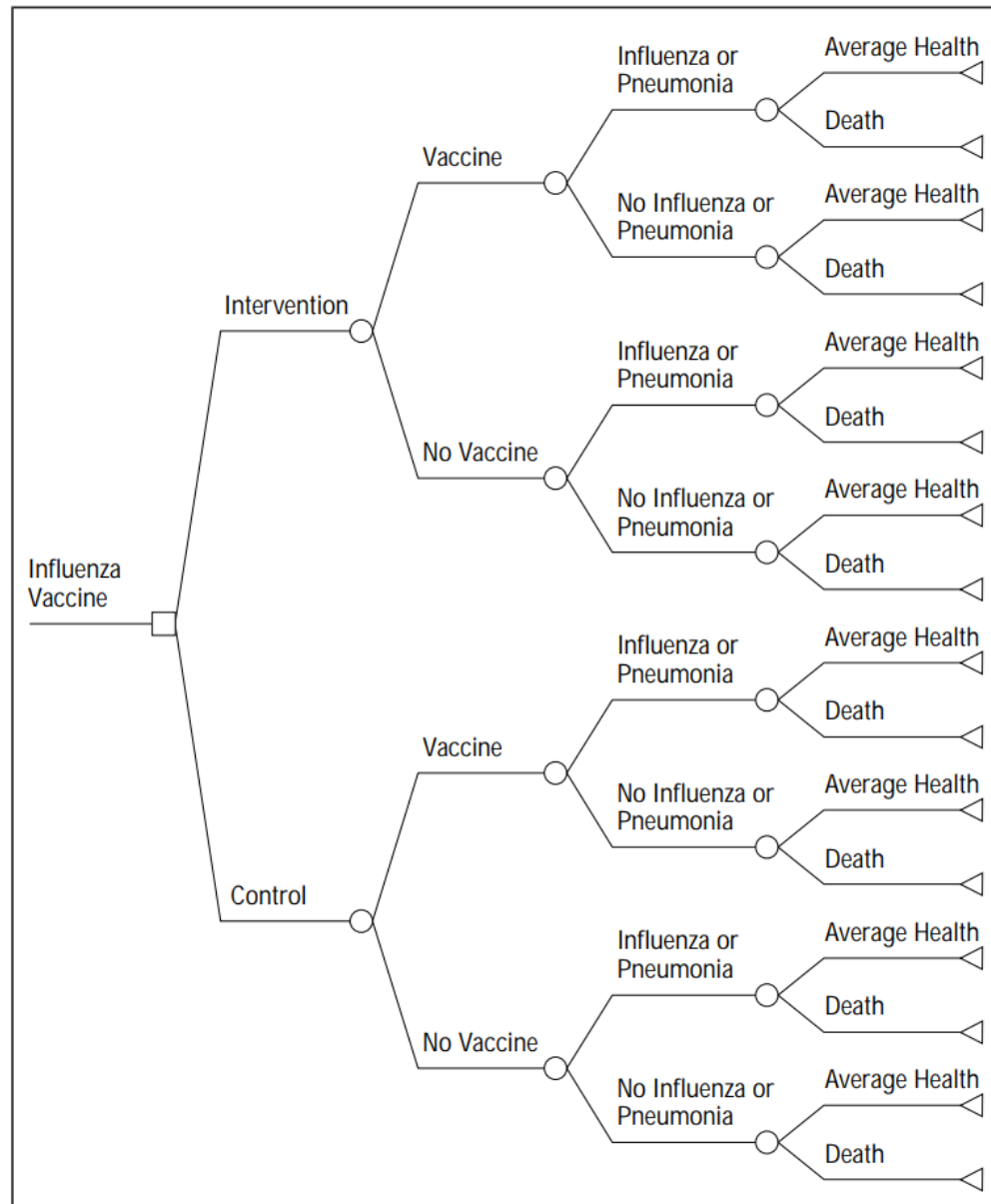


$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I$$



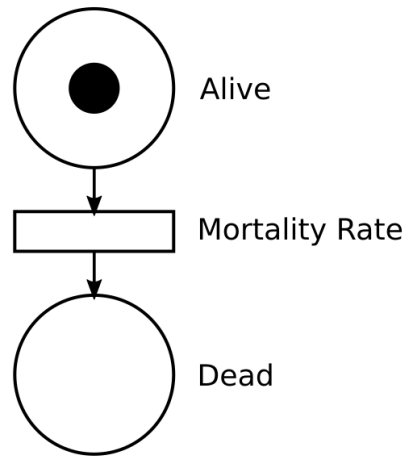
$$\frac{dR}{dt} = \gamma I$$



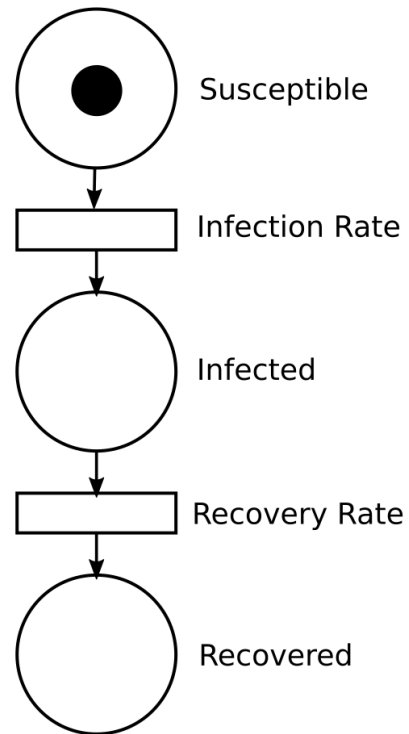


**Figure 3.** Decision tree for an intervention to promote only the influenza vaccine.

