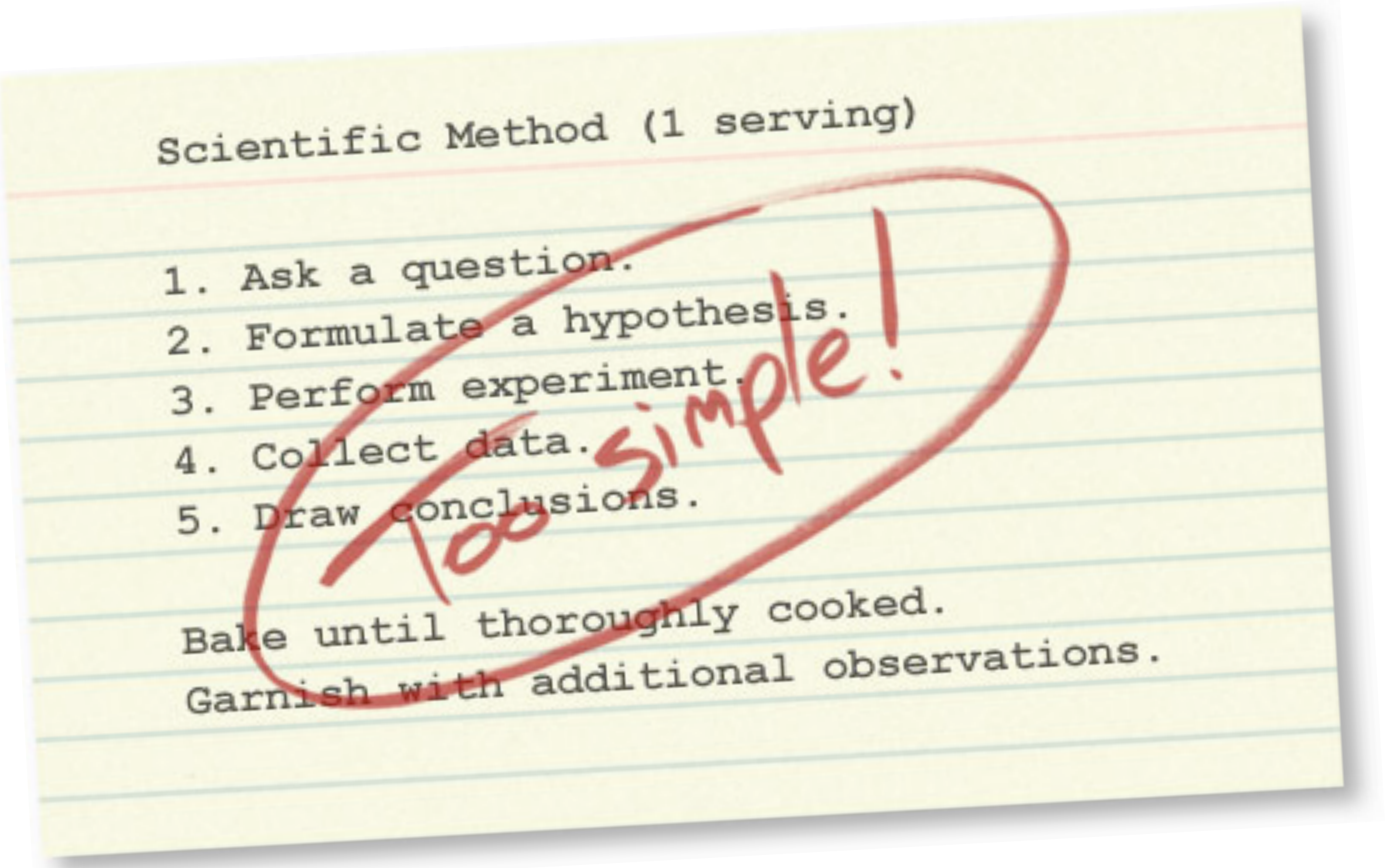
Predictive Models of Development Teams

and the Systems They Build

Robert Smallshire

@robsmallshire


Scientific Method (1 serving)
1. Ask a question.
2. Formulate a hypothesis.
3. Perform experiment.
4. Collect data.
5. Draw conclusions.
Bake until thoroughly cooked.
Garnish with additional observations.
Too simple!

Experimental Science

Randomised controlled trials

- Developers don't like to be watched
- Eliminating extraneous factors
- Toy problems aren't realistic
- No two projects are the same
- Can't do double-blind
- Students have little experience
- Time and money

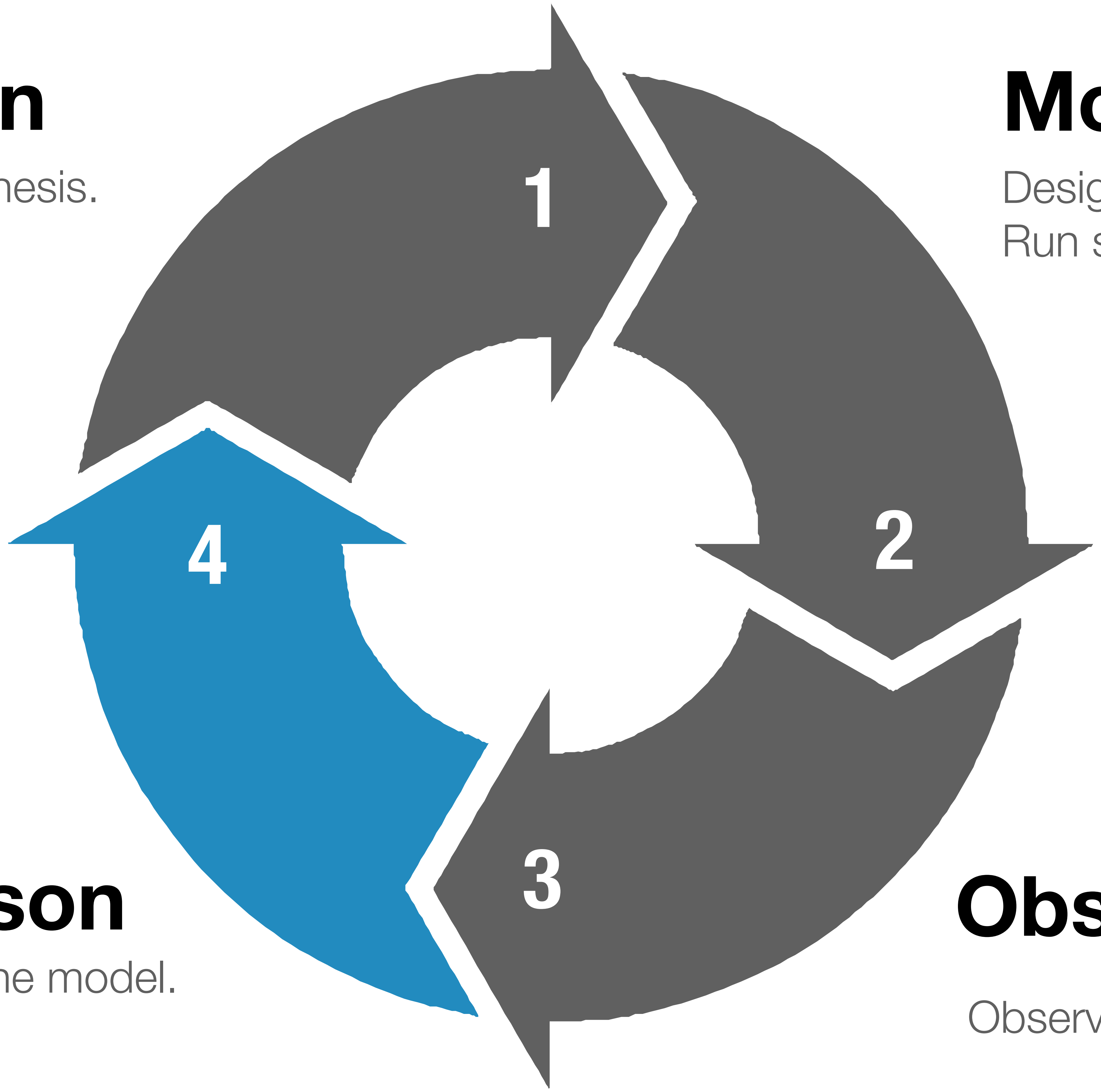
How can we know?

Prediction

Formulate a hypothesis.

Modelling

Design a conceptual model.
Run simulations.



Comparison

Validate or refute the model.

Observation

Observe and record reality.

Modelling system growth 1
How many people work on your system?

Predicting project progress 2
How many people should work on your system?

Software process dynamics 3
How can you construct models and run simulations?

Lifetimes in the software industry

Systems and their architectures are long lived

Sources: Software Lifetime and its Evolution Process over Generations, CEO Succession Practices: 2012 Edition, Investors Chronicle,

Simulating Developer Productivity

Draw teams at random from a productivity distribution

Productivity on 10000 SLOC codebase

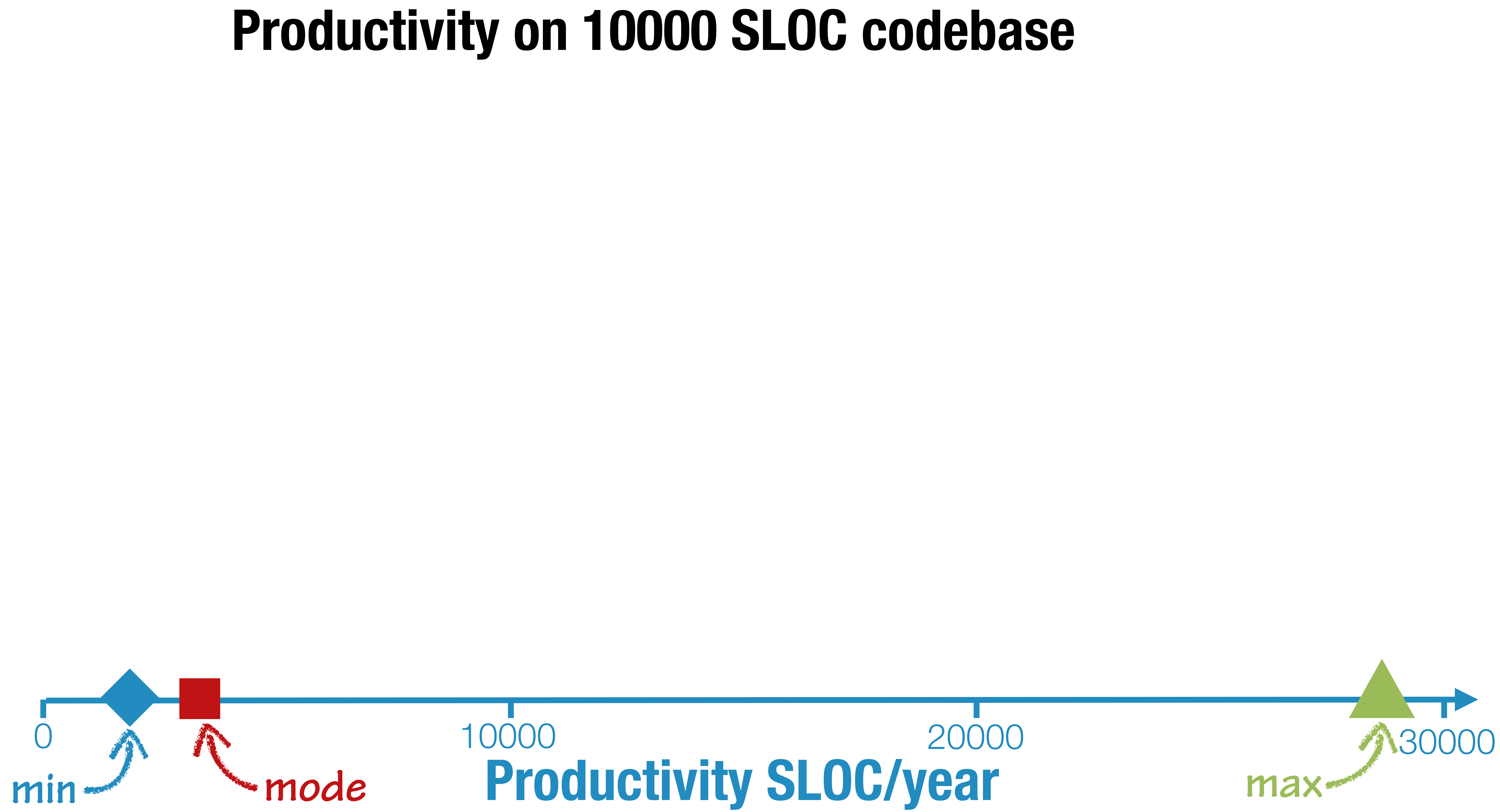
Simulating Developer Productivity

Draw teams at random from a productivity distribution

Productivity on 10000 SLOC codebase

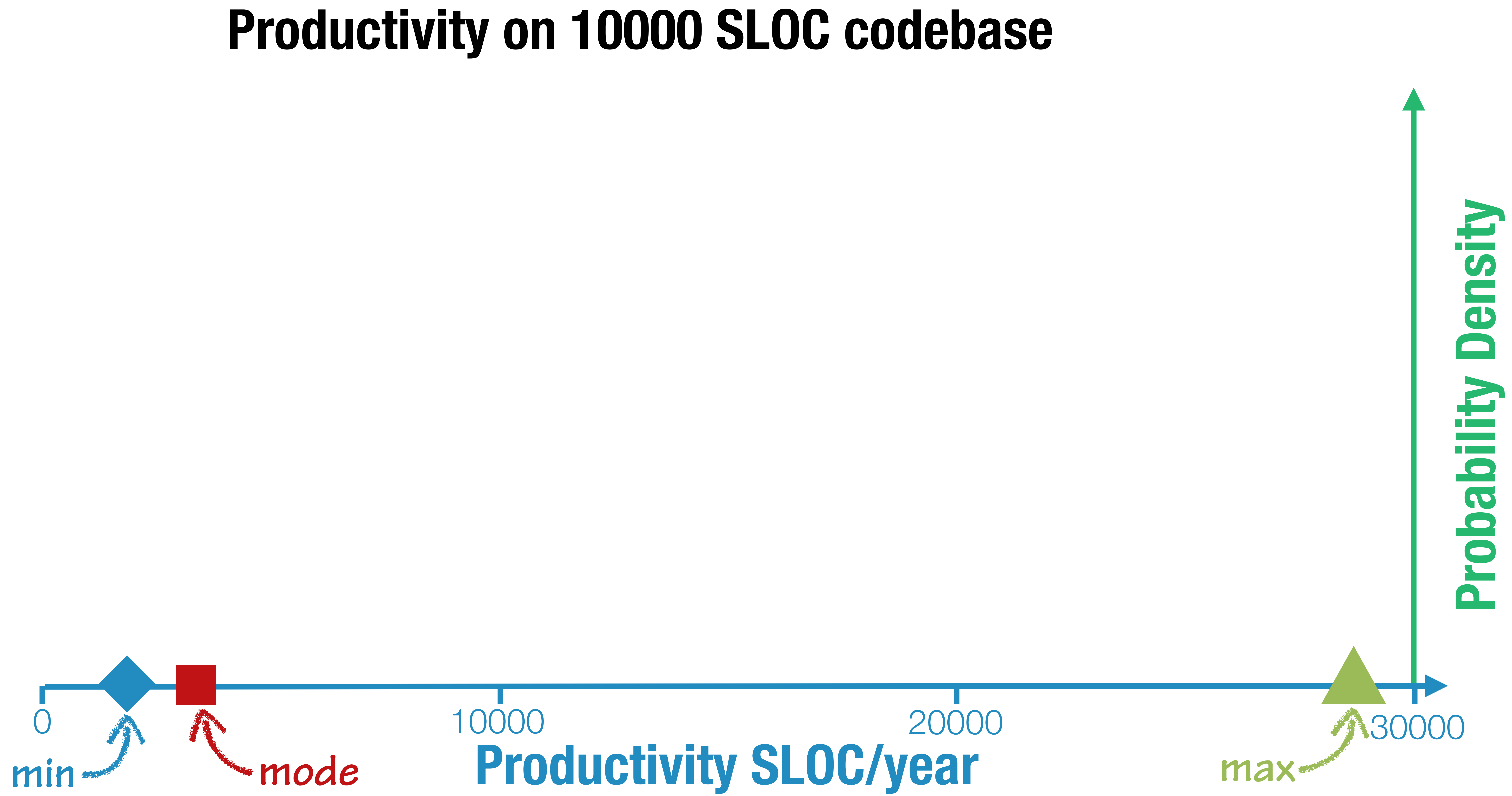
Simulating Developer Productivity

Draw teams at random from a productivity distribution



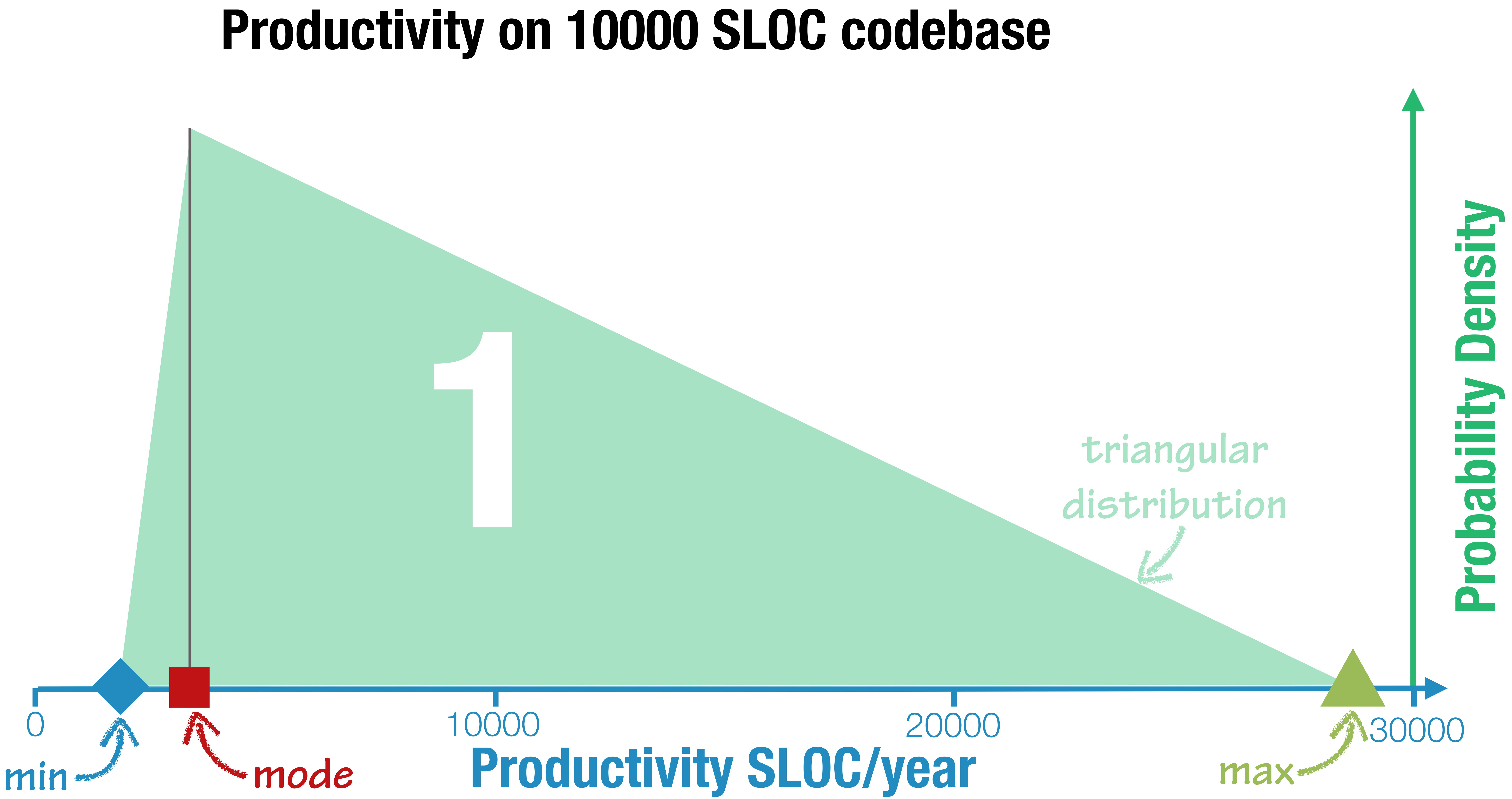
Simulating Developer Productivity

Draw teams at random from a productivity distribution



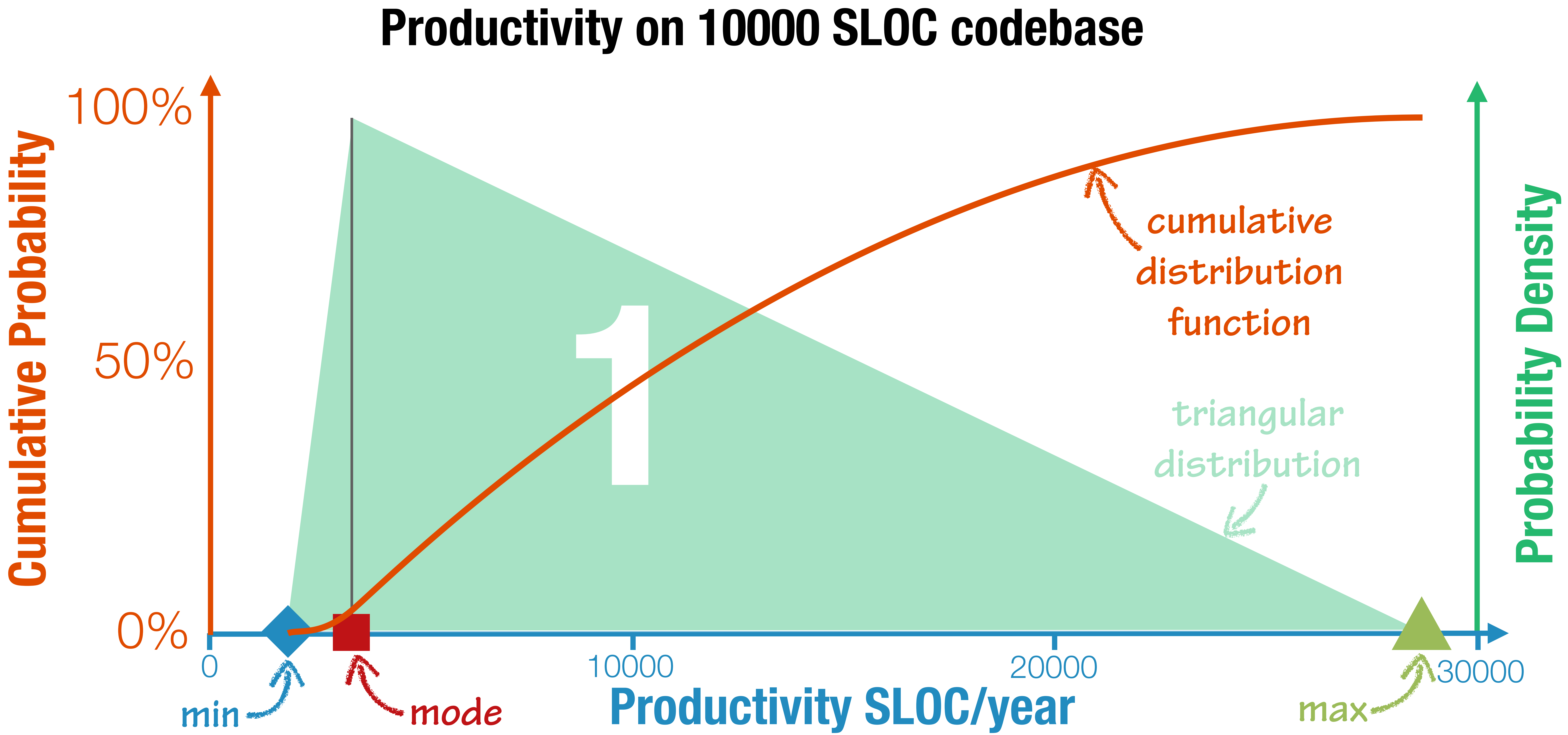
Simulating Developer Productivity

Draw teams at random from a productivity distribution



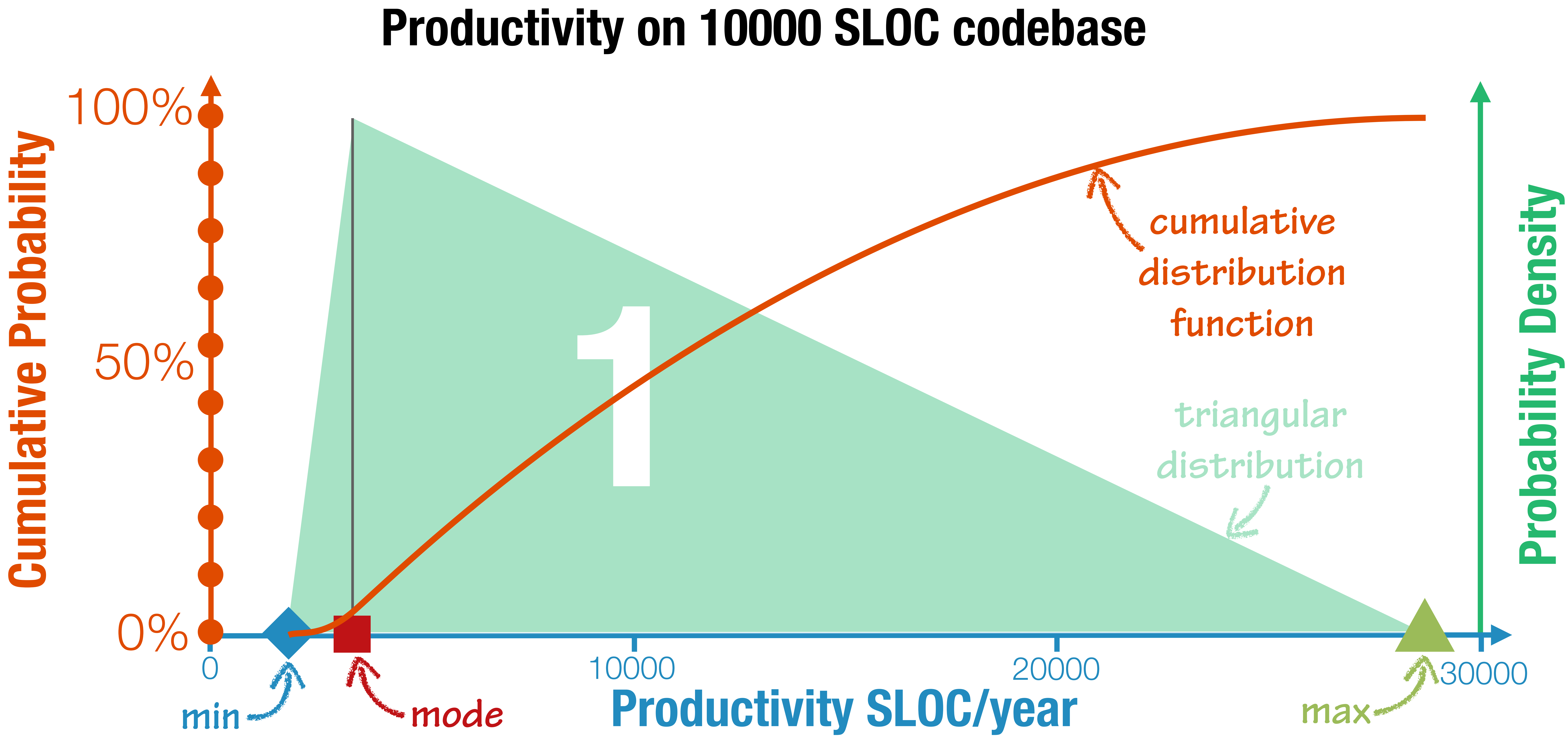
Simulating Developer Productivity

Draw teams at random from a productivity distribution



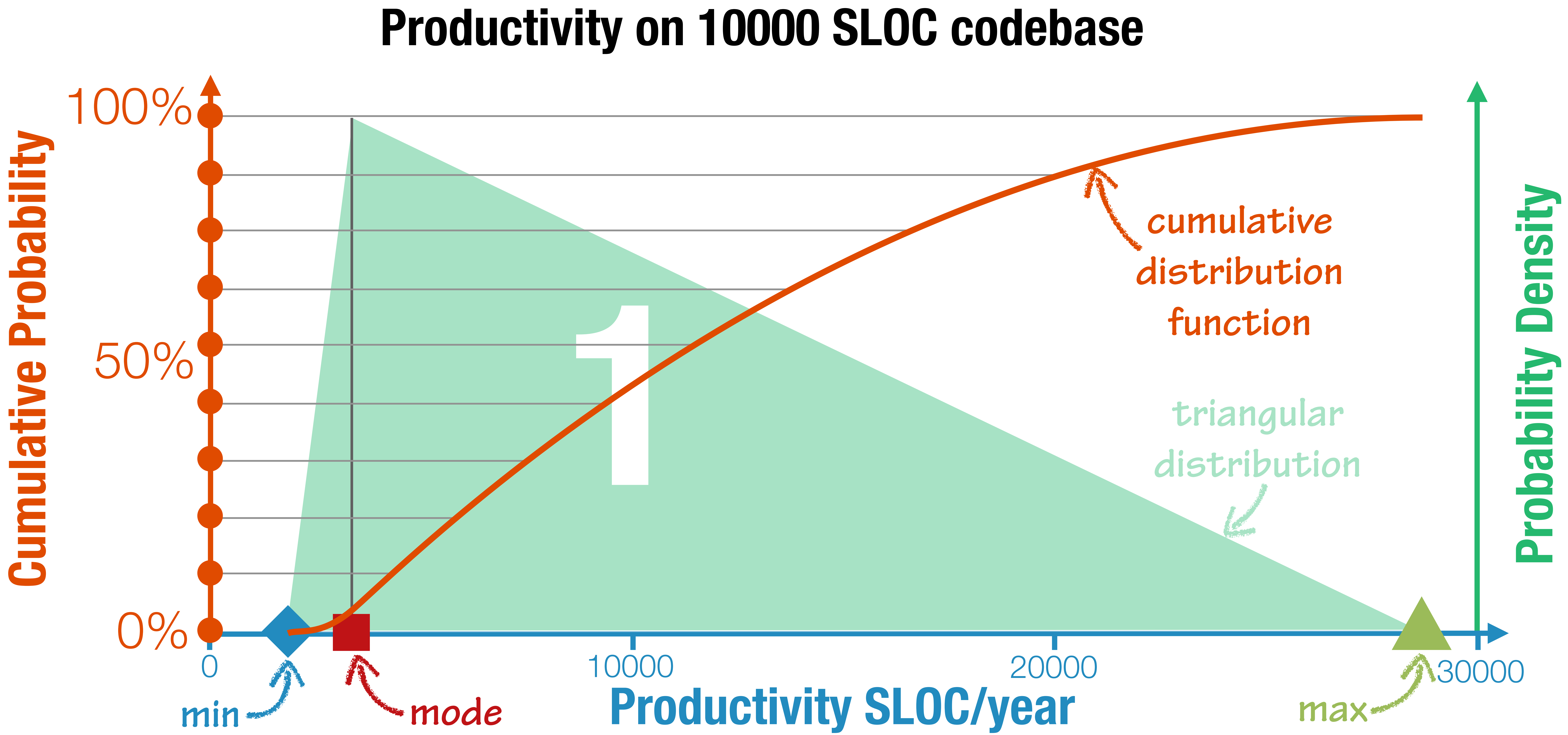
Simulating Developer Productivity

Draw teams at random from a productivity distribution



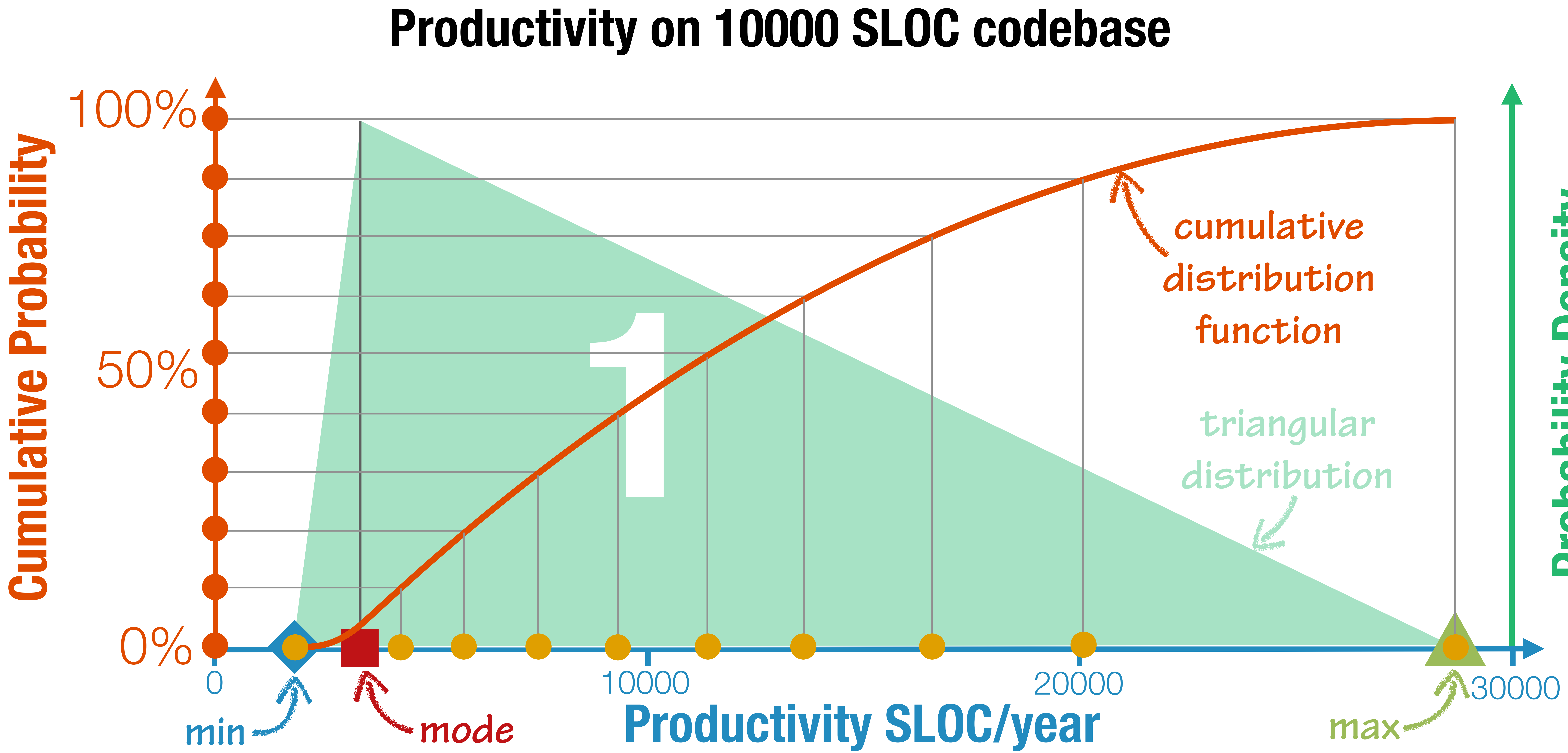
Simulating Developer Productivity

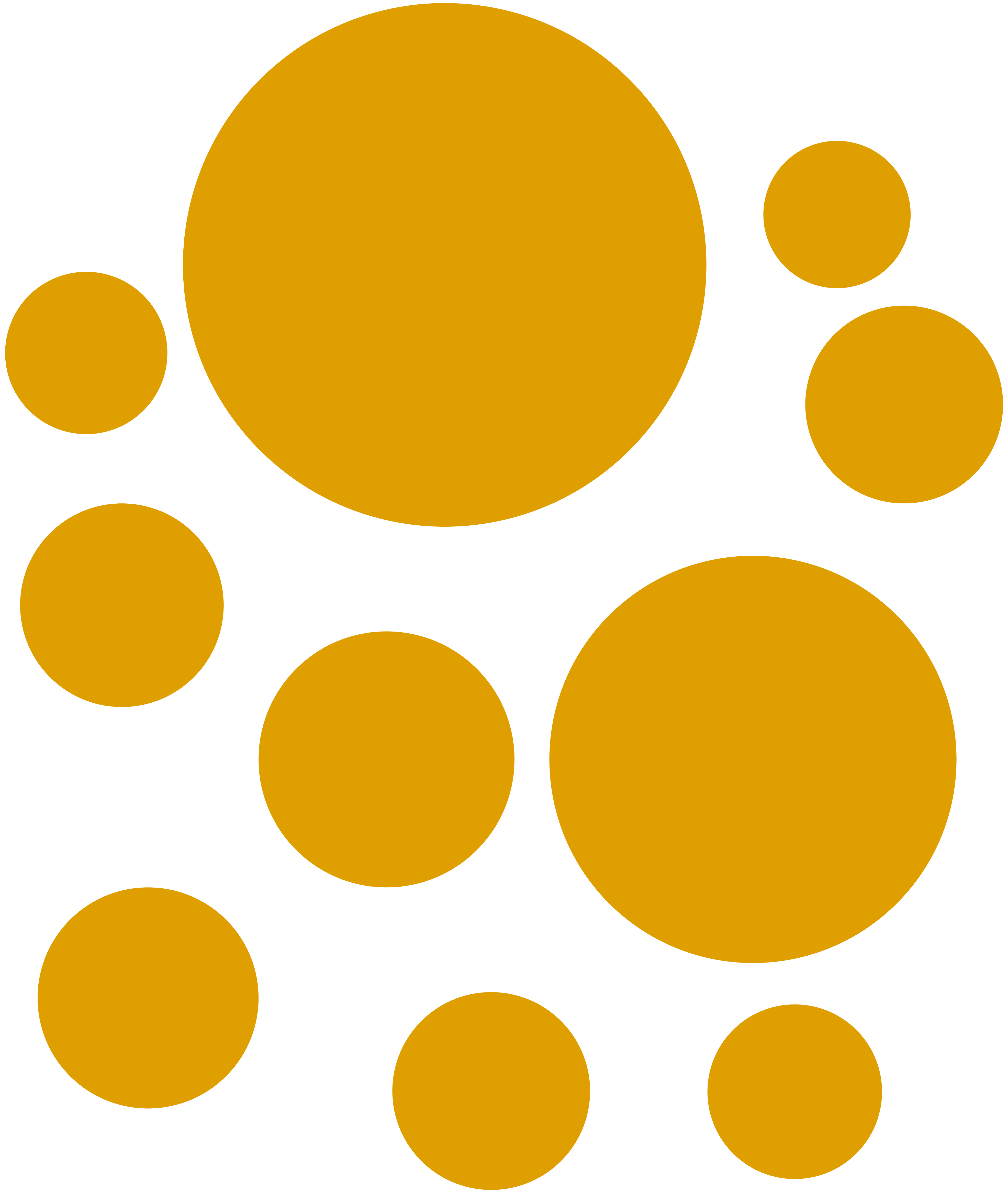
Draw teams at random from a productivity distribution



