



University of Southern California  
**Center for Software Engineering**

---

# CS599 Software Process Modeling

**Ray Madachy**

**December 7, 1999**

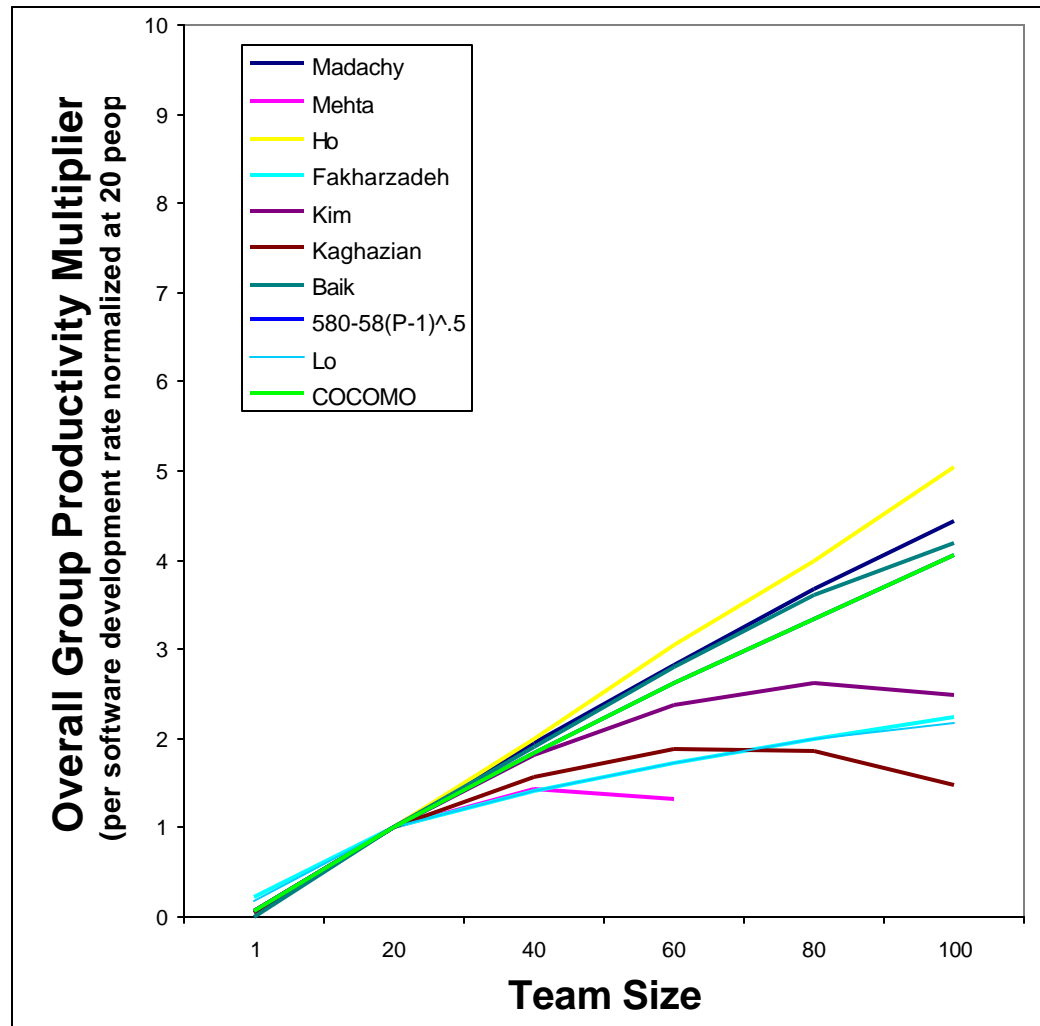


# Outline

- Homework review
- Learning curve
- Resource allocation
- Litton modeling applications overview
- Course outline wrap-up, Q&A
- Future work in the field
- Extra credit problems
- Grades
- Class scheduling
- Course evaluation



# Brooks's Law Model Comparison



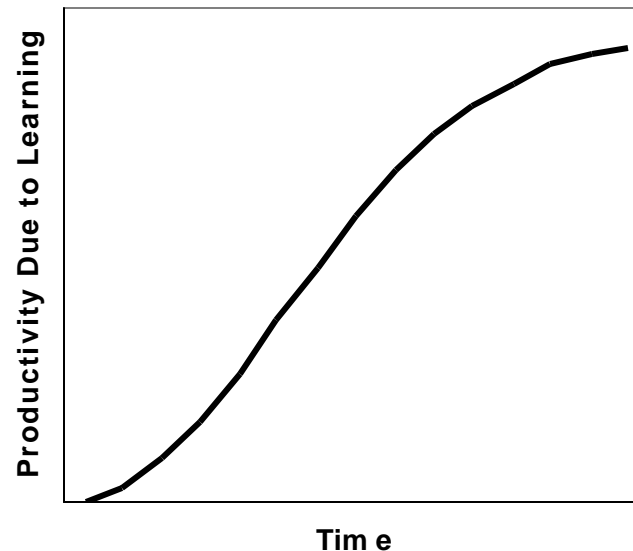


# Learning

- Learning is the act of acquiring skill or knowledge through study, instruction or experience.
- One becomes more proficient at a task after repeated iterations of performing that task.
- In the context of a software process, developers become more productive over the long term due to their accumulated experience.
- The increase in productivity occurs indefinitely.
- A *learning curve* describes the pattern of improvement over time.

# Learning Reference Behavior

- The learning curve shows an initial period of slow learning, a middle period of fast learning followed by a decreasing slope portion that nearly levels out.
- The reduced learning period is due to machine limitations.
  - process constrained by machine-paced tasks such as waiting for compiles, print jobs, computer downtime, network access, etc.





# Representation of Learning

- Possible learning representations:
  - function of absolute time
    - e.g. COCOMO effort multipliers for experience factors
    - typically assumes full-time dedication to learned task
  - function of job percentage completed
  - function of work volume completed
- Learning curve graphs over time are only accurate when time correlates with cumulative output
- Complicated by process disruptions for periods of time that affect productivity
  - e.g. long breaks (vacation, sickness, holidays, etc.) or other work disruptions.