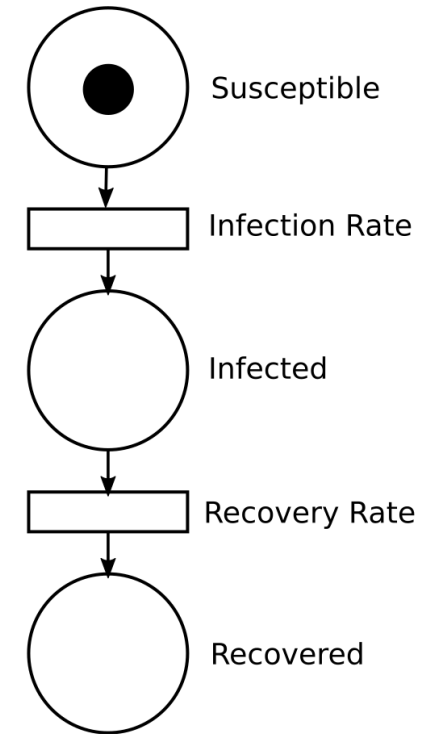
## Mortality



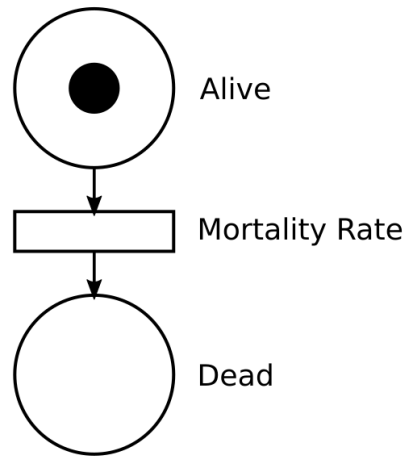
## H1N1



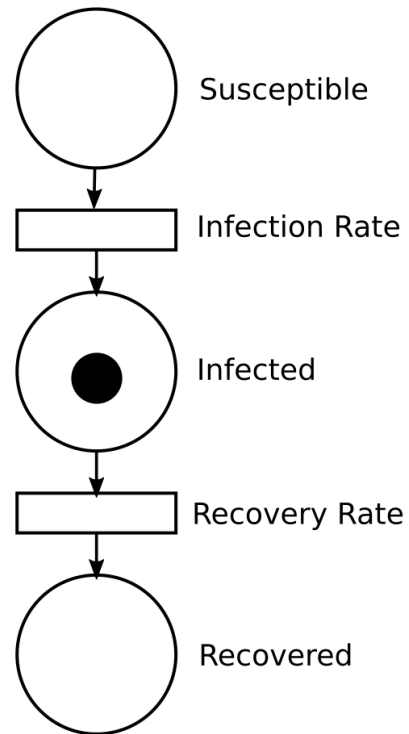
## H3N2



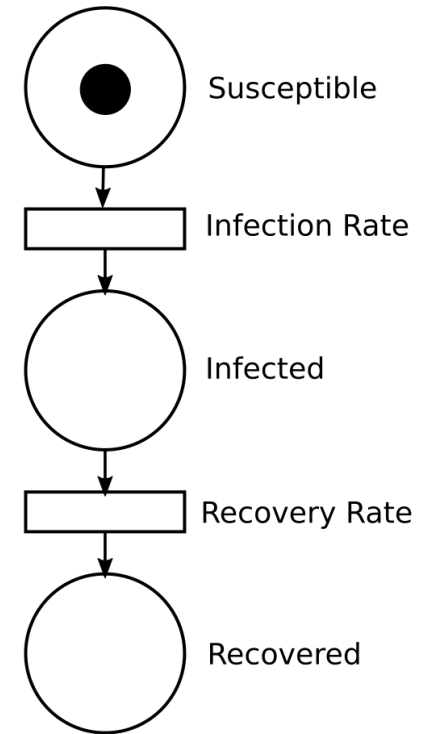
## Mortality



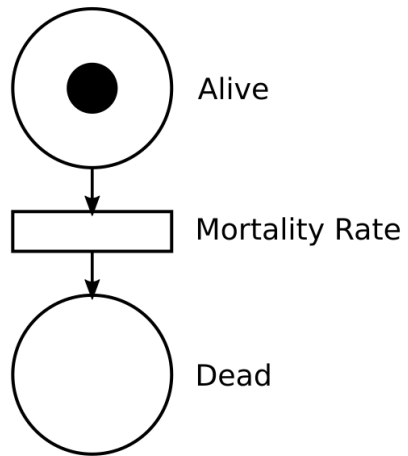