Simulating Developer Productivity

Draw teams at random from a productivity distribution

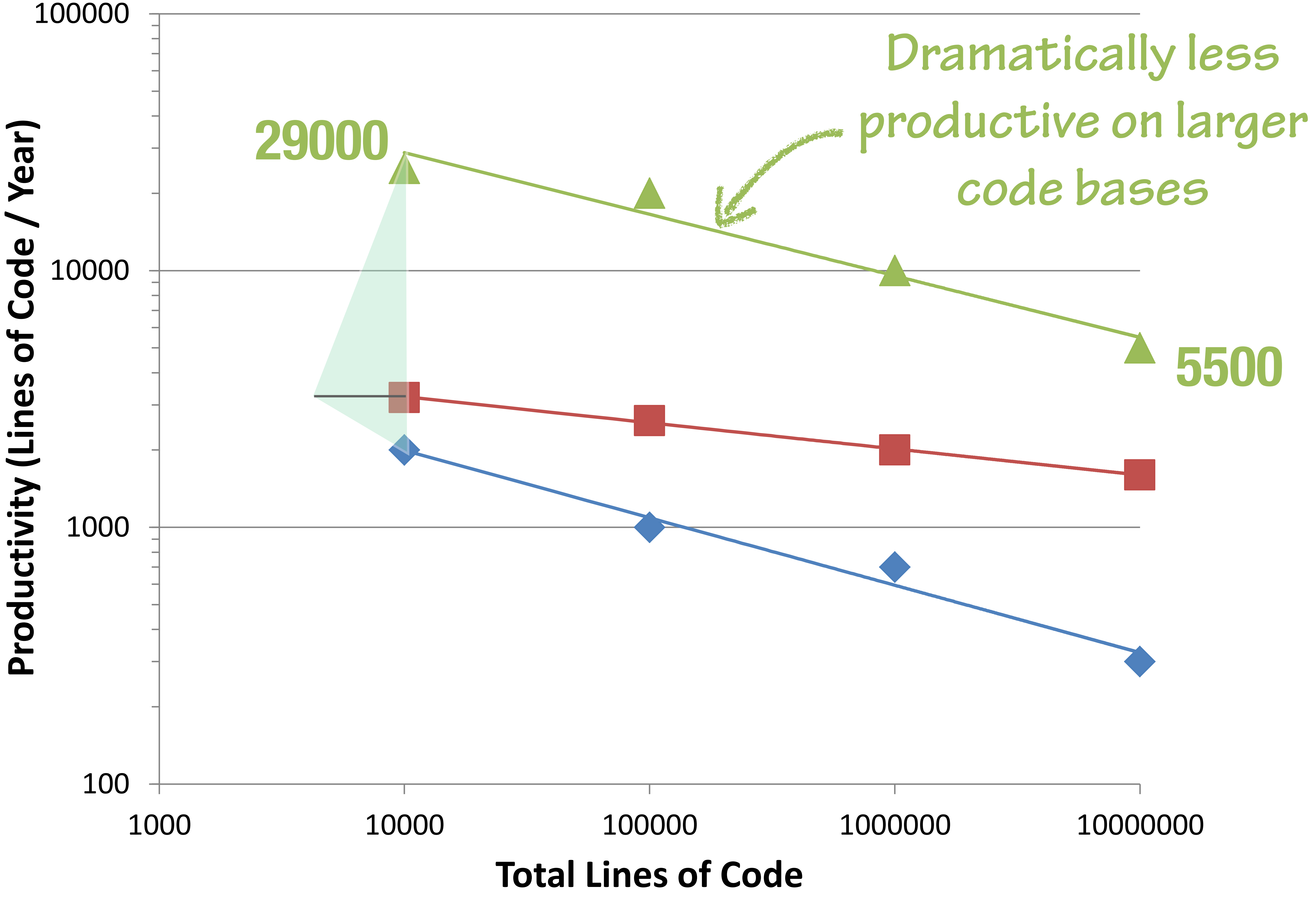




Modelling team and code evolution

Use published productivity data to forward model code size.

At any given system size we can predict a distribution for developer productivity.

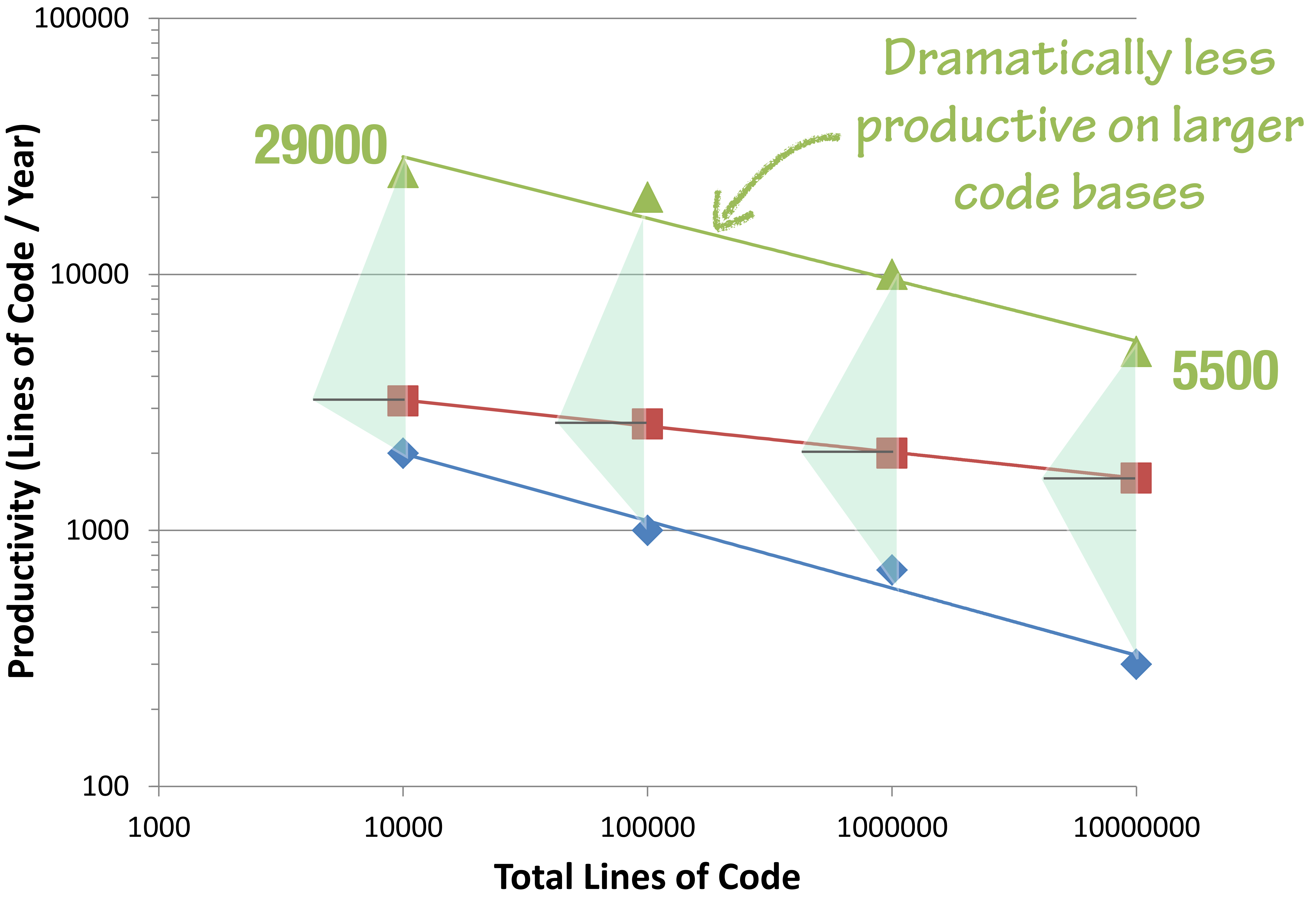


Sources: COCOMO II

Modelling team and code evolution

Use published productivity data to forward model code size.

At any given system size we can predict a distribution for developer productivity.

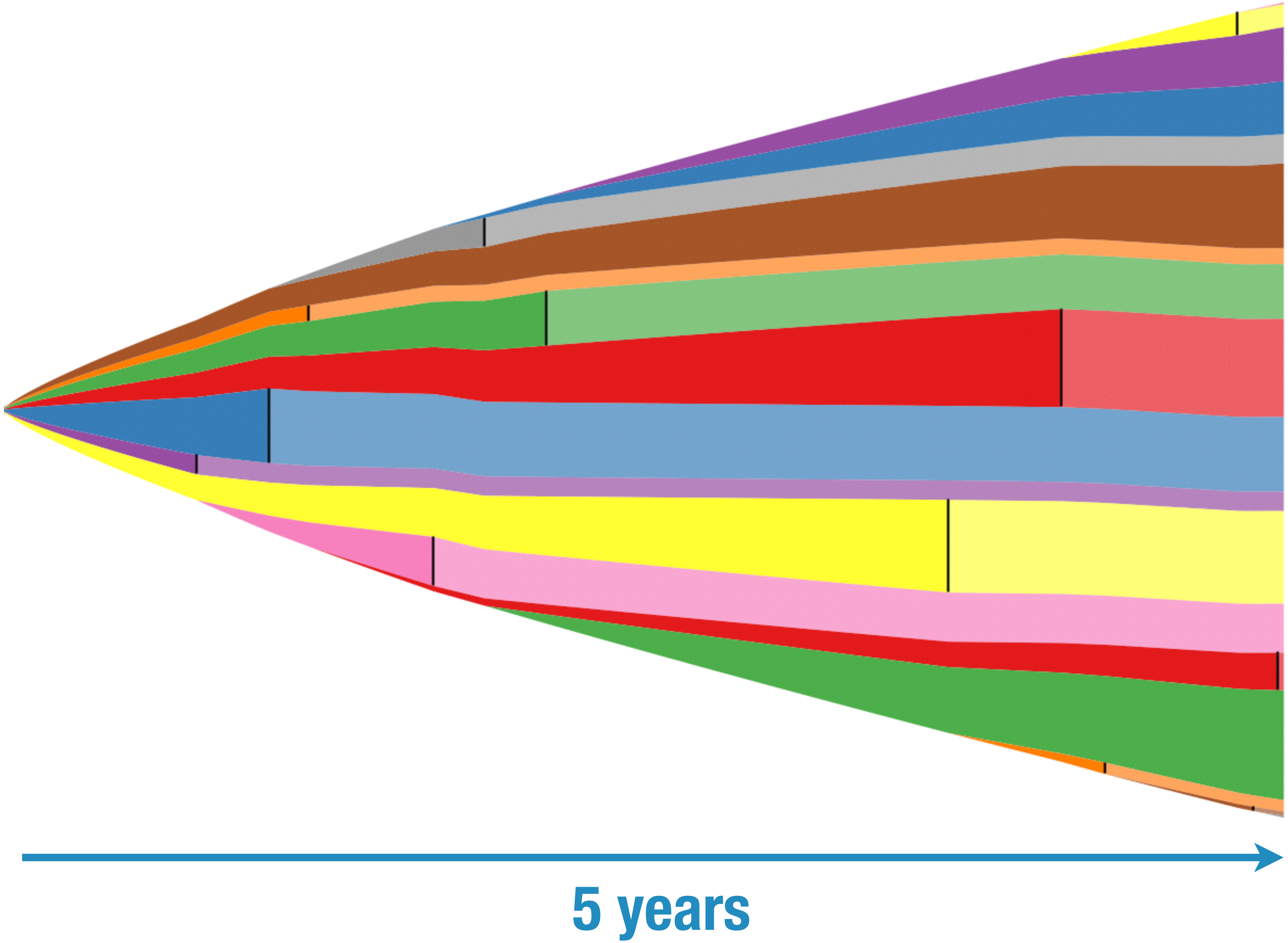


Sources: COCOMO II

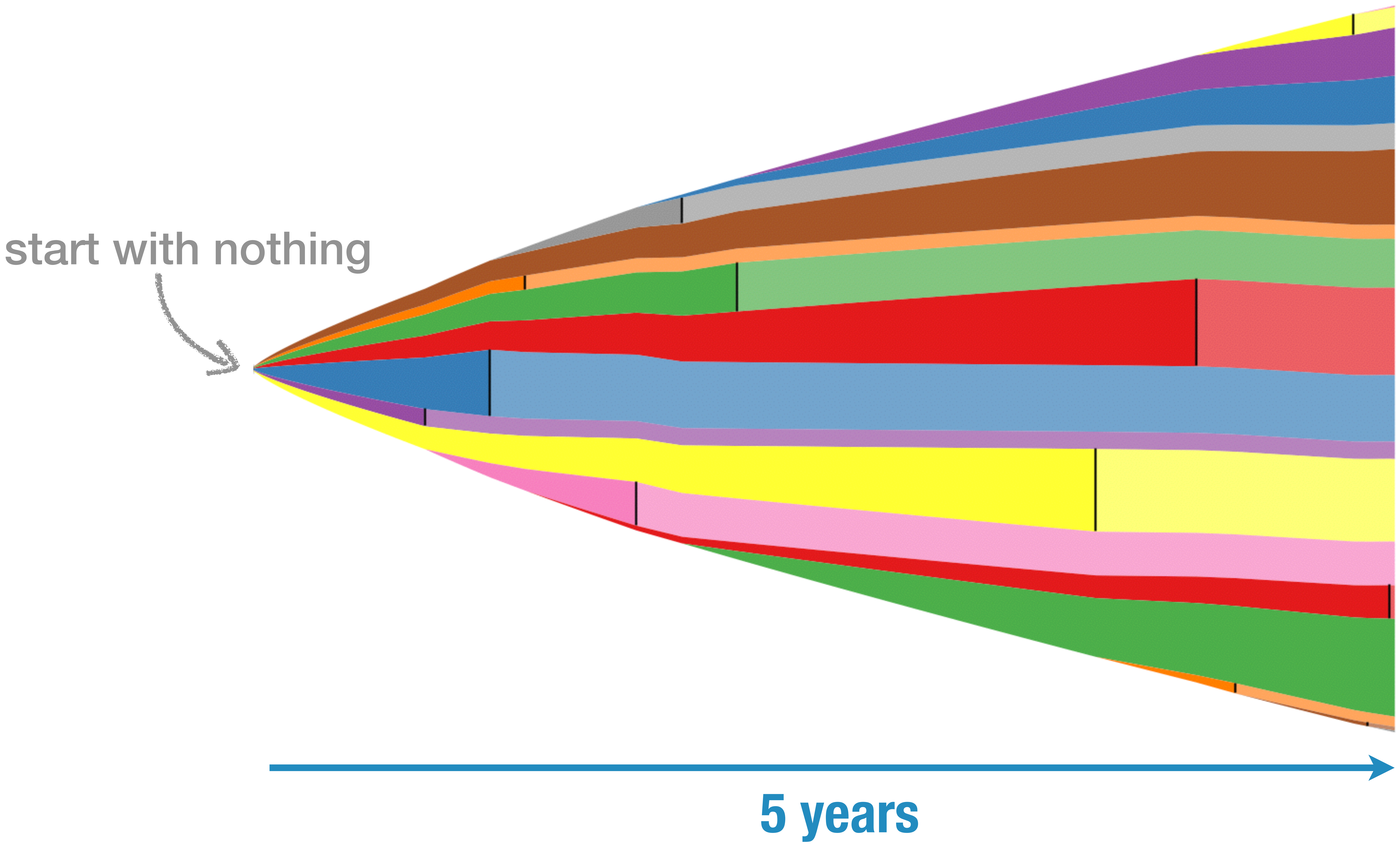
Simulating a team of seven over five years

5 years

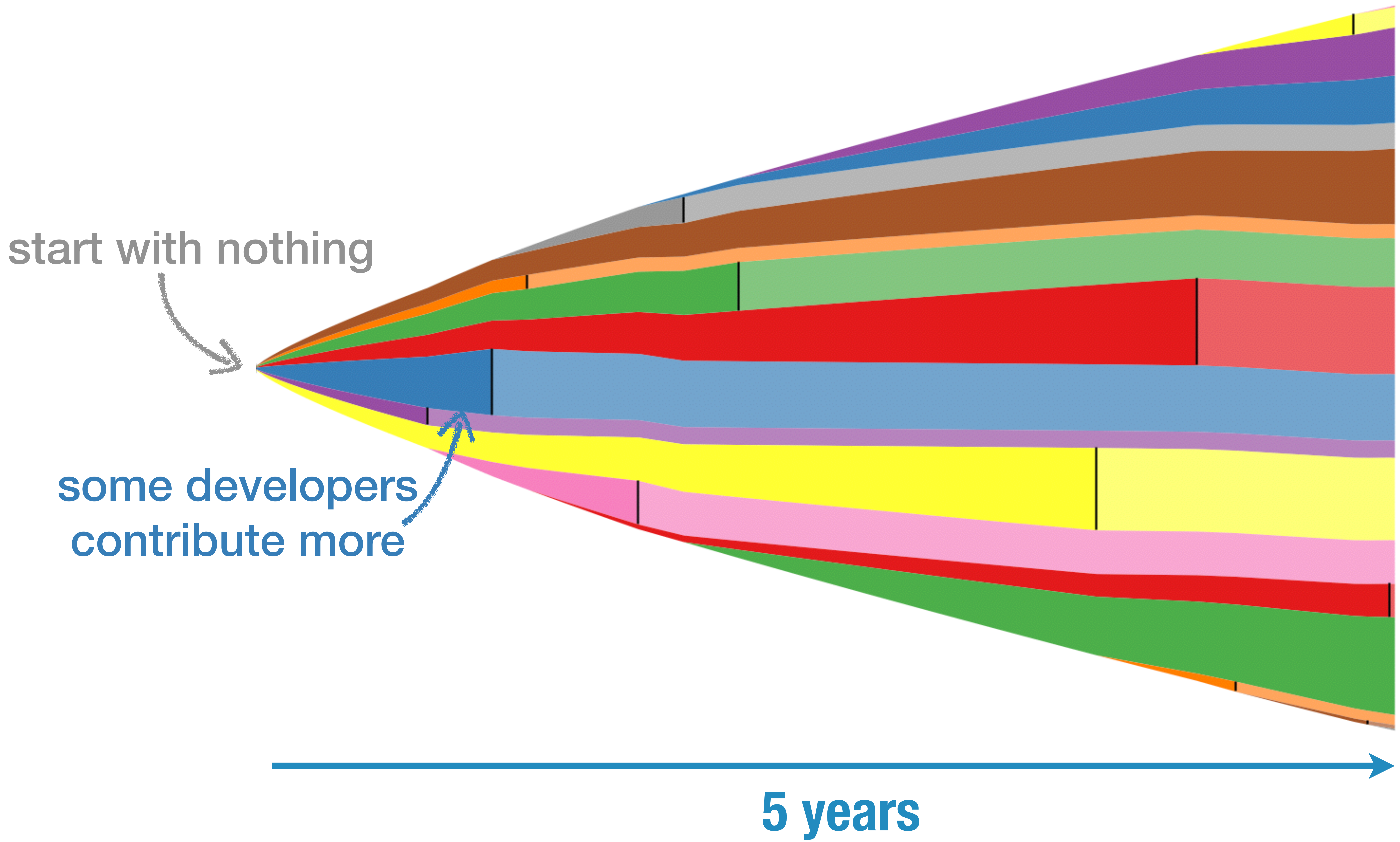
Simulating a team of seven over five years



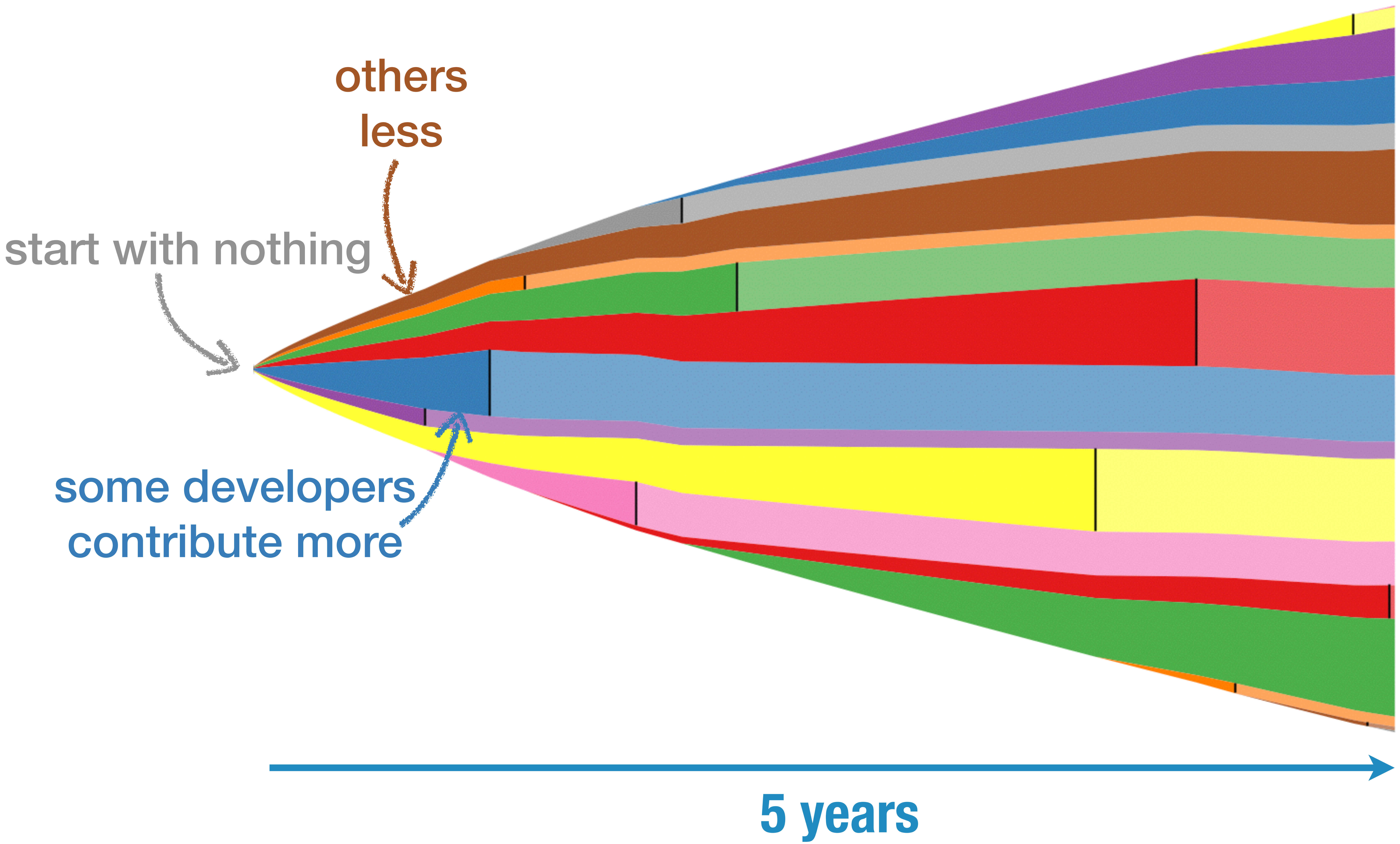
Simulating a team of seven over five years



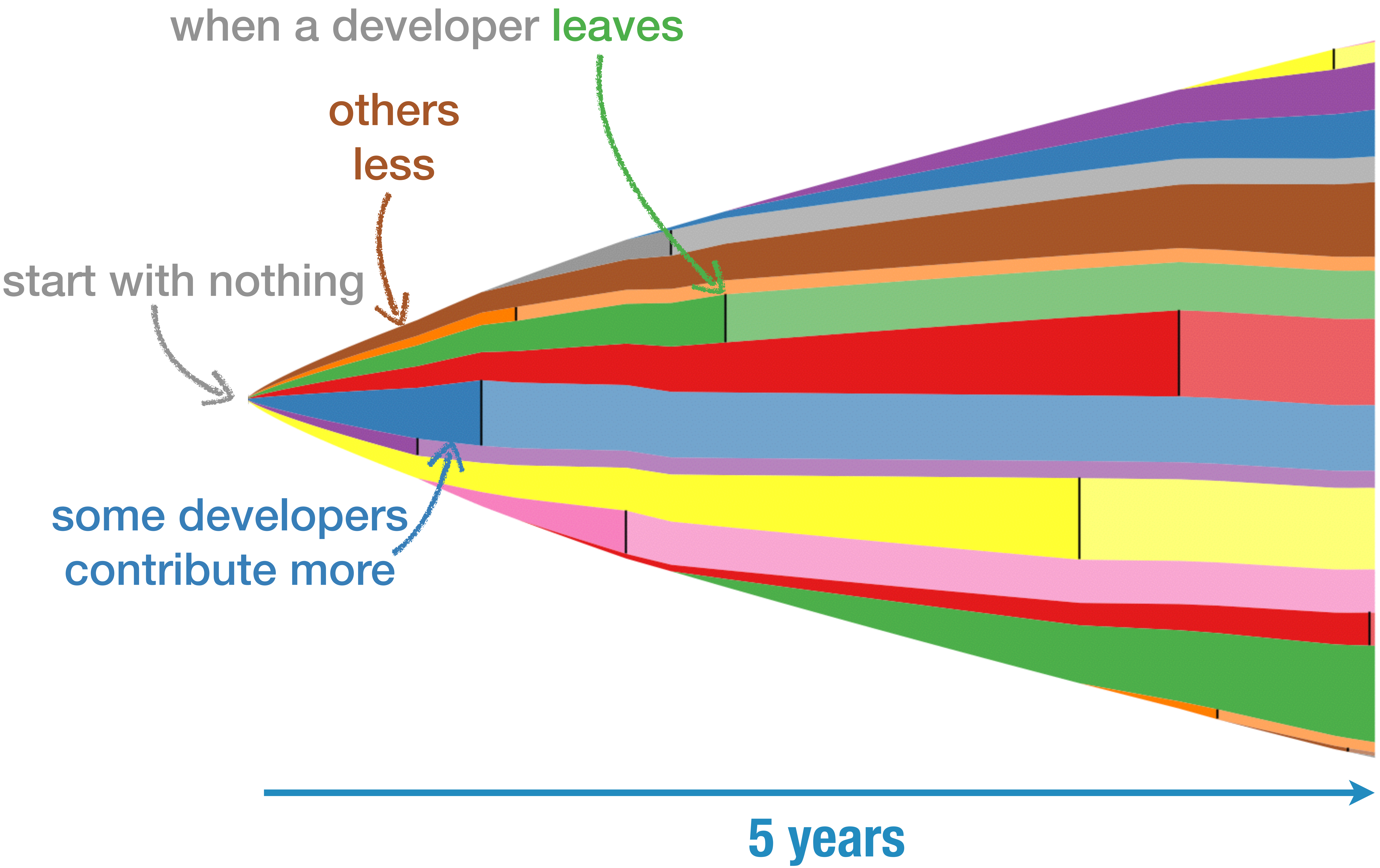
Simulating a team of seven over five years



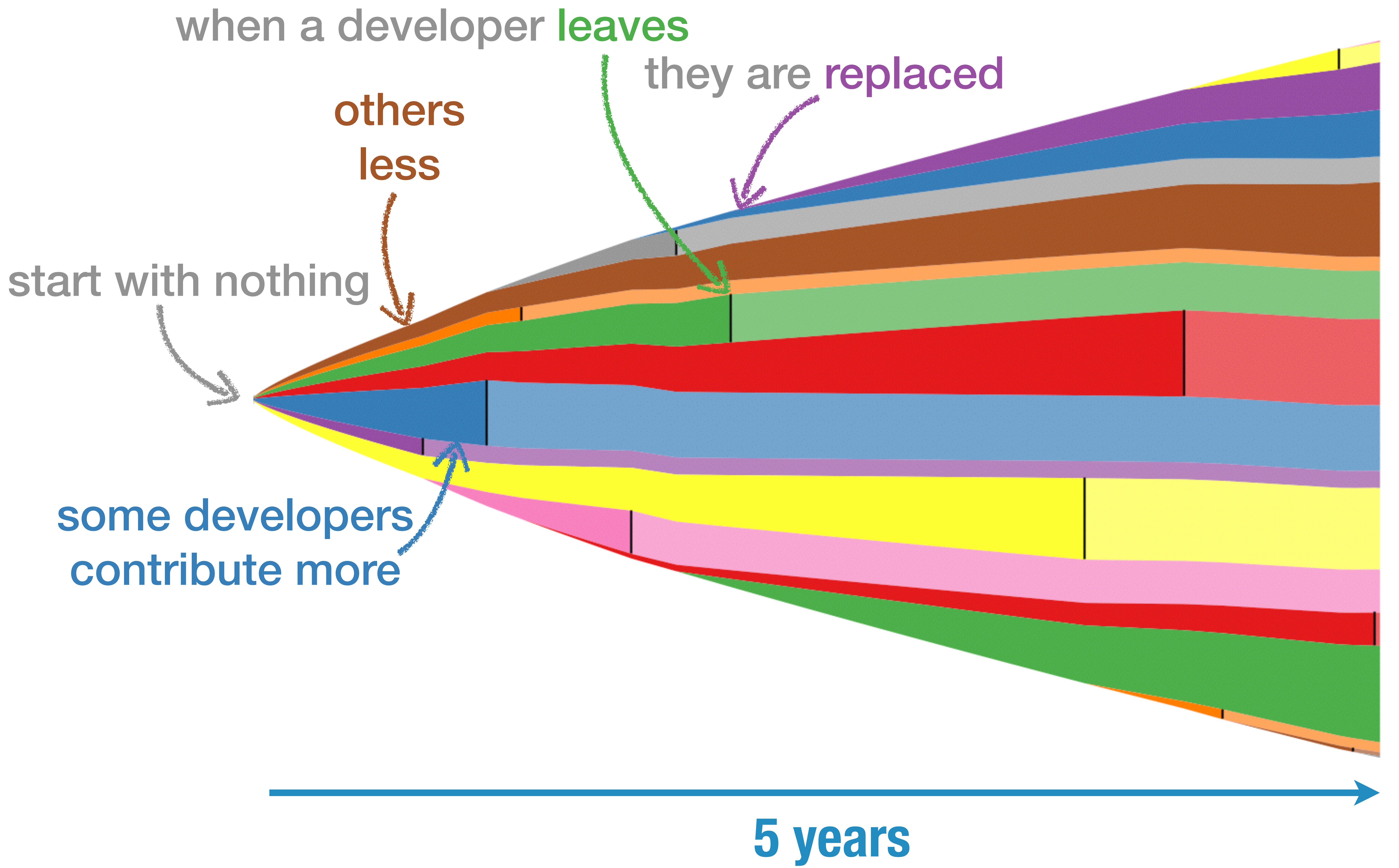
Simulating a team of seven over five years



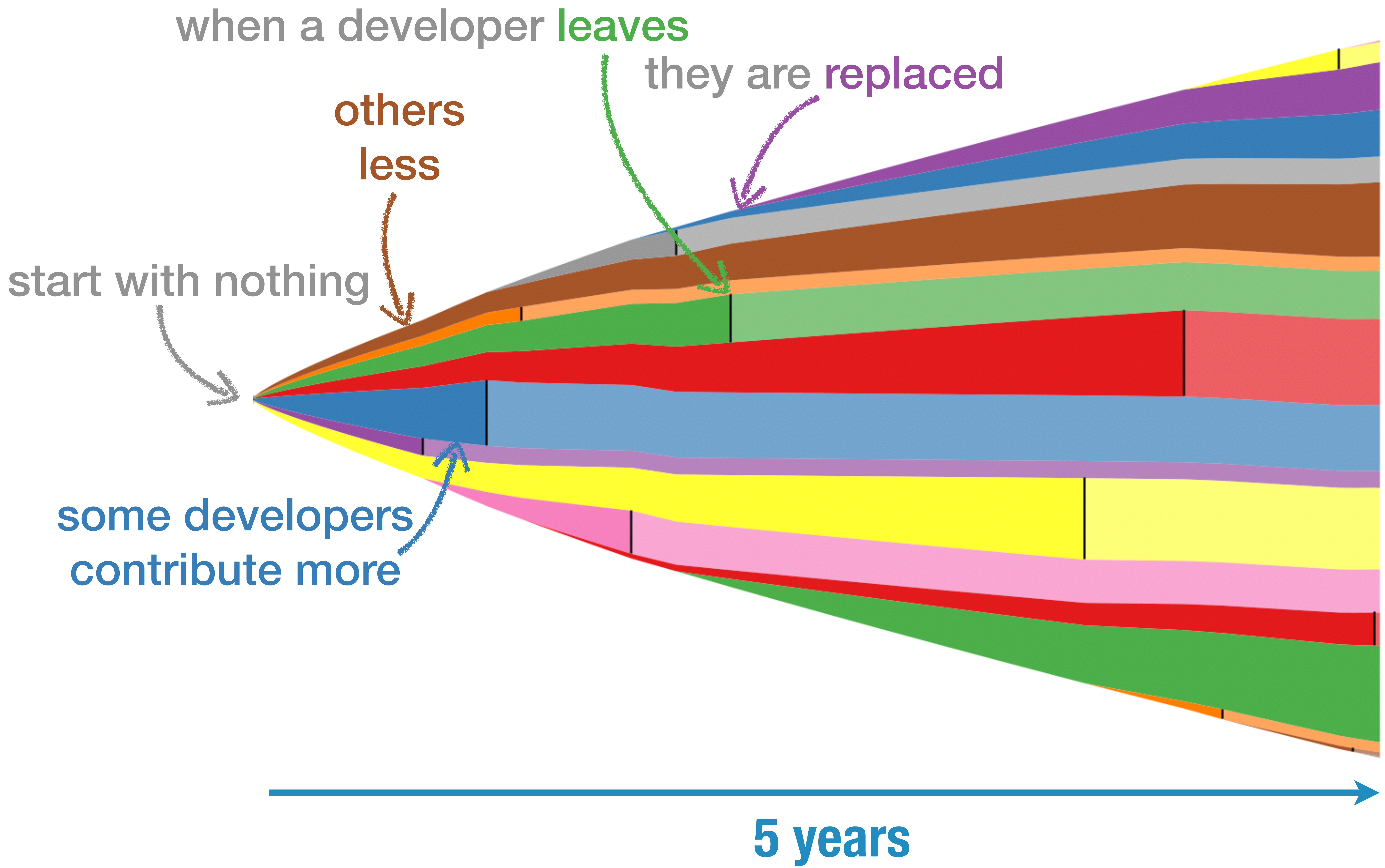
Simulating a team of seven over five years



Simulating a team of seven over five years



Simulating a team of seven over five years



After 5 years we have 235 k lines of code written by a total of 19 people.