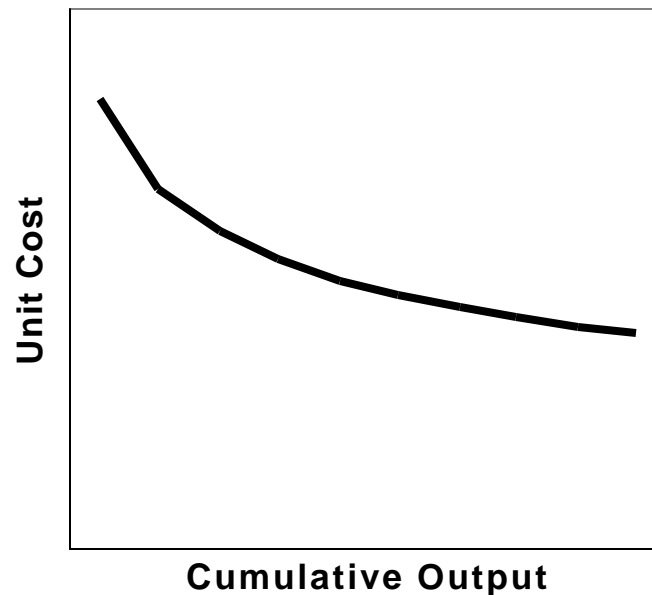


# Representation of Learning (cont.)

- Process bottlenecks may also delay learning, such as when machines pace the work.
- Per [Raccoon 96] handout “When biases affect measurement and time does not correlate with cumulative output, then the units of measurement on the x axis must be translated to cumulative output.”

# Log-linear Learning Curve

- Learning curves are traditionally formulated in terms of the unit costs of production.
- The Log-linear learning curve is expressed by  $y=a(x)^n$ , where  $a$  is the cost of the first unit,  $x$  is the cumulative output, and  $n$  is the learning curve slope.
- Log-linear cost function:

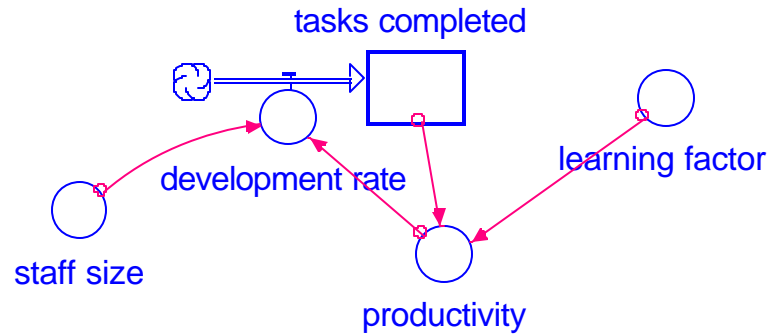




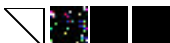
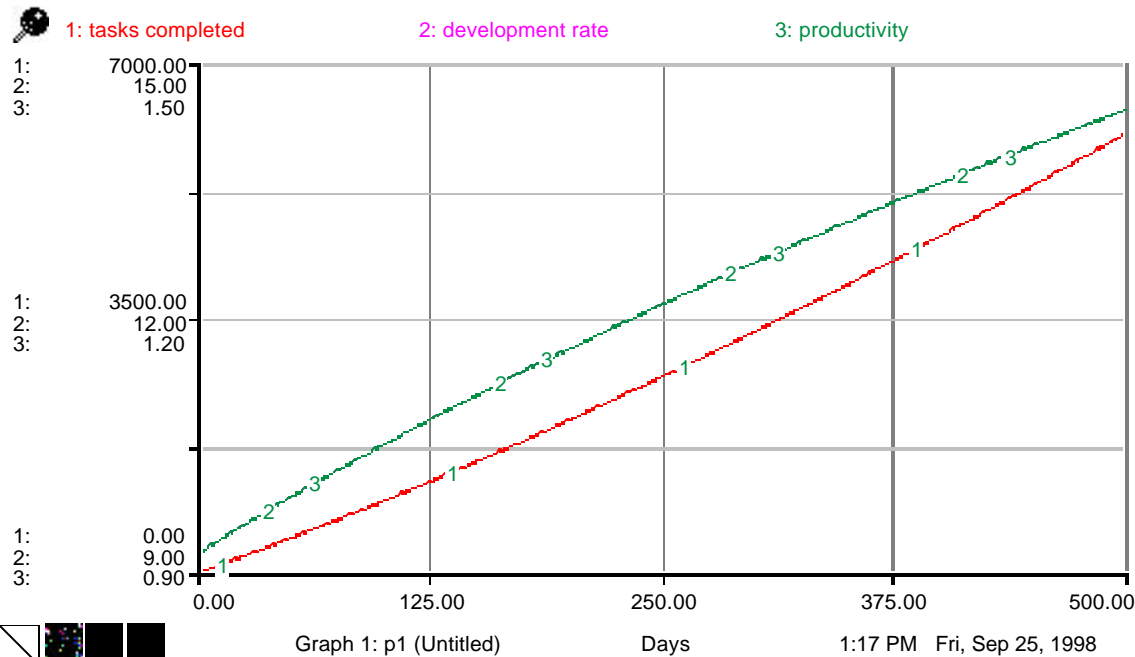
# Learning Rate

- The learning curve slope is related to the learning rate which describes the scaling of unit cost for every doubling of cumulative output.
- A smaller learning rate represents faster learning
- E.g., a learning rate of 75% indicates that the unit cost scales by 75% every doubling of cumulative output. If the unit cost of the 1000<sup>th</sup> line of code is 60 minutes, then the cost of the 2000<sup>th</sup> line would be  $.75(60)=45$  minutes.
- The slope of a learning curve in log-log space is equivalent to  $\log(\text{learning rate})/\log(2) = \log_2(\text{learning rate})$ .
- Software development exhibits ~80% learning rate