## H1N1



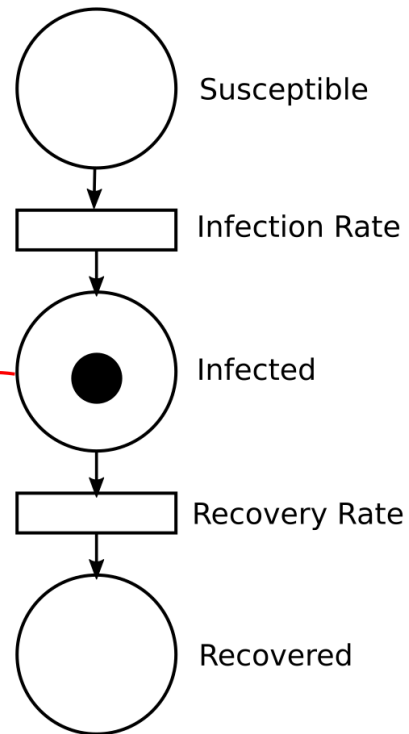
## H3N2



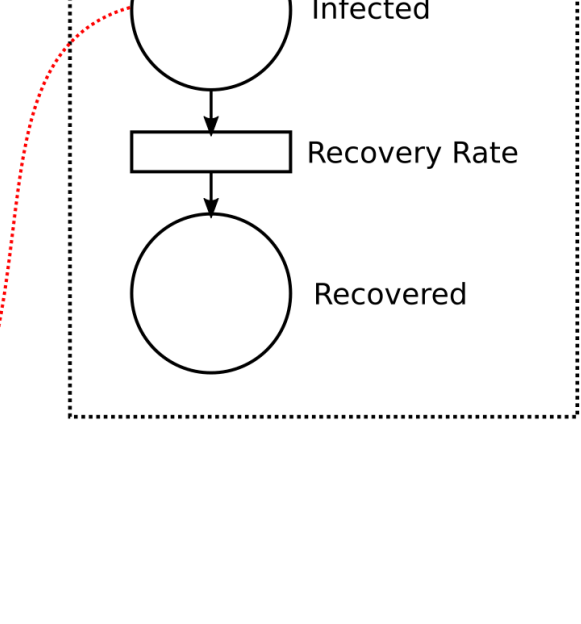
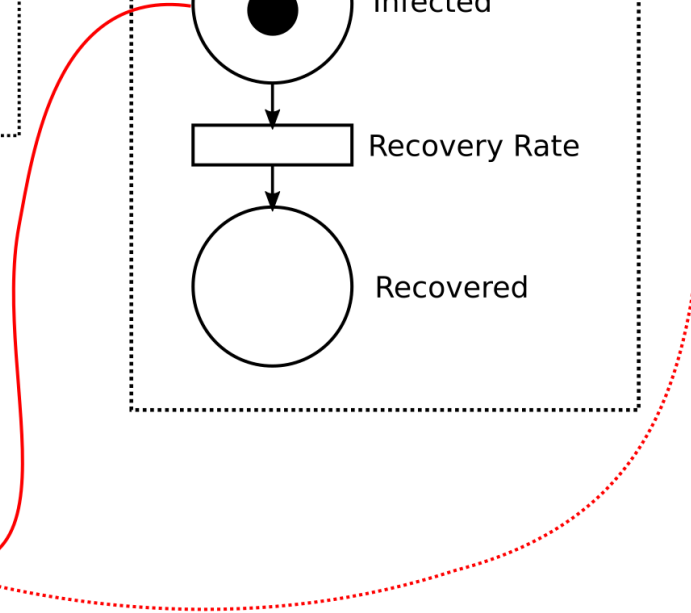
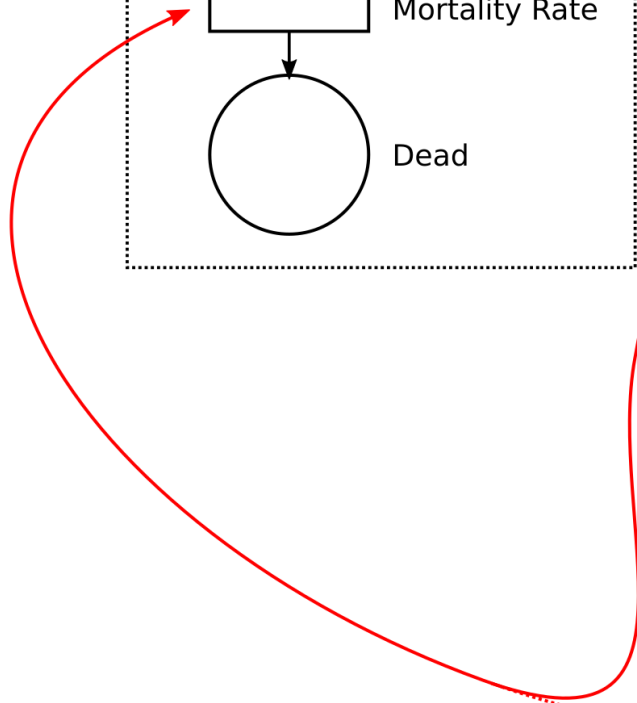
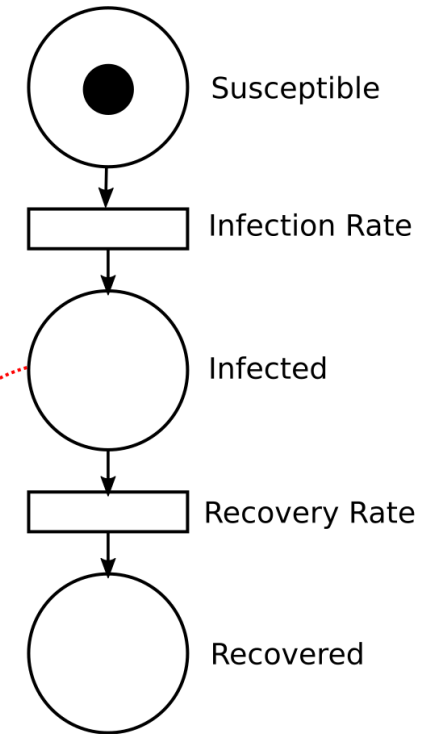
## Mortality