Only 37% of the code is by current team


02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10


02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10
02:35:33:10

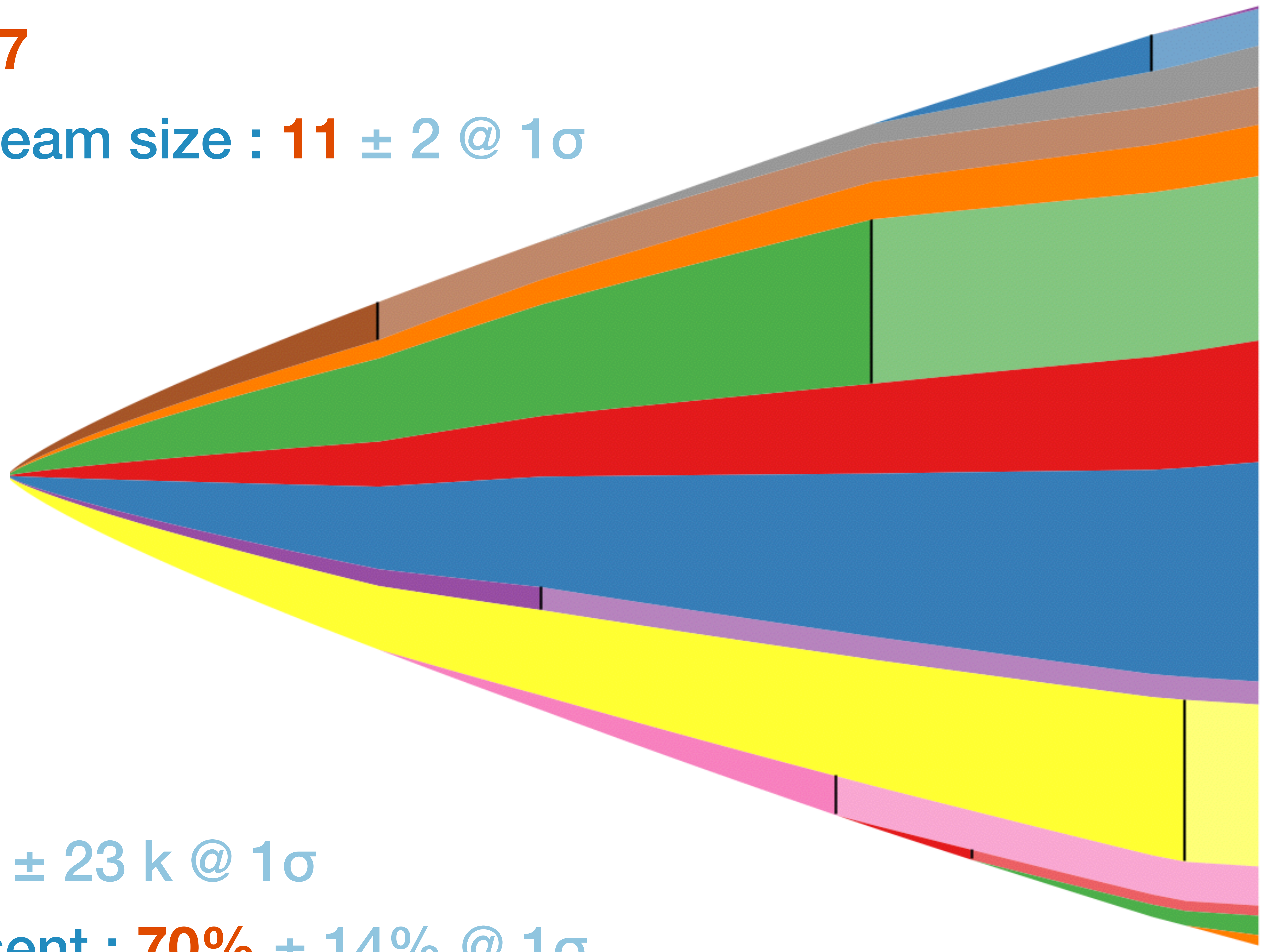
3 years

Team Size : **7**

3 years

Team Size : 7

Cumulative team size : 11 ± 2 @ 1σ



157 kLoC

LoC : 157 k ± 23 k @ 1σ

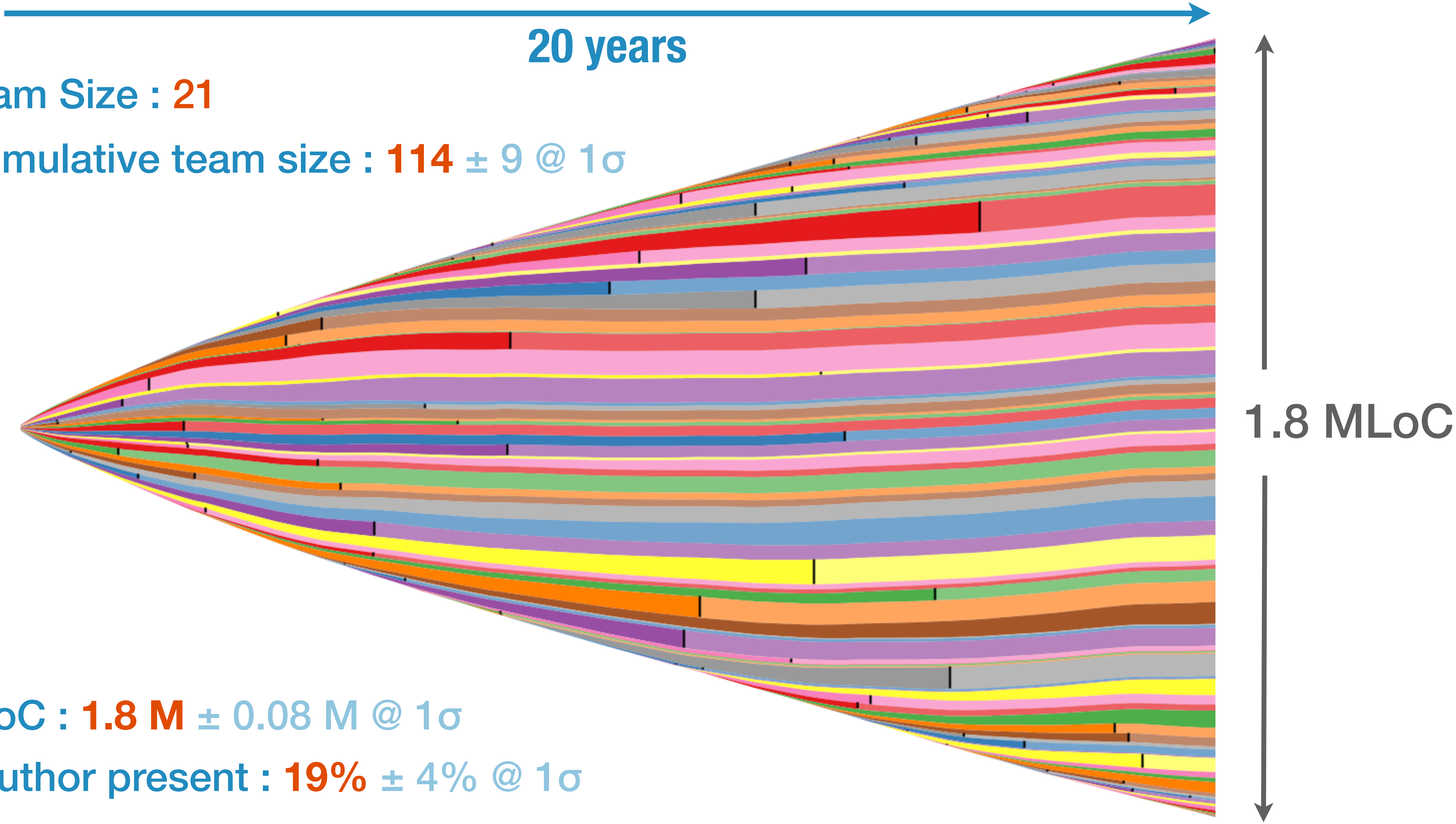
Author present : 70% ± 14% @ 1σ

20 years

Team Size : 21

Team Size : 21

Cumulative team size : 114 ± 9 @ 1σ

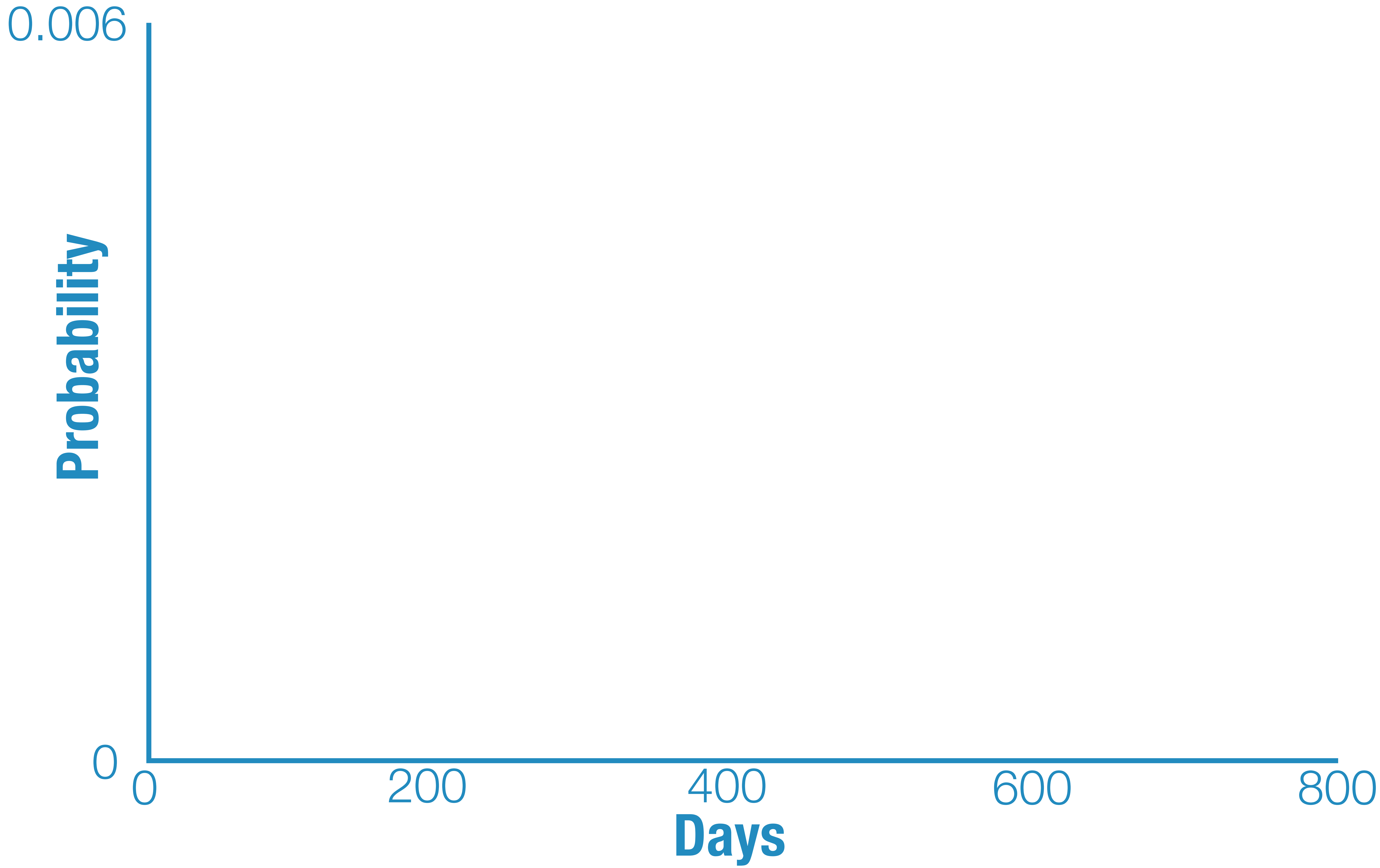


LoC : 1.8 M ± 0.08 M @ 1σ

Author present : 19% ± 4% @ 1σ

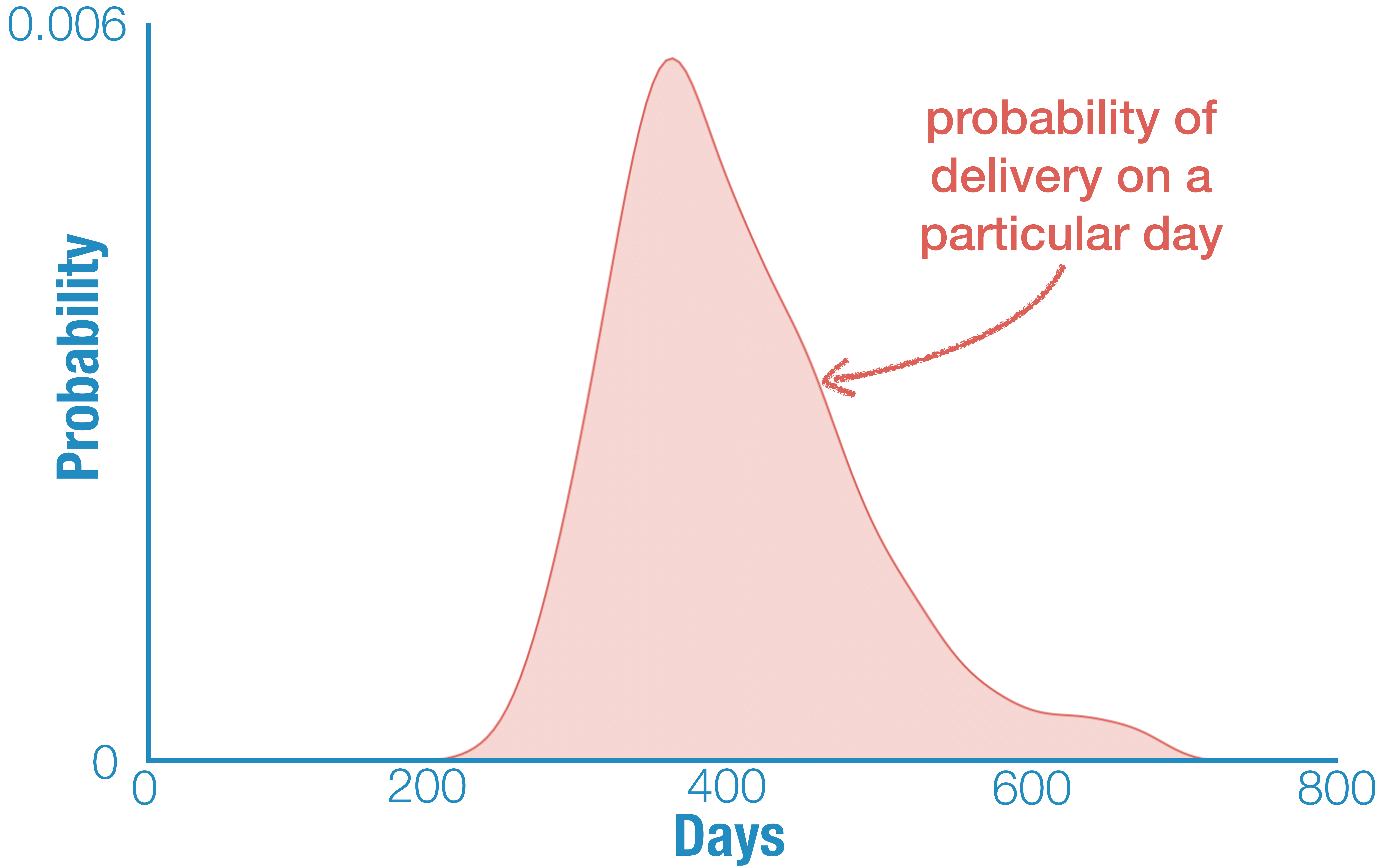
How long for seven to produce 100 000 lines of code?

Probability density from 1000 simulations



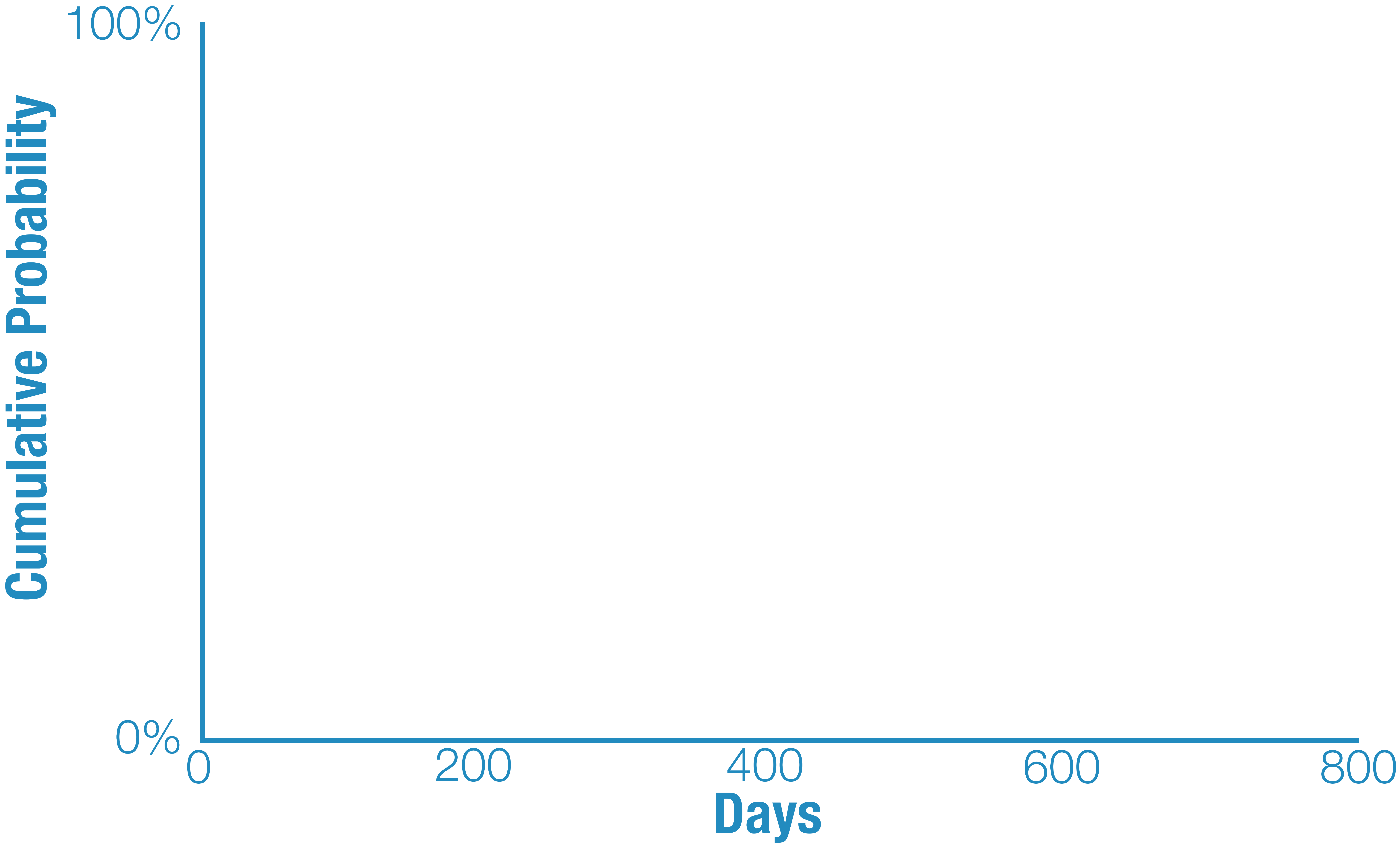
How long for seven to produce 100 000 lines of code?

Probability density from 1000 simulations



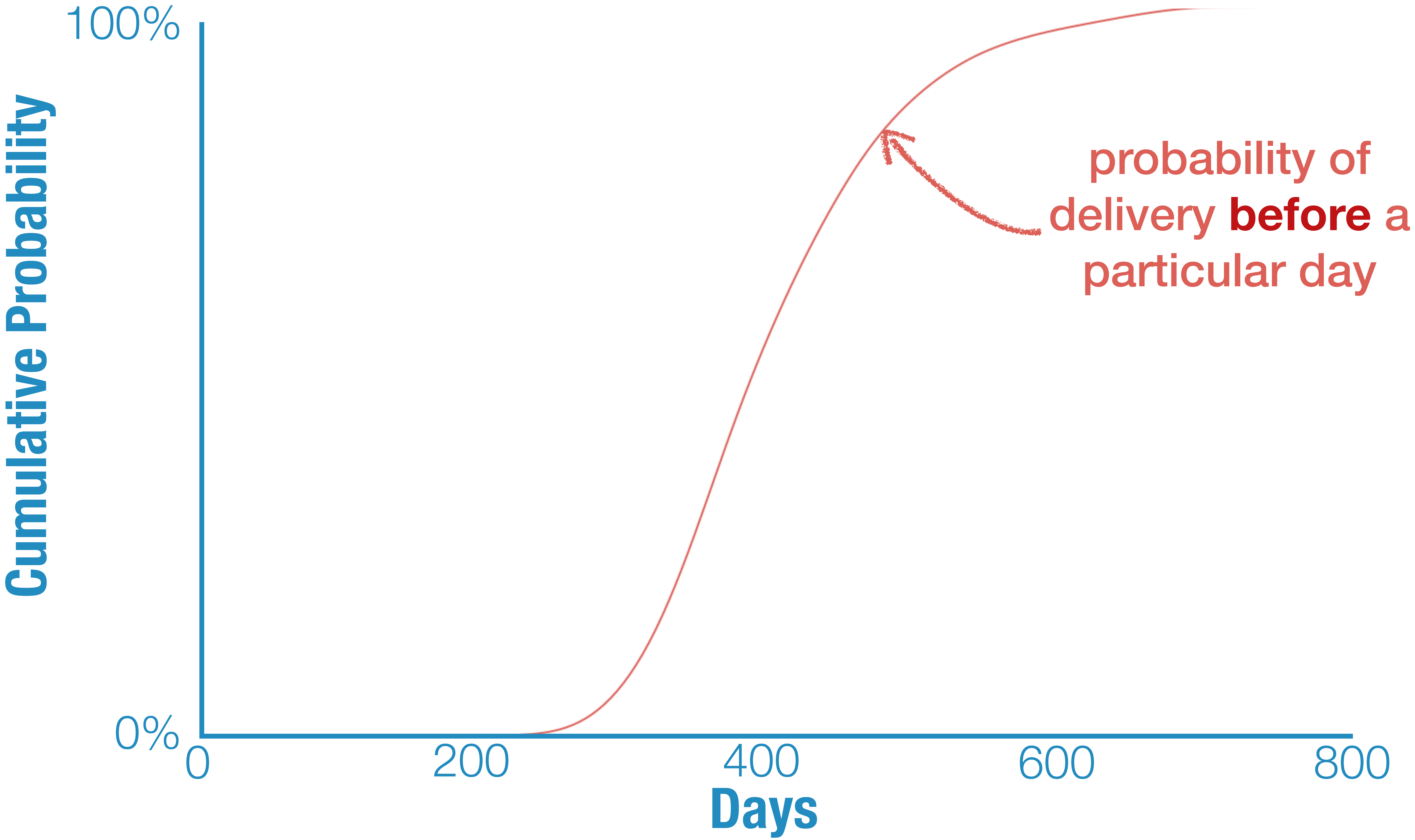
How long for 7 to produce 100 000 lines of code?

Cumulative probability from 1000 simulations



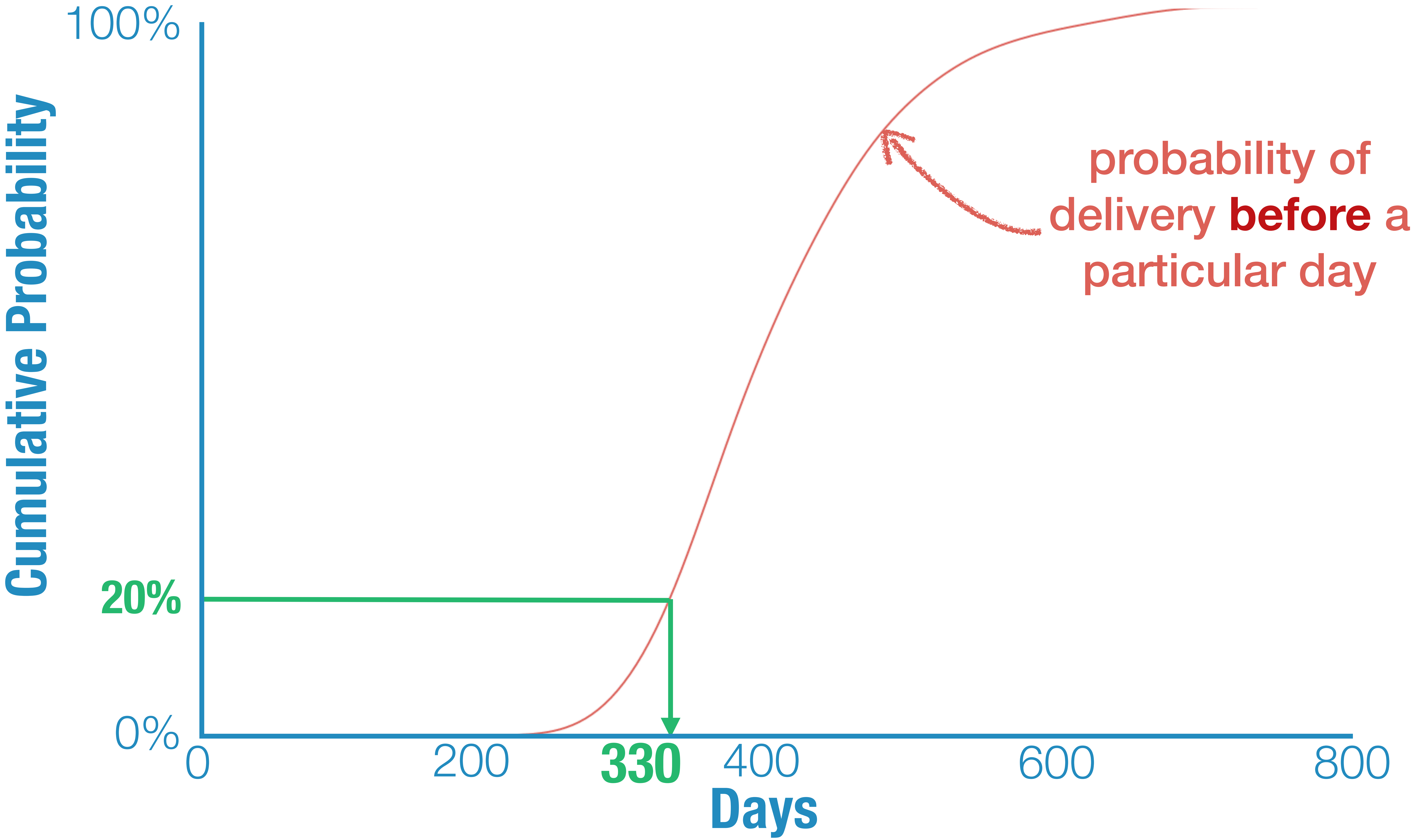
How long for 7 to produce 100 000 lines of code?

Cumulative probability from 1000 simulations



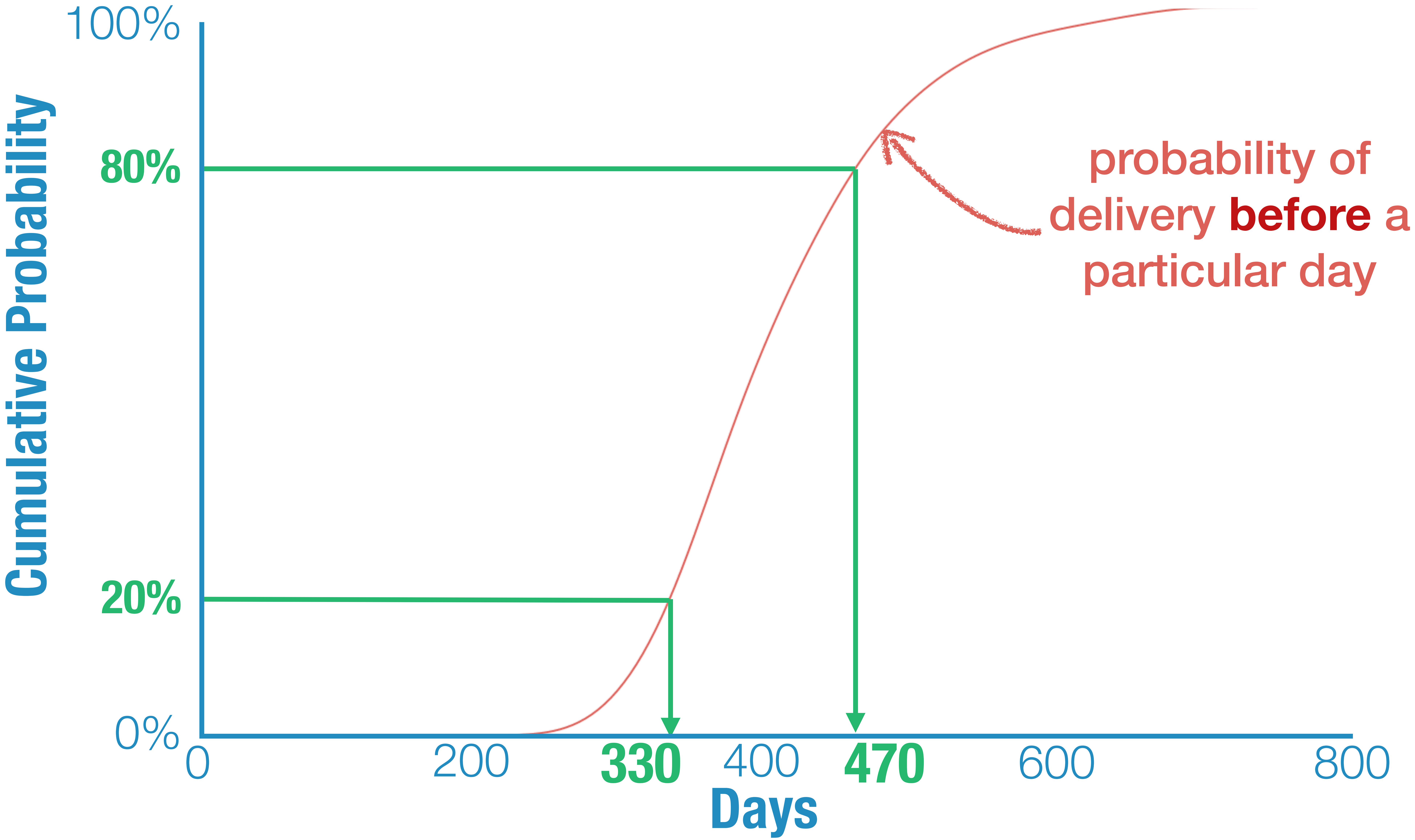
How long for 7 to produce 100 000 lines of code?

Cumulative probability from 1000 simulations



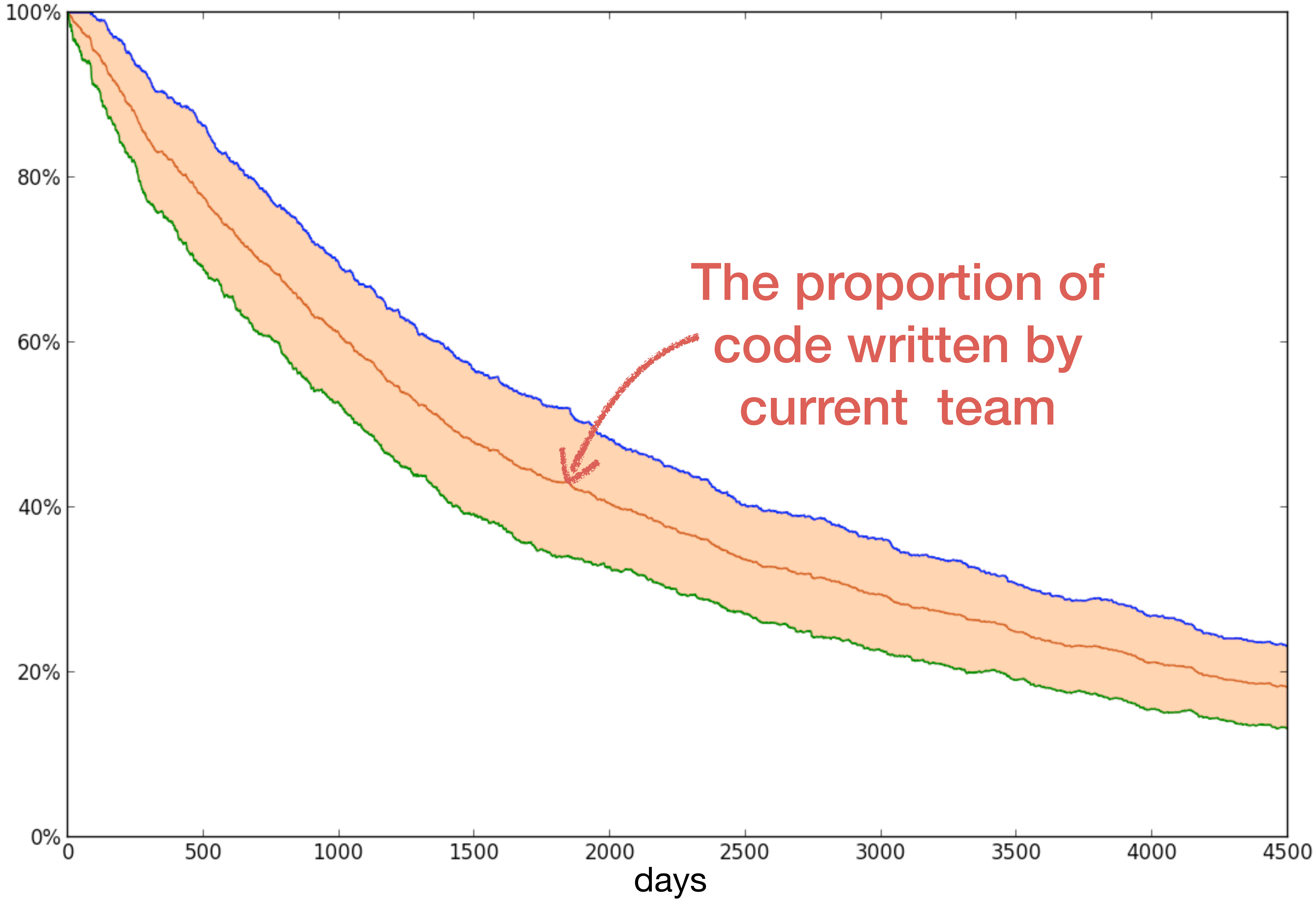
How long for 7 to produce 100 000 lines of code?

Cumulative probability from 1000 simulations



Who can you still talk to?