# Log-linear Learning Curve Model

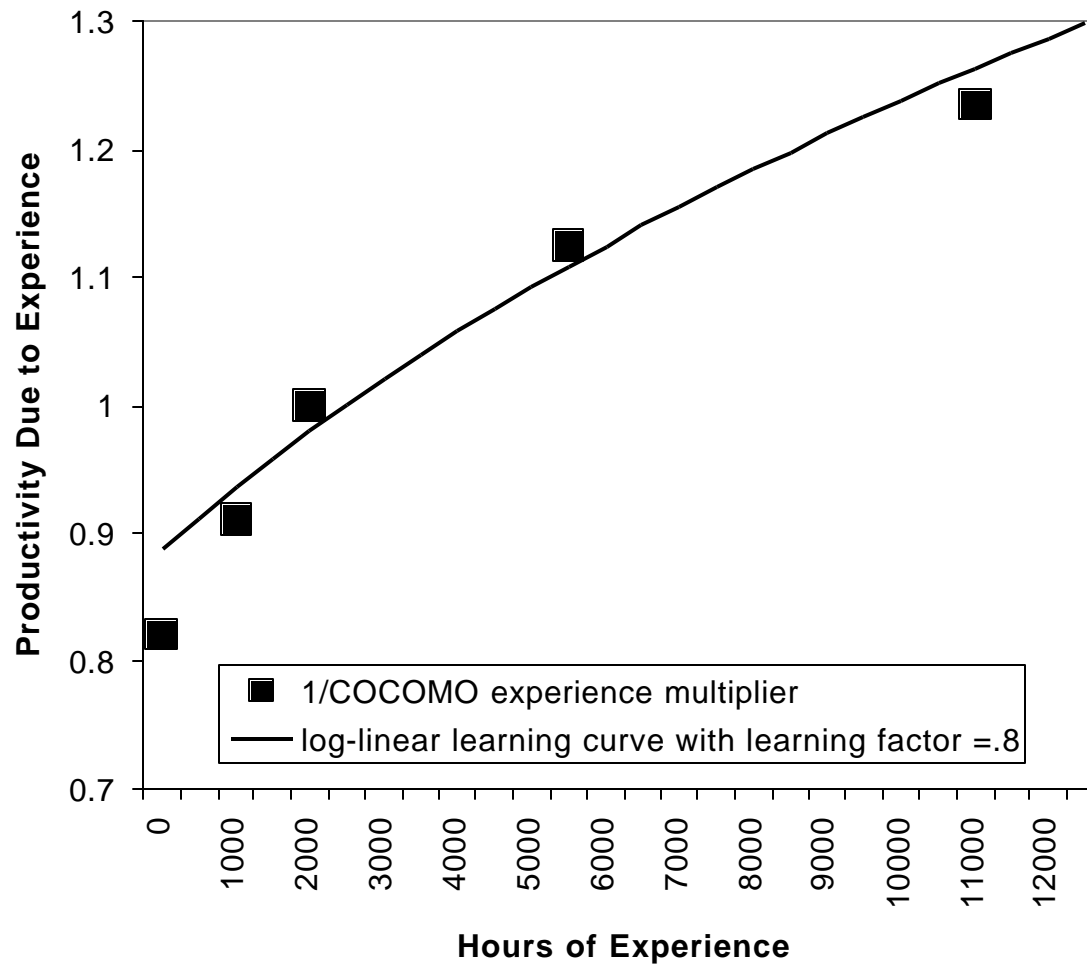


- $tasks\_completed(t) = tasks\_completed(t - dt) + (development\_rate) * dt$   
 INIT  $tasks\_completed = 1$   
 INFLOWS:  
⚙  $development\_rate = staff\_size * productivity$
- $learning\_factor = .8$
- $productivity = 1 / (100 * (tasks\_completed + 2000) ^ (\log_{10}(learning\_factor) / \log_{10}(2)))$
- $staff\_size = 10$