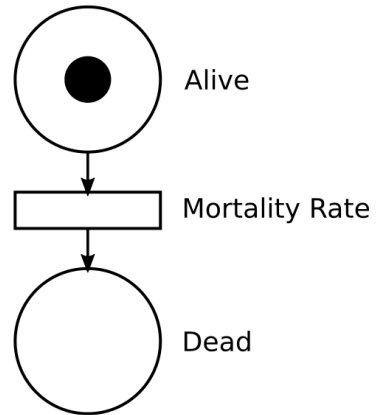
## H1N1



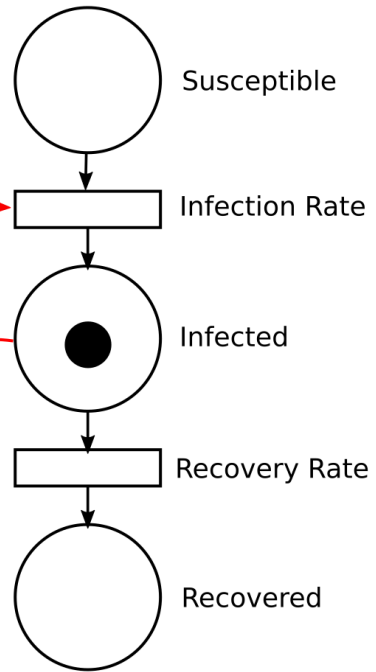
## H3N2



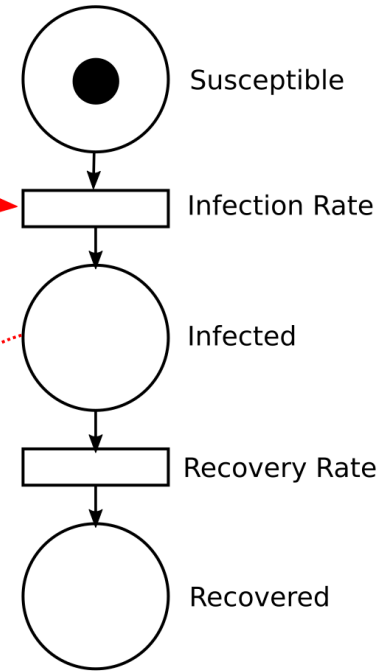
## Mortality



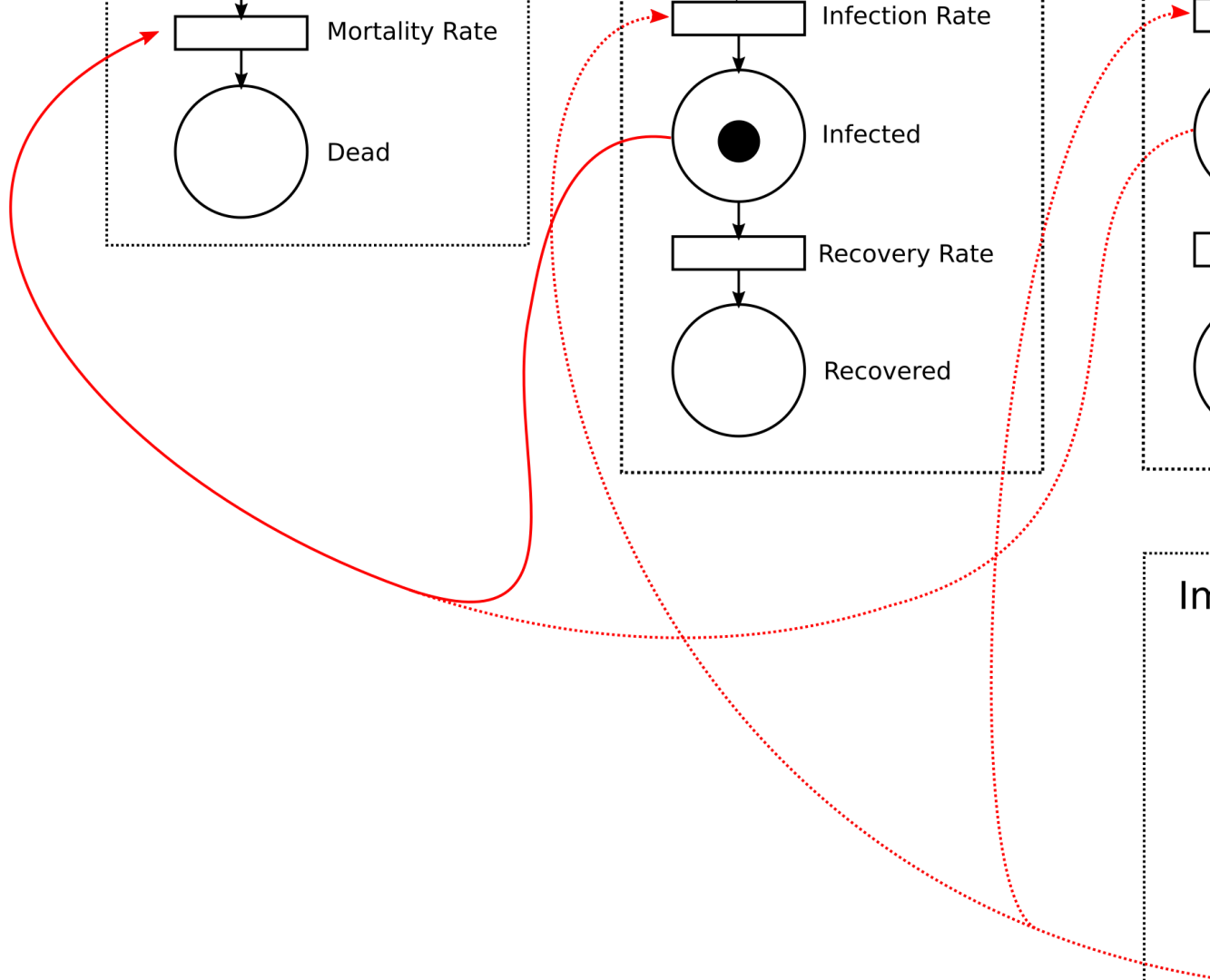
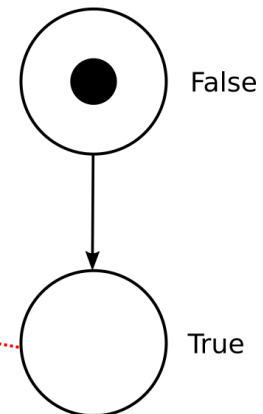
## H1N1