Most authors of your product quit way back when



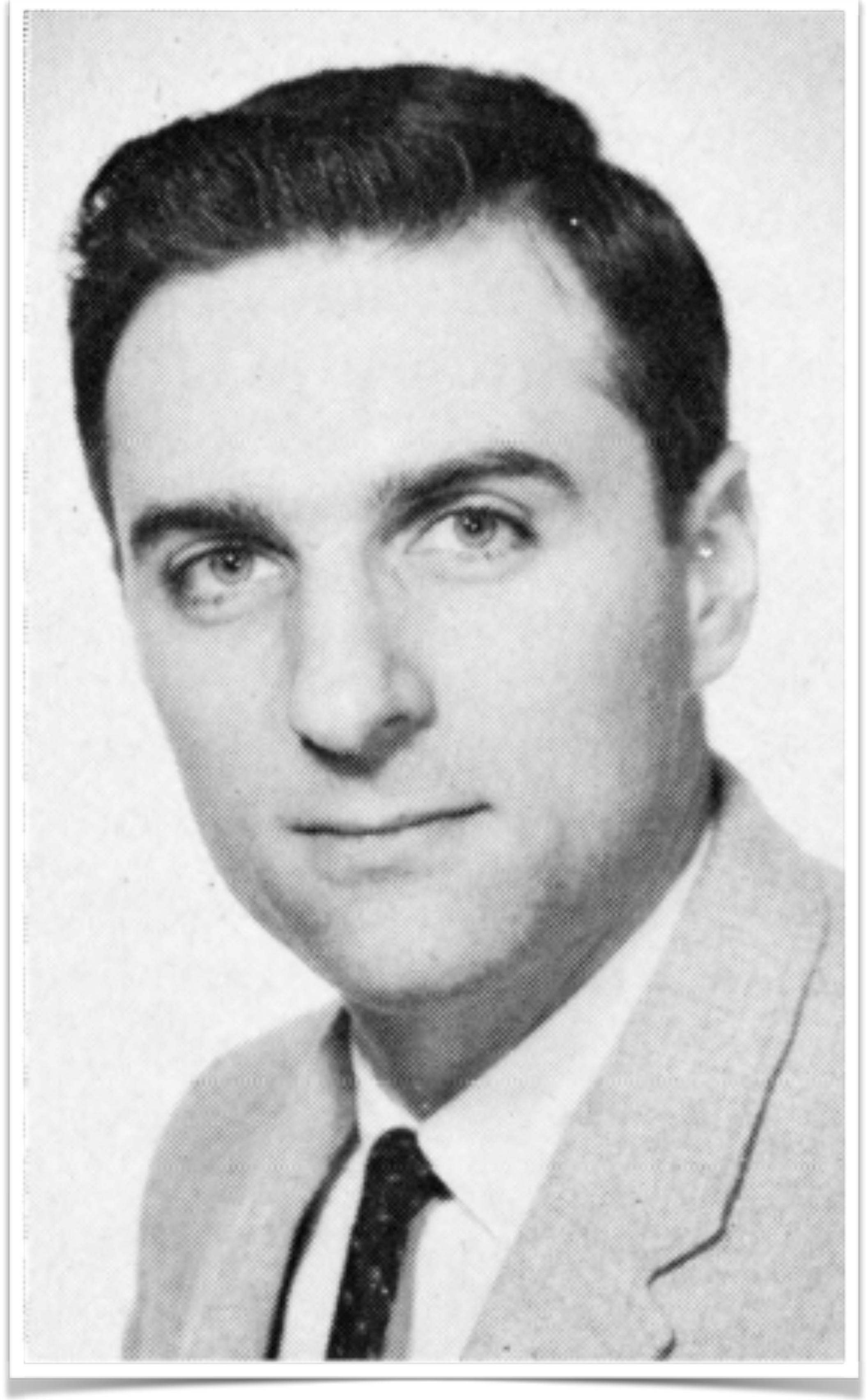
20% after 20 years

Conway's Law

from the 1968 paper *How do committees invent?*

"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure"

integrated over time



Melvin Conway

Modelling system growth 1

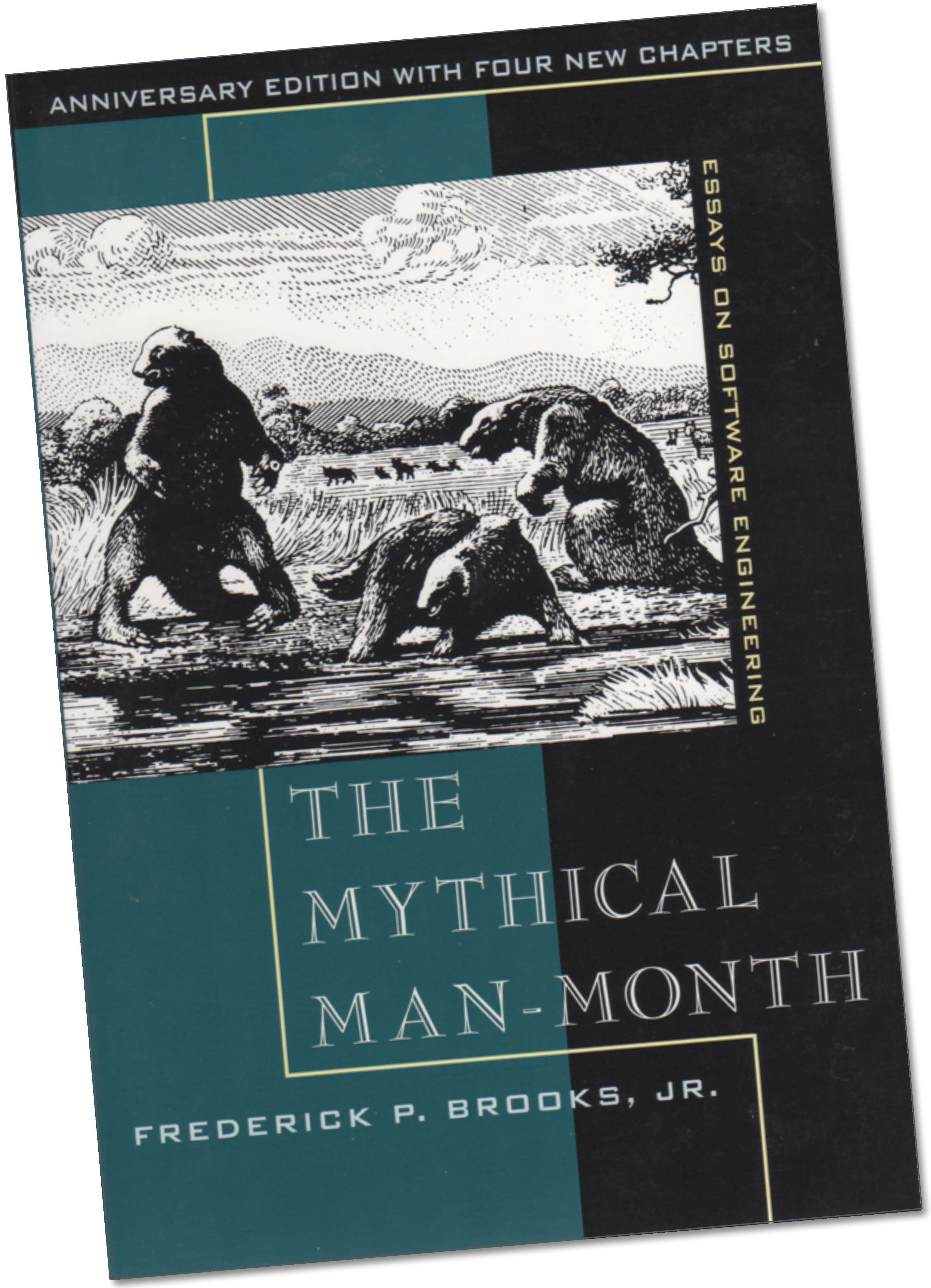
How many people work on your system?

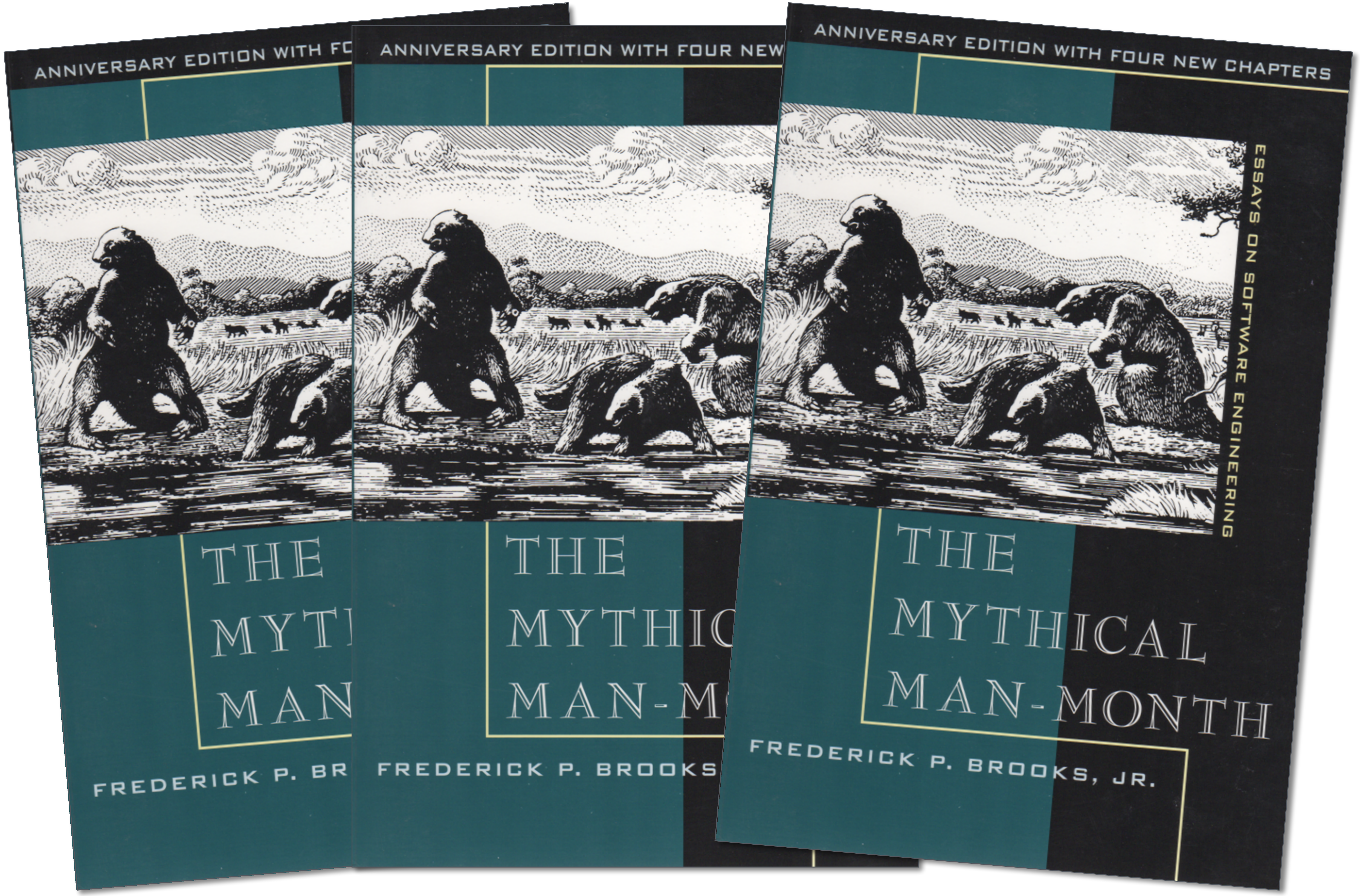
Predicting project progress 2

How many people should work on your system?

Software process dynamics 3

How can you construct models and run simulations?


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.

Charles R Knight (1921) *Rancho la Brea Tar Pool*

“Adding manpower to a late software project makes it later.”

Fred Brooks / The Mythical Man-Month

Wikimedia Commons

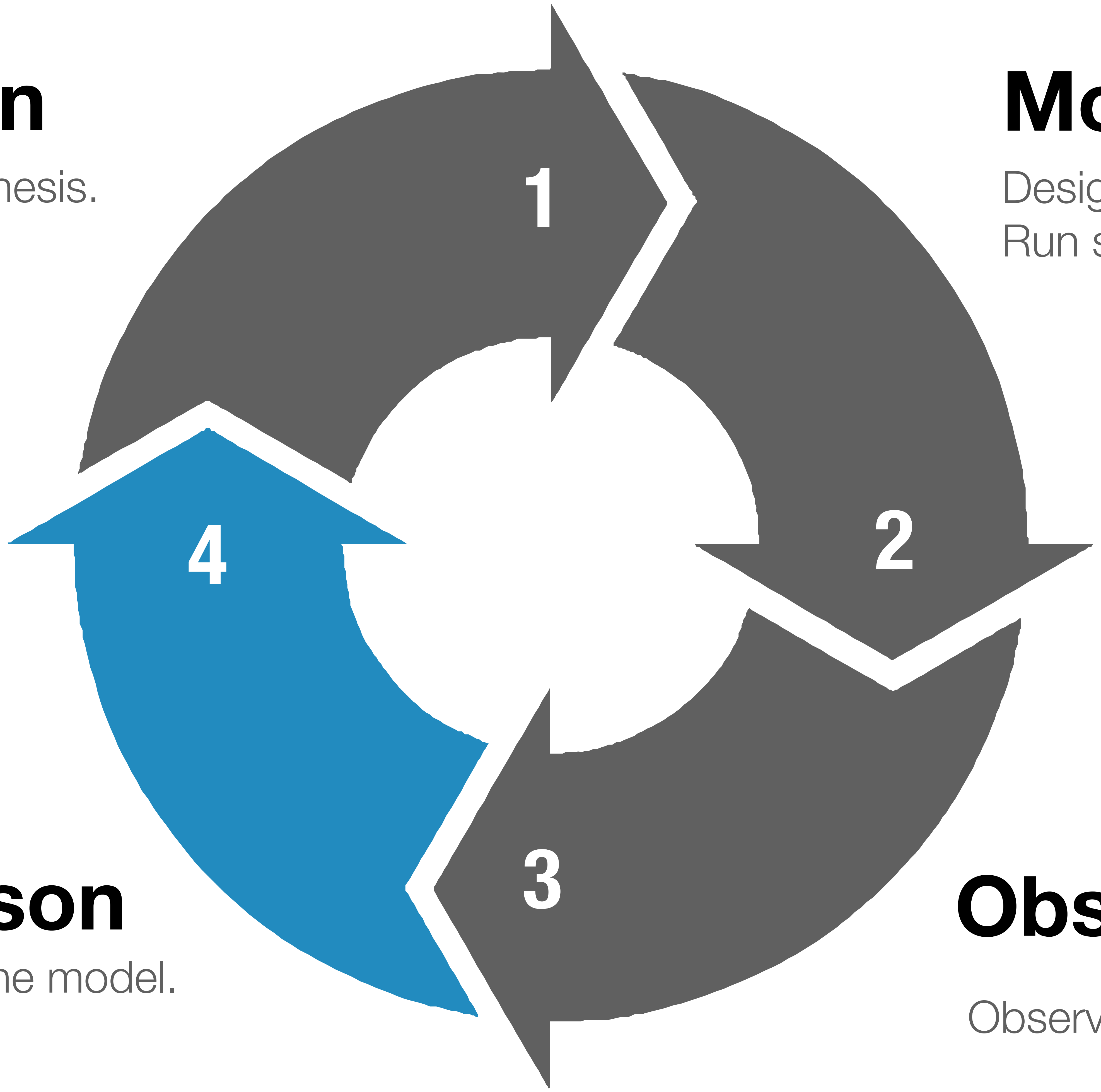
How can we know?

Prediction

Formulate a hypothesis.

Modelling

Design a conceptual model.
Run simulations.



Comparison

Validate or refute the model.

Observation

Observe and record reality.

System dynamics simulations

Model systems for improving structures, policies and interventions

- Define problem dynamically – over time
- Endogenous view of significant dynamics
- Model reproduces problem of concern
- Derive understanding

Discrete versus continuous modelling

Events or equations?

Discrete versus continuous modelling

Events or equations?

Discrete

- Individuals
- Populations
- Definite events
- Probability distributions
- Stochastic
- Concrete scenarios
- Harder to formulate as code

Discrete versus continuous modelling

Events or equations?

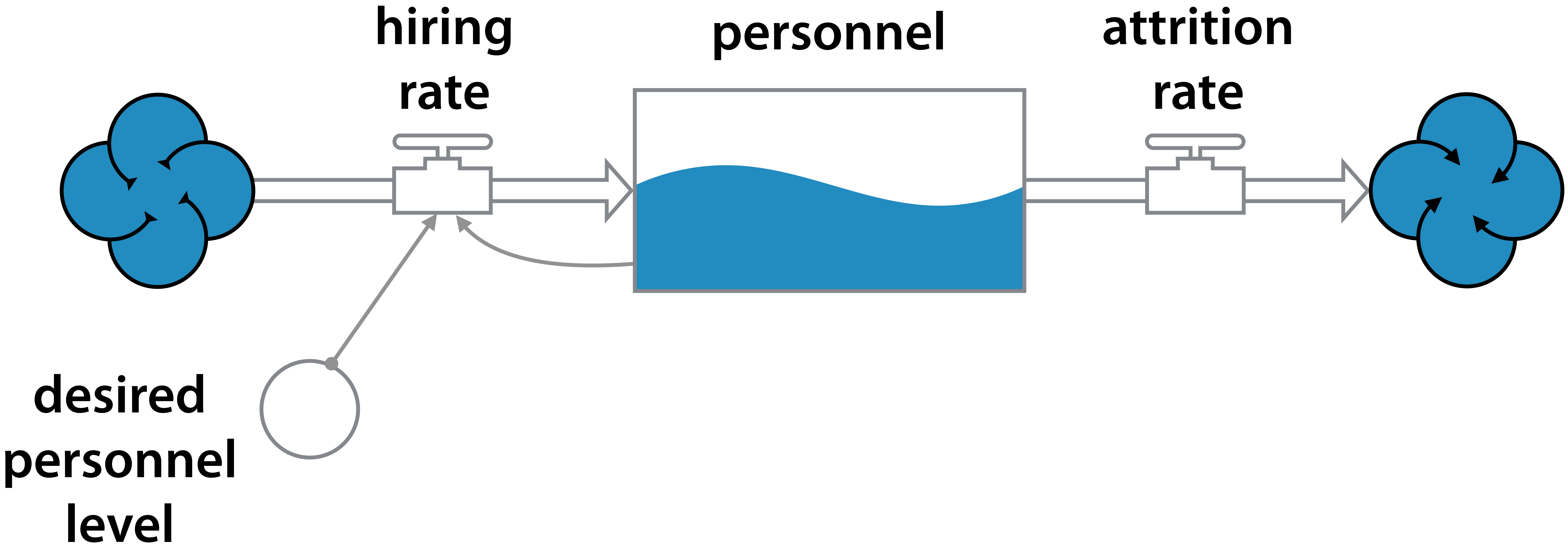
Discrete

- Individuals
- Populations
- Definite events
- Probability distributions
- Stochastic
- Concrete scenarios
- Harder to formulate as code

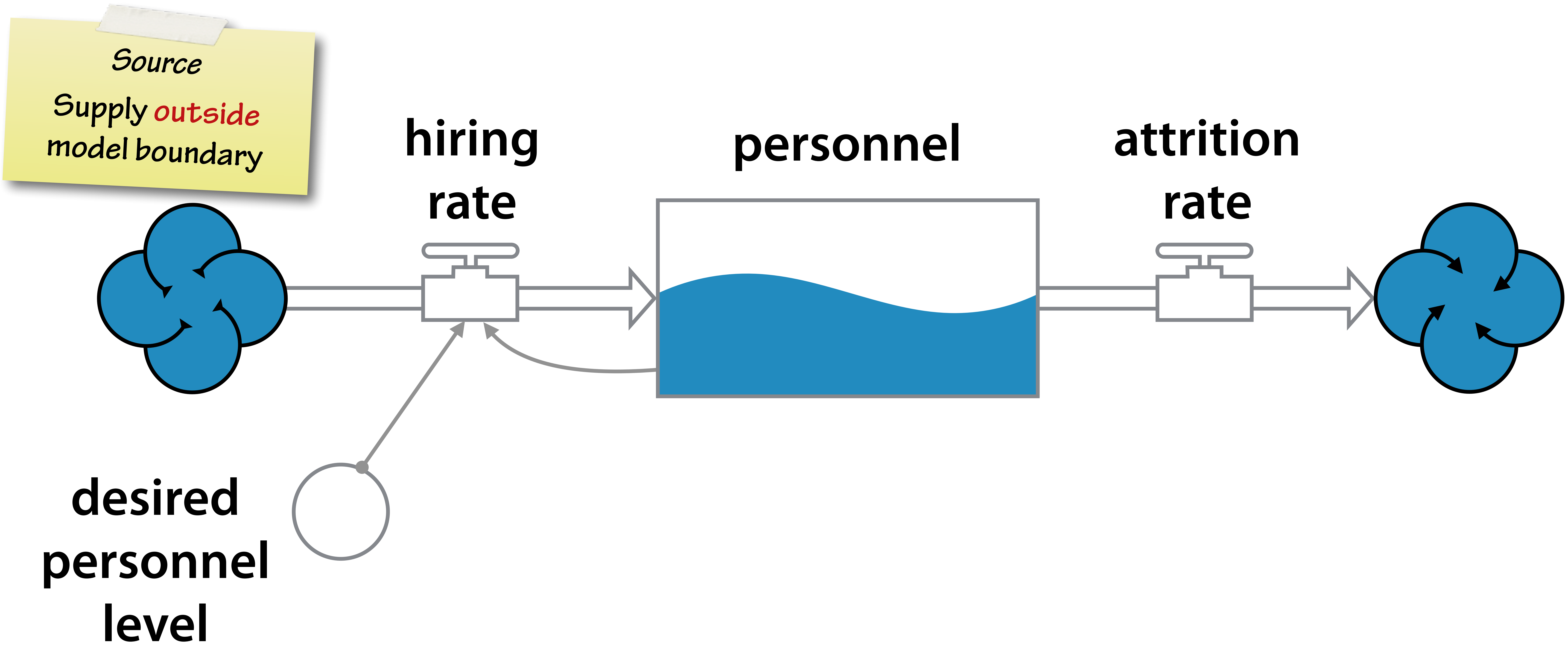
Continuous

- Aggregates
- Levels of quantities
- Flow rates
- Equations
- Numerical / analytical solutions
- More abstract
- Easier to formulate as code

