CT561: Systems Modelling & Simulation

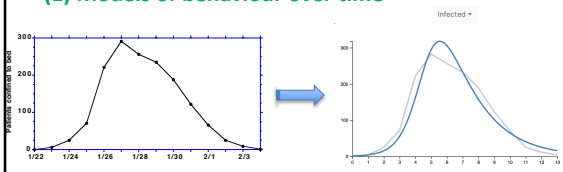
Lecture 6: Formulating Effects and the Rework Cycle

Prof. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.
https://github.com/JimDuggan/SDMR

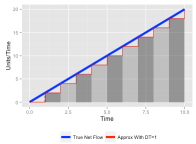
1

Course Recap

(1) Models of behaviour over time

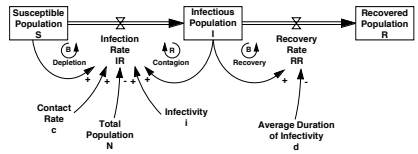


(2) Integration (Calculus)

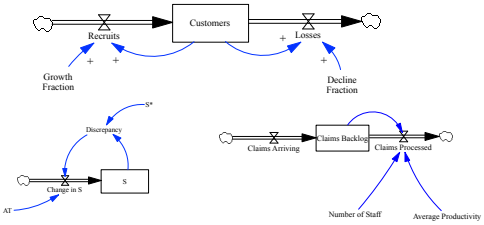


$$S_t = S_{t-dt} + NF_{t-dt} \times DT$$

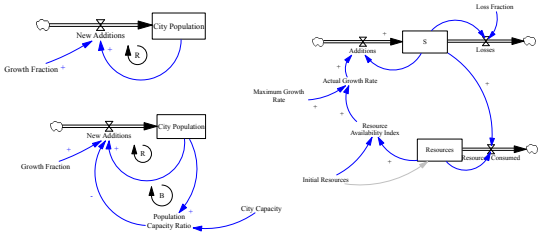
(3) Feedback (Positive & Negative)



(4) Formulating Flows (4)



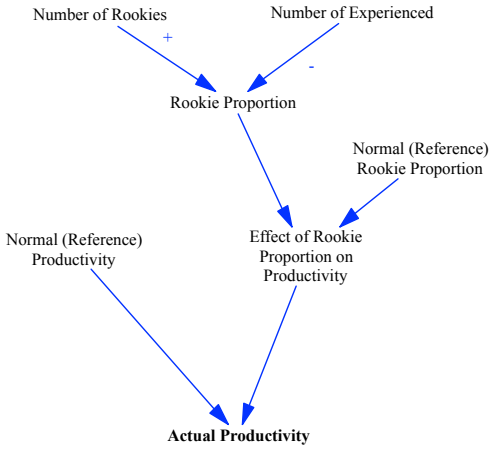
(5) Limits to Growth

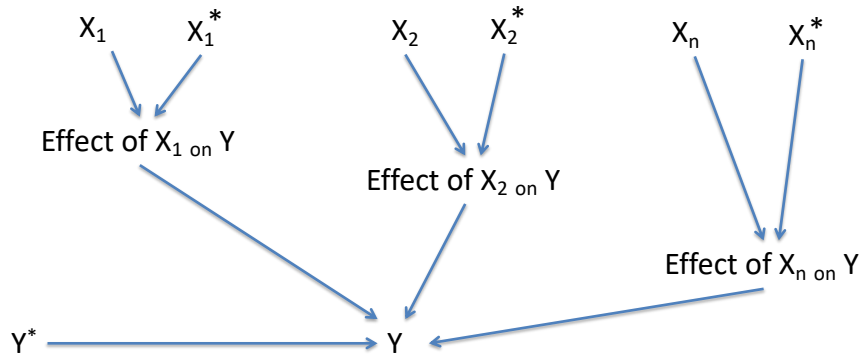


2

Formulating Effects

- An important building block for models is to capture how variables influence one another over time.
- System dynamics offers a convenient structure for modeling effect variables (Sterman 2000).
- These can be used to help simulate more complex feedback structures