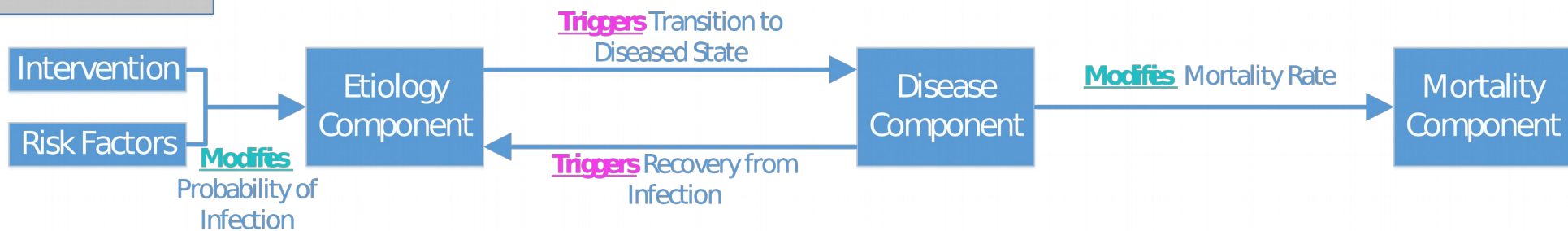
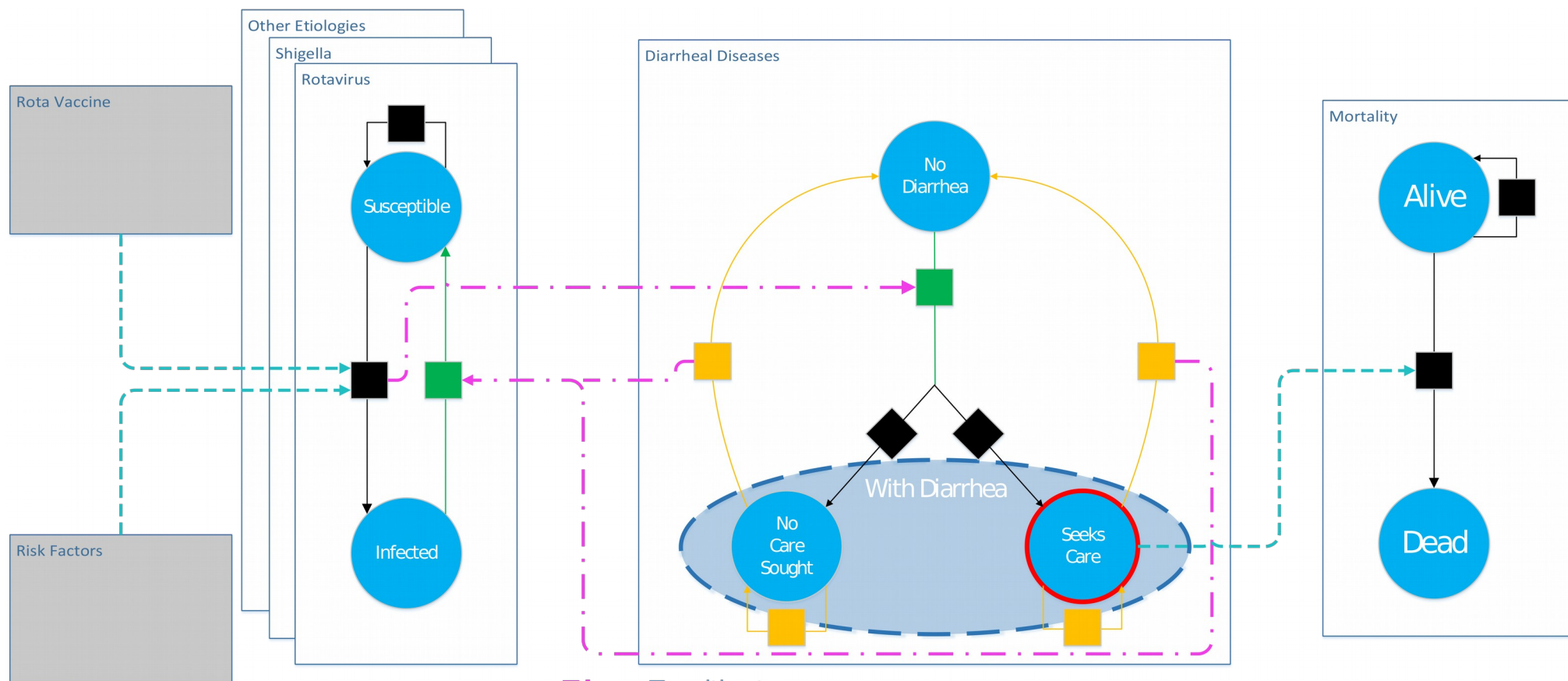
## H3N2