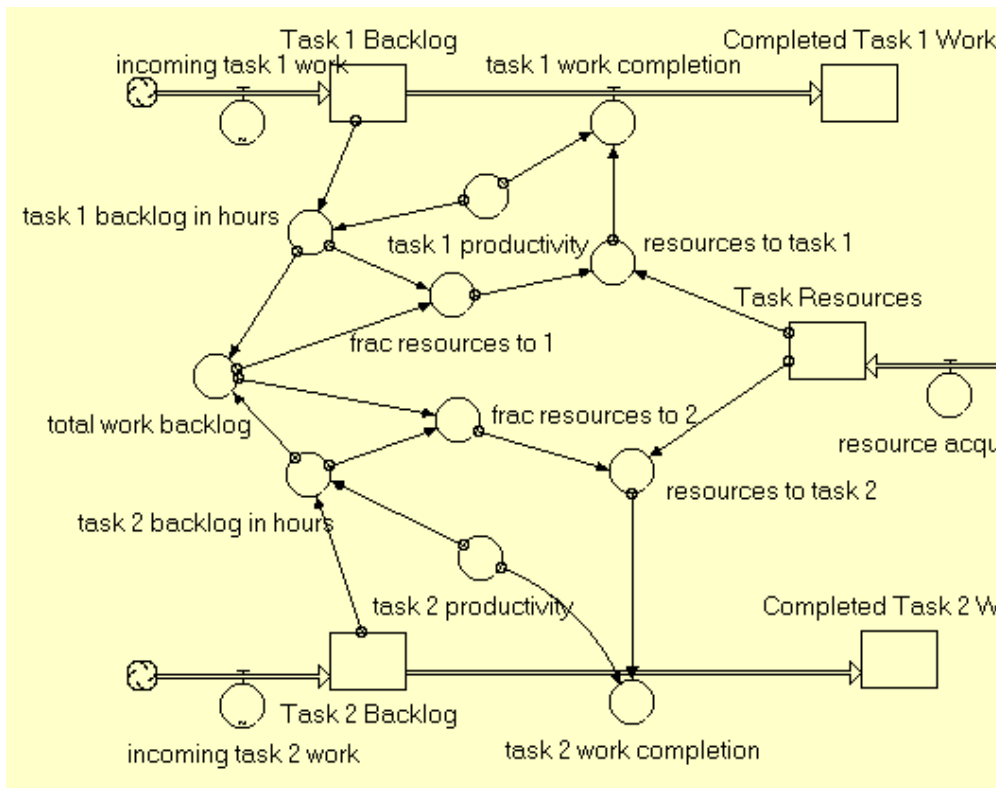




# Learning Curve Comparison to COCOMO Multipliers



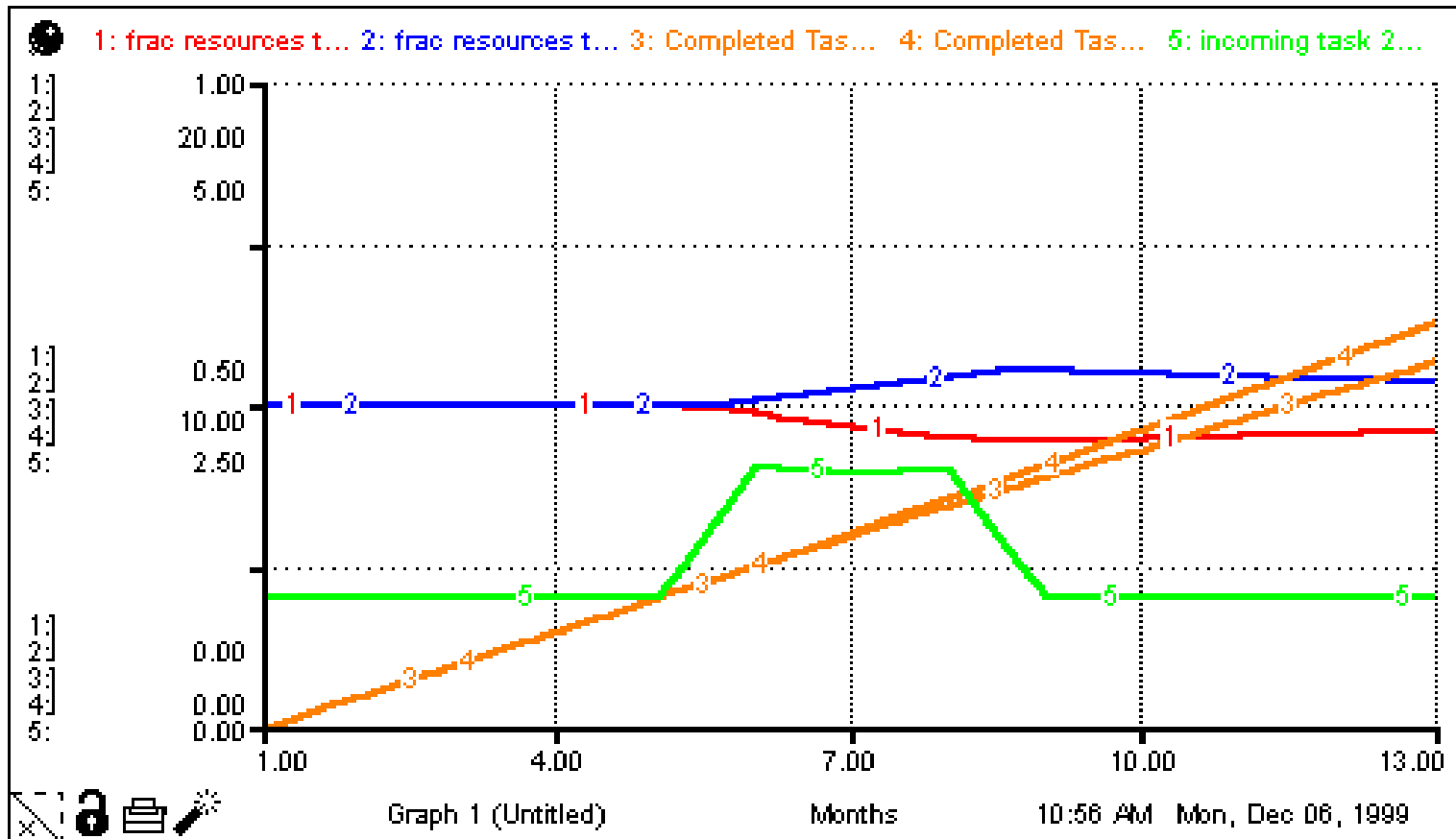
# Resource Allocation Model



- $Completed\_Task\_1\_Work(t) = Completed\_Task\_1\_Work(t - dt) + (task\_1\_work\_completion) * dt$   
 INIT  $Completed\_Task\_1\_Work = 0$   
 INFLOWS:  
 ☞  $task\_1\_work\_completion = resources\_to\_task\_1 * task\_1\_productivity$
- $Completed\_Task\_2\_Work(t) = Completed\_Task\_2\_Work(t - dt) + (task\_2\_work\_completion) * dt$   
 INIT  $Completed\_Task\_2\_Work = 0$   
 INFLOWS:  
 ☞  $task\_2\_work\_completion = resources\_to\_task\_2 * task\_2\_productivity$
- $Task\_1\_Backlog(t) = Task\_1\_Backlog(t - dt) + (incoming\_task\_1\_work - task\_1\_work\_completion) * dt$   
 INIT  $Task\_1\_Backlog = 10$   
 INFLOWS:  
 ☞  $incoming\_task\_1\_work = GRAPH(time)$   
 (1.00, 1.00), (2.00, 1.00), (3.00, 1.00), (4.00, 1.00), (5.00, 1.00), (6.00, 1.00), (7.00, 1.00), (8.00, 1.00), (9.00, 1.00), (10.0, 1.00), (11.0, 1.00), (12.0, 1.00), (13.0, 1.00)  
 OUTFLOWS:  
 ☞  $task\_1\_work\_completion = resources\_to\_task\_1 * task\_1\_productivity$
- $Task\_2\_Backlog(t) = Task\_2\_Backlog(t - dt) + (incoming\_task\_2\_work - task\_2\_work\_completion) * dt$   
 INIT  $Task\_2\_Backlog = 10$   
 INFLOWS:  
 ☞  $incoming\_task\_2\_work = GRAPH(time)$   
 (1.00, 1.00), (2.00, 1.00), (3.00, 1.00), (4.00, 1.00), (5.00, 1.00), (6.00, 1.00), (7.00, 1.00), (8.00, 1.00), (9.00, 1.00), (10.0, 1.00), (11.0, 1.00), (12.0, 1.00), (13.0, 1.00)  
 OUTFLOWS:  
 ☞  $task\_2\_work\_completion = resources\_to\_task\_2 * task\_2\_productivity$
- $Task\_Resources(t) = Task\_Resources(t - dt) + (resource\_acquisition) * dt$   
 INIT  $Task\_Resources = 4$   
 INFLOWS:  
 ☞  $resource\_acquisition = 0$
- $frac\_resources\_to\_1 = task\_1\_backlog\_in\_hours / total\_work\_backlog$
- $frac\_resources\_to\_2 = task\_2\_backlog\_in\_hours / total\_work\_backlog$