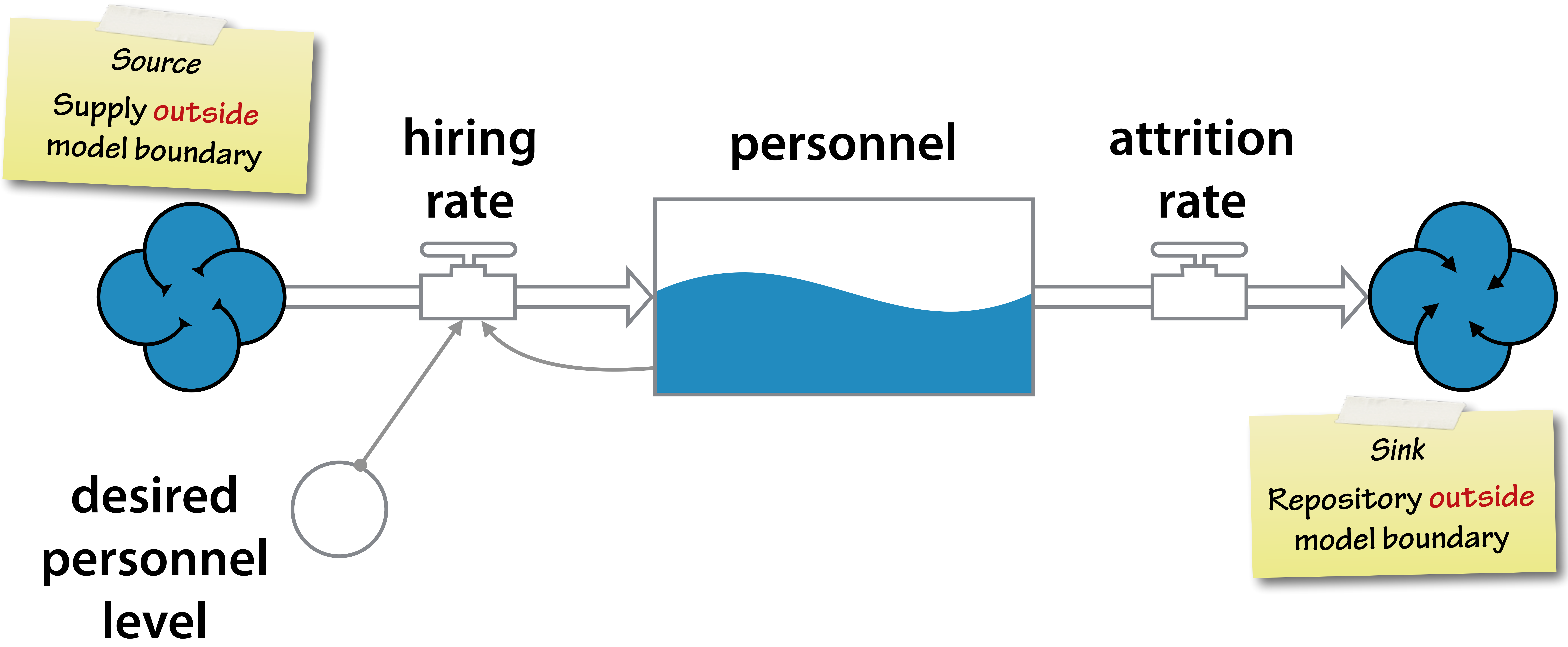
Elements of continuous models



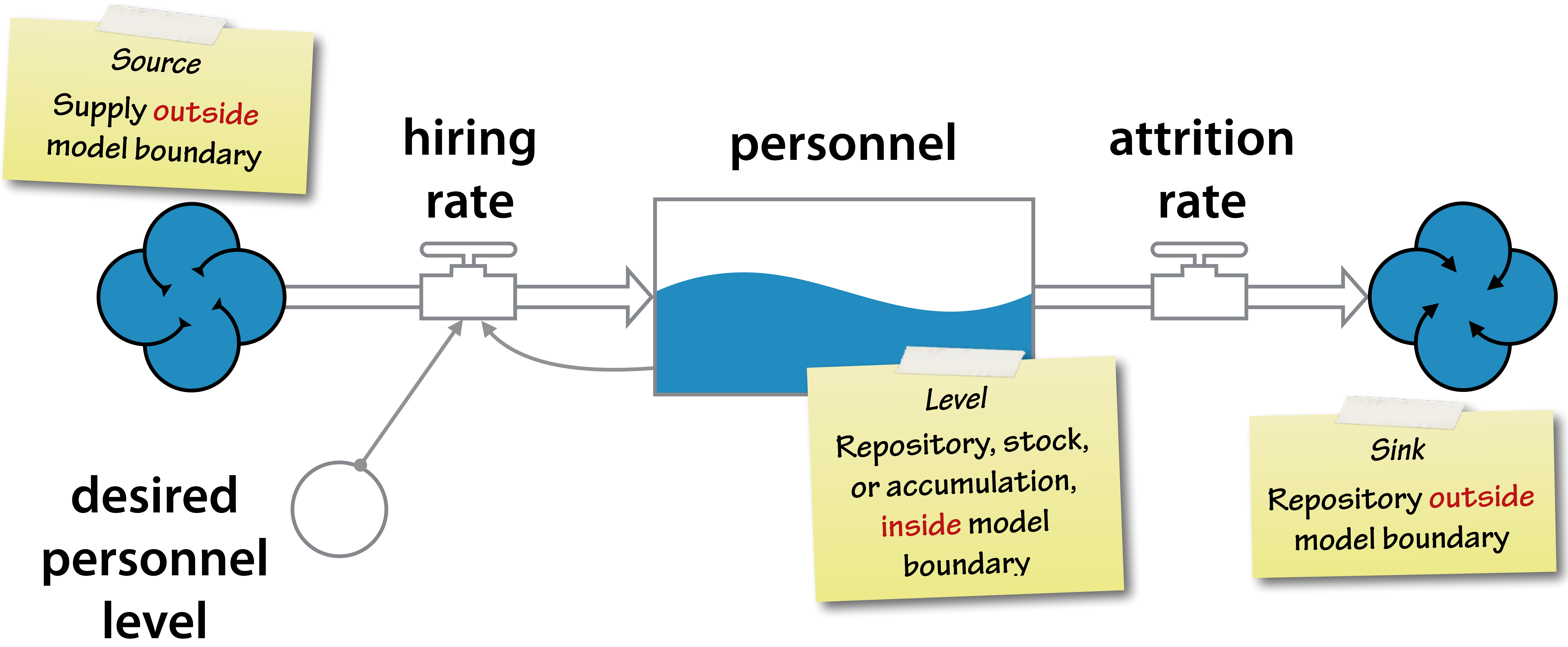
Elements of continuous models



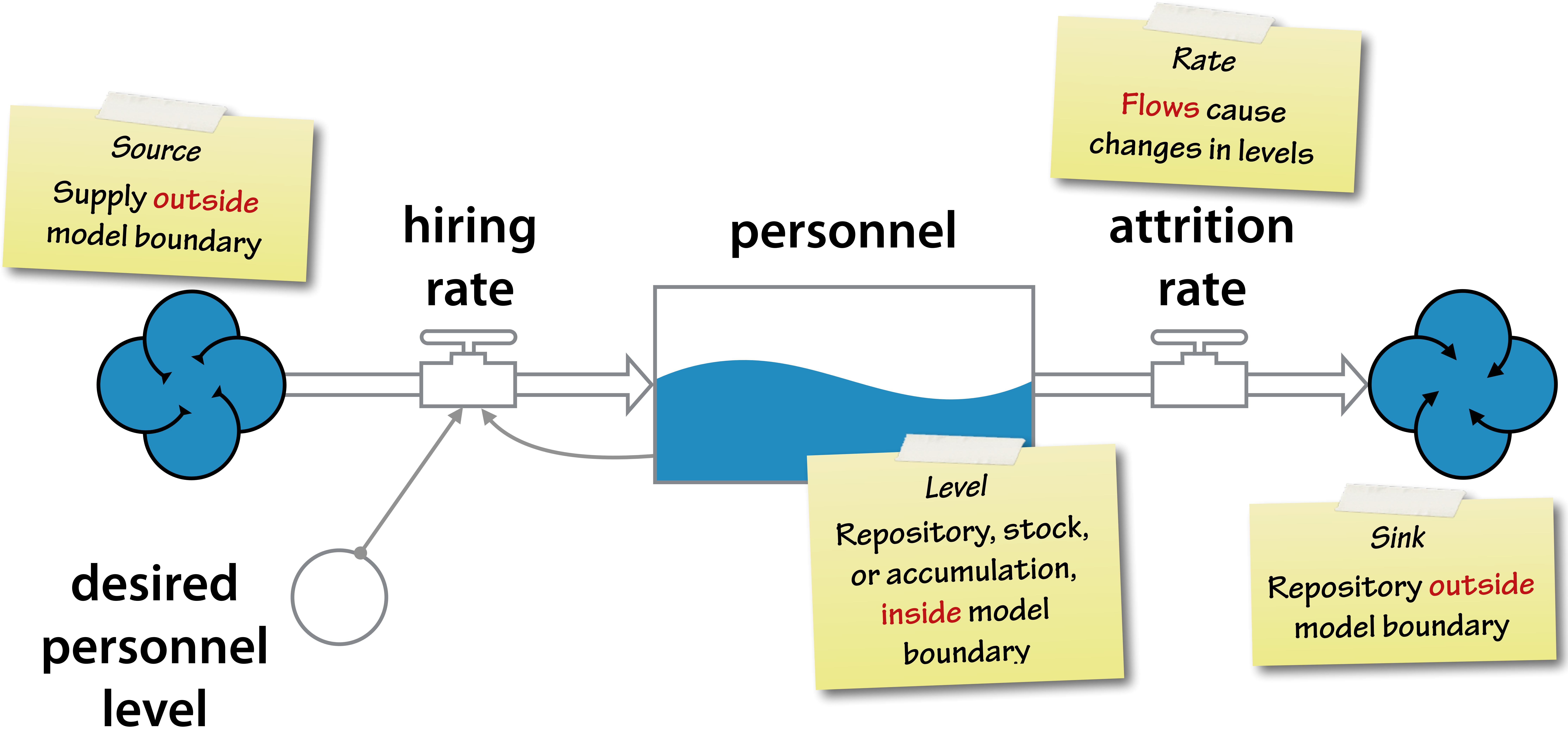
Elements of continuous models



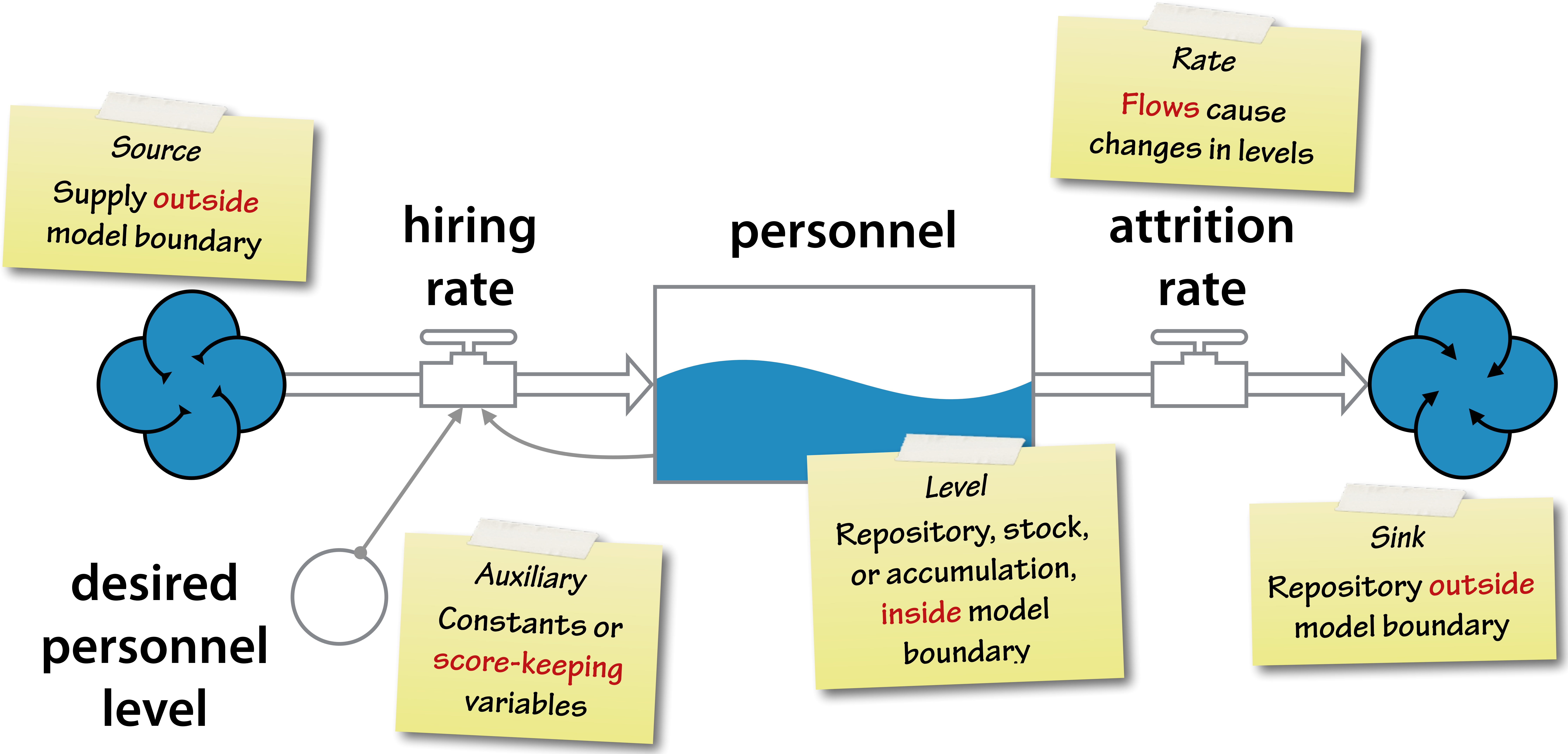
Elements of continuous models



Elements of continuous models



Elements of continuous models

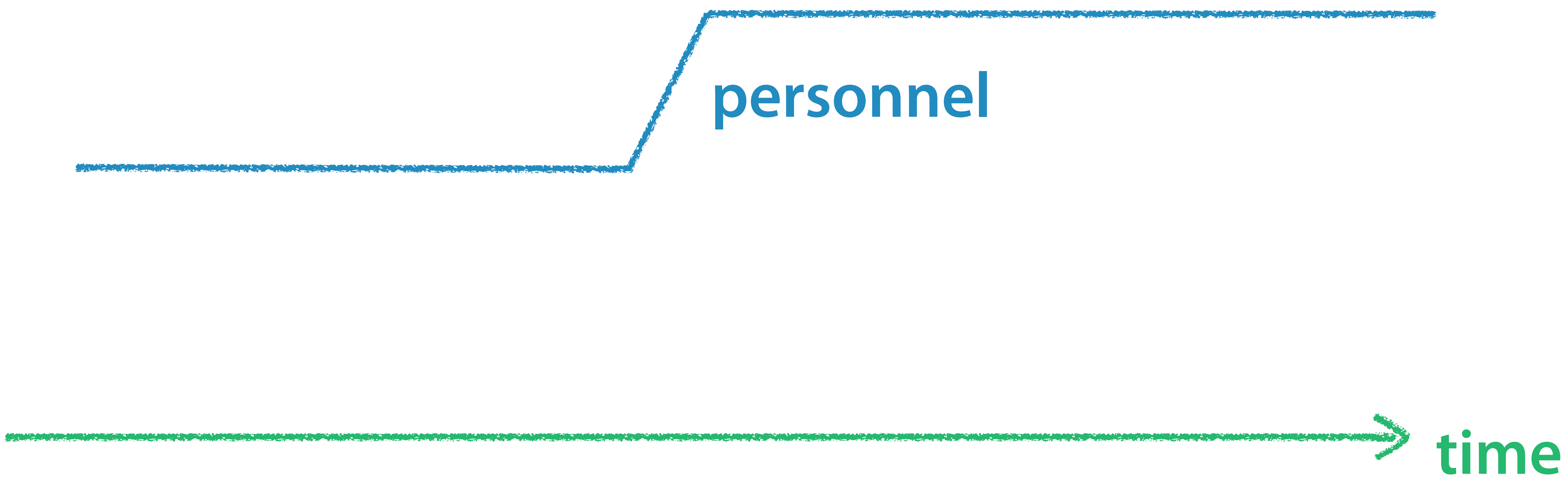


Brooks' Law

Reference behaviour

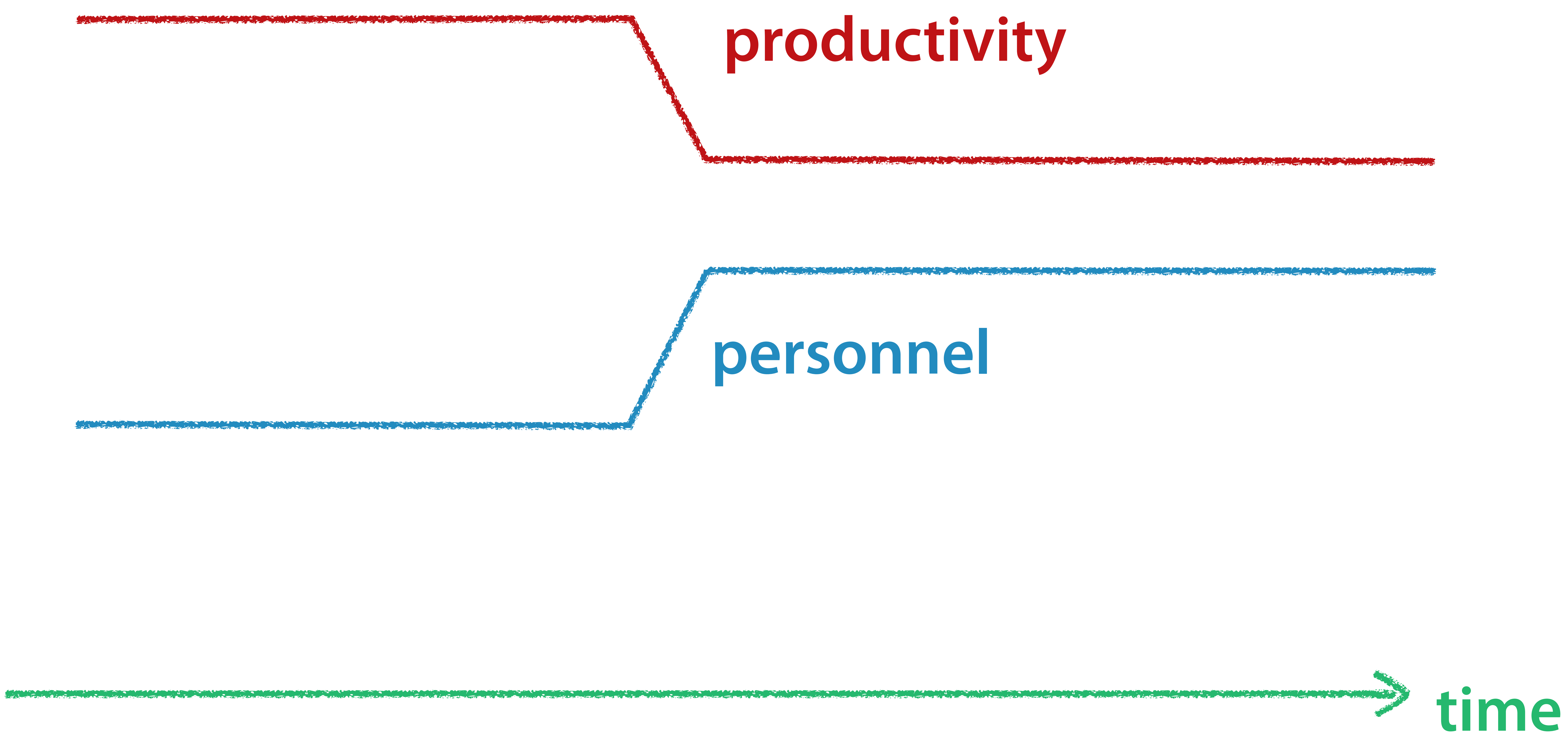
Brooks' Law

Reference behaviour

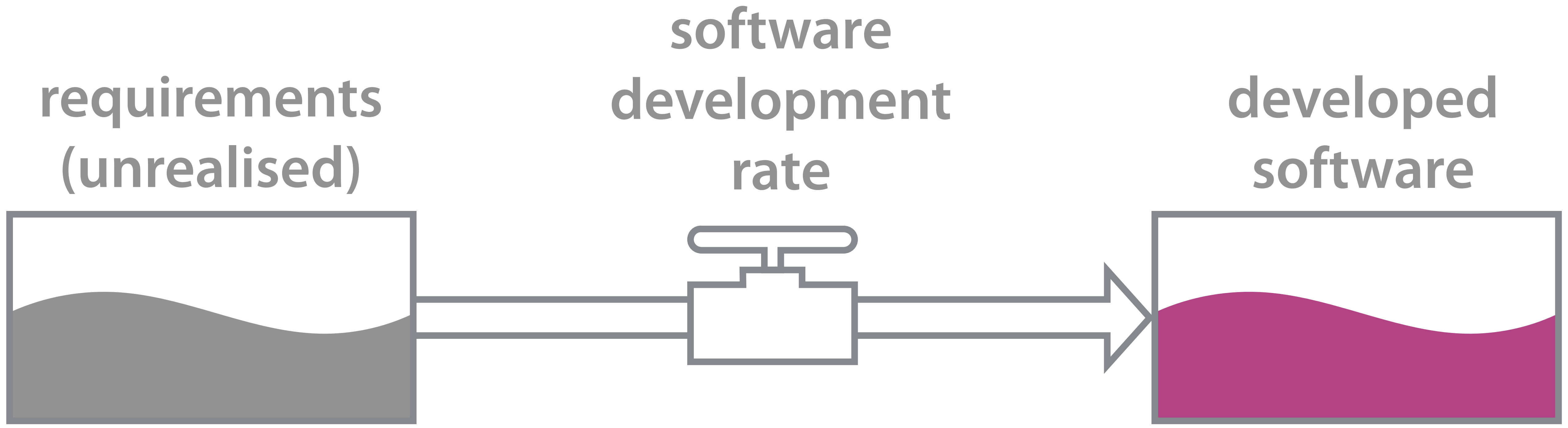


Brooks' Law

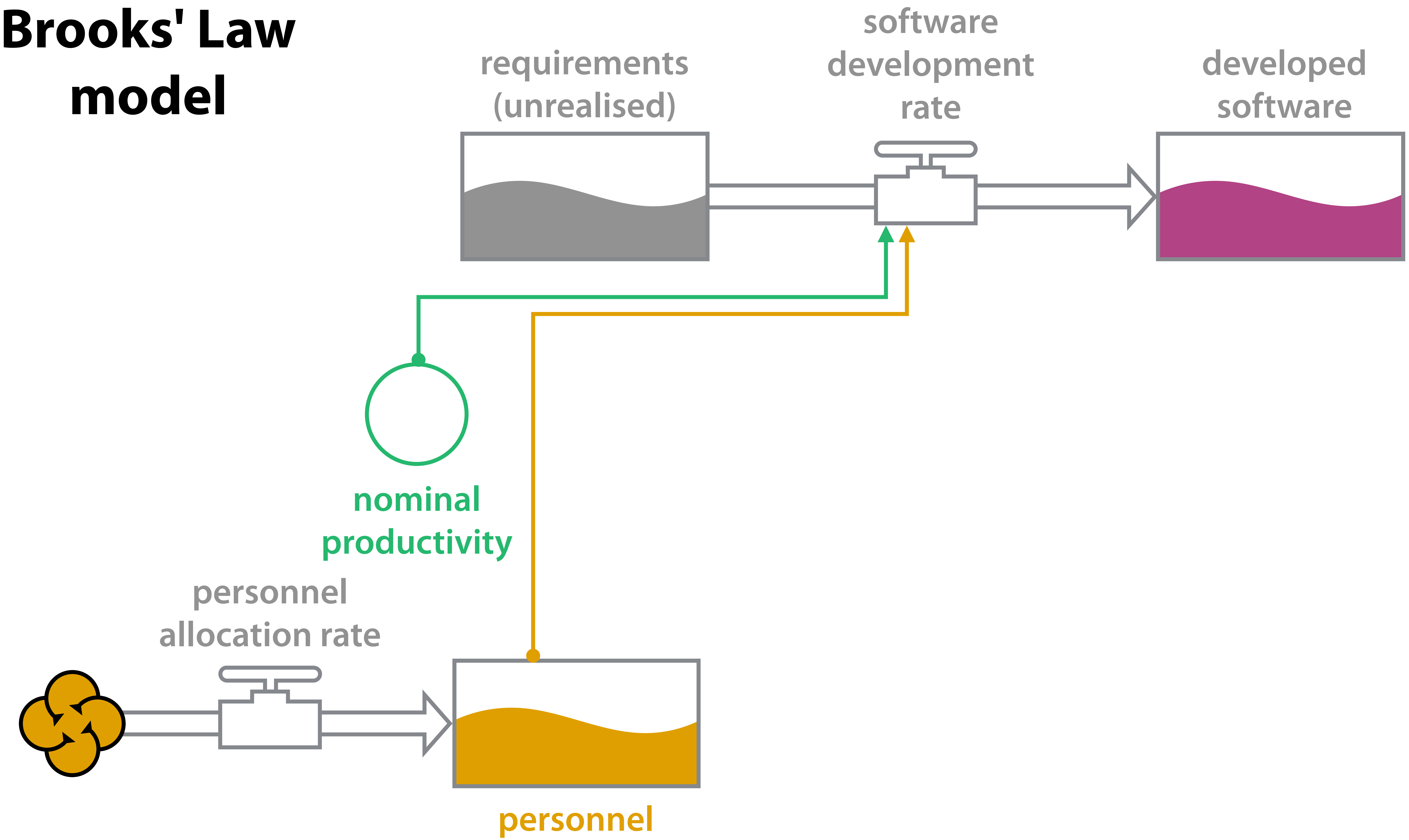
Reference behaviour

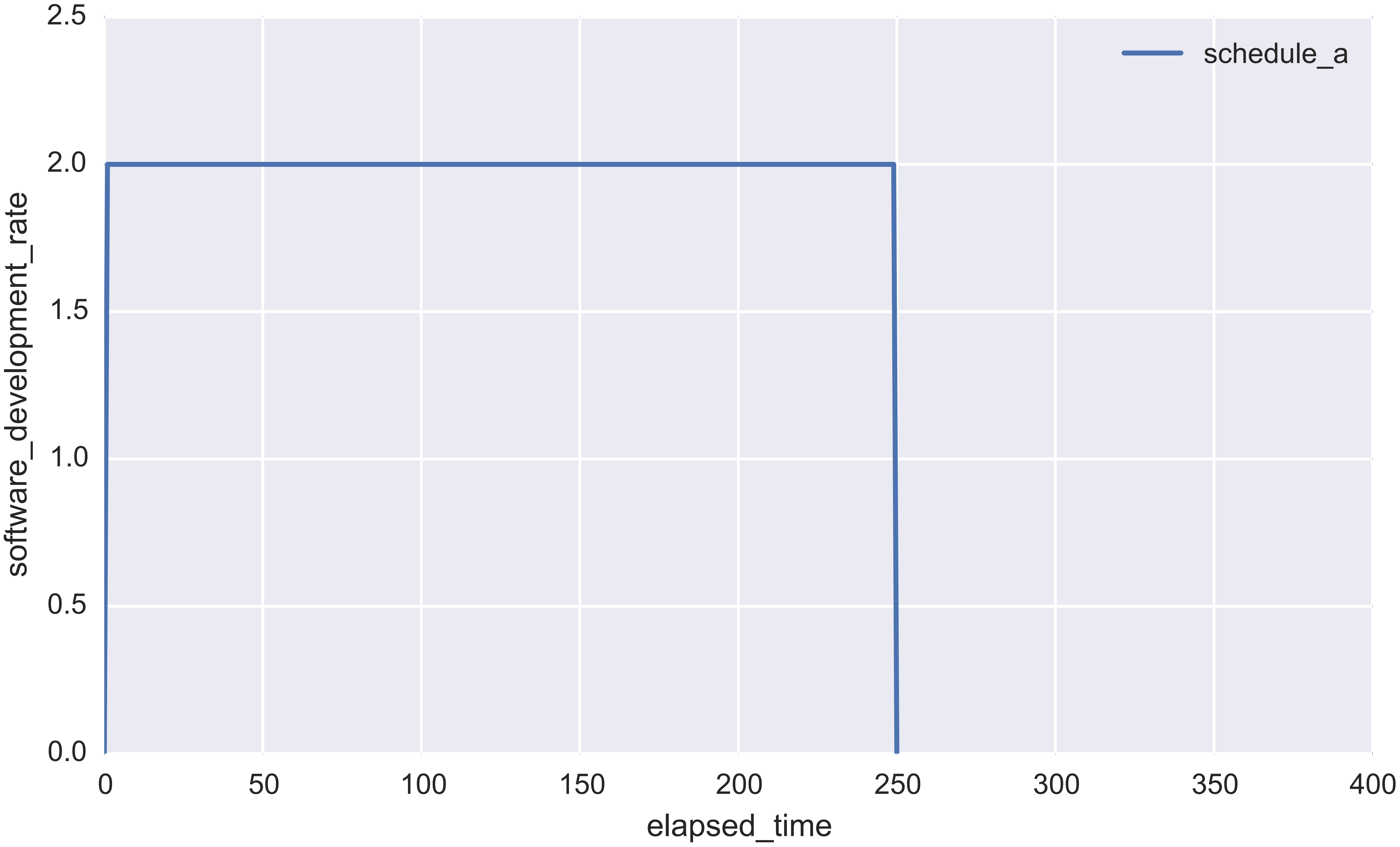


Brooks' Law model

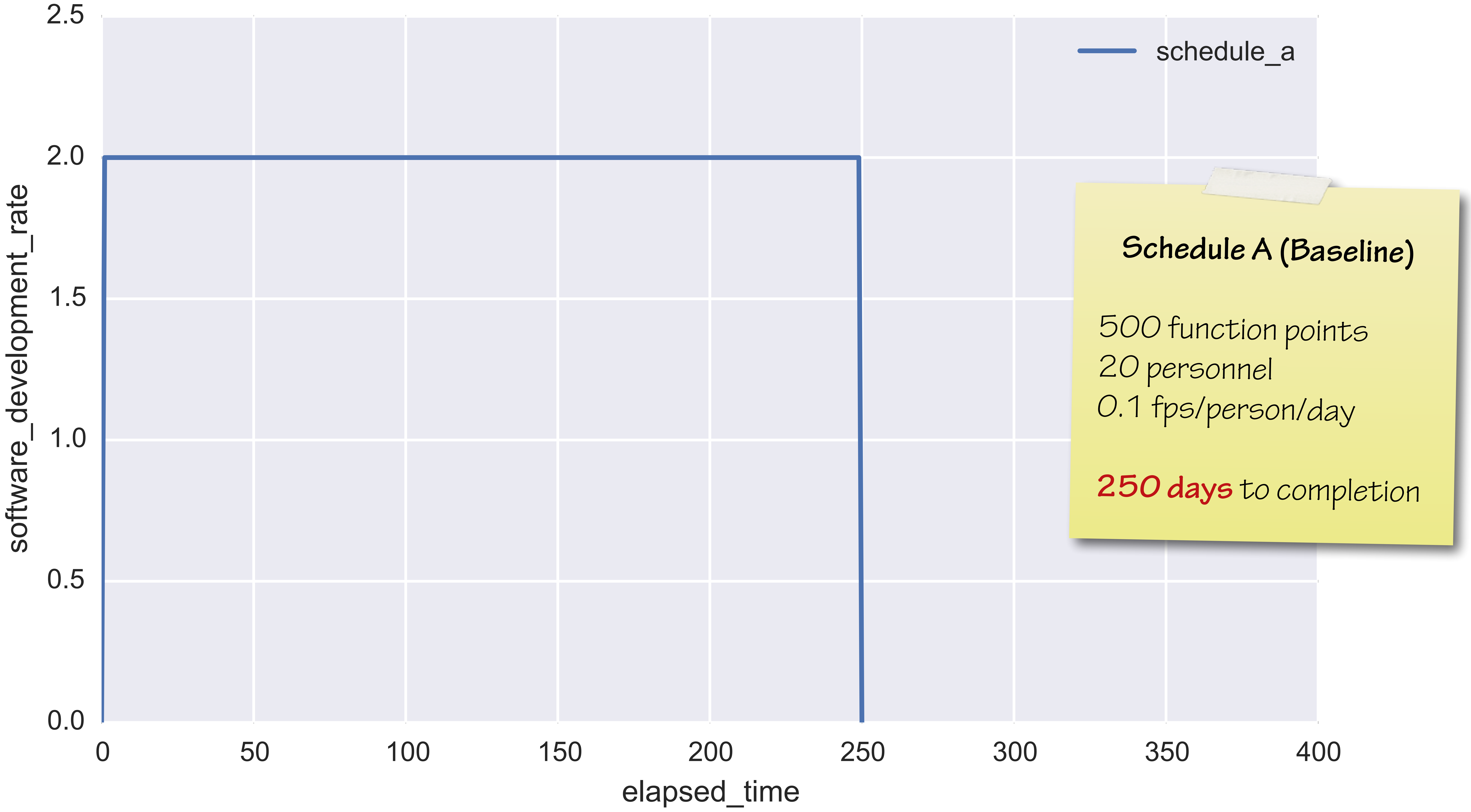


Brooks' Law model



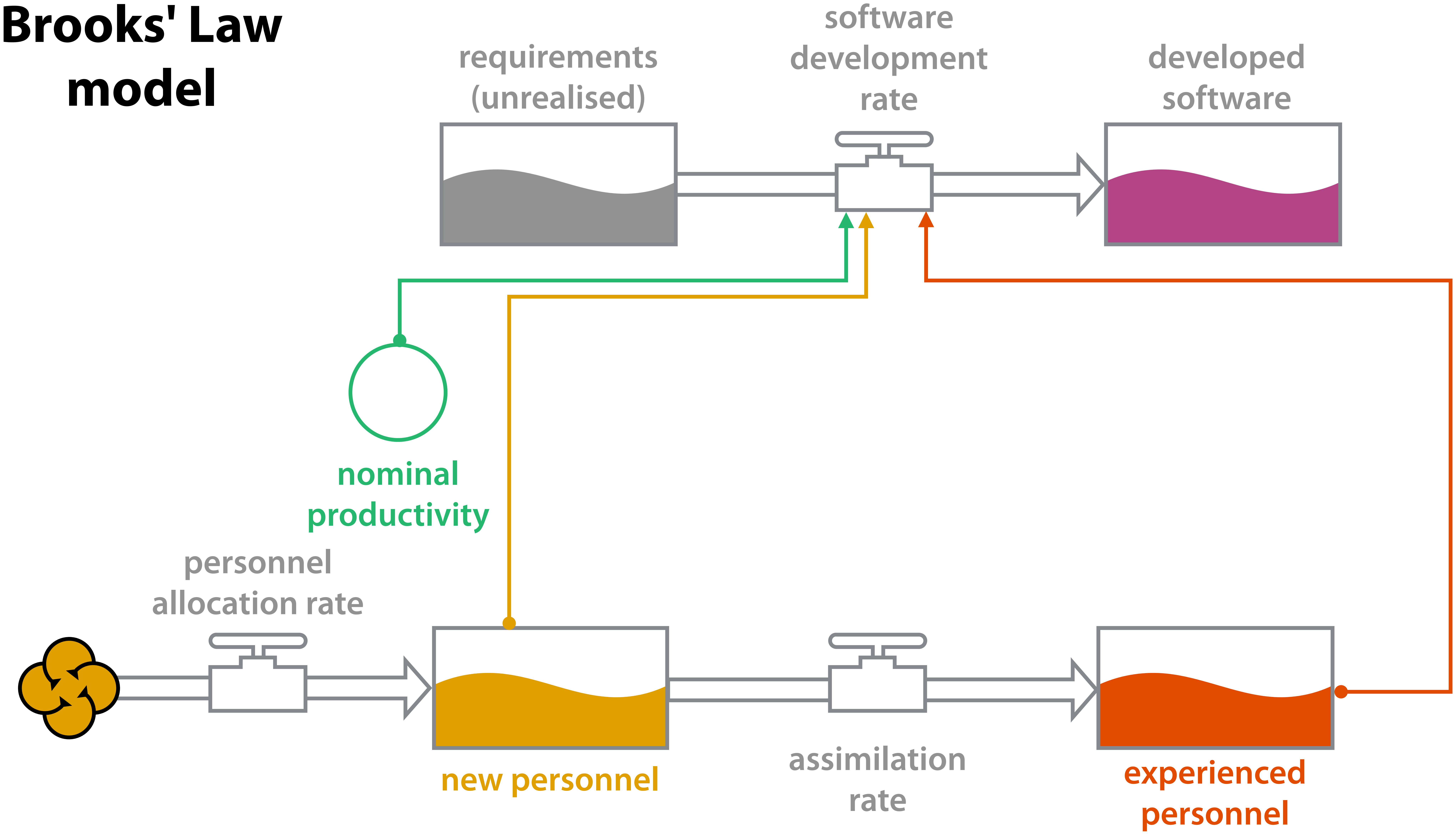


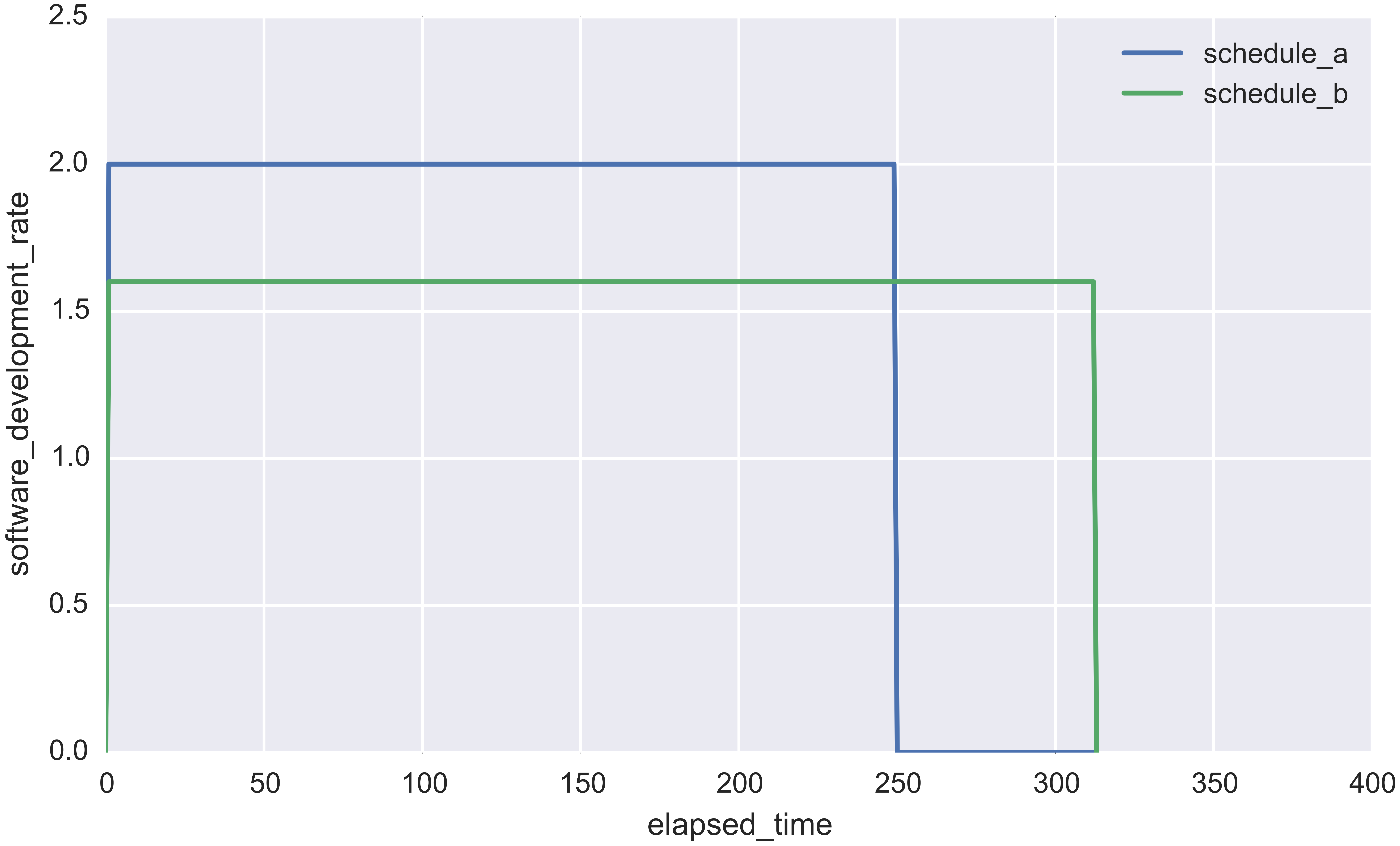
schedule_a
software_development_rate
elapsed_time
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
350
400

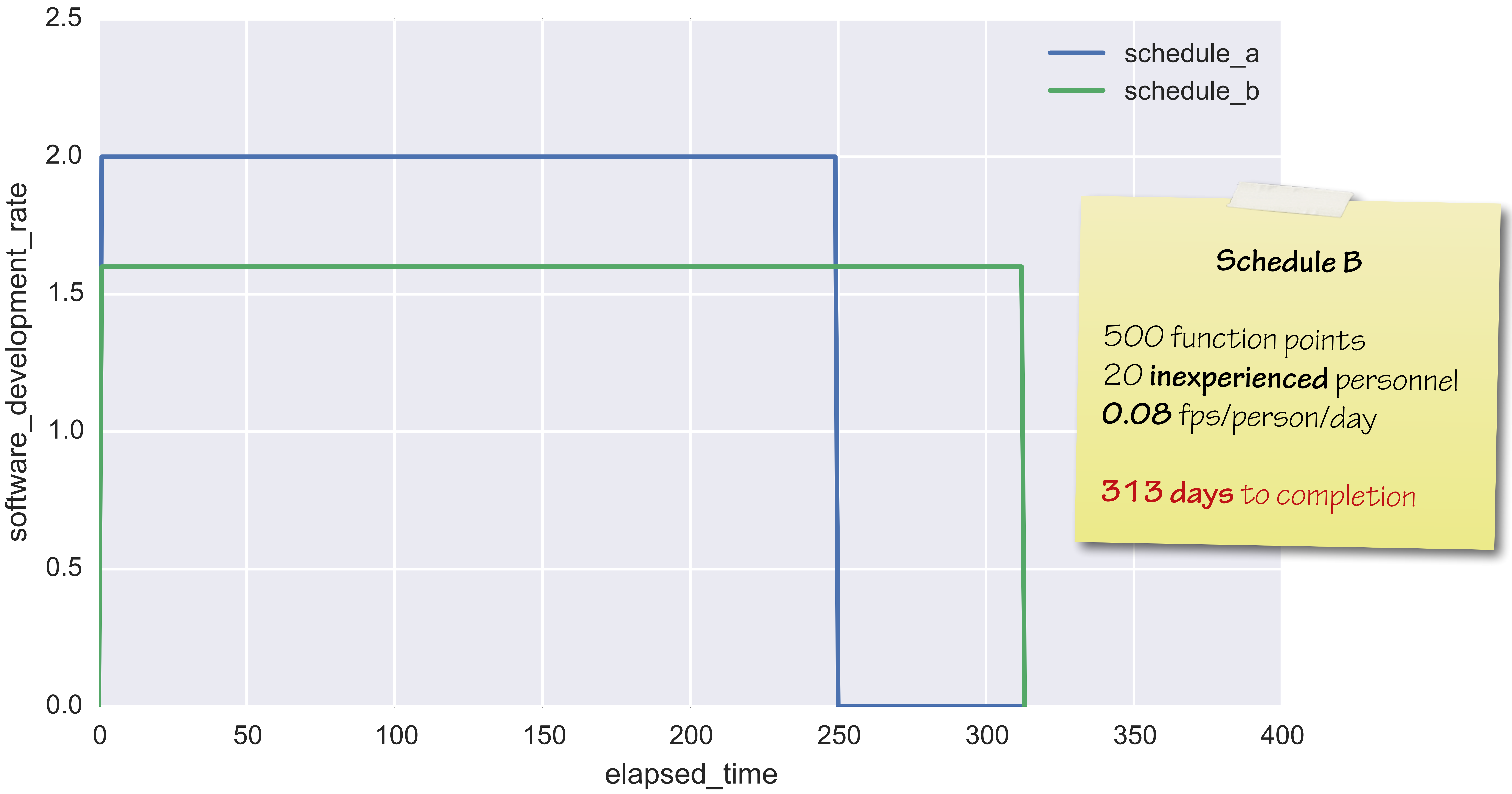


2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_a
Schedule A (Baseline)
500 function points
20 personnel
0.1 fps/person/day
250 days to completion

Brooks' Law model

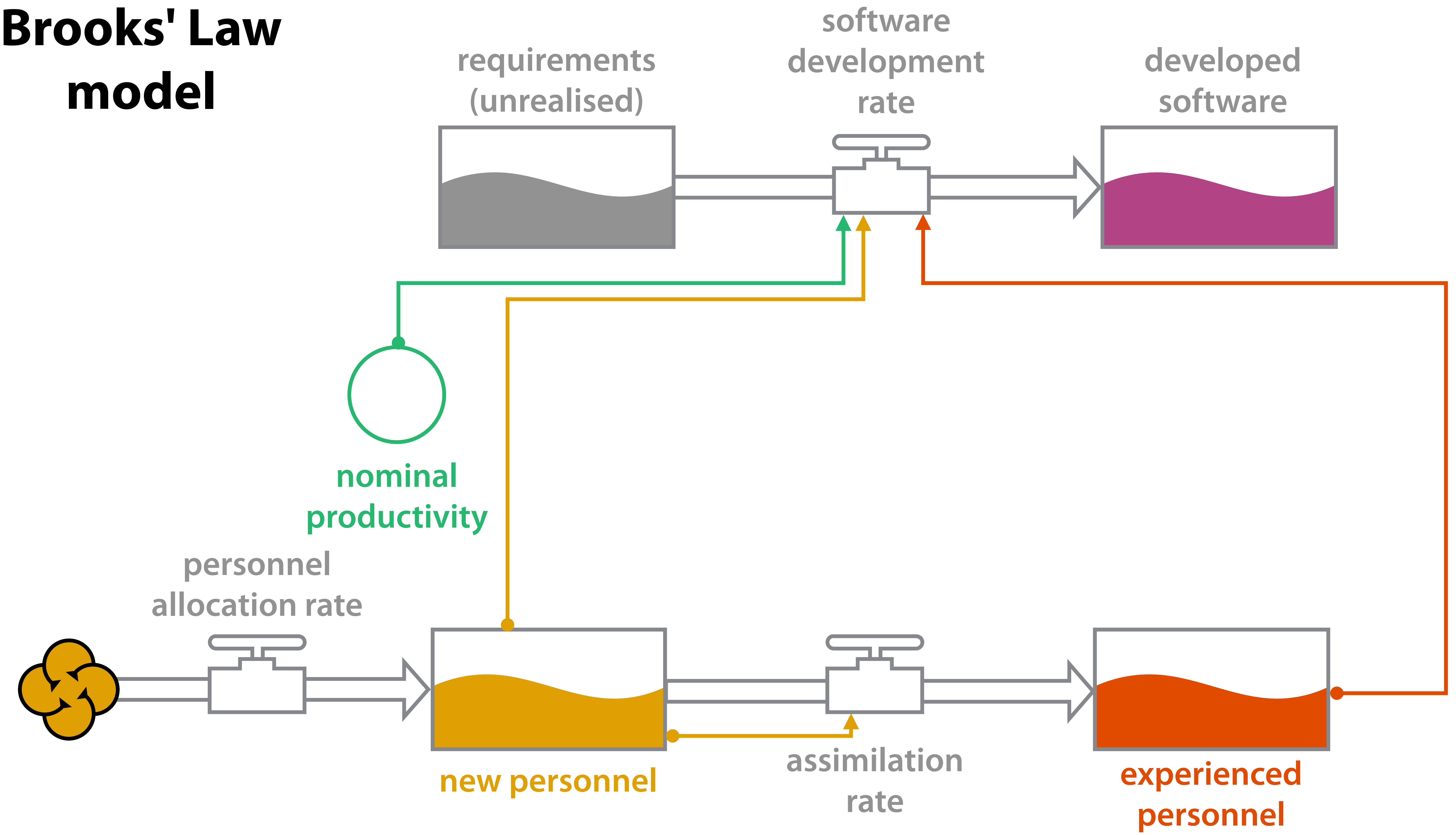


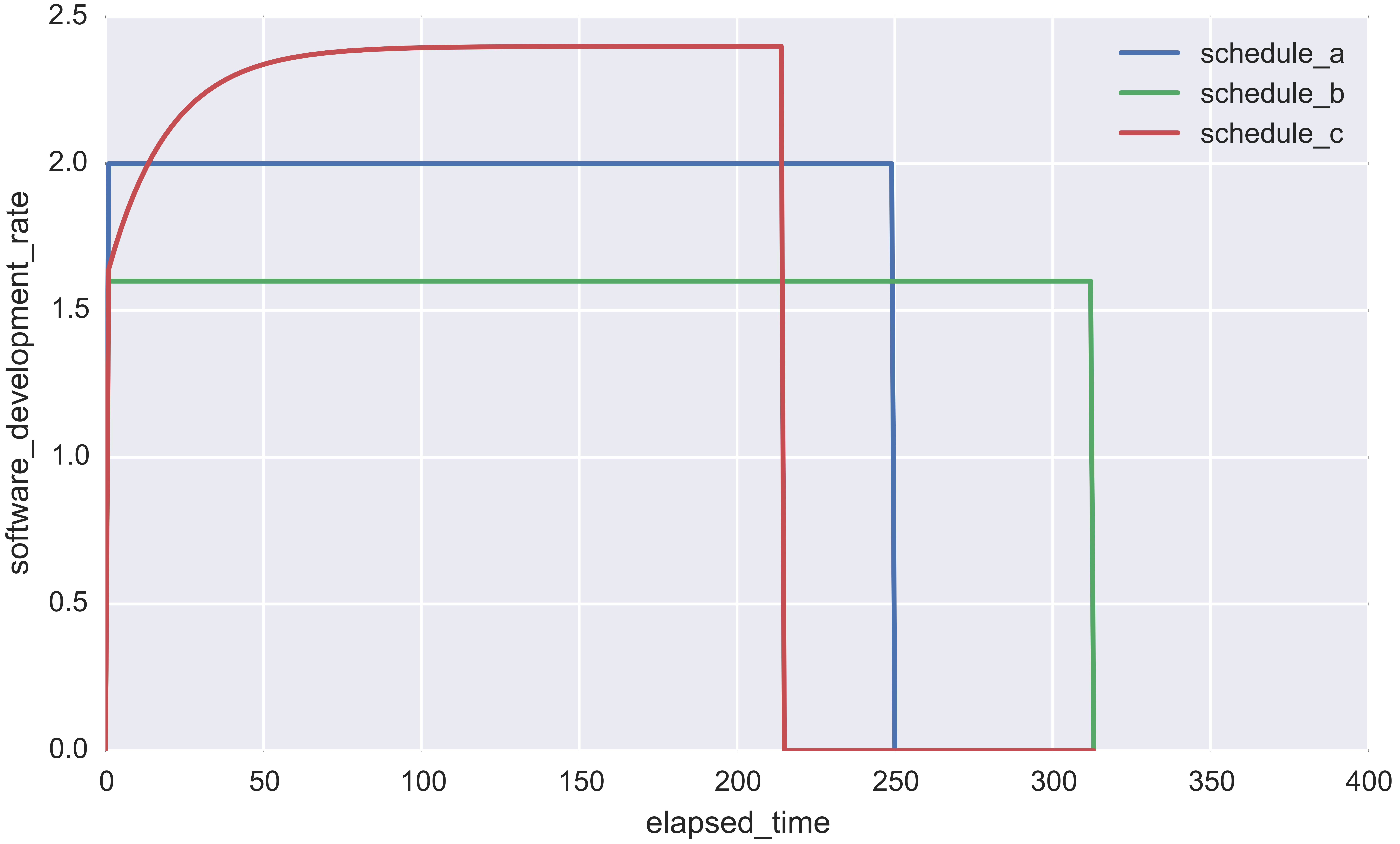

schedule_a
schedule_b
2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time



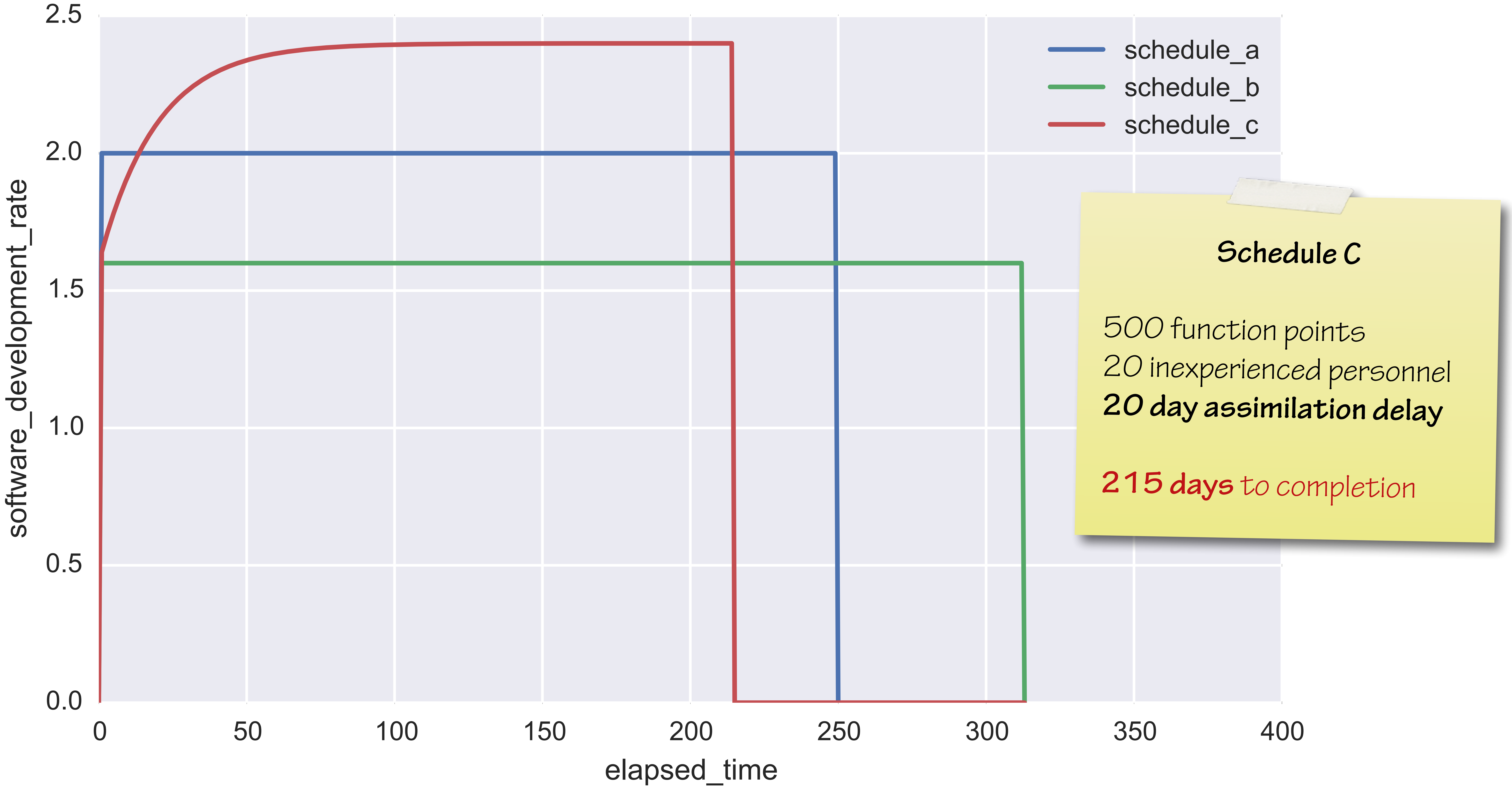
2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_a
schedule_b
Schedule B
500 function points
20 inexperienced personnel
0.08 fps/person/day
313 days to completion