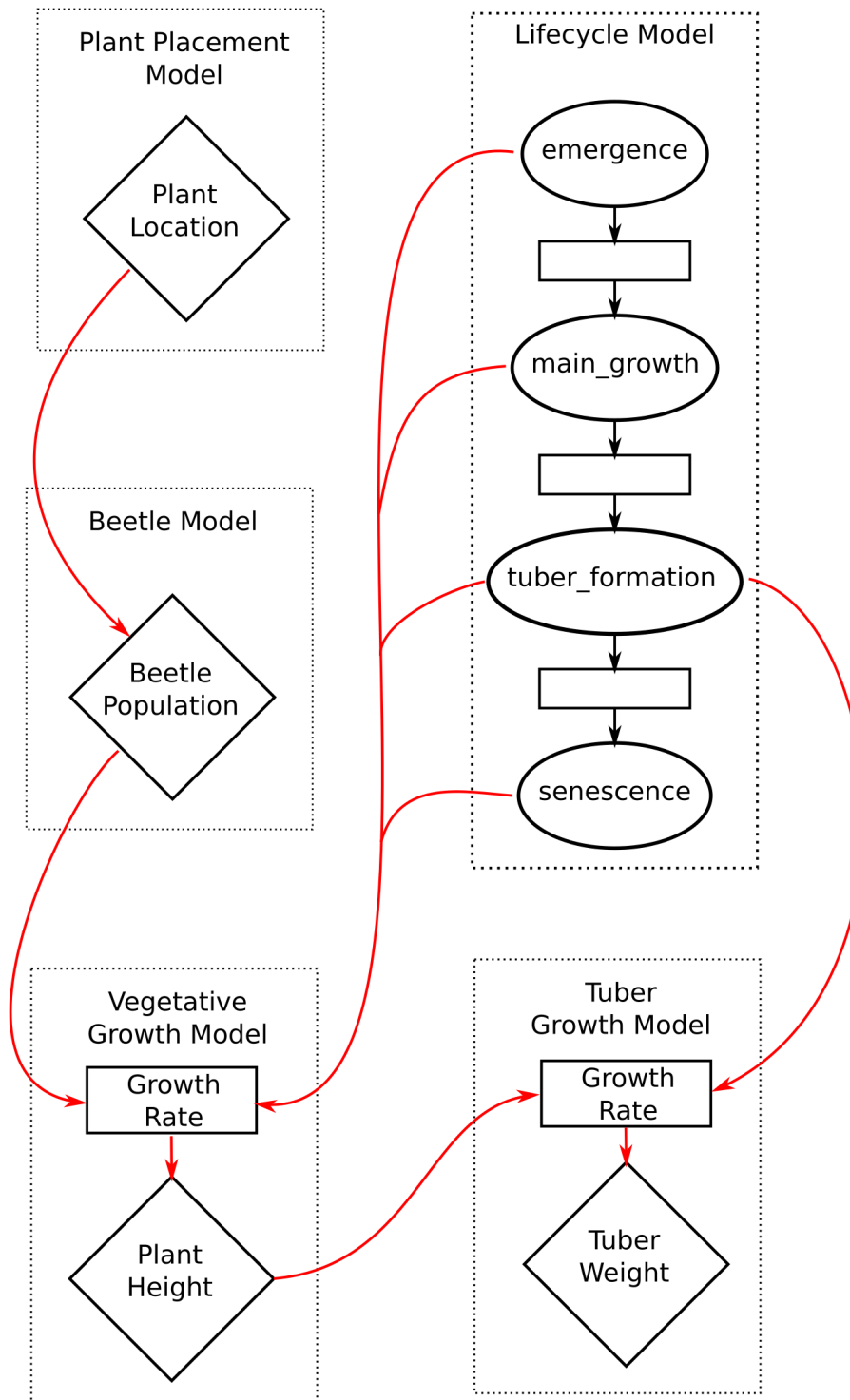
## Immunosuppressed







What's Vivarium then?



```
do_some_setup()
```

```
while current_time < end_time:  
    execute_all_components()
```

```
output_results()
```

```
@listens_for('time_step')
@uses_columns(['plant_height'])
def growth(self, event):
    effective_daily_growth = self.daily_growth(event.index)

    population = event.population

    population['plant_height'] += effective_daily_growth

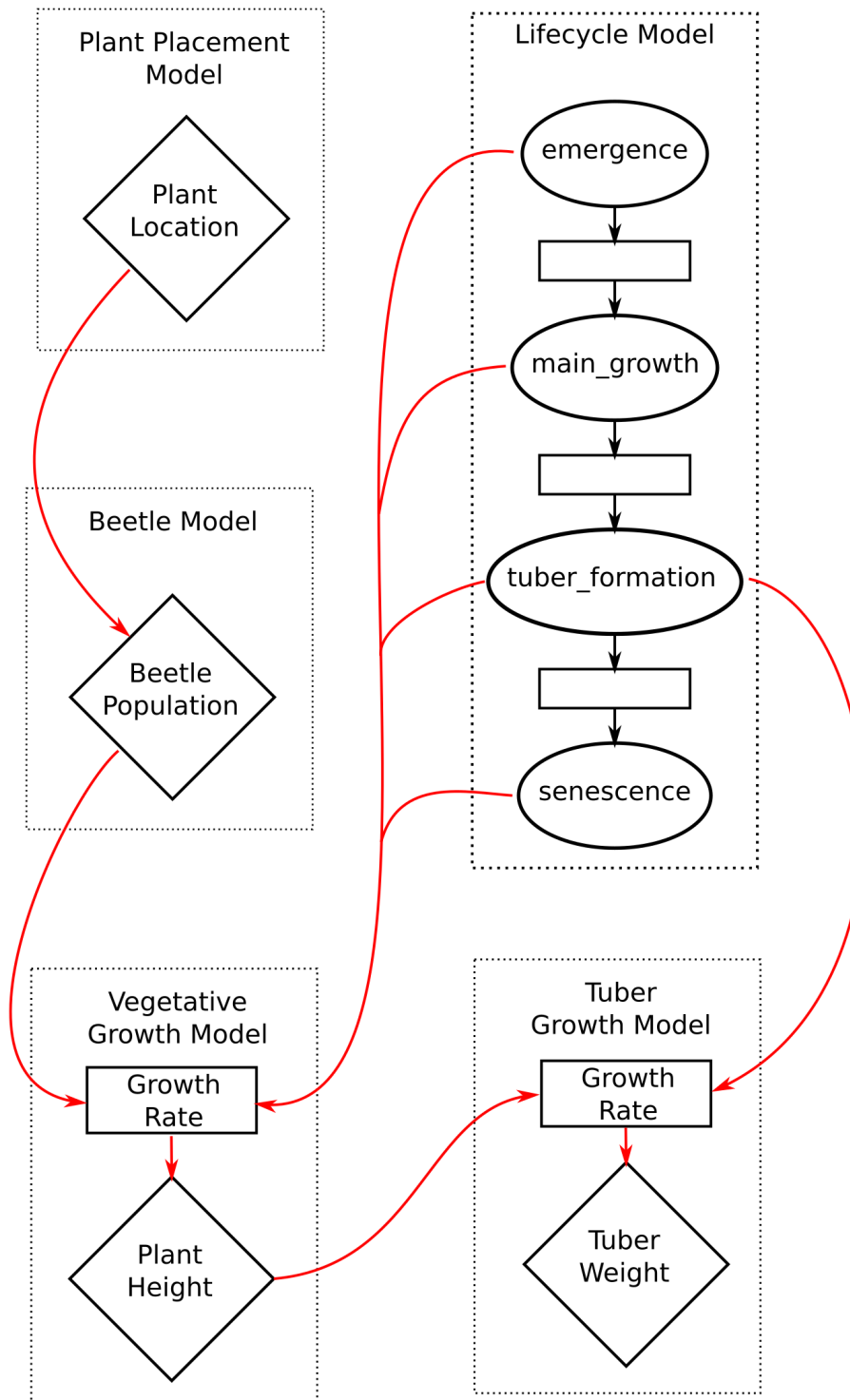
    event.population_view.update(population)
```

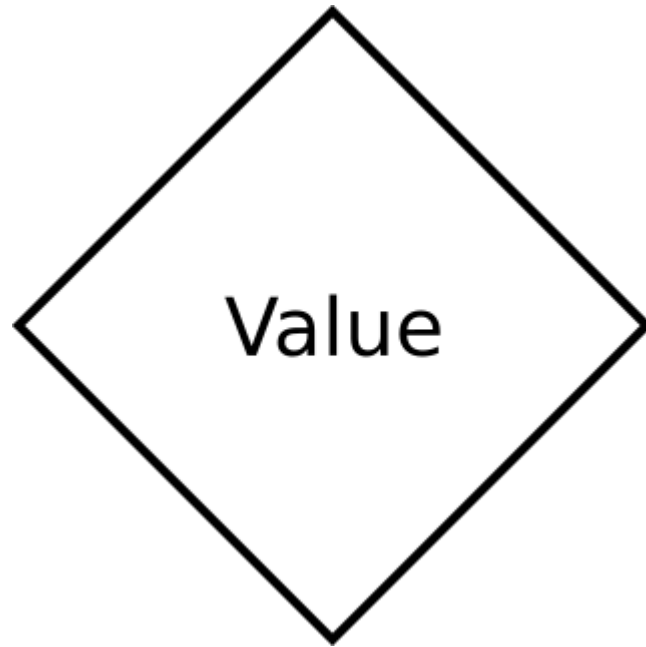
id	plant_height	growth_phase	tuber_weight	location_x	location_y	beetle_population
0	16	main_growth	0	0	0	2
1	20	tuber_formation	10	10	10	104
2	15	main_growth	0	11	10	30

