



University of Southern California
Center for Software Engineering

CS599 Software Process Modeling

Week 6

Ray Madachy

October 5, 1999



Outline

- Quiz feedback
- Standard test inputs and outputs
- Brooks's Law Model Testing
 - demonstration of various modeling techniques
 - developing relative graph functions
 - sensitivity runs
- Process Concurrency revisited
- Exhaustion model in Abdel-Hamid
- Homework

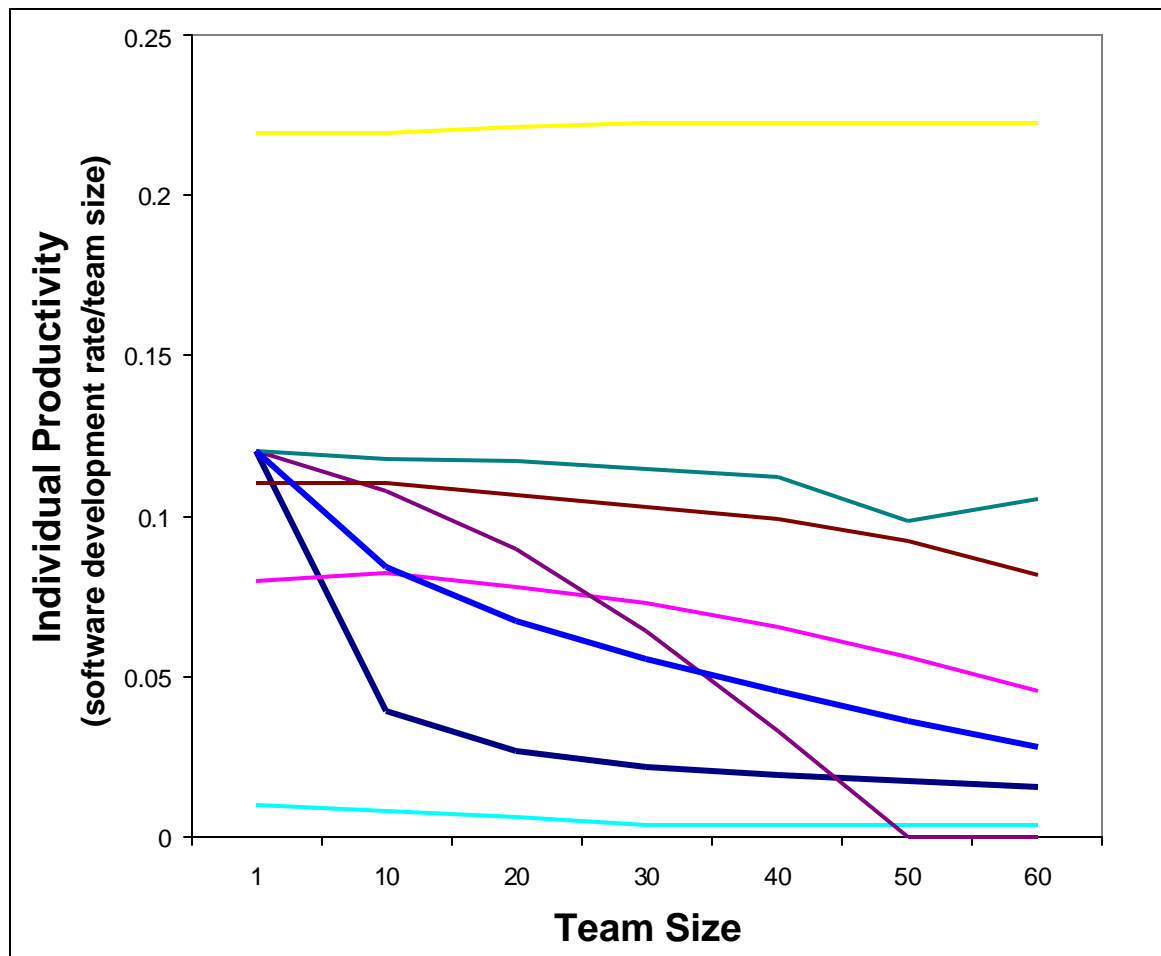


Test Input Demonstrations

- Pulse
- Constant