Brooks' Law model

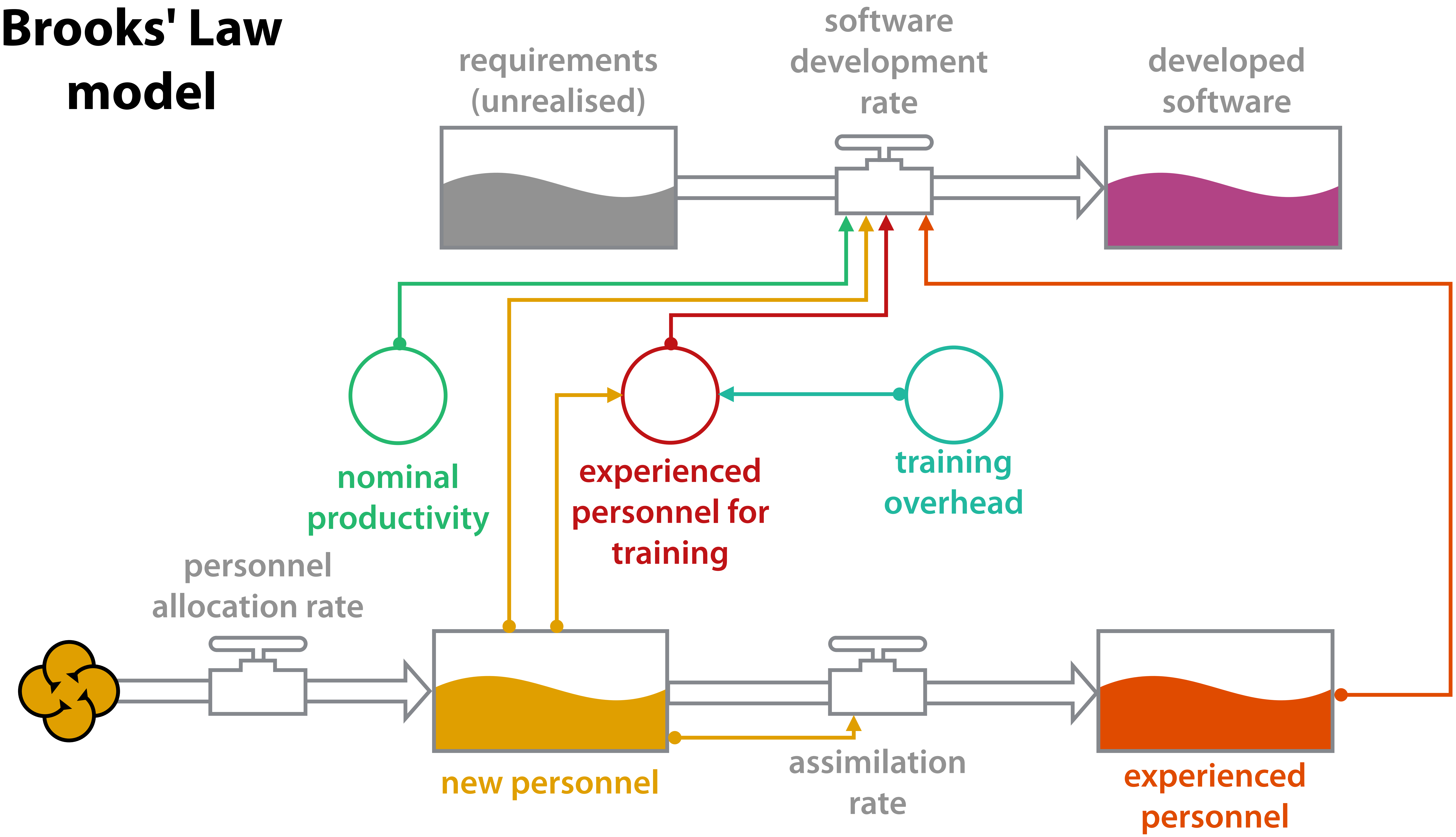


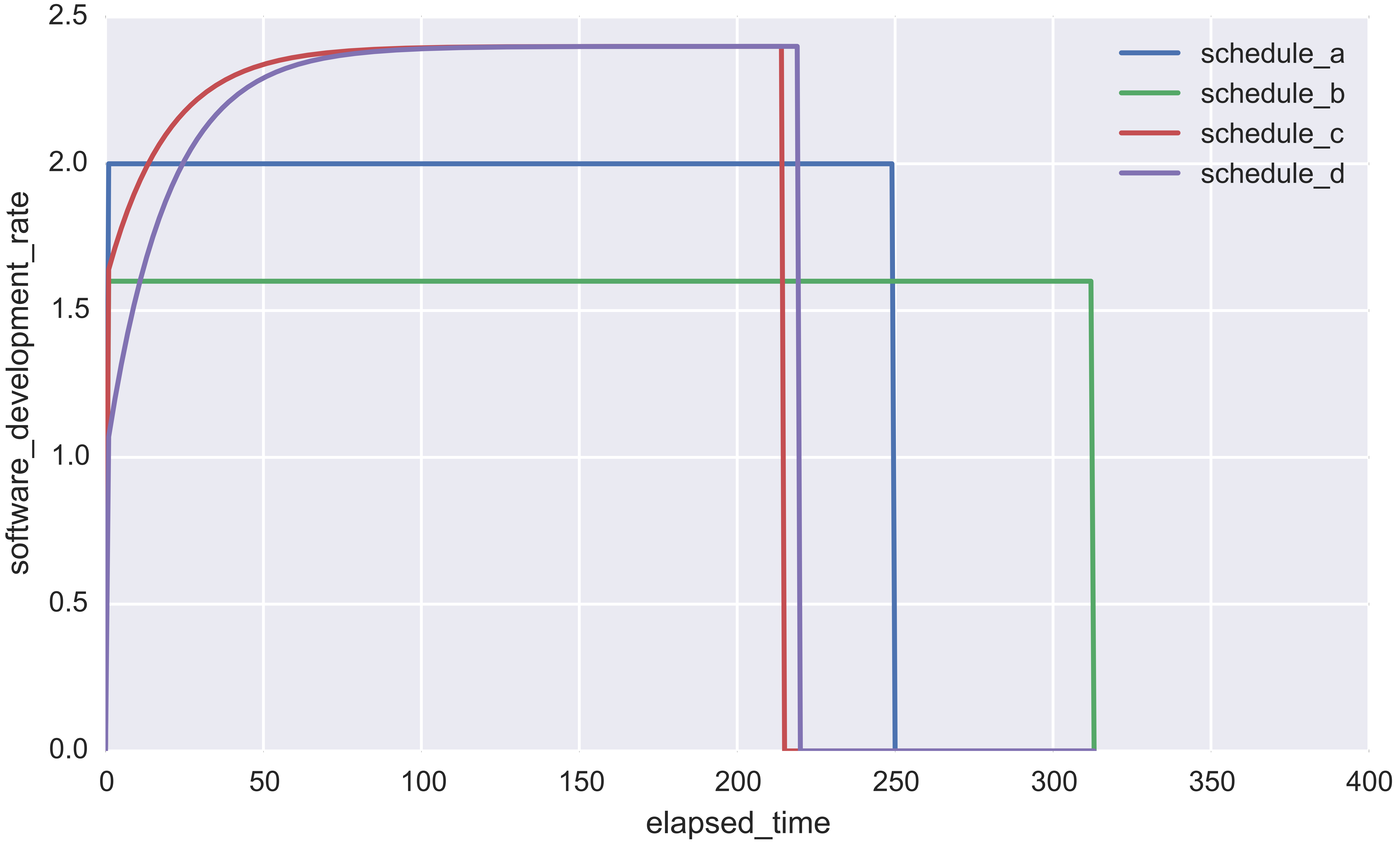


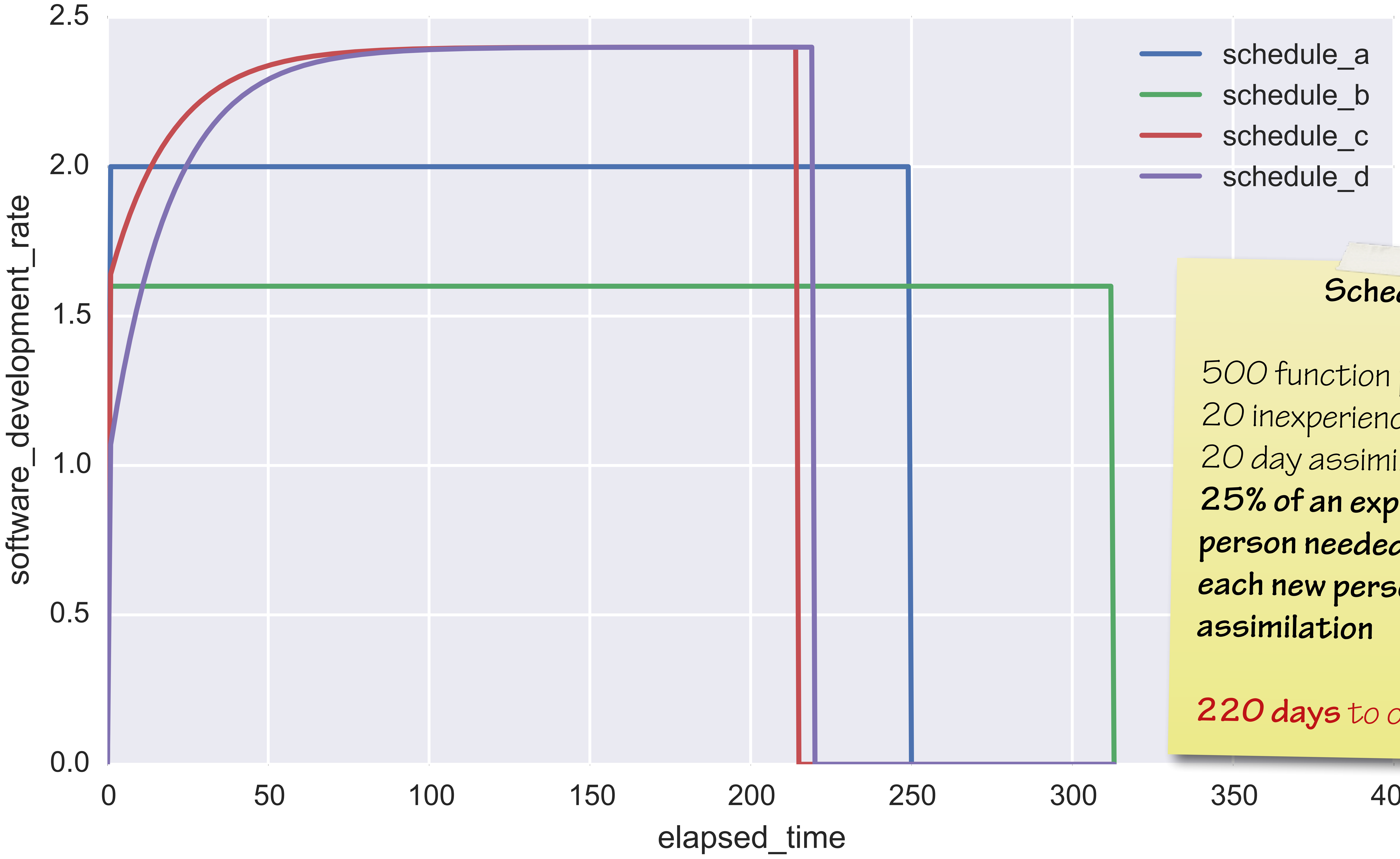
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300
350
400
elapsed_time
software_development_rate
schedule_a
schedule_b
schedule_c


2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_a
schedule_b
schedule_c
Schedule C
500 function points
20 inexperienced personnel
20 day assimilation delay
215 days to completion

Brooks' Law model




schedule_a
schedule_b
schedule_c
schedule_d
software_development_rate
elapsed_time
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300
350
400

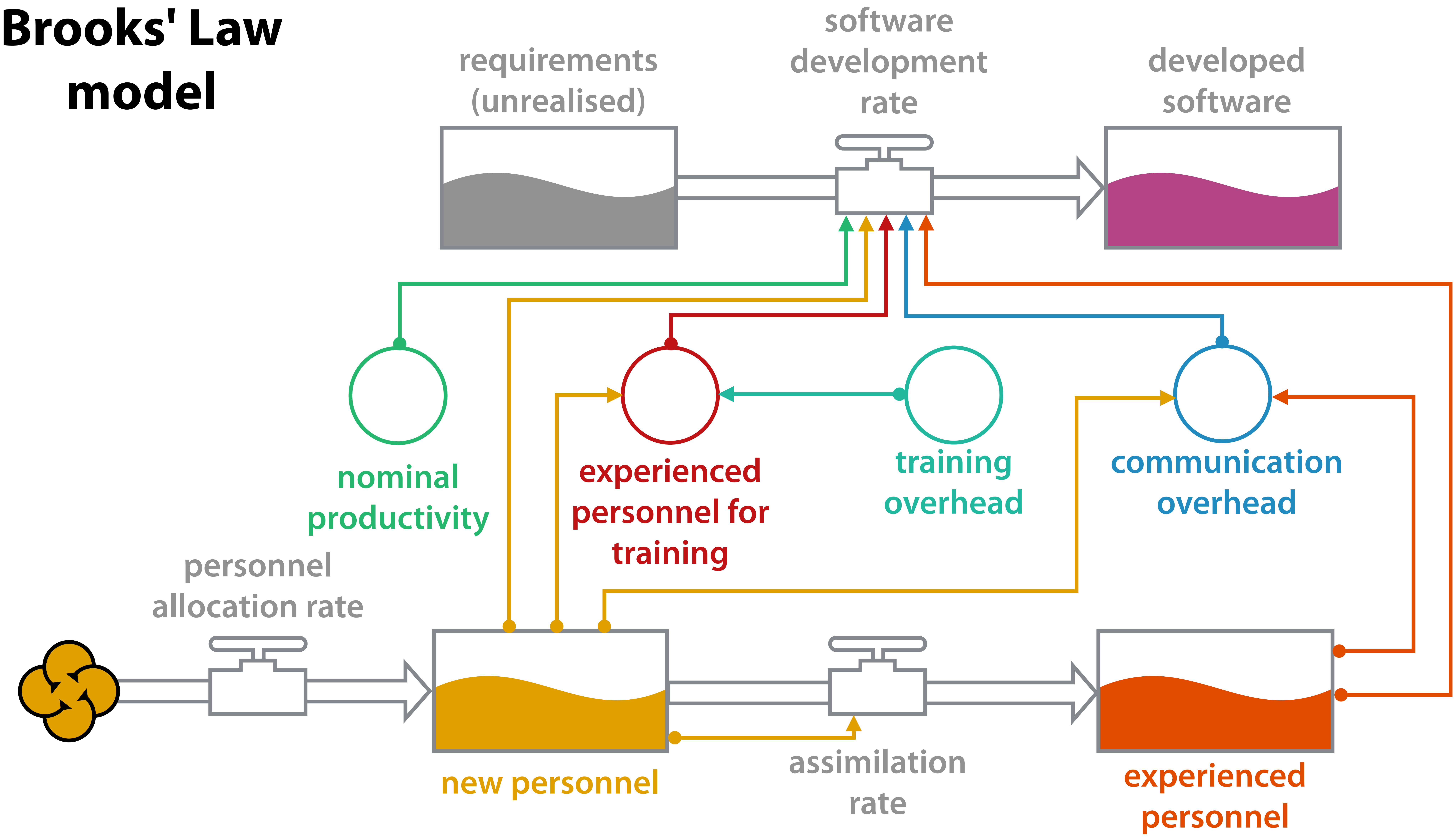


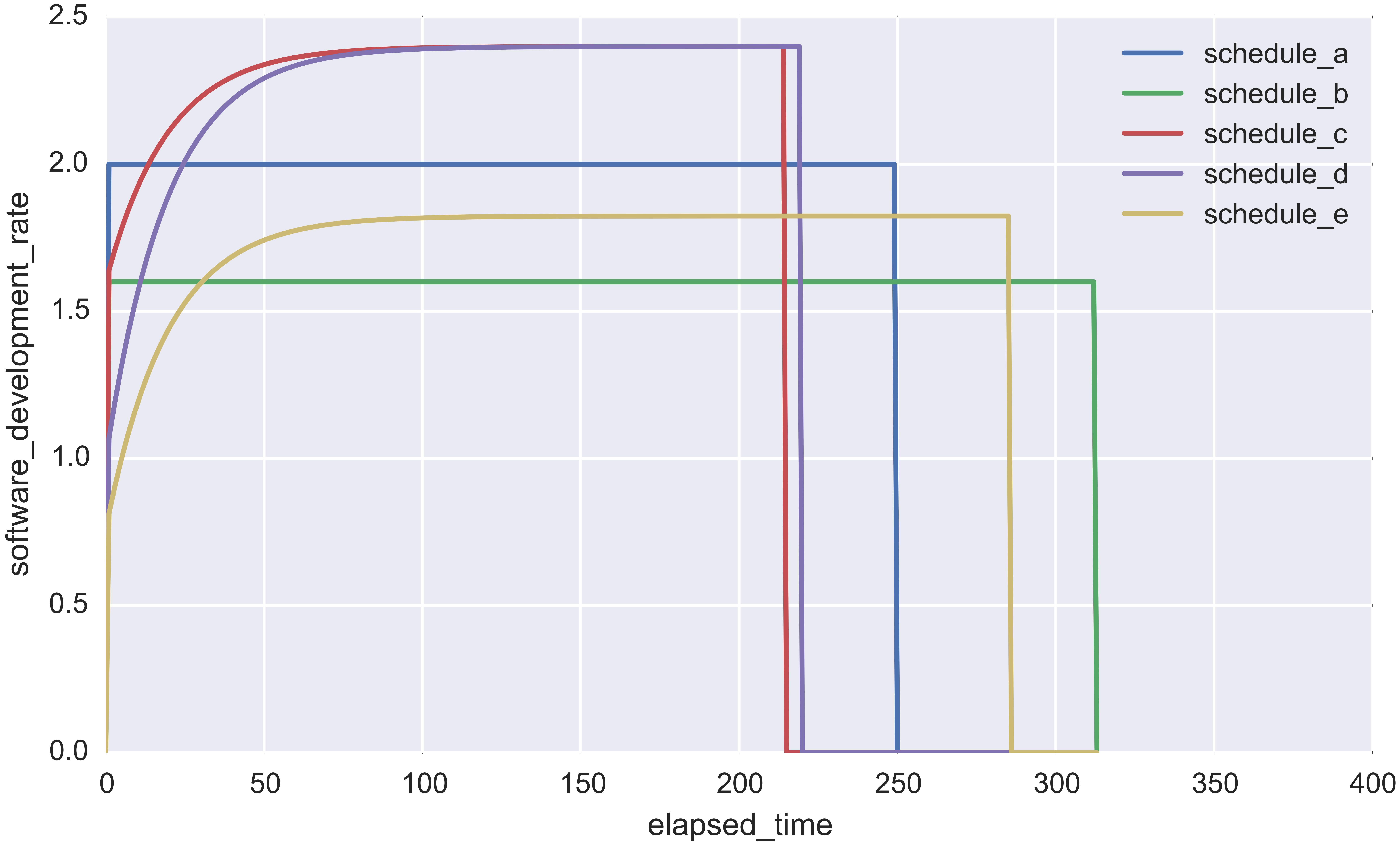
Schedule D

500 function points
20 inexperienced personnel
20 day assimilation delay
25% of an experienced person needed for training each new person during assimilation

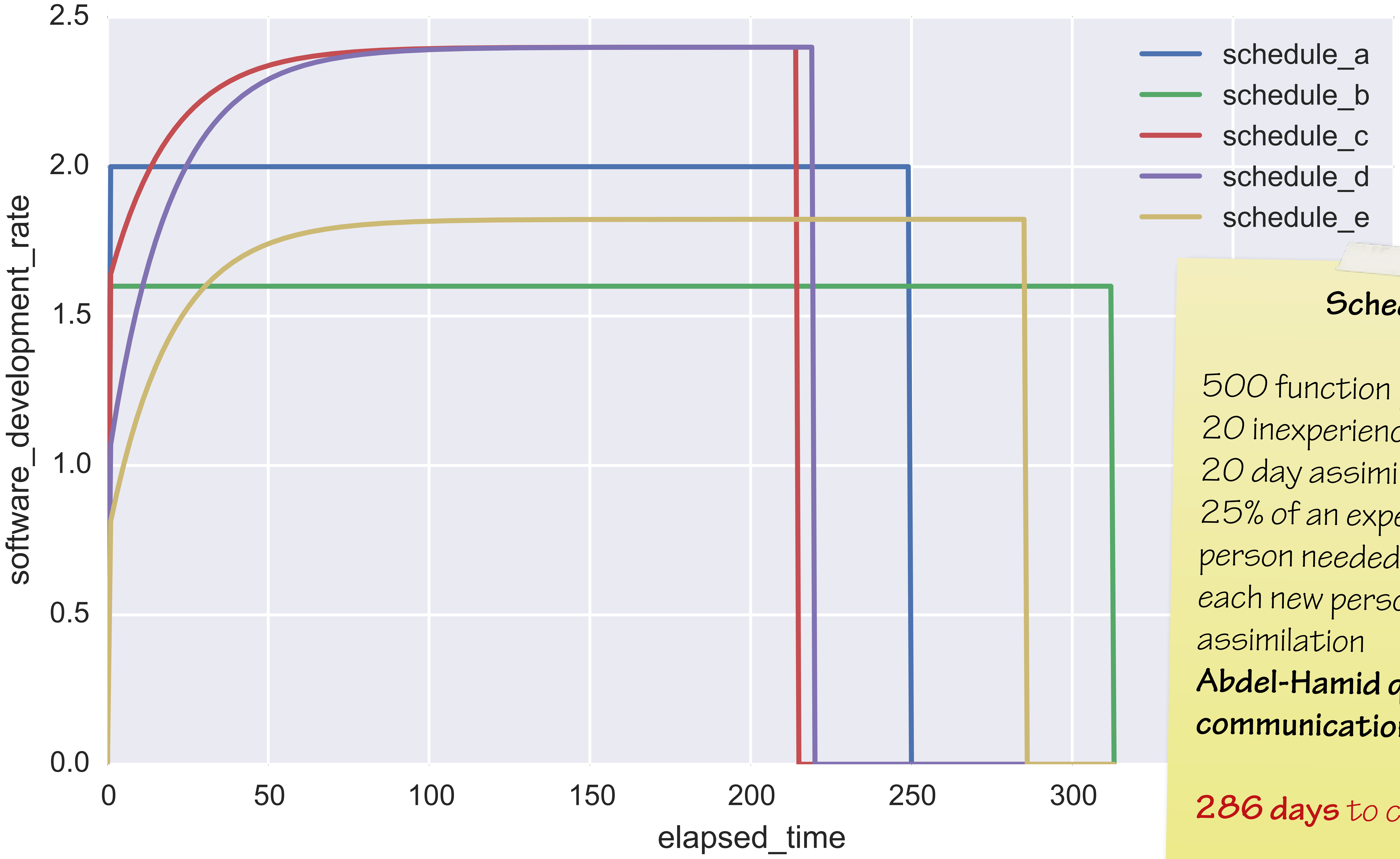
220 days to completion

Brooks' Law model





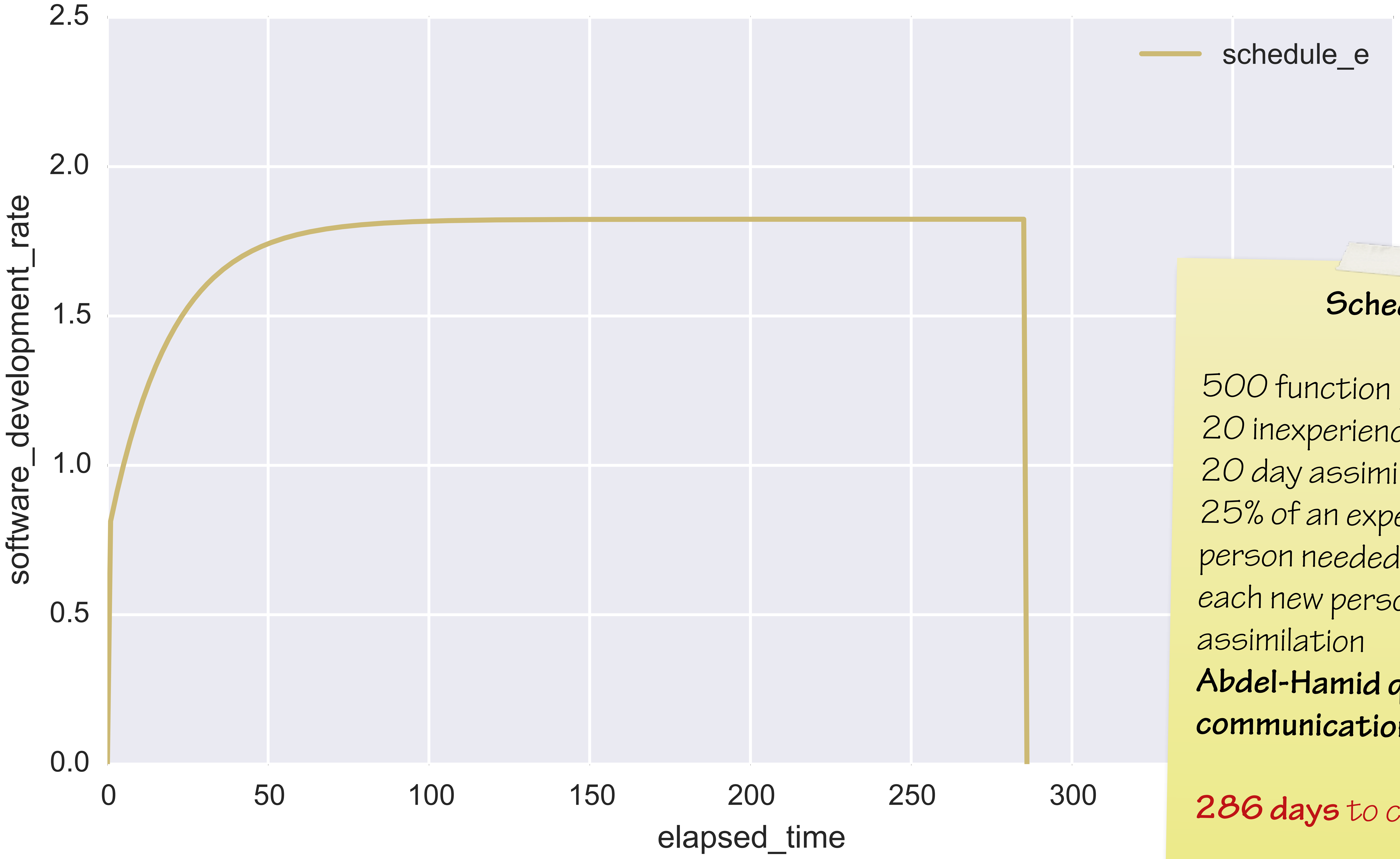
software_development_rate
elapsed_time
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
350
400
schedule_a
schedule_b
schedule_c
schedule_d
schedule_e



Schedule E

500 function points
20 inexperienced personnel
20 day assimilation delay
25% of an experienced person needed for training each new person during assimilation
Abdel-Hamid quadratic communication overhead

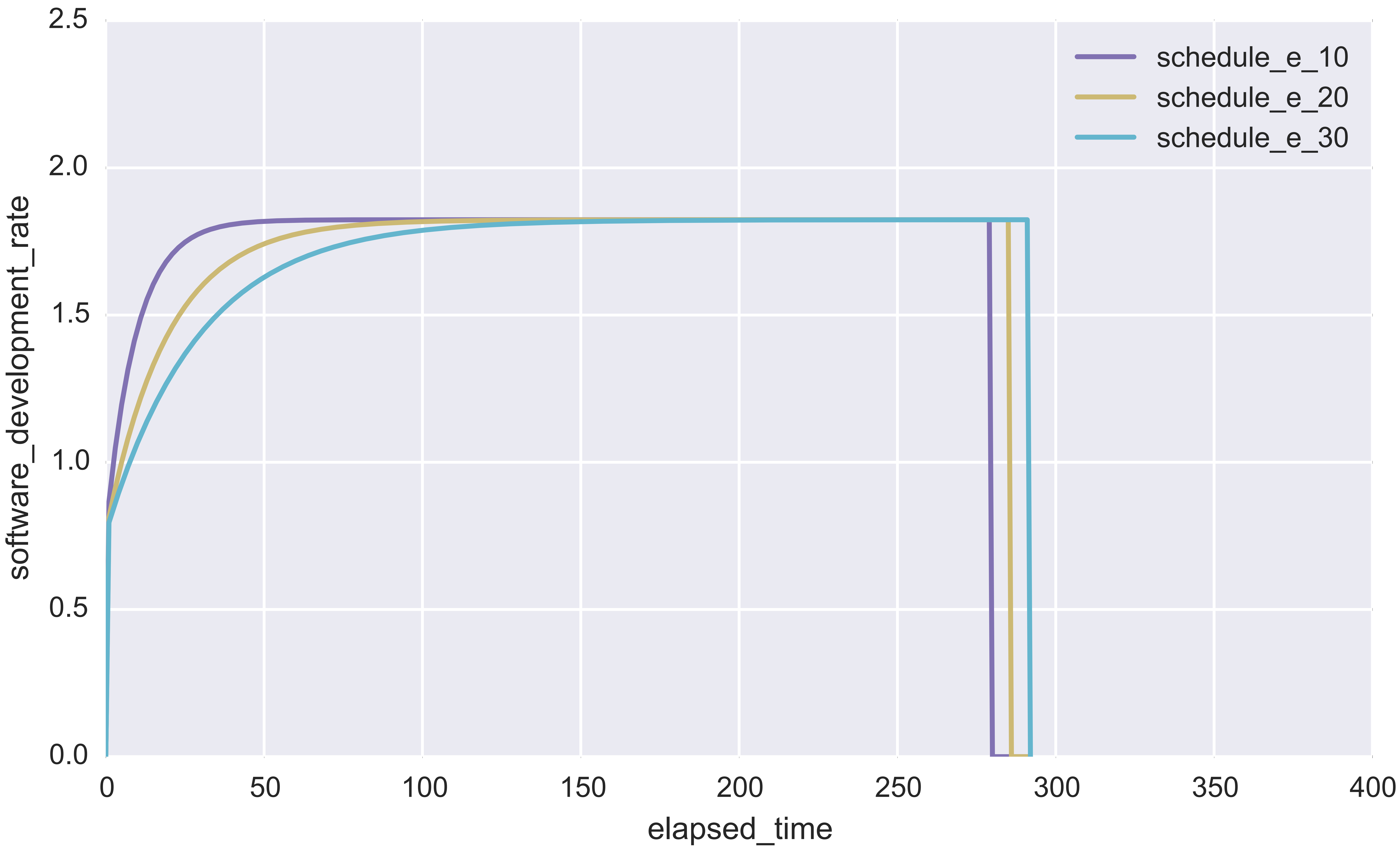
286 days to completion



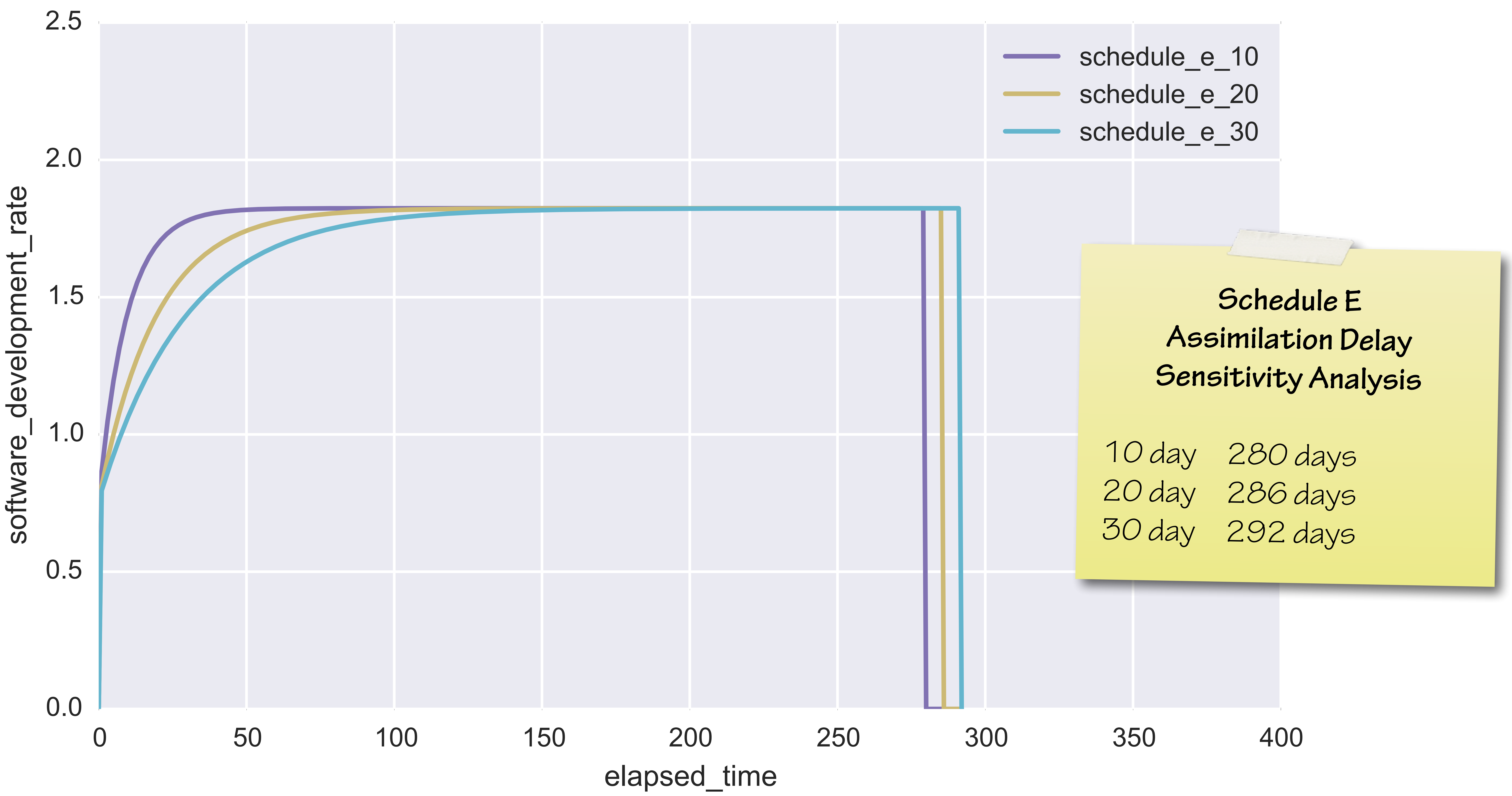
Schedule E

500 function points
20 inexperienced personnel
20 day assimilation delay
25% of an experienced person needed for training each new person during assimilation
Abdel-Hamid quadratic communication overhead