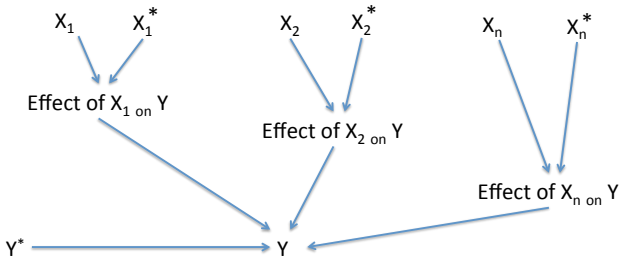


$$Y = Y^* \times Effect\,(X_1 on\ Y) \times \ldots \times Effect(X_n on\ Y)$$

$$Effect\,(X_i\ on\ Y) = f\left(X_i / X_i^*\right)$$

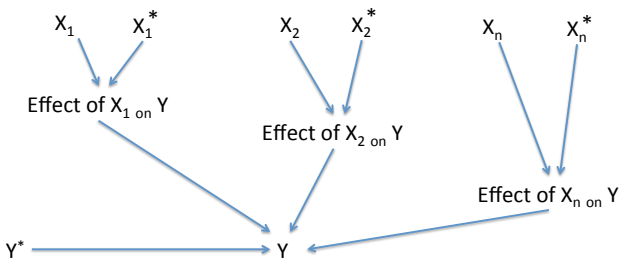
Effects structure (1)

- There is a variable *Y* that is the dependent variable of a causal relationship, and this depends on a set of *n* independent variables (X_1, X_2, …, X_n)
- The variable *Y* has a reference value Y^*, and this is multiplied by a sequence of *effect functions* that are calculated based on the normalized ratio of (X_i/X_i^*), where X_i^* is the reference value, and X_i is the actual value.



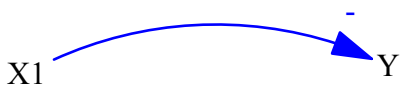
Effects structure (2)

- The effect function (y-axis) has the normalized ratio (X/X^*) on its x-axis, and always contains the point (1,1) although the function itself can be either linear or non-linear around this point.
- This point (1,1) is important for the following reason: if *X* equals its reference value X^*, then the effect function will be 1, and therefore *Y* will then equal its reference value Y^*.



Challenge 6.1

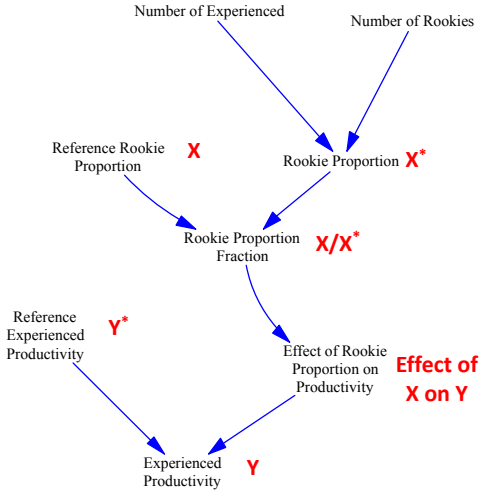
- Work through the logic of the effect structure, starting with the following structures.



7

Software Engineering Example

- Reference productivity is 100 loc/person/day
- This assumes a reference rookie proportion in the team (say 20%)
- If we have exactly 20% Rookies
  - Actual Productivity = Reference Productivity
- If we have > 20% Rookies
  - Experienced Productivity < Reference Productivity
- If we have < 20% Rookies
  - Experienced Productivity > Reference Productivity



8

The equation (for experienced productivity)

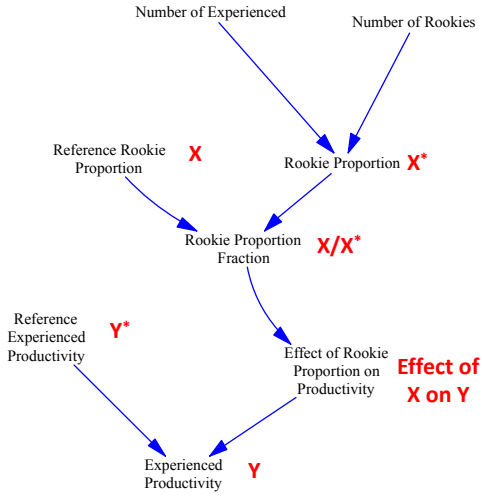
Experienced Productivity = Reference Experienced Productivity * Effect of Rookie Proportion on Productivity