- $resources\_to\_task\_1 = Task\_Resources * frac\_resources\_to\_1$
- $resources\_to\_task\_2 = Task\_Resources * frac\_resources\_to\_2$
- $task\_1\_backlog\_in\_hours = Task\_1\_Backlog / task\_1\_productivity$
- $task\_1\_productivity = 5$
- $task\_2\_backlog\_in\_hours = Task\_2\_Backlog / task\_2\_productivity$
- $task\_2\_productivity = 5$
- $total\_work\_backlog = task\_1\_backlog\_in\_hours + task\_2\_backlog\_in\_hours$



# Resource Allocation Model Execution



- Extension for software process models



# Introduction to Software Process

## Modeling at Litton

- Litton GCS has used process modeling in the areas of managerial training, project and organizational modeling
- Using system dynamics to create mostly small-scale models
- SEPG is responsible for organizational analysis and training, and develops the models
- Many of these efforts are in the early stages<sub>14</sub>



# Process Maturity

- GCS is SEI-certified at CMM Level 4
- Process simulation is being used to support continued improvements
- Existing process performance baselines provide leverage in developing and calibrating meaningful simulation models



# Overview of Models

- Project level models
  - planning of specific projects
  - Brooks's Law and hiring issues
  - earned value model
  - requirements volatility
  - detailed (peer review) walkthrough model
- Multi-project or departmental level models
  - domain learning
  - product-line reuse processes
  - resource contention among projects



# Overview of Models (cont.)

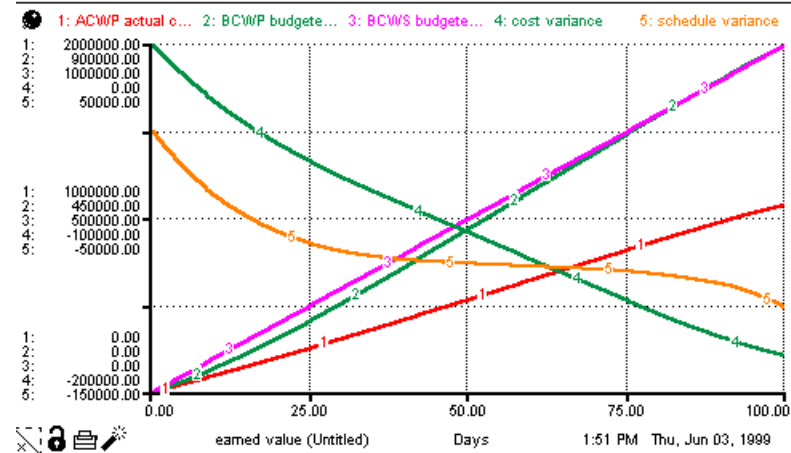
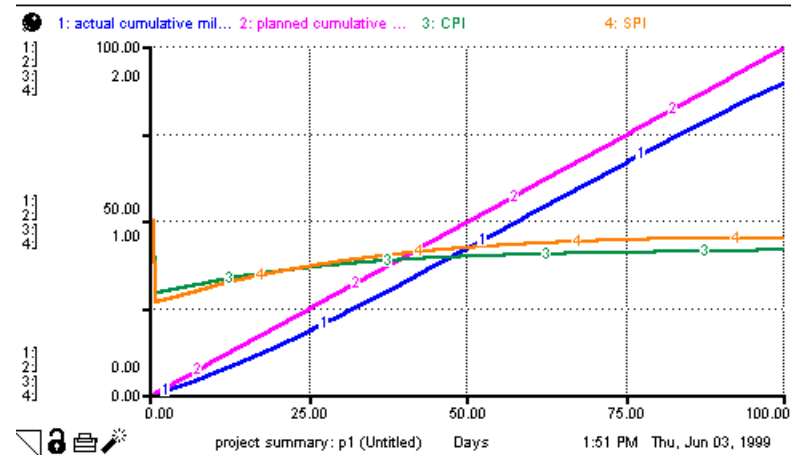
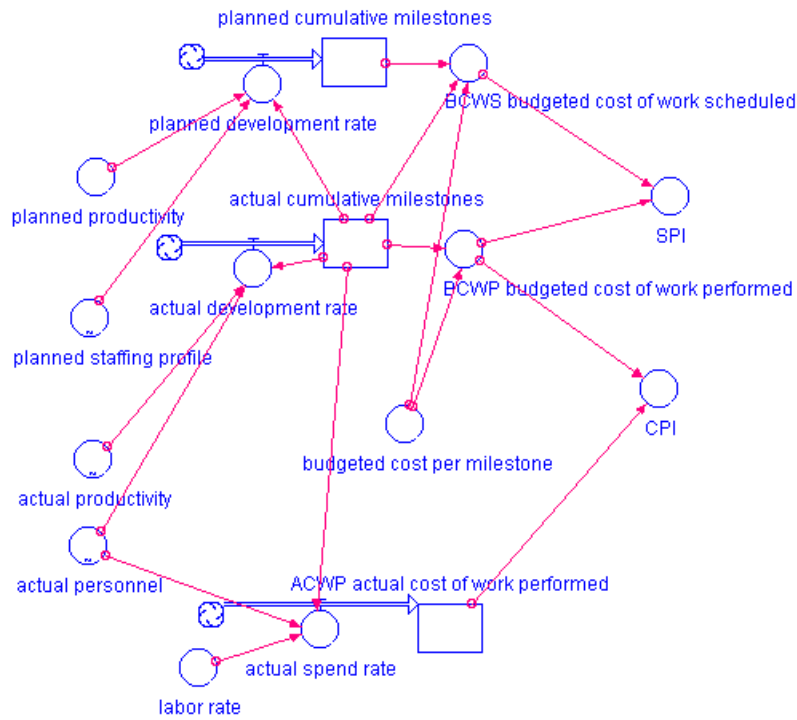
- Training applications
  - earned value techniques
  - productivity estimation
  - requirements volatility effects
  - extrapolation of project tracking indicators
  - project control
- Some of these are interactive “flight training” simulations