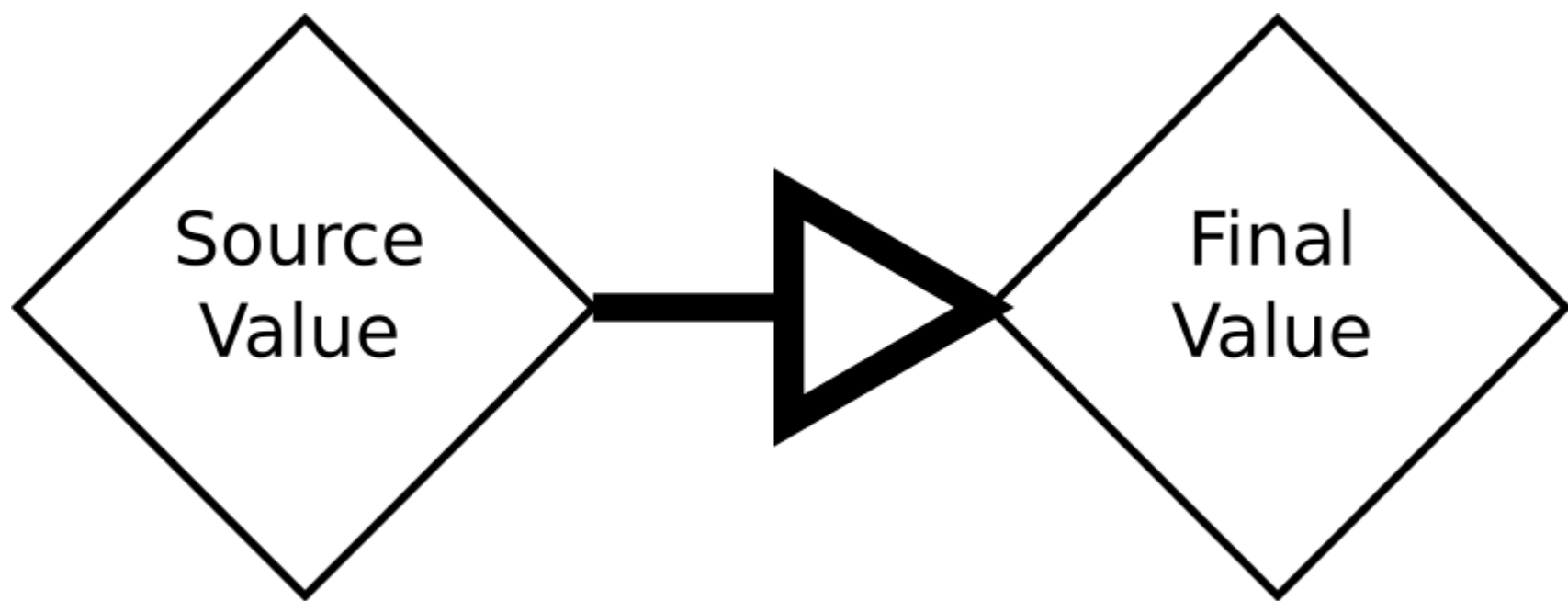
```
@listens_for('initialize_simulants')
@uses_columns(['tuber_weight'])
def create_initial_weight(self, event):
    event.population_view.update(pd.Series(0.0, index=event.index))
```