- Step
- Ramp



Brooks's Law Simulation Results against Conte et al. Equations

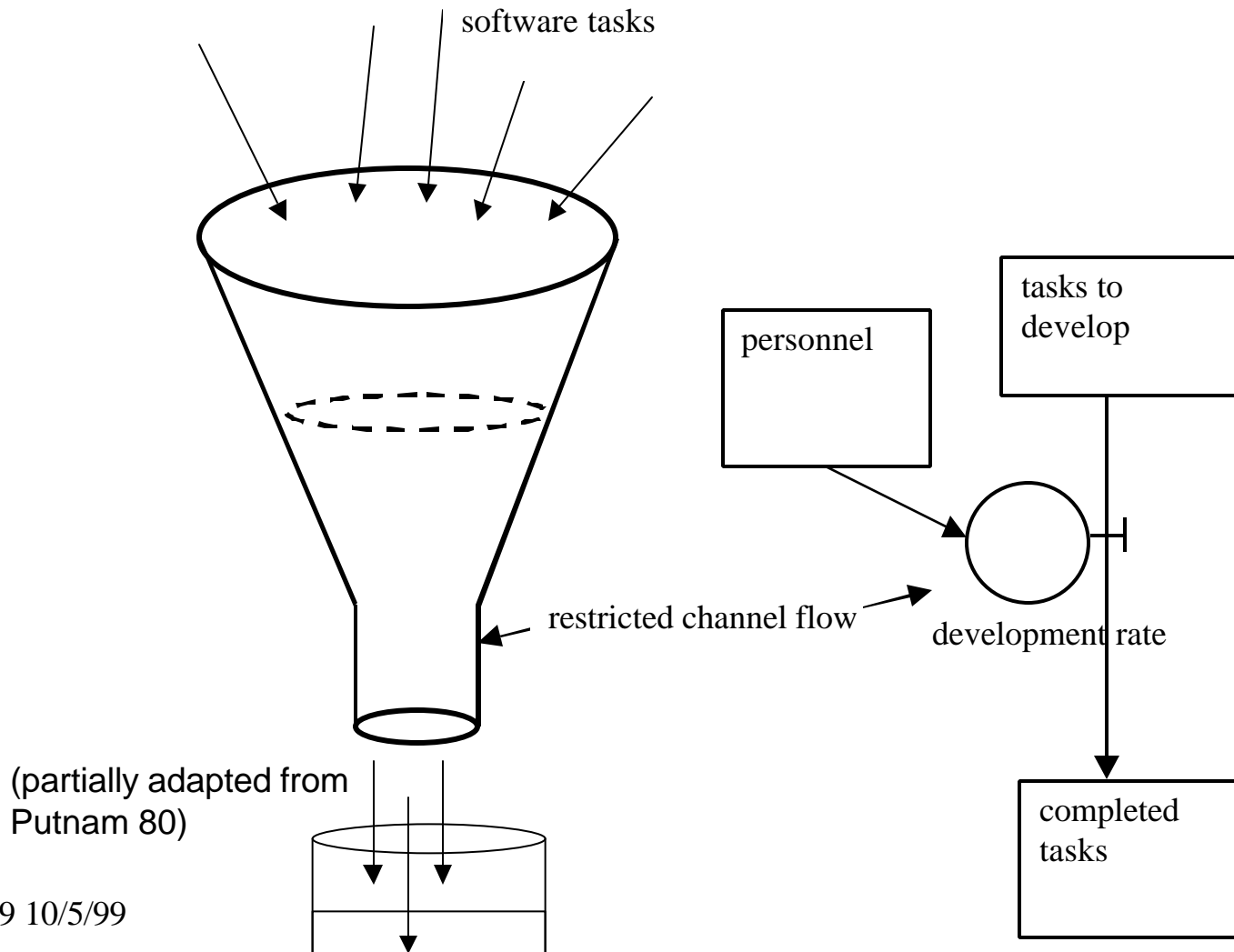




Process Concurrency

- Process concurrency refers to interdependency constraints between tasks, both within and between phases.
- Describes how much work becomes available for completion based on previous work accomplished.
- Bottlenecks on the availability of work
- Concurrency relations can be sequential, parallel, partially concurrent, or other dependent relationships.