Reference Experienced Productivity (Y^*)	Reference Rookie Proportion (X^*)	Actual Rookie Proportion (X)	X/X^*	Effect Multiplier	Actual Experienced Productivity (Y)	Comments
100	20%	20%	1	1	100	No effect on experienced productivity, as the benchmark value of 100 is measured when we have 20% rookies in the team.
100	20%	40%	> 1	< 1	< 100	Effect < 1 and experienced productivity goes down, as we have more rookies which will require increased feedback from experienced productivity
100	20%	10%	< 1	> 1	> 100	Effect > 1 and experienced productivity rises, as experienced programmers have more time to focus on coding efforts

9

The Effect Equation

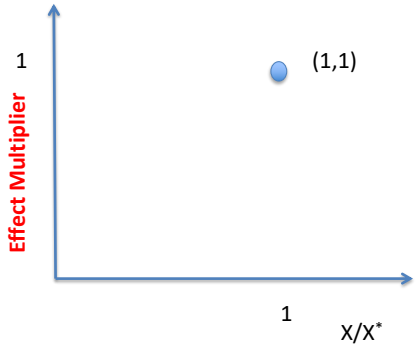
- Actual Productivity = Reference Productivity * Effect of Rookie Proportion on Productivity
- Effect of X on Y = $F(X/X^*)$
- Normalised Value
- *When $X = X^*$, $F(X) = 1$*
- *X^* and Y^* are reference values*



10

Example

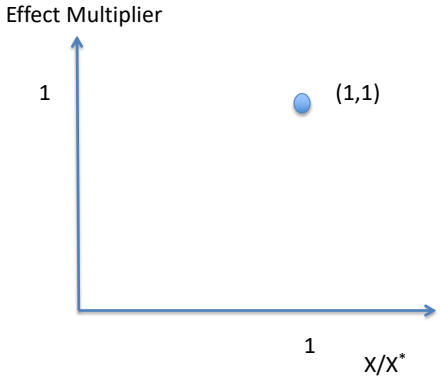
- X = Rookie Proportion
- X^* = Reference Rookie Percentage
- Impact on productivity?
- (1,1) is always on the line



11

Thinking about the effects…

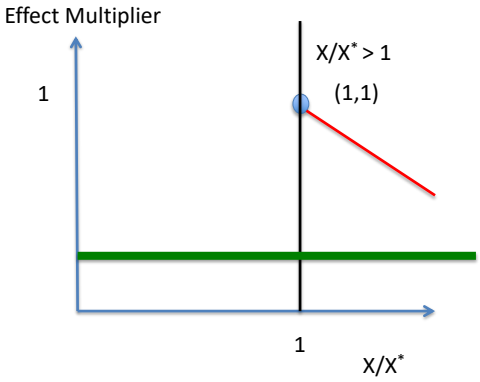
- X = Actual Rookie Proportion
- X^* = Reference Rookie Proportion (i.e. the number at which our experienced productivity is at its reference value)
- Scenarios:
  - If $X > X^*$, Effect?
  - If $X < X^*$, Effect?



12

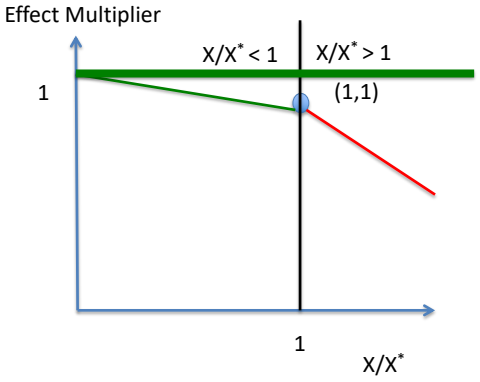
Sketching the relationship, More rookies than reference value

- $X > X^*$
  - We have more Rookies than our target level
  - This will reduce our experienced productivity
  - More work to train rookies
  - Effect will be lower than 1
  - Decide on minimum value (0.25)

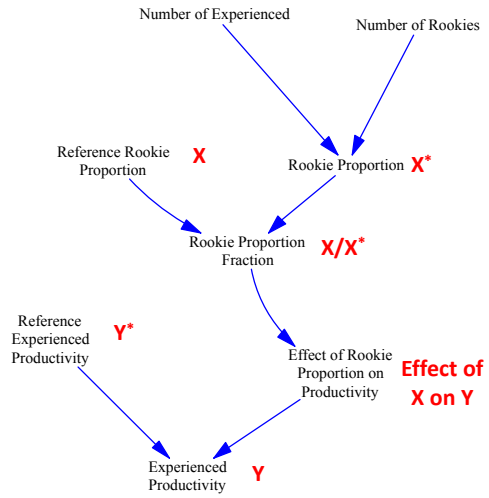


Sketching the relationship, Less rookies than reference value

- $X < X^*$
  - We have less Rookies than our target level
  - This will increase our experienced productivity
  - Less work to train rookies
  - Effect will be greater than 1
  - Decide on a maximum value (1.8)