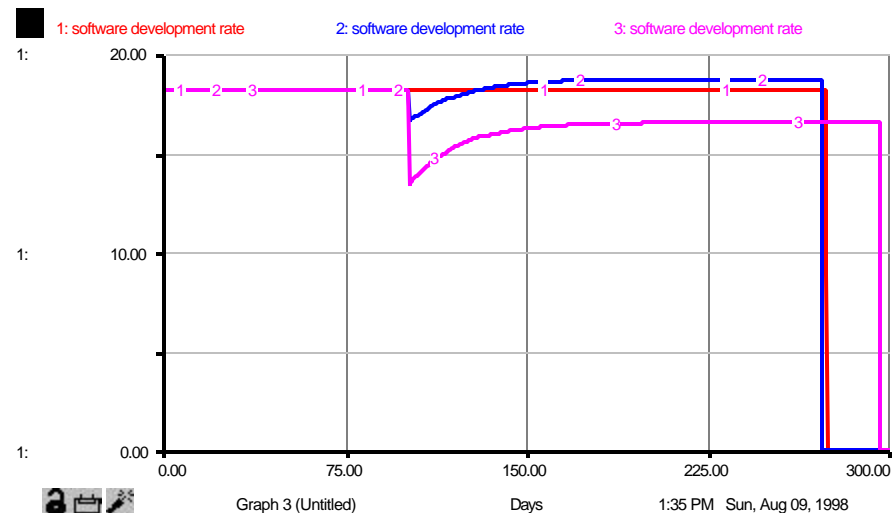
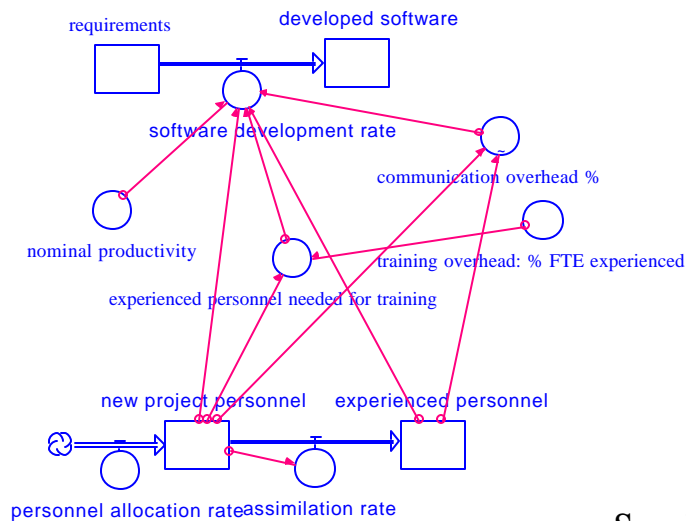
# Characterization of Case Studies

Scope / Purpose	Portion of lifecycle	Development project	Multiple, concurrent projects	Long-term product evolution	Long-term organization
Strategic management				product-line reuse strategies	
Planning	stage-based cost/schedule estimation	staffing project cost/schedule/quality estimation	reuse costs		
Control and operational management	stage tracking	earned value tracking			
Process improvement and technology adoption	peer review optimization	peer review effects on project	inter-project reuse processes	product-line reuse processes	
Understanding		requirements volatility	core reuse dynamics		
Training and learning	managerial metrics training	managerial metrics training			

# Earned Value Model



# Brooks's Law Model



Sensitivity of Software Development Rate to Varying Personnel Allocation Pulses  
(1: no extra hiring, 2: add 5 people on 100<sup>th</sup> day, 3: add 10 people on 100<sup>th</sup> day)

## Basic model assumptions:

- new personnel require training by experienced personnel to come up to speed
- more people on a project entail more communication overhead
- experienced personnel are more productive than new personnel, on average