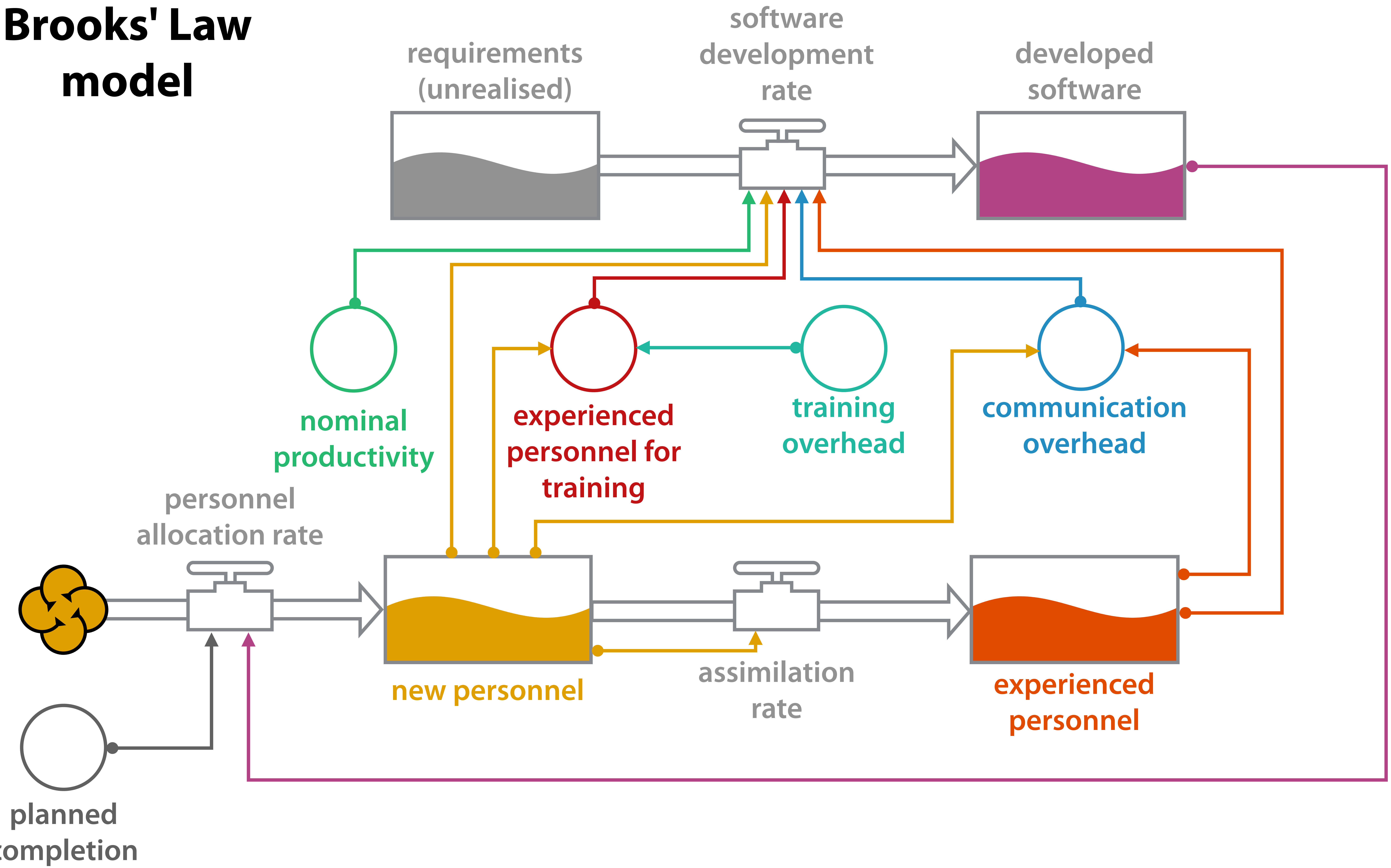
286 days to completion



schedule_e_10
schedule_e_20
schedule_e_30
software_development_rate
elapsed_time
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
350
400


2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_e_10
schedule_e_20
schedule_e_30
Schedule E
Assimilation Delay
Sensitivity Analysis
10 day 280 days
20 day 286 days
30 day 292 days

Brooks' Law model

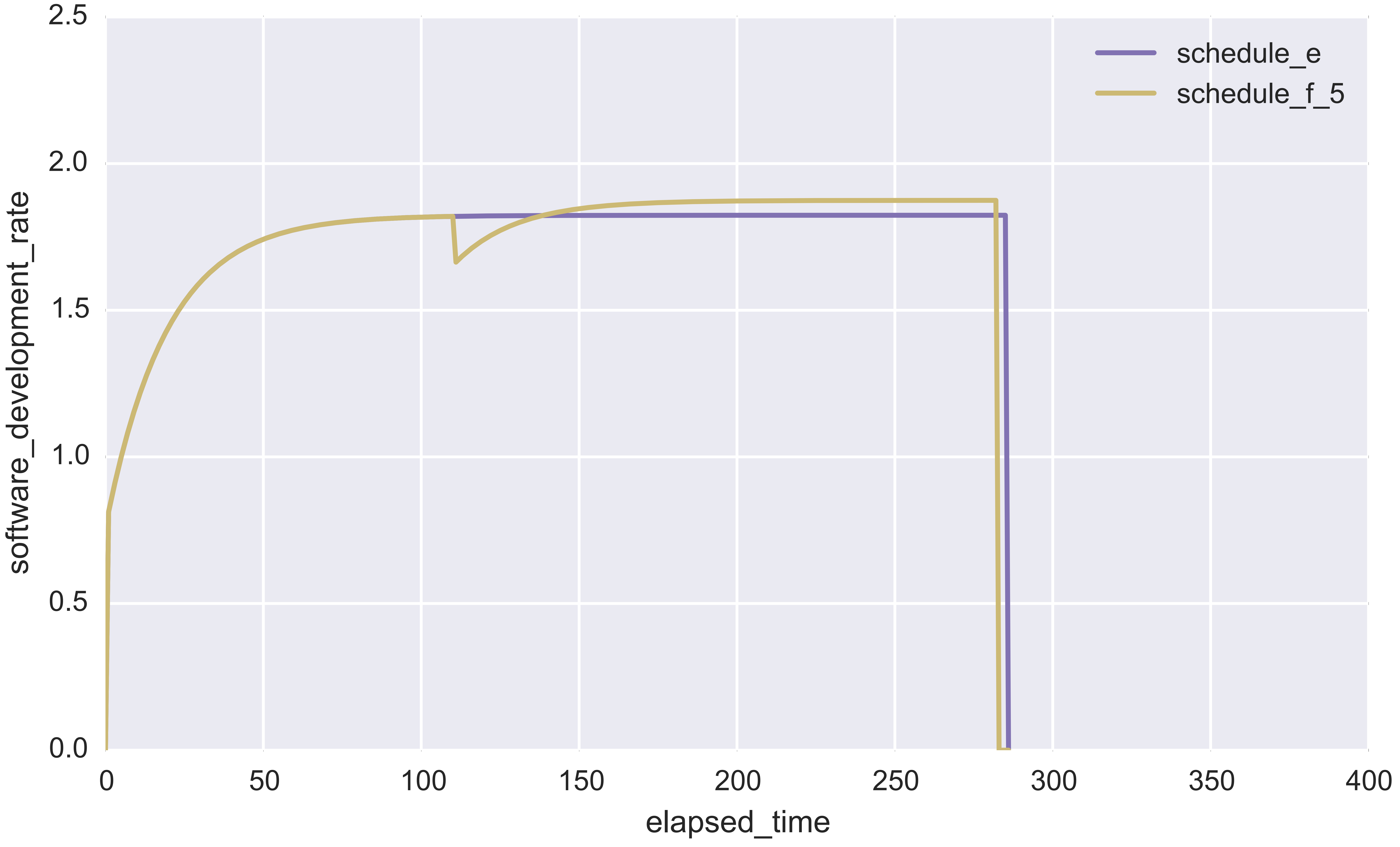


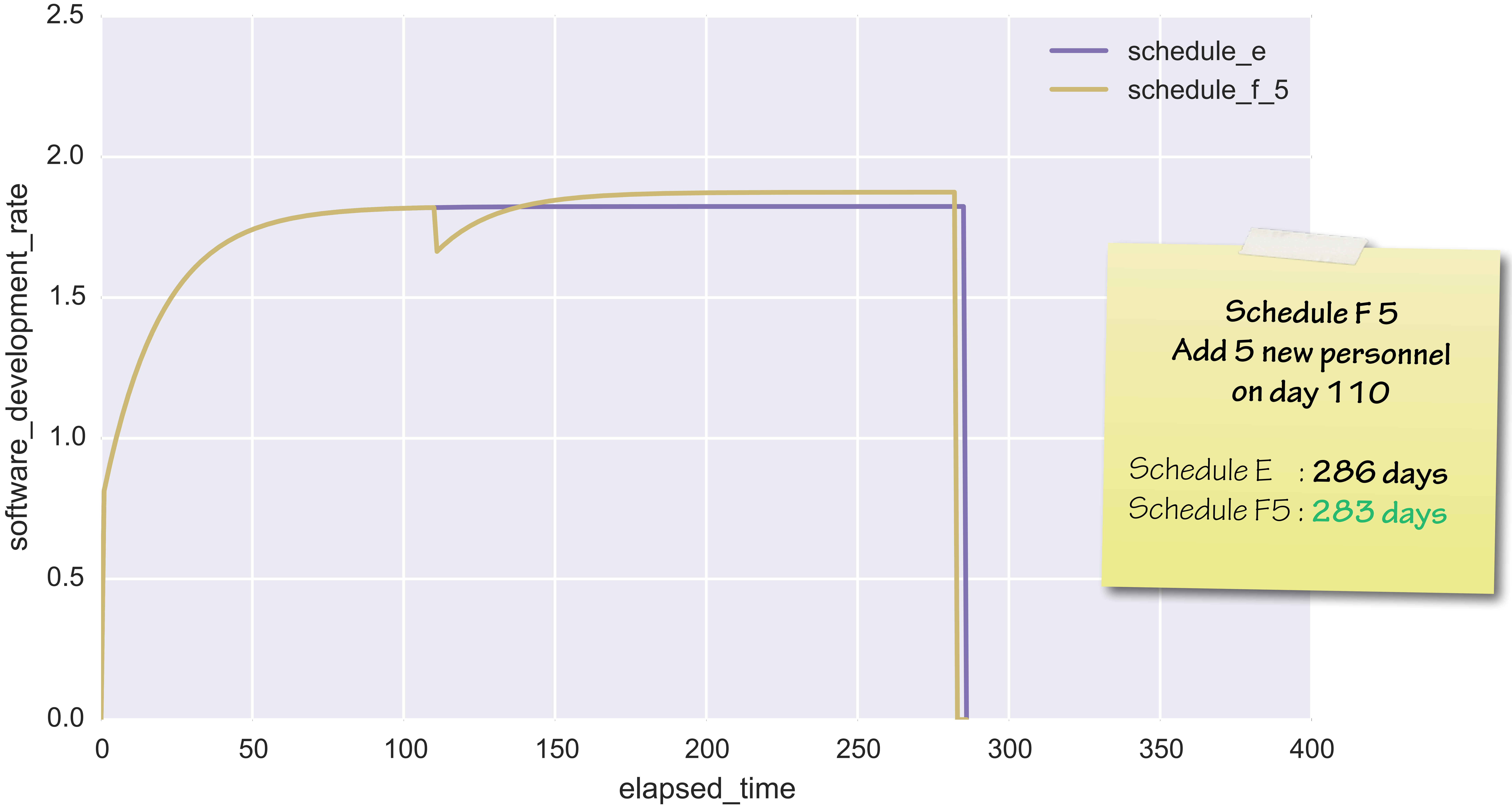
schedule_e.py

```python
import brooks.communication


def initial():
    """Configure the initial model state."""
    return dict(
        step_duration_days=1,
        num_function_points_requirements=500,
        num_function_points_developed=0,
        num_new_personnel=20,
        num_experienced_personnel=0,
        personnel_allocation_rate=0,
        personnel_assimilation_rate=0,
        assimilation_delay_days=20,
        nominal_productivity=0.1,
        new_productivity_weight=0.8,
        experienced_productivity_weight=1.2,
        training_overhead_proportion=0.25,
        communication_overhead_function=brooks.communication.quadratic_overhead_proportion,
        software_development_rate=None,
    )


def intervene(step_number, elapsed_time, state):
    """Intervene in the current step before the main simulation step is executed."""
    return state


def is_complete(step_number, elapsed_time_seconds, state):
    """Determine whether the simulation should end."""
    return state.num_function_points_developed >= state.num_function_points_requirements


def complete(step_number, elapsed_time_seconds, state):
    """Finalise the simulation state for the last recorded step."""
    state.software_development_rate = 0
    return state
```

schedule_f_5.py

```python
import brooks.communication


def initial():
    """Configure the initial model state."""
    return dict(
        step_duration_days=1,
        num_function_points_requirements=500,
        num_function_points_developed=0,
        num_new_personnel=20,
        num_experienced_personnel=0,
        personnel_allocation_rate=0,
        personnel_assimilation_rate=0,
        assimilation_delay_days=20,
        nominal_productivity=0.1,
        new_productivity_weight=0.8,
        experienced_productivity_weight=1.2,
        training_overhead_proportion=0.25,
        communication_overhead_function=brooks.communication.quadratic_overhead_proportion,
        software_development_rate=None,
    )


def intervene(step_number, elapsed_time, state):
    """Intervene in the current step before the main simulation step is executed."""
    if elapsed_time == 110:
        state.num_new_personnel += 5
    return state


def is_complete(step_number, elapsed_time_seconds, state):
    """Determine whether the simulation should end."""
    return state.num_function_points_developed >= state.num_function_points_requirements


def complete(step_number, elapsed_time_seconds, state):
    """Finalise the simulation state for the last recorded step."""
    state.software_development_rate = 0
    return state
```

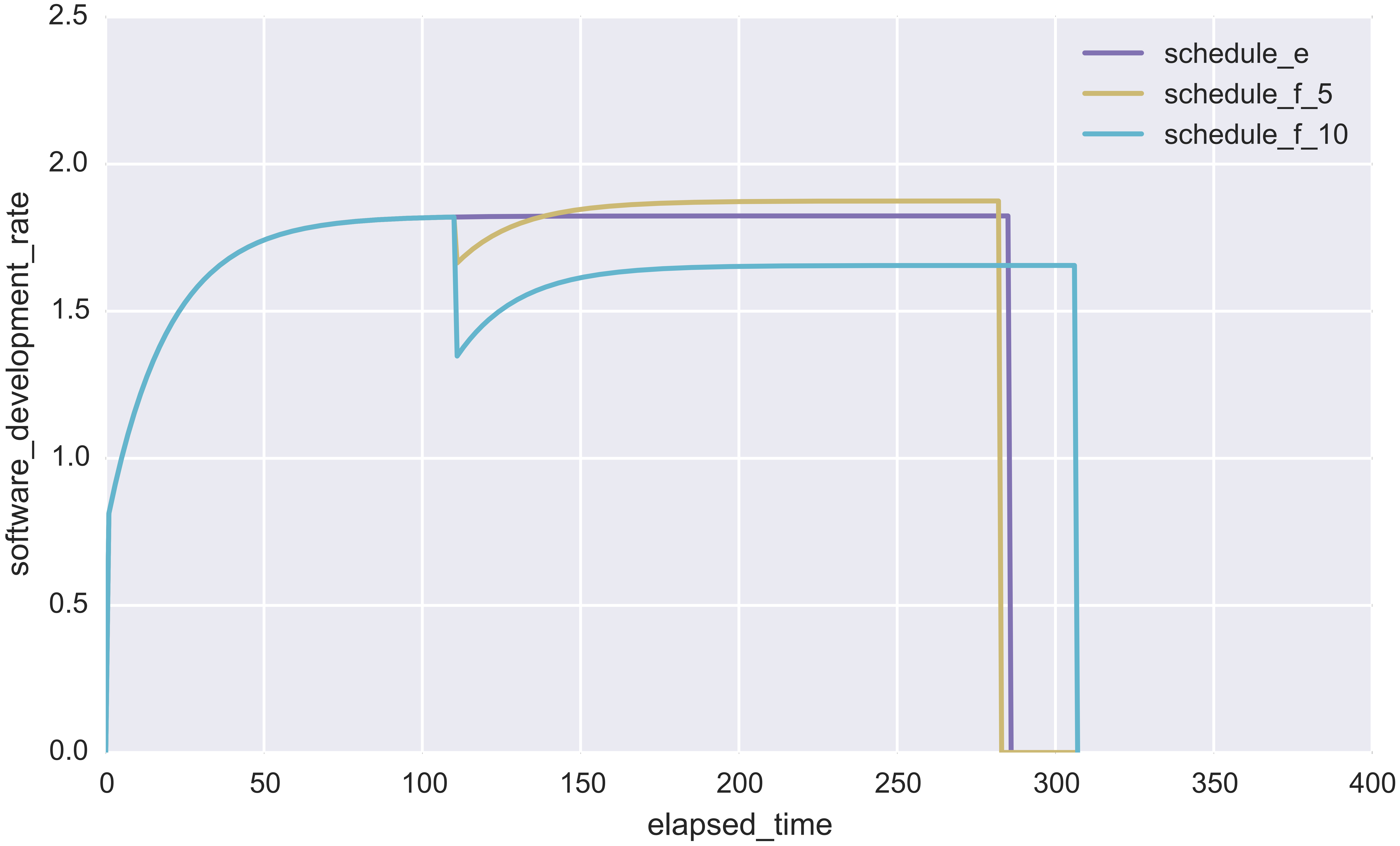

schedule_e
schedule_f_5
software_development_rate
elapsed_time
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
350
400



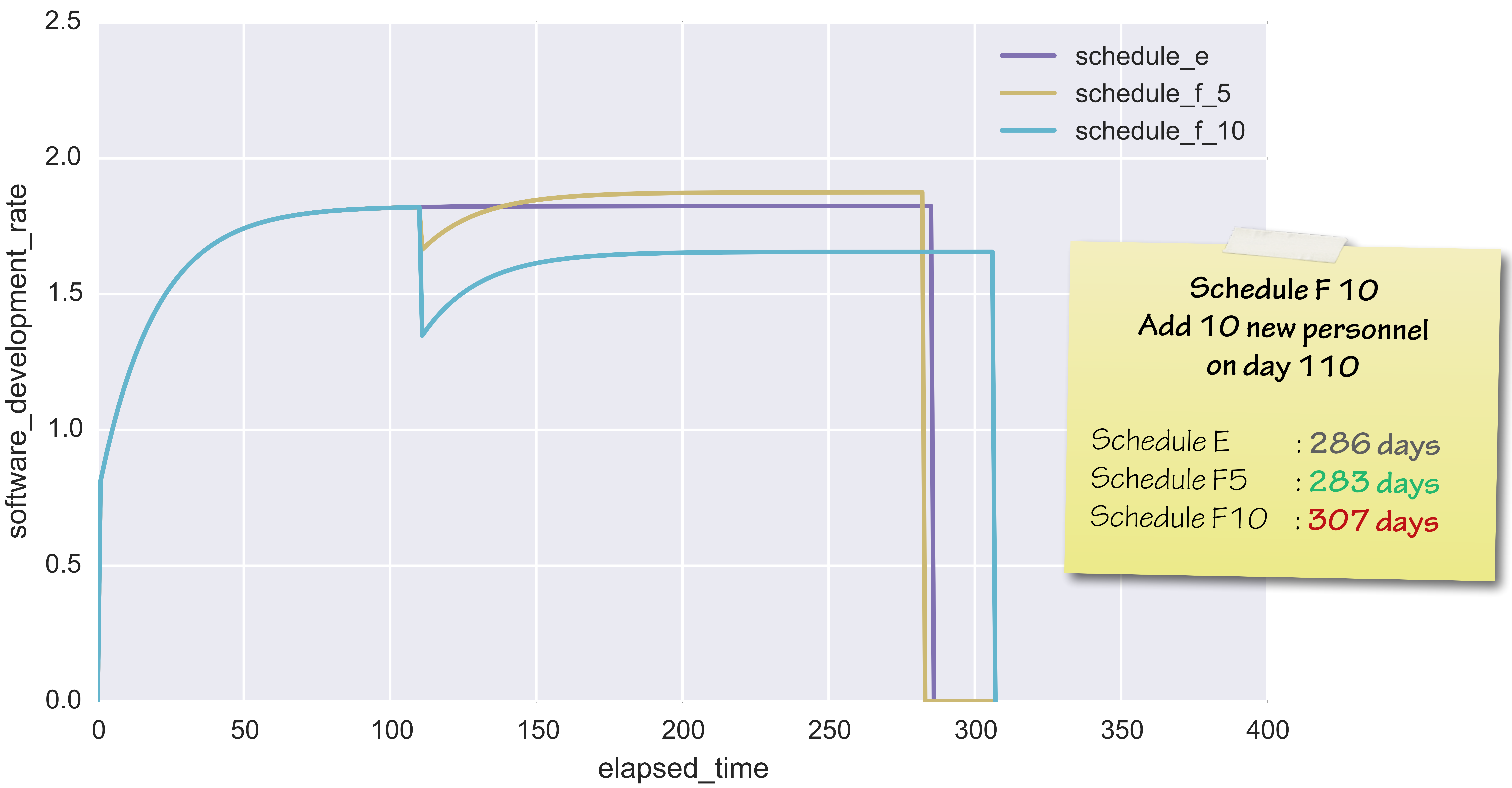
2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_e
schedule_f_5
Schedule F 5
Add 5 new personnel
on day 110
Schedule E : 286 days
Schedule F5 : 283 days

Fred Brooks was WRONG!

Actually…



schedule_e
schedule_f_5
schedule_f_10
software_development_rate
elapsed_time
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
350
400


2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_e
schedule_f_5
schedule_f_10
Schedule F 10
Add 10 new personnel
on day 110
Schedule E : 286 days
Schedule F5 : 283 days
Schedule F10 : 307 days

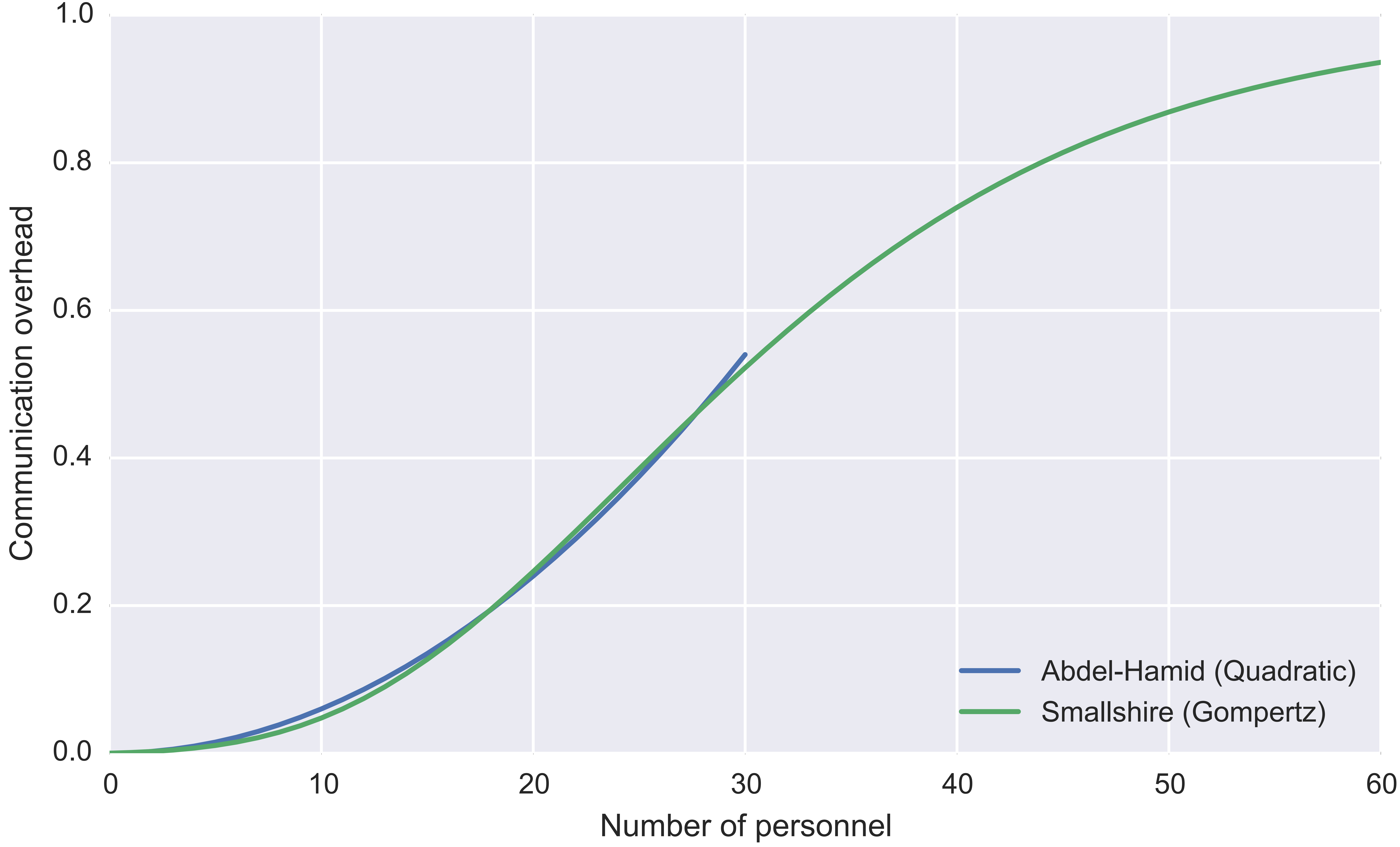
Fred Brooks was RIGHT!

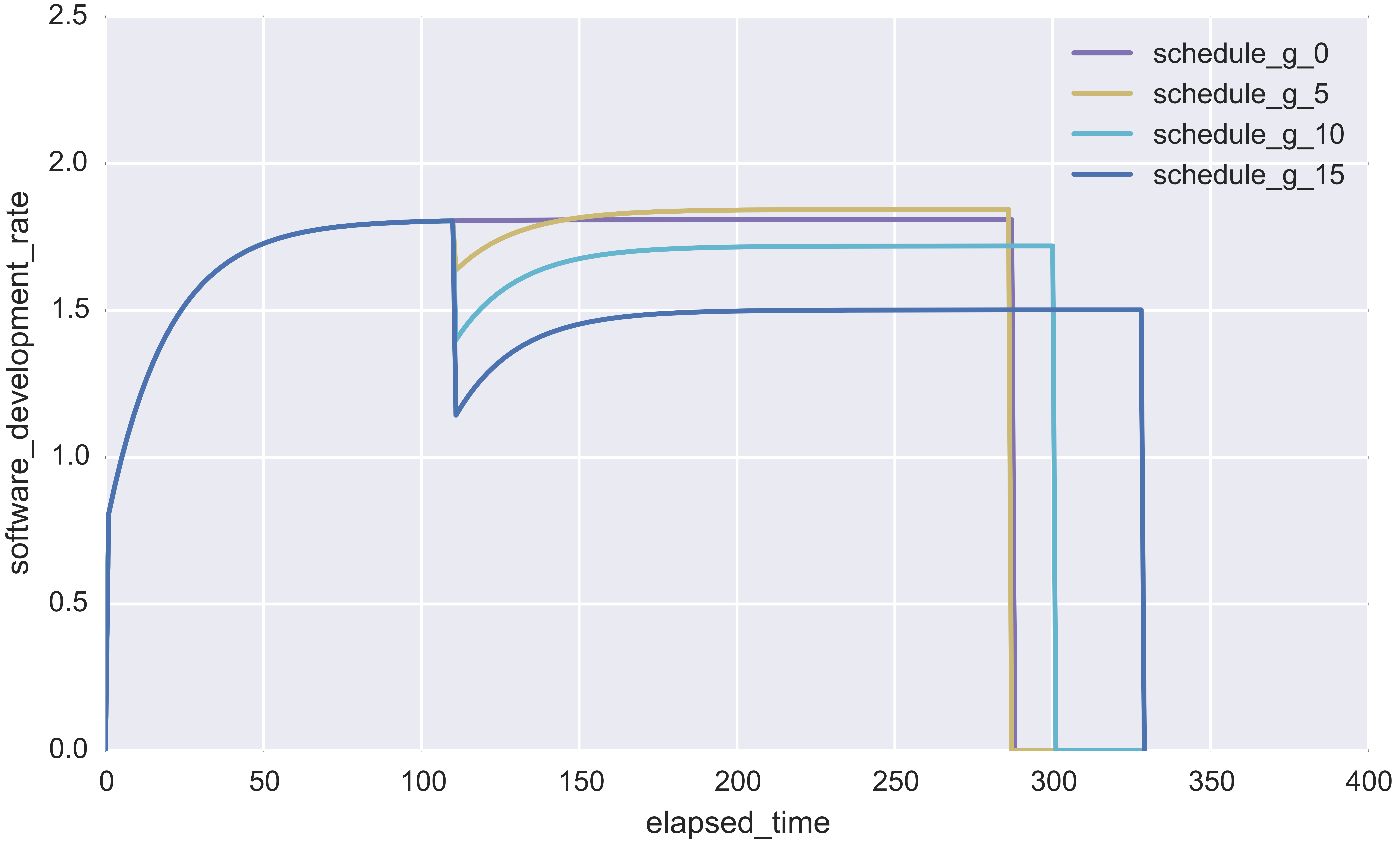
```
ValueError: Communication overhead
proportion personnel number 34.9 out
of range
```

```
ValueError: Communication overhead proportion personnel number 34.9 out of range
```

Model limitations

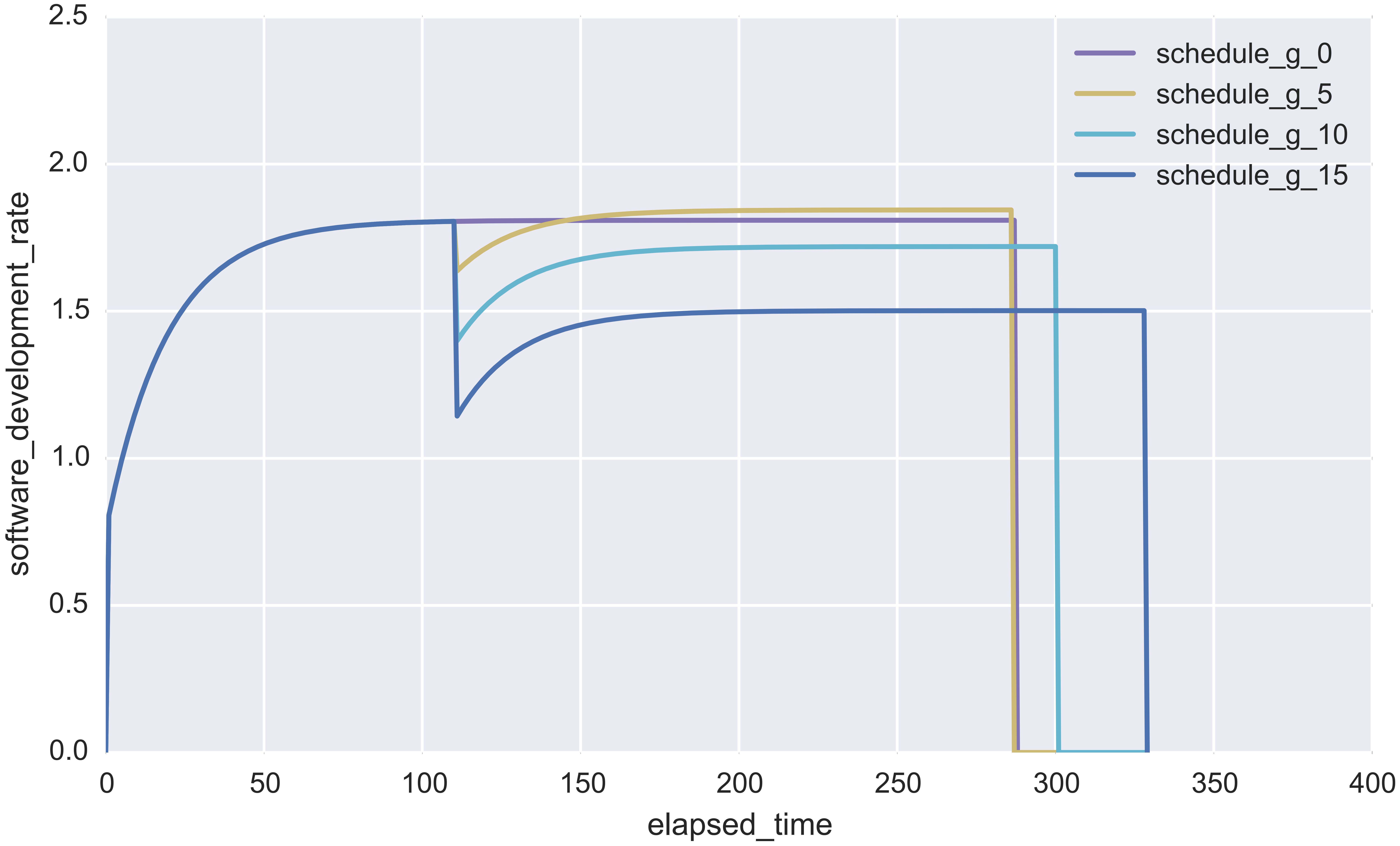
Prevent extrapolation outside reasonable bounds!