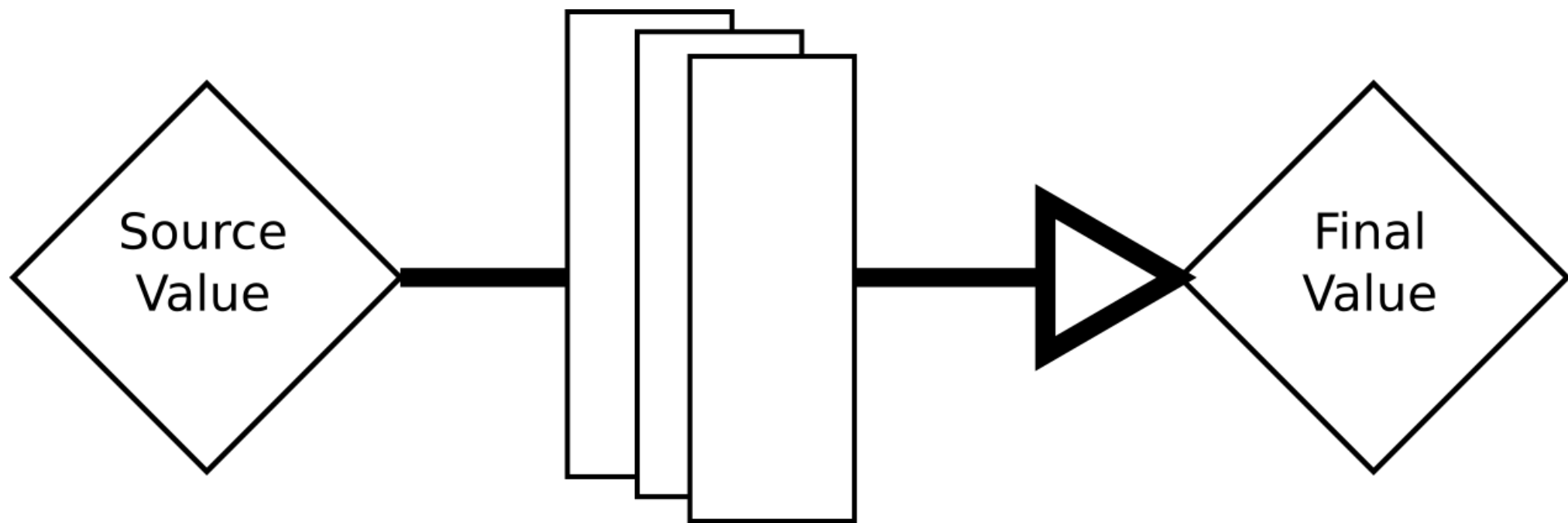


Framework		Growth		Placement		Flea Beetle
id	plant_height	growth_phase	tuber_weight	location_x	location_y	beetle_population
0	16	main_growth	0	0	0	2
1	20	tuber_formation	10	10	10	104
2	15	main_growth	0	11	10	30









```

@produces_value('potato.daily_tuber_growth')
def tuber_growth_base_rate(self, index):
    population = self.population_view.get(index)

    daily_growth_in_ounces = 1.8
    return pd.Series((population[self._model] == self.state_id) * daily_growth_in_ounces, index=index)

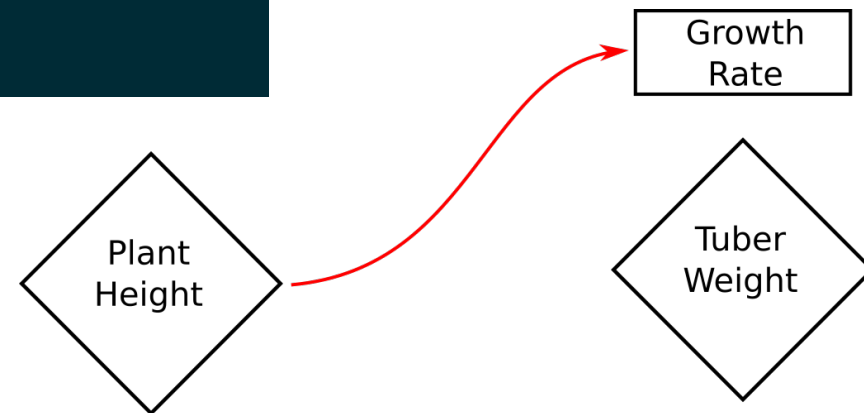
```

```

@modifies_value('potato.daily_tuber_growth')
@uses_columns(['plant_height'])
def modify_tuber_growth(self, index, rate, population_view):
    pop = population_view.get(index)
    multiplier = pop.plant_height / 24

    return rate * multiplier

```



```

@listens_for('time_step')
@uses_columns(['tuber_weight'])
def growth(self, event):
    effective_daily_growth = self.daily_growth(event.index)

    population = event.population

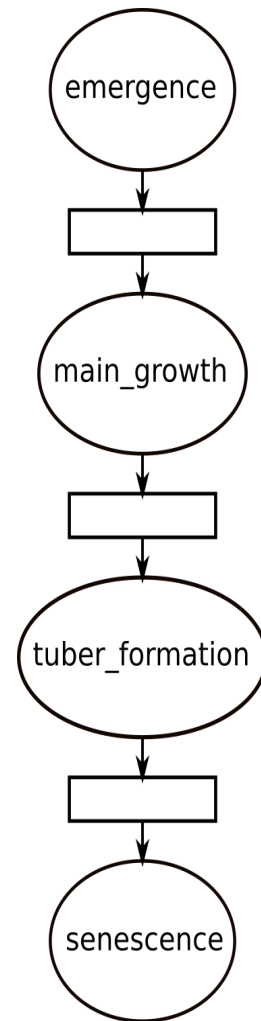
    population['tuber_weight'] += effective_daily_growth

    event.population_view.update(population)

```

```
emergence = GrowthPhaseState('emergence', 0.0)
main_growth = GrowthPhaseState('main_growth', 1.0)
tuber_formation = TuberFormationPhase(0.1)
senescence = GrowthPhaseState('senescence', 0.0)

emergence.transition_set.append(RateTransition(emergence, main_growth, main_growth_rate))
main_growth.transition_set.append(RateTransition(main_growth, tuber_formation, tuber_formation_rate))
tuber_formation.transition_set.append(RateTransition(tuber_formation, senescence, senescence_rate))
```





```

class VegetativeGrowthComponent:
    def setup(self, builder):
        self.daily_growth = builder.value('potato.daily_vegetative_growth')
        self.daily_growth.source = lambda index: pd.Series(0.2, index=index)

    @listens_for('initialize_simulants')
    @uses_columns(['plant_height'])
    def create_initial_height(self, event):
        event.population_view.update(pd.Series(0.0, index=event.index))

    @listens_for('time_step')
    @uses_columns(['plant_height'])
    def growth(self, event):
        effective_daily_growth = self.daily_growth(event.index)

        population = event.population

        population['plant_height'] += effective_daily_growth

        event.population_view.update(population)

    @modifies_value('potato.daily_tuber_growth')
    @uses_columns(['plant_height'])
    def modify_tuber_growth(self, index, rate, population_view):
        pop = population_view.get(index)
        multiplier = pop.plant_height / 24

        return rate * multiplier

    @listens_for('simulation_end')
    @uses_columns(['plant_height'])
    def metrics(self, event):
        print(f'Average Plant Height: {event.population.plant_height.mean()} in')

```

Growth  
Rate

Plant  
Height

```

class PlantingComponent:
    configuration_defaults = {
        'planting': {
            'density': 0.05 # Plants per square foot
        }
    }

    def setup(self, builder):
        self.density = builder.configuration.planting.density
        self.randomness = builder.randomness('placement')

    @listens_for('initialize_simulants')
    @uses_columns(['location_x', 'location_y'])
    def create_initial_population(self, event):
        garden_edge_length = np.sqrt(len(event.index) / self.density)
        x = self.randomness.get_draw(event.index, 'x') * garden_edge_length
        y = self.randomness.get_draw(event.index, 'y') * garden_edge_length
        placements = pd.DataFrame({'location_x': x, 'location_y': y})
        event.population_view.update(placements)

```