# Project Planning and Control

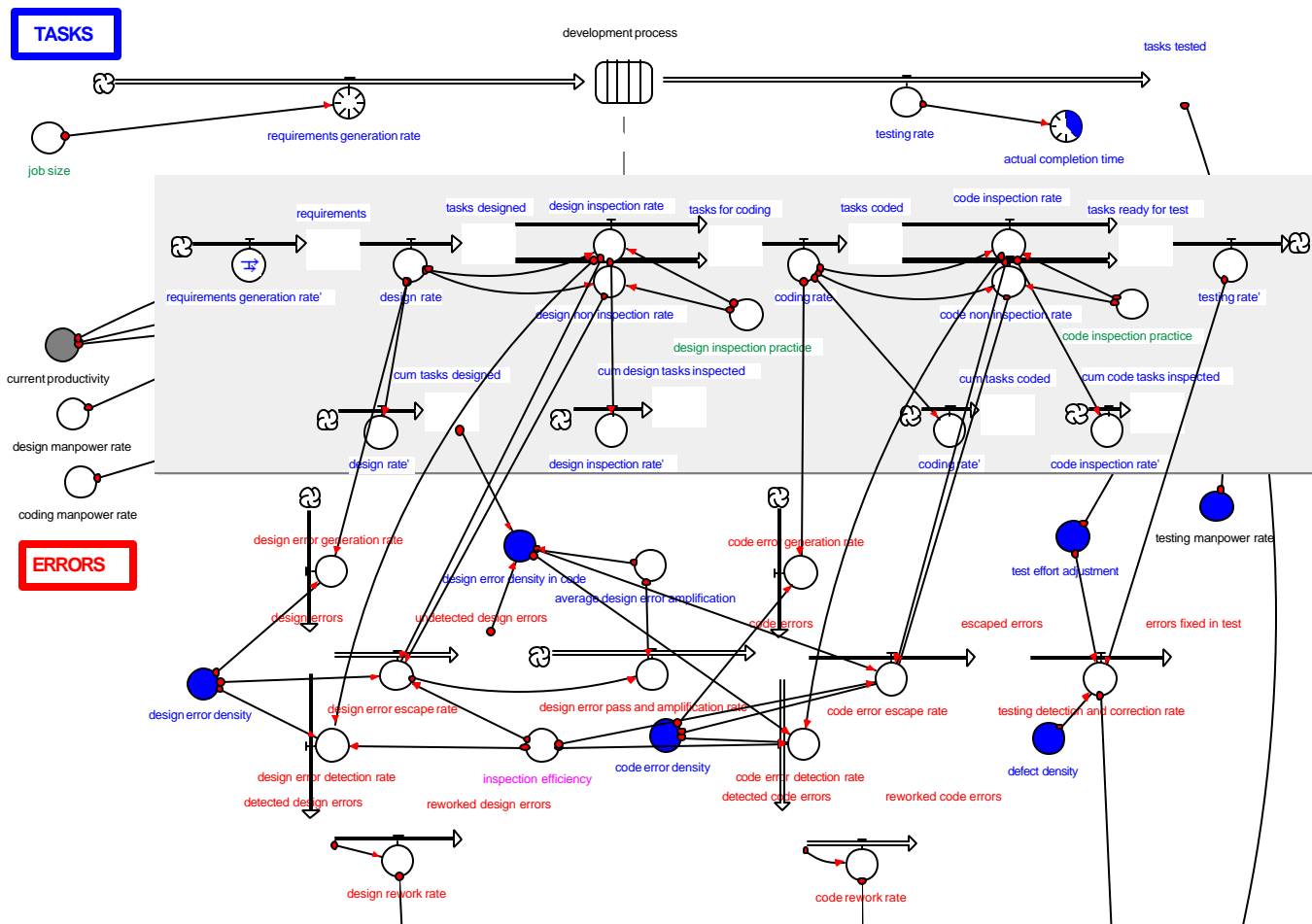
- Using pilot to raise the visibility of certain planning issues and to monitor the project
- Major model elements
  - incremental development
    - incremental COCOMO staffing profile translated into manpower addition and transfer rates
  - Brooks's Law effects
  - hiring delays
  - earned value
  - requirements volatility



# Peer Review Model

- An inspection model has been modified for other types of peer reviews, particularly walkthroughs
- Basic calibration parameters
  - nominal productivity
  - review efficiency
  - design defect density
  - code defect density
  - average design defect amplification
- Project specific parameters
  - job size
  - COCOMO effort parameter
  - schedule constraint
- Management decision parameters
  - design walkthrough practice (percent of design packages reviewed)
  - code walkthrough practice (percent of code reviewed)