Equations



Reference Rookie Proportion = 0.2

Reference Experienced Productivity = 100

Number of Experienced = 100

Number of Rookies = 20

Rookie Proportion = Number of Rookies / (Number of Experienced + Number of Rookies)

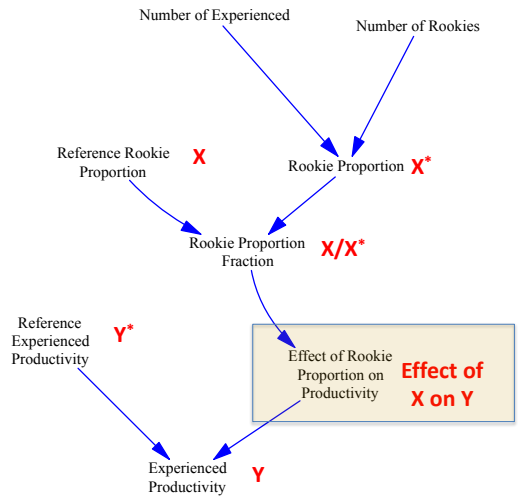
Rookie Proportion Fraction = Rookie Proportion / Reference Rookie Proportion

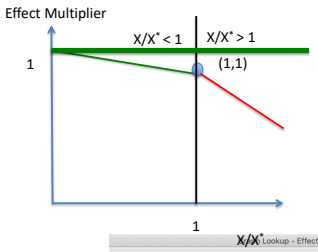
Effect of Rookie Proportion on Productivity = WITH LOOKUP(Rookie Proportion Fraction
, ([(0,0)-(5,2)],(0,1.8),(1,1),(1.5,0.8),(2,0.7),(3,0.5),(4,0.45)
))

Experienced Productivity = Reference Experienced Productivity * Effect of Rookie Proportion on Productivity

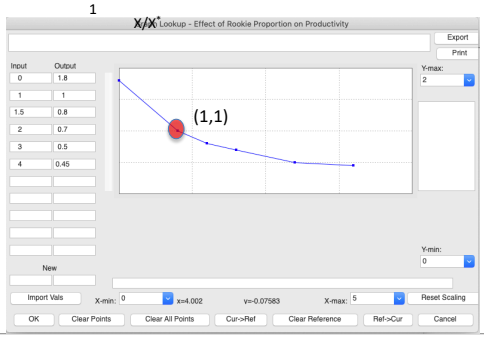
15

Sample Effect Function (interpolated)





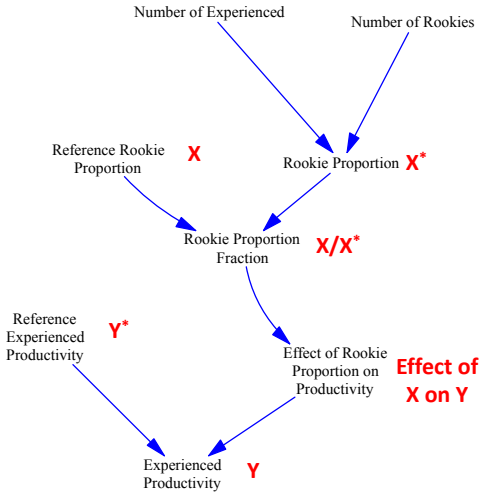
Effect of Rookie Proportion on Productivity = **WITH LOOKUP**(Rookie Proportion Fraction, ([(0,0)(5,2)], (0,1.8), (1,1), (1.5,0.8), (2,0.7), (3,0.5), (4,0.45)))



16

Challenge 6.2

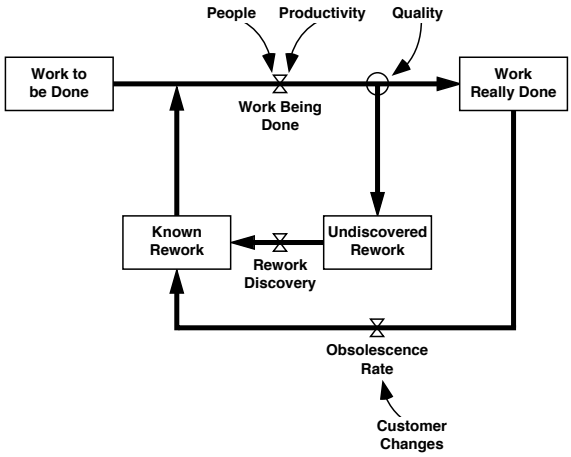
- Explore the model in Vensim and observe the impact of changing X for the three scenarios
- Extend the model to include the following effect variables on experienced productivity:
  - Average time to promotion
  - Average length of working week