1.0
0.8
0.6
0.4
0.2
0.0
Communication overhead
0
10
20
30
40
50
60
Number of personnel
Abdel-Hamid (Quadratic)
Smallshire (Gompertz)

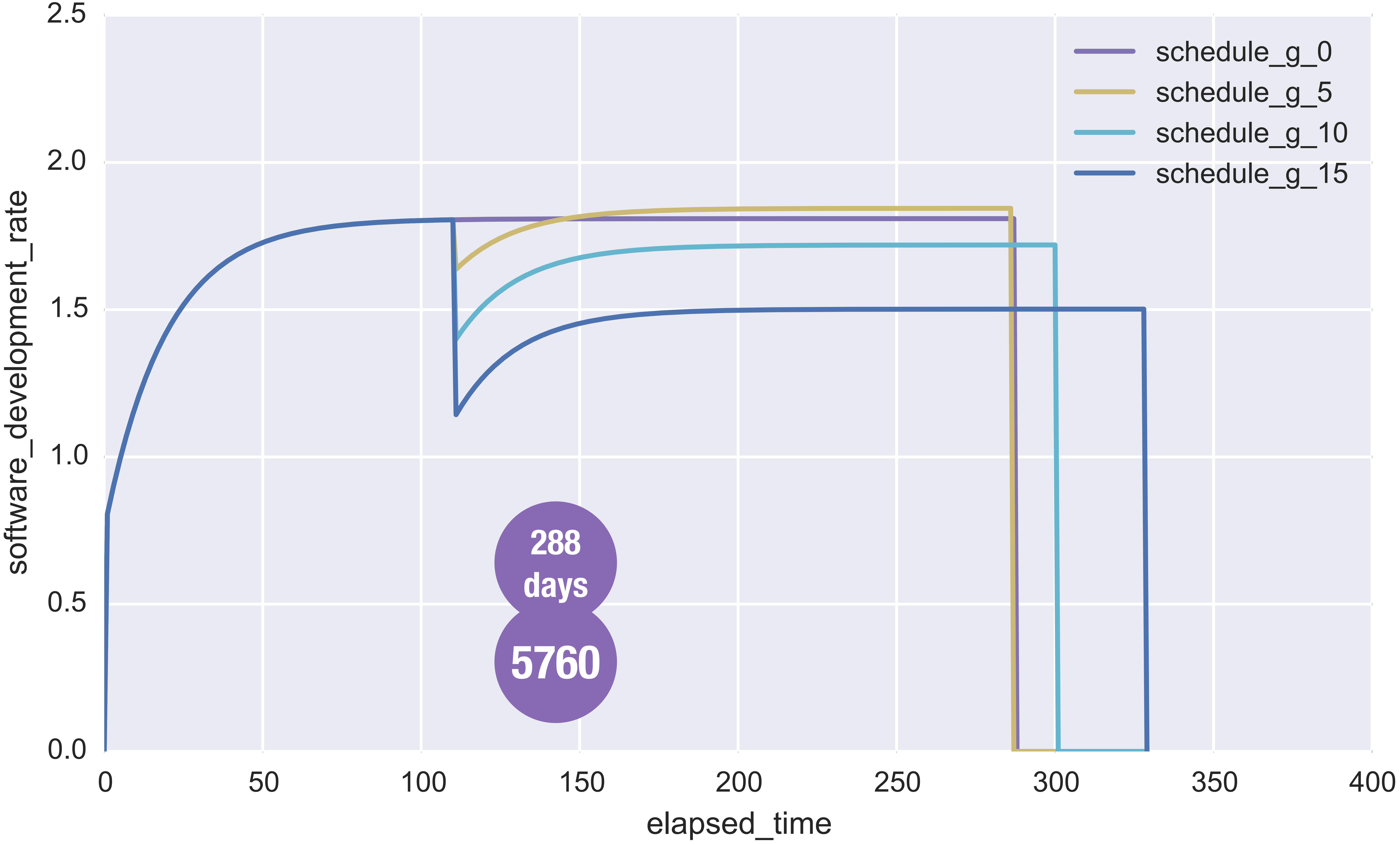


2.5
2.0
1.5
1.0
0.5
0.0
software_development_rate
0
50
100
150
200
250
300
350
400
elapsed_time
schedule_g_0
schedule_g_5
schedule_g_10
schedule_g_15

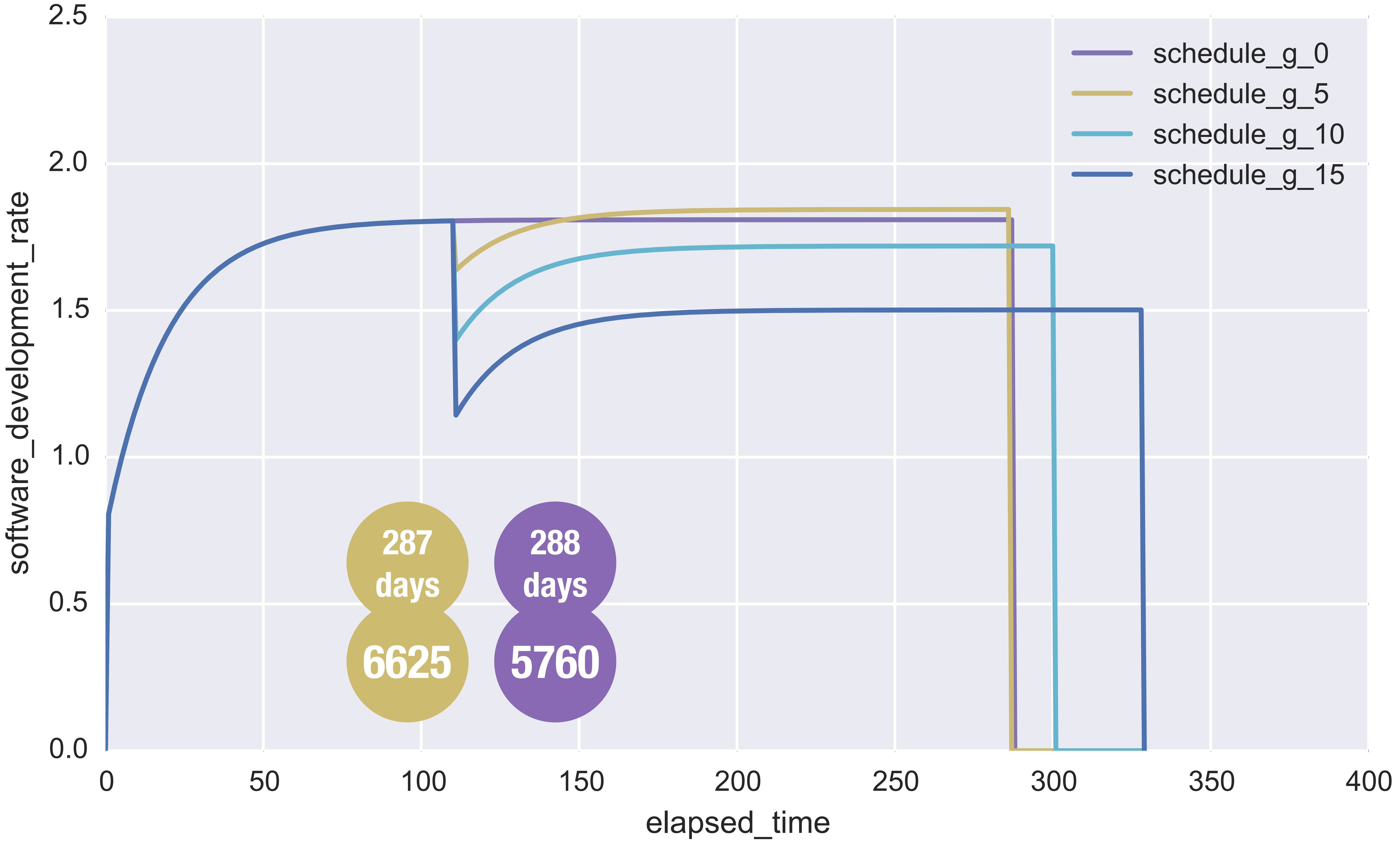
What about cost?



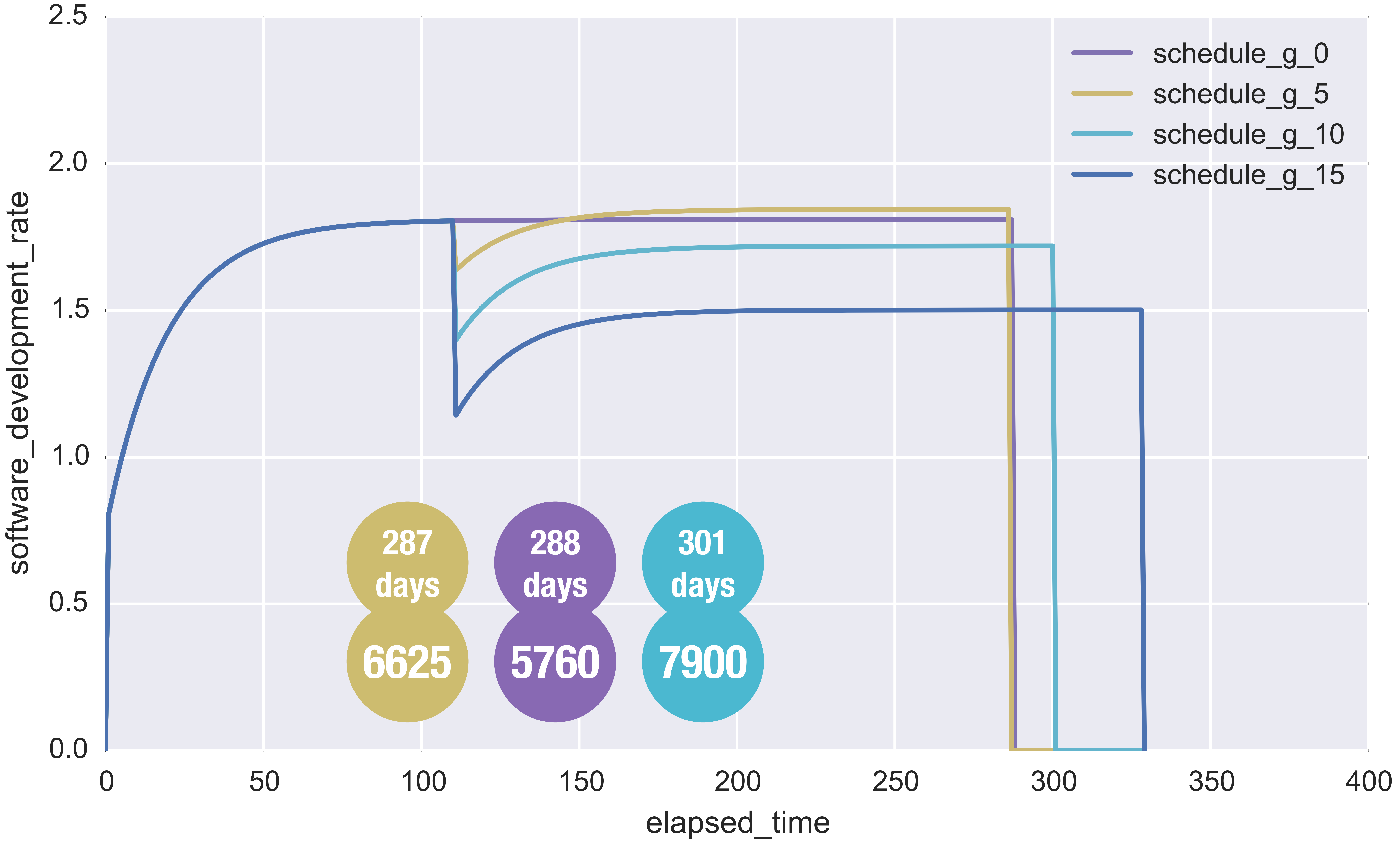
What about cost?



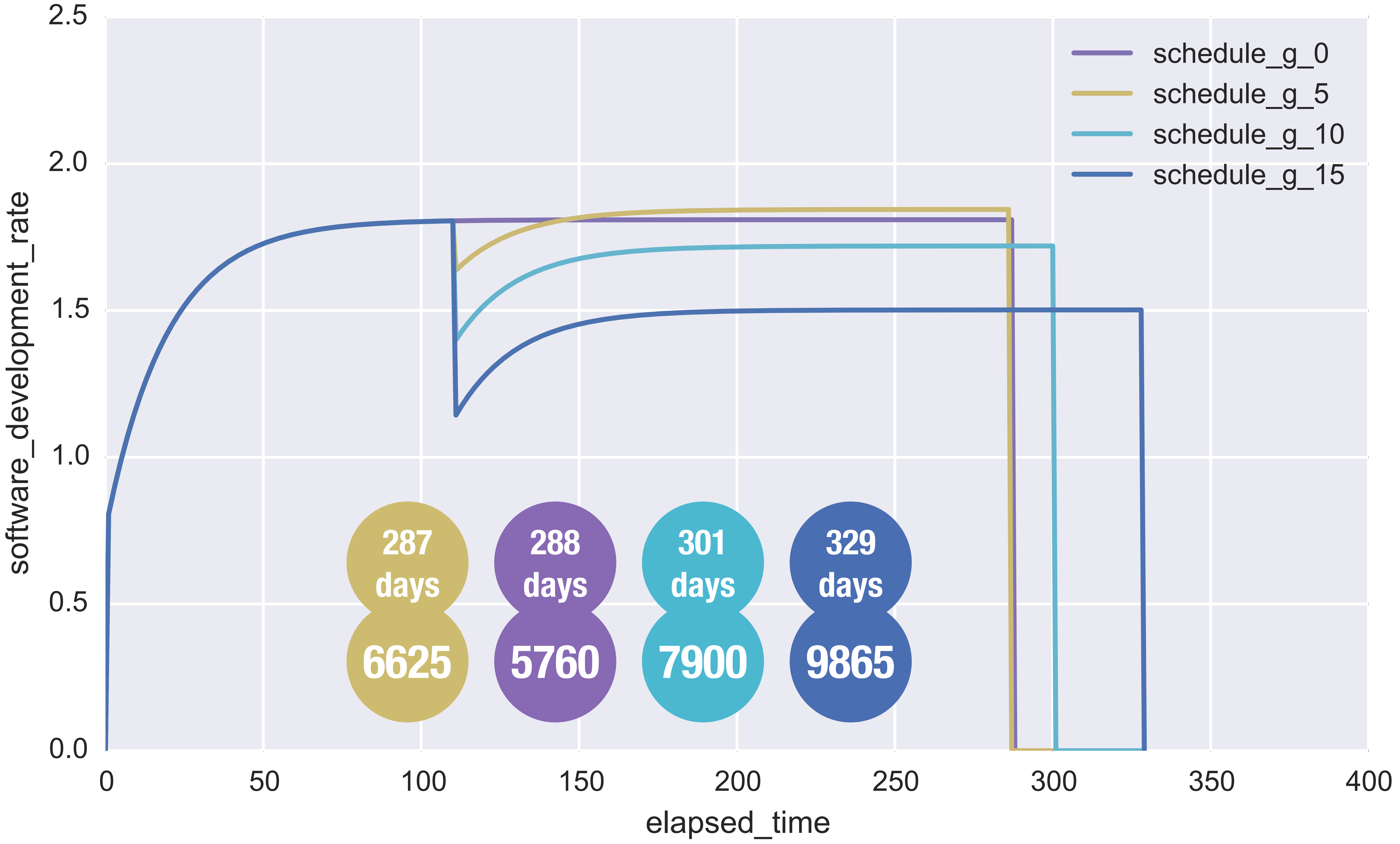
What about cost?



What about cost?



What about cost?



Modelling system growth
How many people work on your system?

1

Predicting project progress
How many people should work on your system?

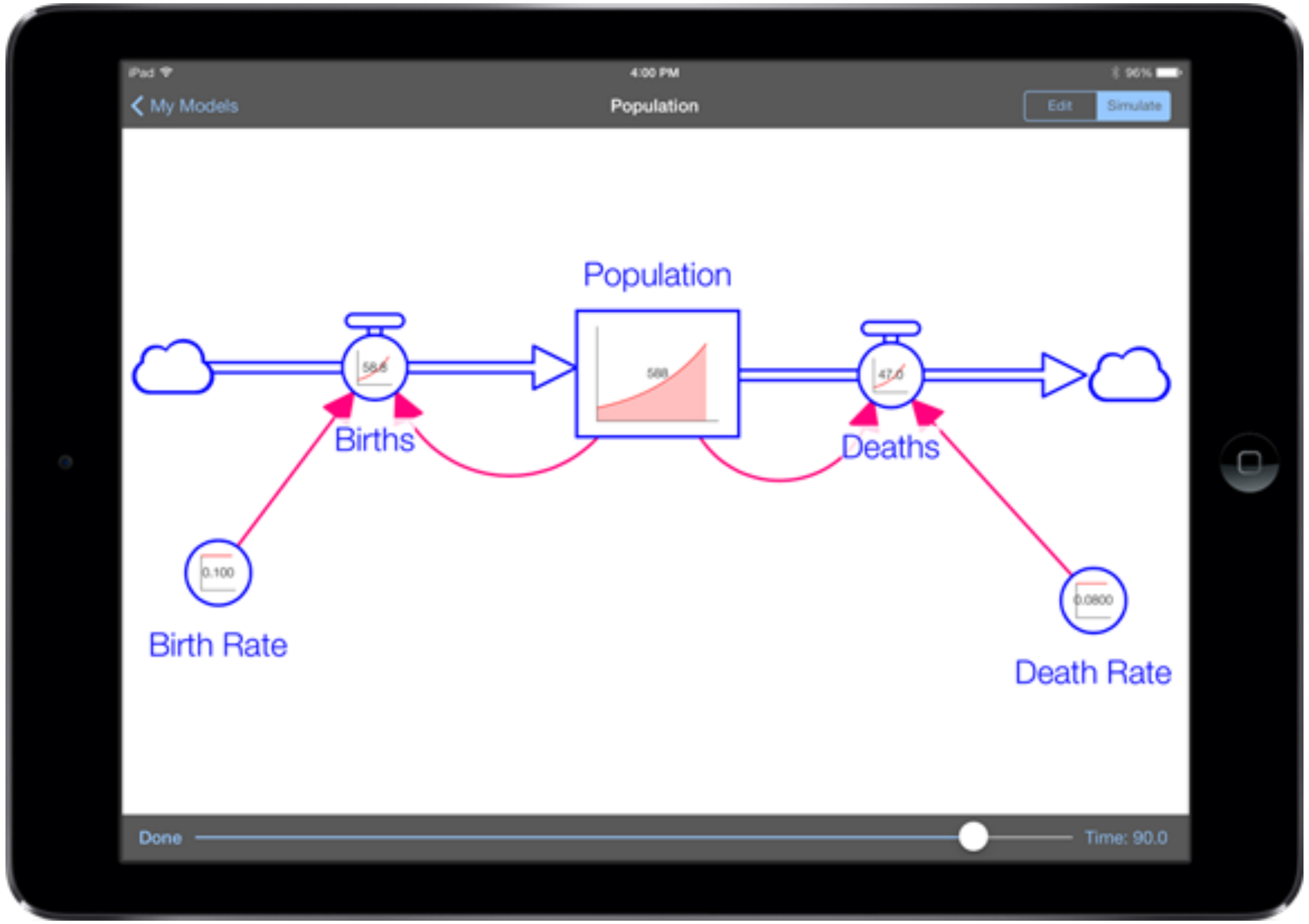
2

Software process dynamics
How can you construct models and run simulations?

3

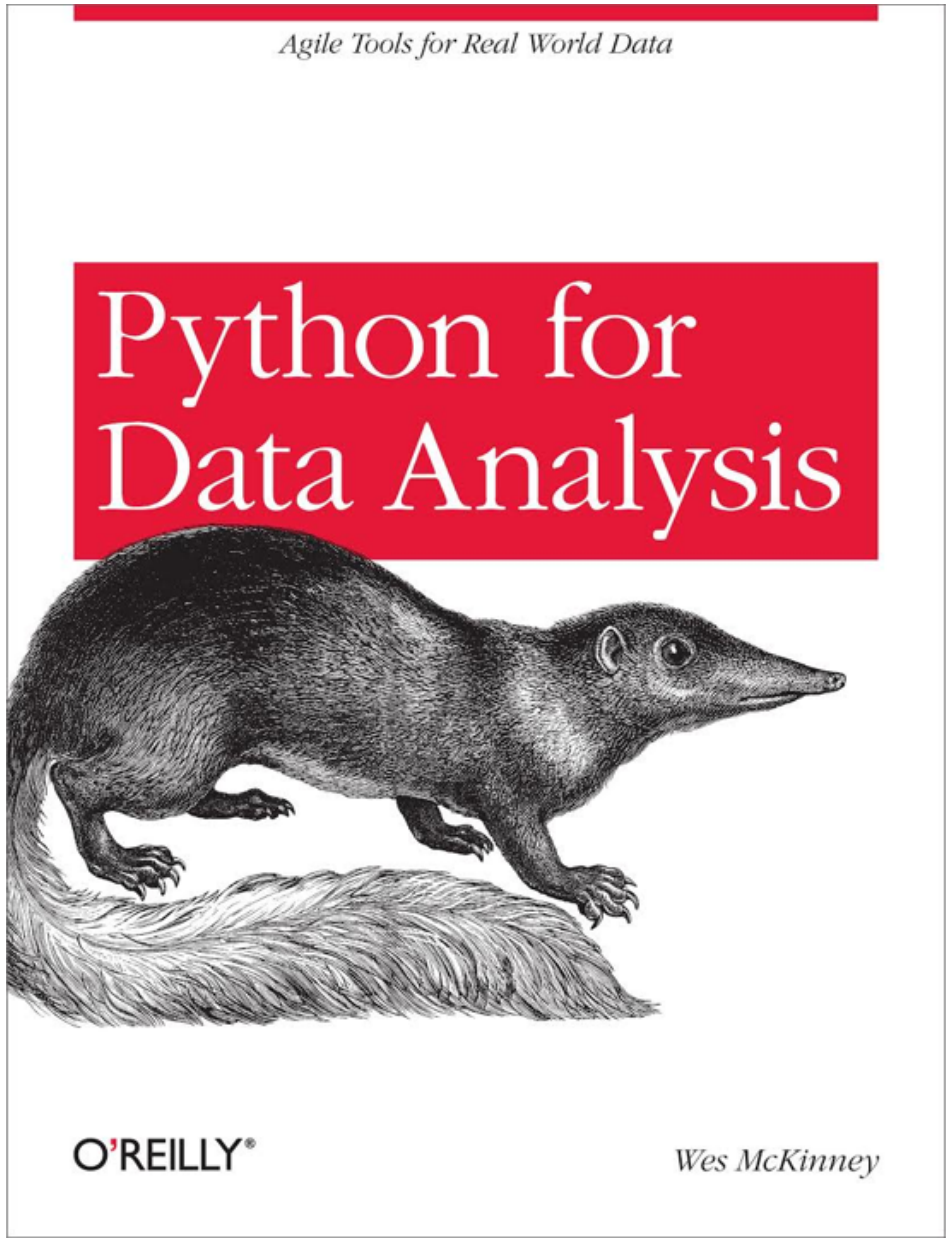
Simulation Tools

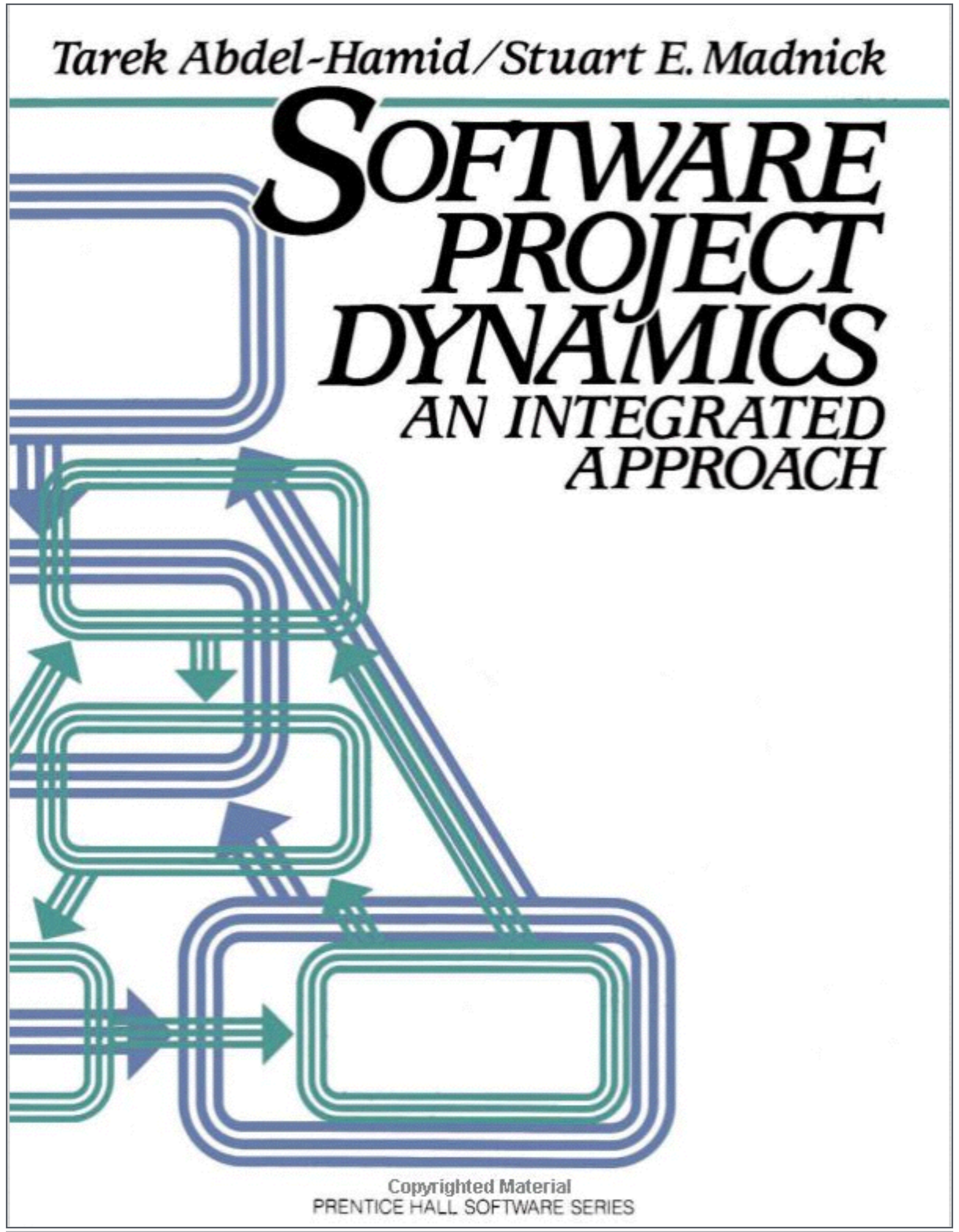
- iThink / Stella
- Vensim
- Excel
- PowerSim
- Simile
- *etc*

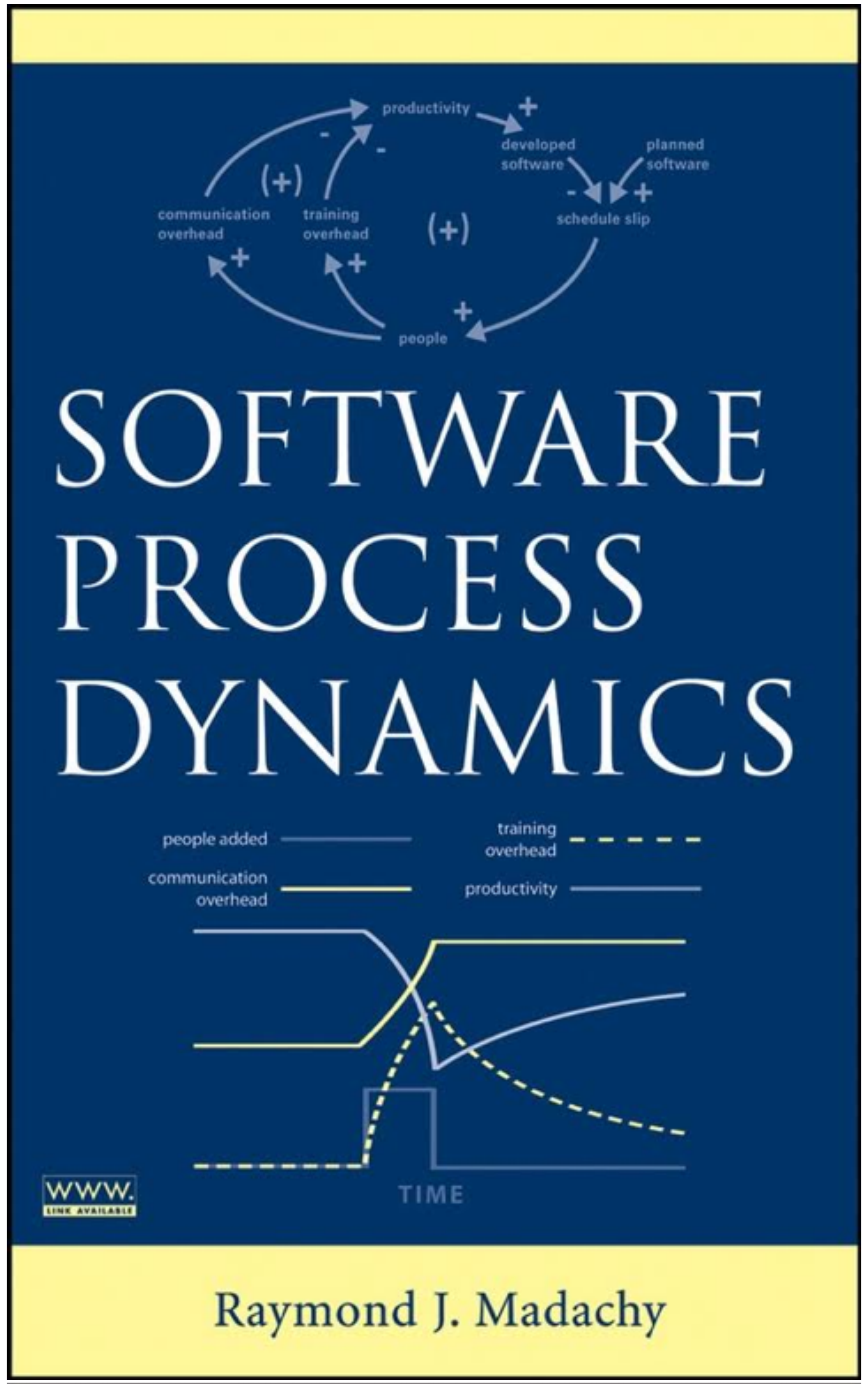


Program it yourself

- Python
- Matplotlib (charting)
- Pandas (tables, time-series)
- Numpy (fast numerics)




Tarek Abdel-Hamid/Stuart E. Madnick
SOFTWARE PROJECT DYNAMICS
AN INTEGRATED APPROACH
Copyrighted Material
PRENTICE HALL SOFTWARE SERIES


productivity
developed software
planned software
communication overhead
training overhead
schedule slip
people
(+)
(+)
SOFTWARE PROCESS DYNAMICS
people added
training overhead
communication overhead
productivity
TIME
Raymond J. Madachy

Model implementation

`https://github.com/sixty-north/brooks`

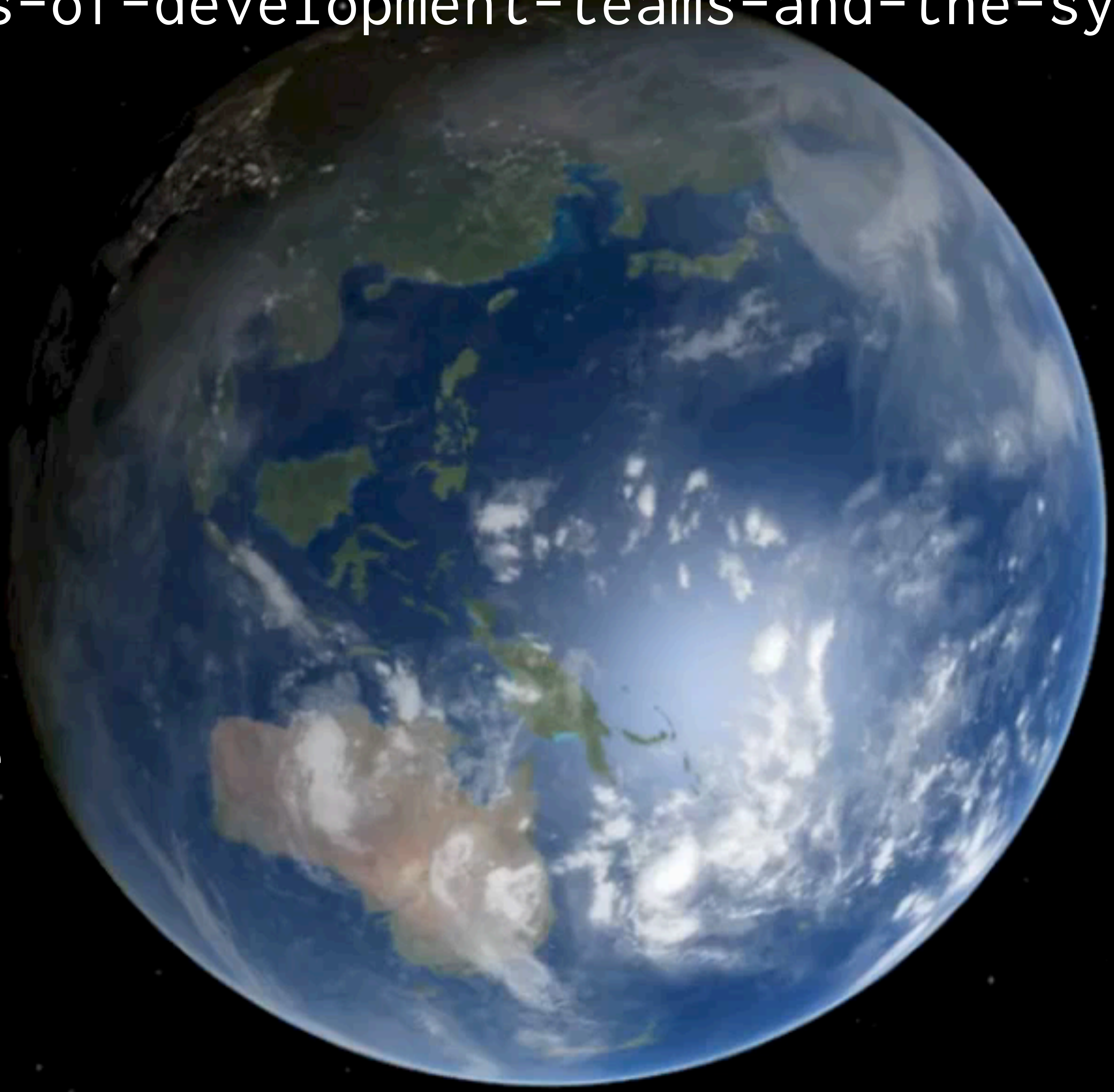
Software Process Dynamics

Sure it's fun! But is it useful?

Software Process Dynamics

- Secure **buy-in** for modelling and models
- **Parameterise** the model
- As **simple** as possible, but no simpler
- Be clear on system boundary / **assumptions**
- **Experiment!**
- **Discuss** results

http://sixty-north.com/blog/predictive-models-of-development-teams-and-the-systems-they-build



Thank you!

Robert Smallshire

@robsmallshire

SixtyNORTH

@sixty_north

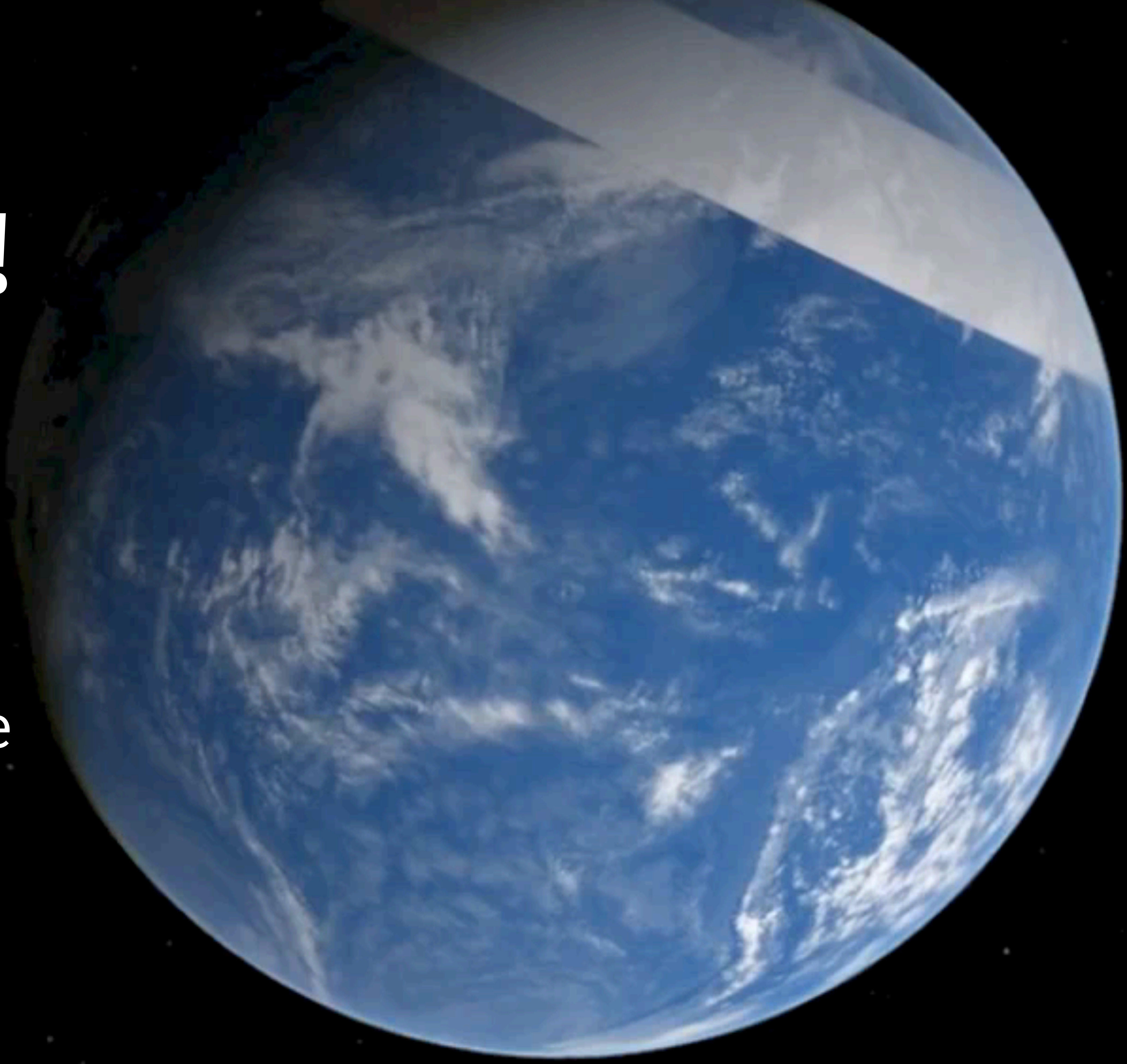
http://sixty-north.com/blog/predictive-models-of-development-teams-and-the-systems-they-build

Thank you!

Robert Smallshire

@robsmallshire

http://sixty-north.com/blog/predictive-models-of-development-teams-and-the-systems-they-build



Thank you!

Robert Smallshire
@robsmallshire

SixtyNORTH

@sixty_north

http://sixty-north.com/blog/predictive-models-of-development-teams-and-the-systems-they-build

Thank you!

Robert Smallshire

@robsmallshire

SixtyNORTH