# Peer Review Model Diagram

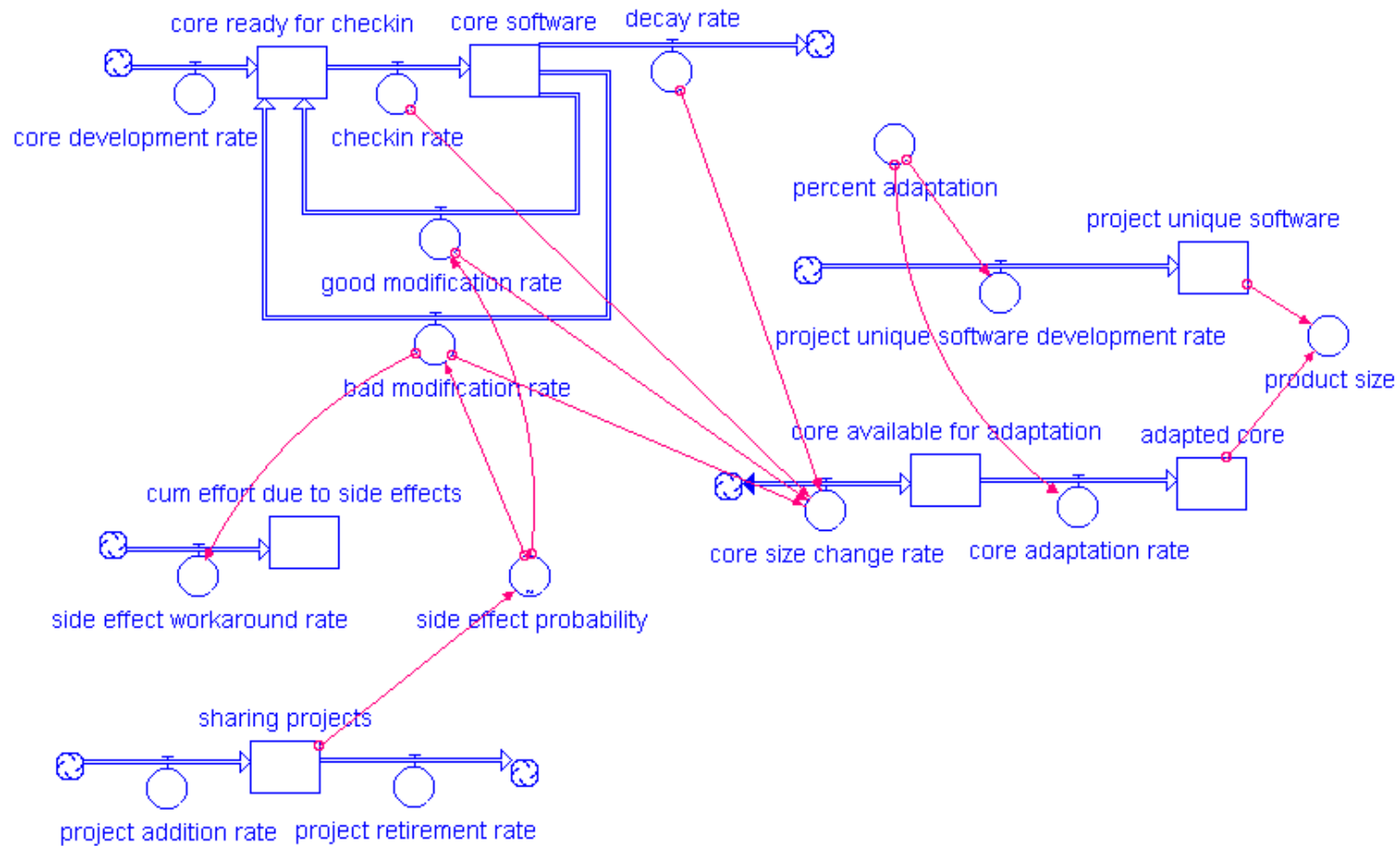




# Core Software Reuse Study

- A product-line reuse model is being developed to analyze the dynamics of reusing software shared among many projects
- Problem statement: product-line reuse is a major risk item
  - planned reuse levels are rarely met on a project
  - economics not well-known
  - large impact due to side effects of changes
  - the half-life of reuse components is usually underestimated
- A core software library is shared among projects within a specific product line
  - over a dozen projects use the core reuse library simultaneously
  - changes to the core by one project often adversely affect other projects, since side effects create new problems that often lead to cost and schedule overruns.
- The reuse process is currently being instrumented in order to parameterize the model

# Product Line Reuse Model





# Lessons Learned

- Simulation enables sharing of a process vision and discussion against common models
  - focussed studies on common issues and problems have improved our management vision and actions
  - helps managers understand the key factors in complex scenarios
  - improves planning and management processes
- Even small models are highly valuable for providing insight into dynamic trends
  - smaller is often better
- Simulation supports both organizational and individual learning
- Advantages to using simulation in the classroom vs. traditional methods
  - impart information in a more meaningful and dynamic way
  - live demonstrations keep up student interest
  - hands-on interactivity serves to drill in the learning experience
  - fun!



# Future Work

- Complete studies in progress
- Continue integrating simulation into process improvement initiatives
  - raise visibility with management and identify advantages of simulation
  - spread results to rest of organization
  - per Barry Richmond: “10% of the organization should model, the other 90% use the models”
- Plan for model evolution
- Document the modeling process and lessons



# CS599 Major Topic Outline

- Basic terminology (8/31)
- System dynamics fundamentals (9/7)
- Brooks's Law intro (9/7)
- RAD strategies (9/7)
- Software process lifecycles (9/14)
- RAD hypothesis testing (9/14)
- Modeling heuristics (9/14)



# CS599 Major Topic Outline (cont.)

- Abdel-Hamid model (9/28)
- Process concurrence (9/28, 10/5)
- Earned value (9/28, 11/2)
- Test inputs (10/5)
- Calibration example (10/12)
- General system behaviors (10/12)
- Rayleigh model (10/12, 11/2)



# CS599 Major Topic Outline (cont.)

- Common structures (10/19, 11/2)
- Past applications (11/2)
- Sensitivity analysis (11/30 reading)



# Directions for Future Work

- Model structures
- Common models and component reusability
- Usability
- Process model selection
- Knowledge-based techniques
- Object orientation
- Related simulation research
- Networked and distributed simulations
- Industrial data analysis



# Extra Credit Homework

- 3) Rayleigh curve version of COCOMO
  - develop a general purpose staffing curve generator for an incremental lifecycle process
  - calibrate a Rayleigh curve model based on the 10/12 representation to a reasonable midpoint solution of COCOMO II (or a chosen COCOMO extension)
  - adhere to COCOMO II schedule constraint relationships in light of the Rayleigh shape factor
  - see example in Boehm: *Software Engineering Economics* pp. 67-69
    - develop a similar default capability to clip the front and backends of the Rayleigh curve
  - optionally allow the user to specify the clipping relationship
    - possible inputs include % of Td, % of peak staff for initial point, absolute number of staff for initial point, others
  - allow user to specify the number and sizes of different increments (10 maximum)
  - allow user to specify arbitrary phase shifting between increments
    - e.g. increment 2 starts when increment 1 is x% through, or increment 2 starts y months after increment 1 starts



# Grades

SS #	HW #1 Degree Progress (10 points)	HW #2 Brooks's Law #1 (20 points)	HW #3 DPRS (15 points)	HW #4 Brooks's Law #2 (10 points)	HW #5 Process Concurrence #1 (15 points)	HW #6 Process Concurrence #2 (10 points)	Quiz #1 (12 points)	Quiz #2 (12 points)	Quiz #3 (16 points)	Term Project	Total Points
7134	9.4	17	12.5	8.5	13	9	12	11	14		106.4
4866	<b>9</b>	18	14	7	10	8.5	9	9	11		95.5
9077	9.4	15	12	8.25	11	9.5	12	7	10.5		94.65
2558	9.4	15	13.5	6.5	7	9.5	10	6	13.5		90.4
2245	9.4	10	9	9.5	10		1	5	13		66.9
6190	8.2	14	12	7	10	7.5	0	1.5	6.5		66.7
5928	<b>10</b>	13	9.5	7	7	6	3	4	7		66.5



# Remaining Milestones

- December 14 - Final Presentations
- December 17 - Extra Credit Homework Due
- December 20 - Final Report Due
- December 23 - Grades Due to CS Office