17

The Rework Cycle (Sterman 2000)

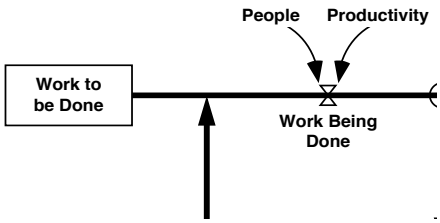
- Productivity of workers
- Capacity of team
- Key Variables
  - Workflow
    - Work to do
    - Work completed
    - Rework: Undiscovered and Known
  - Resources
    - Experience
    - Speed
    - Quality



18

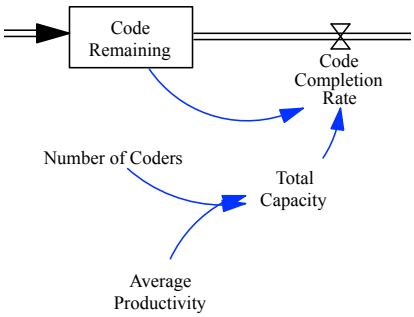
Work Being Done… (Flow)

- WBD = Resource * Productivity
- Production = Labour Force * Average Productivity
- (Units/Period) = (People) * ((Units/Period)/Person)
- Labour Force * Average Productivity also gives the system's ***Capacity***



19

Software Construction



Code Remaining= INTEG (
New Code Added-
Code Completion Rate,
0)

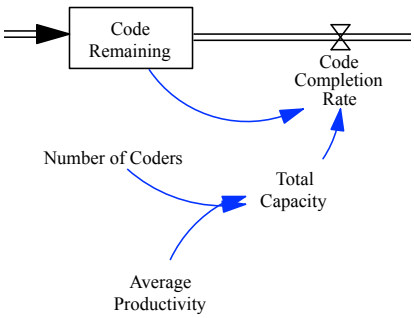
Average Productivity= 50

Number of Coders = 10

Total Capacity= Number of Coders*Average Productivity

20

Formulating the Outflow



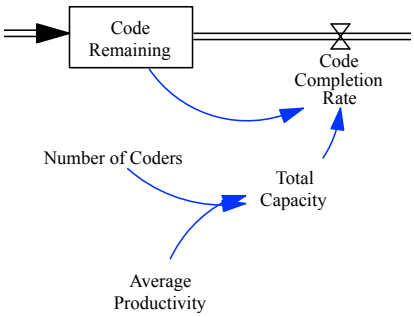
- Stock cannot go negative!
- We cannot produce more than our capacity

Code Remaining	Capacity	Outflow
1,000	200	200
50	200	50

Outflow = f(Code Remaining, Capacity)

21

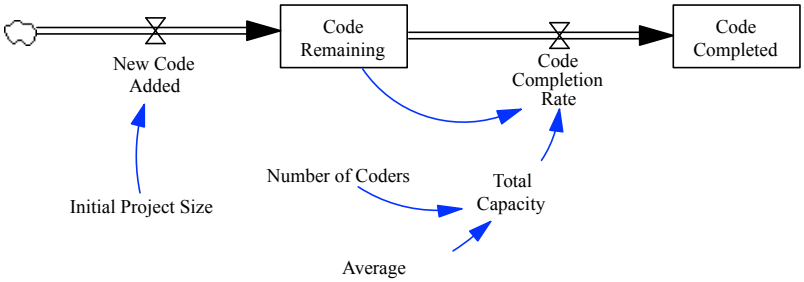
Finalising Outflow (simpler version of "first order control")



Completion Rate= MIN(
Code Remaining,
Total Capacity)

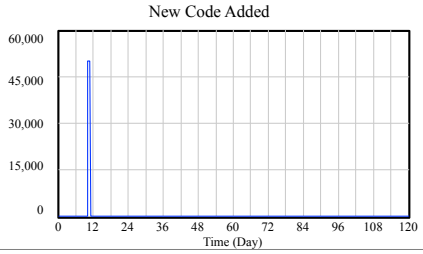
22

Extended Model



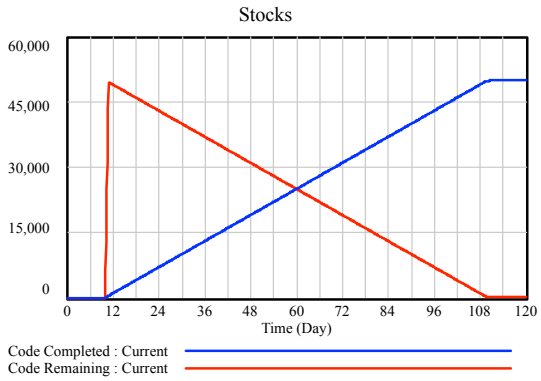
Initial Project Size = 50000

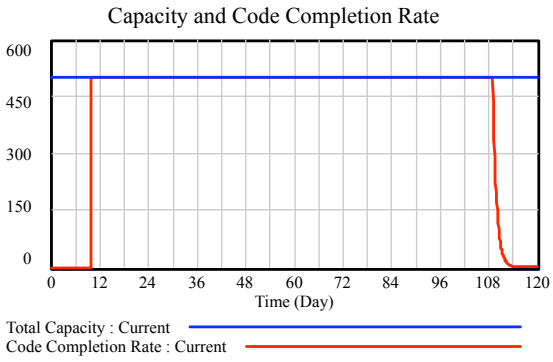
New Code Added = PULSE(10, 1) * Initial Project Size



23

Simulation Output

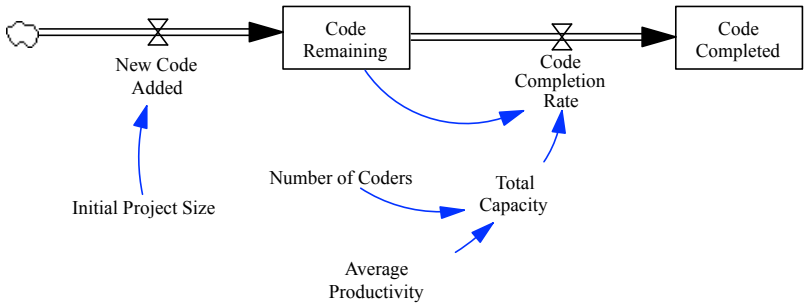




24

Challenge 6.3

- Implement the model in Vensim, and explore its behaviour



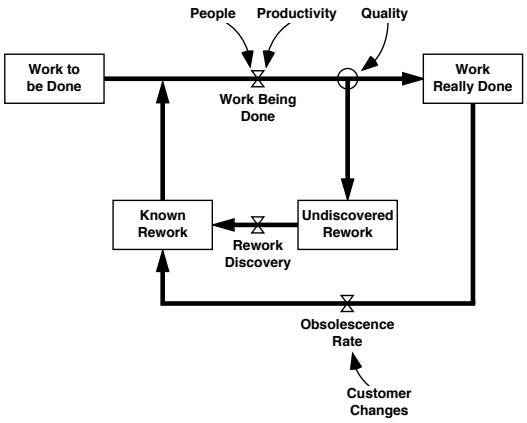
25

Extension… the Rework Cycle

- Introduce the notion of quality
- All processes have a concept of Yield
  - Output/Input

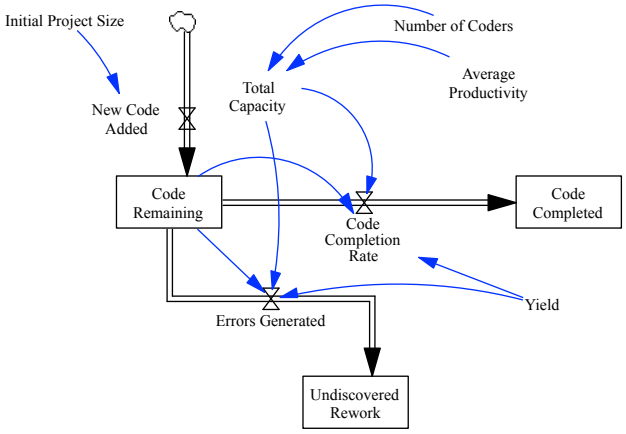
Input	Output	Yield
1,000	879	87.9%

Input	Output	Defects
1,000	879	12.1%



26

An initial model

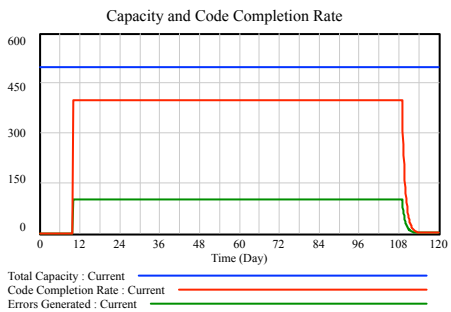


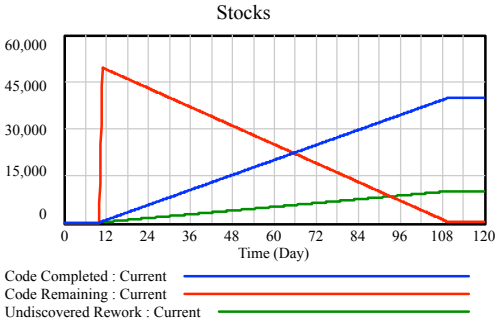
Code Completion Rate=
min(Code Remaining,
Total Capacity)*Yield

Errors Generated=
min(Code Remaining,
Total Capacity) * (1-Yield)

27

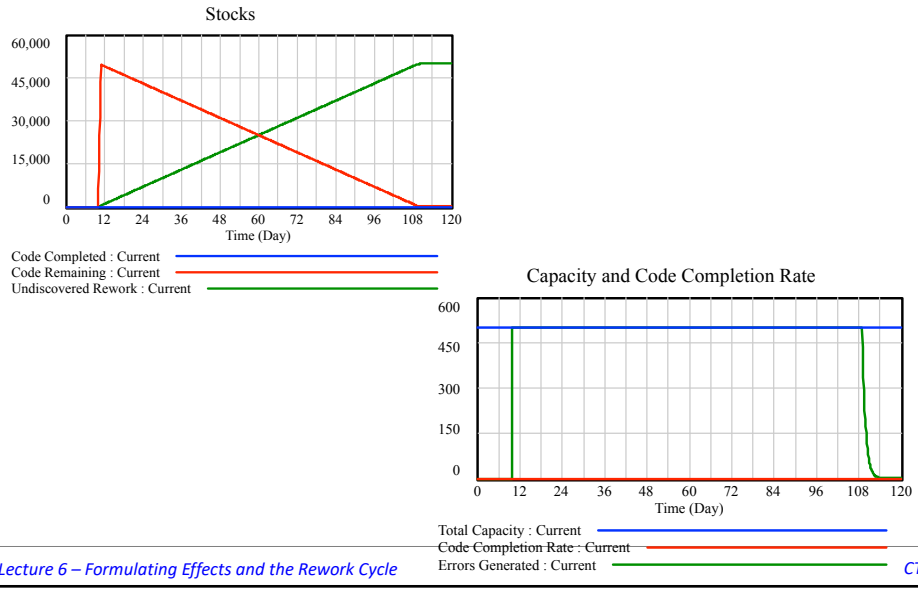
Model Behaviour





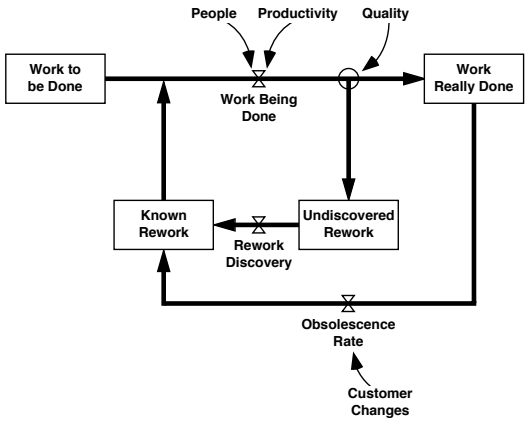
28

Model Test: Set Yield = 0



Challenge 6.4

- Complete a model for the software rework cycle.
- Use fractional decrease rate for rework discovery and customer changes
- Assume 100,000 LOC to be done at the start (no need for inflow into *Work to be Done*).



Summary: Effects and the Rework Cycle

- Effects are an important building block for models is to capture how variables influence one another over time.
- Rework cycle and important structure for modelling projects
- Productivity and quality amongst different groups can also be applied to the rework cycle.