Trying to Accelerate Software Production

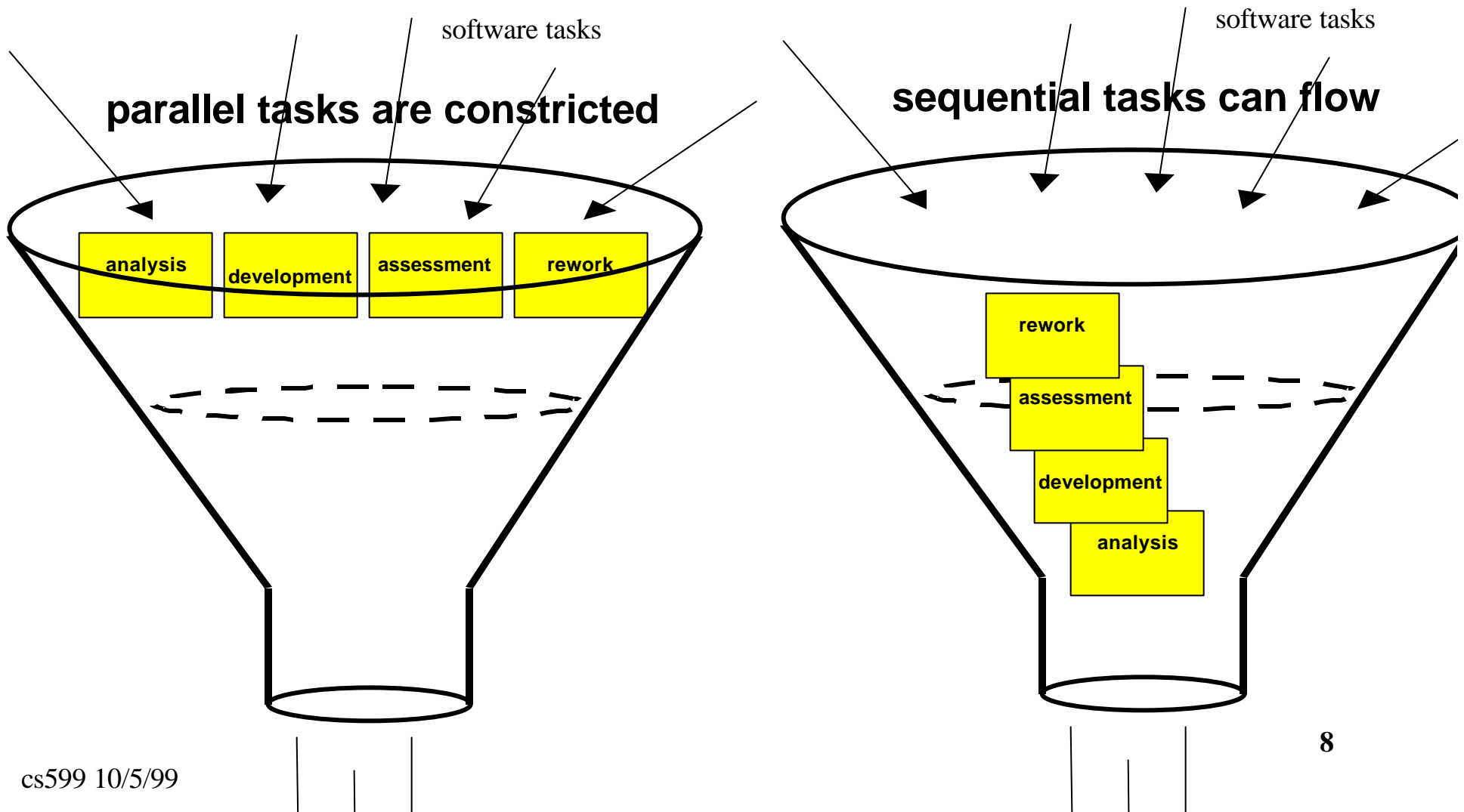




Limited Parallelism of Software Activities

- There are always sequential constraints independent of phase:
 - analysis and specification
 - figure out what you're supposed to do
 - development of something (architecture, design, code, test plan, etc.)
 - assessment
 - verify/validate/review/debug
 - possible rework recycle of previous activities
- These can't be done in parallel with more applied people
 - Different people can perform the different activities with limited parallelism, but downstream activities will always have to follow some of⁷

Funnel View of Limited Parallelism





Relevance of Brooks in *The Mythical Man-Month*

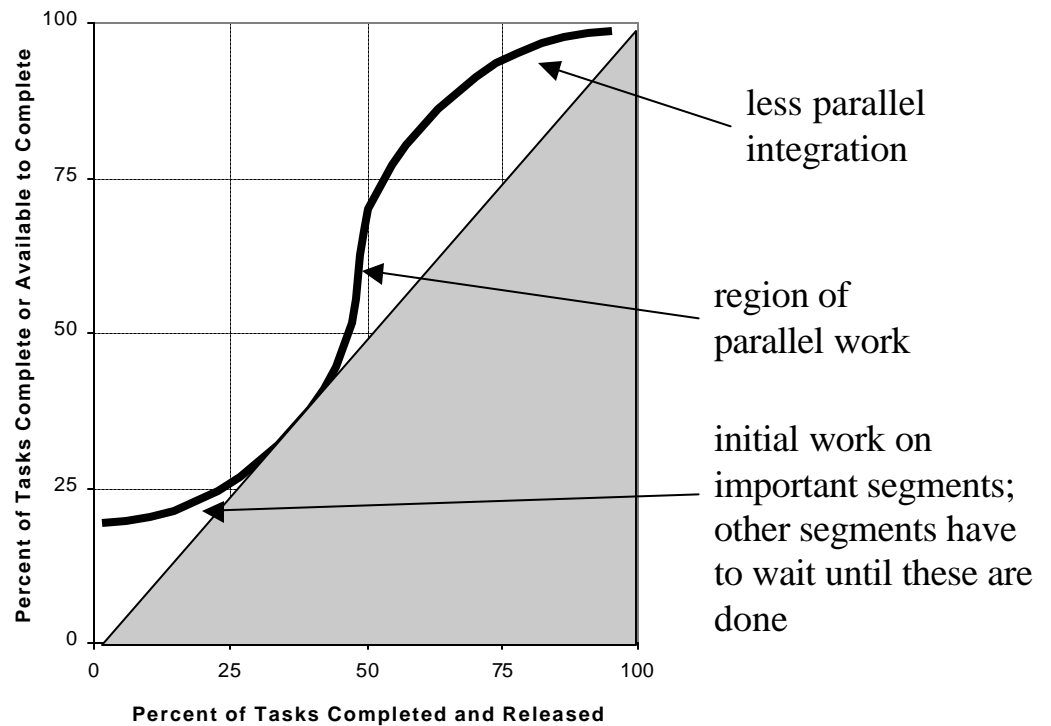
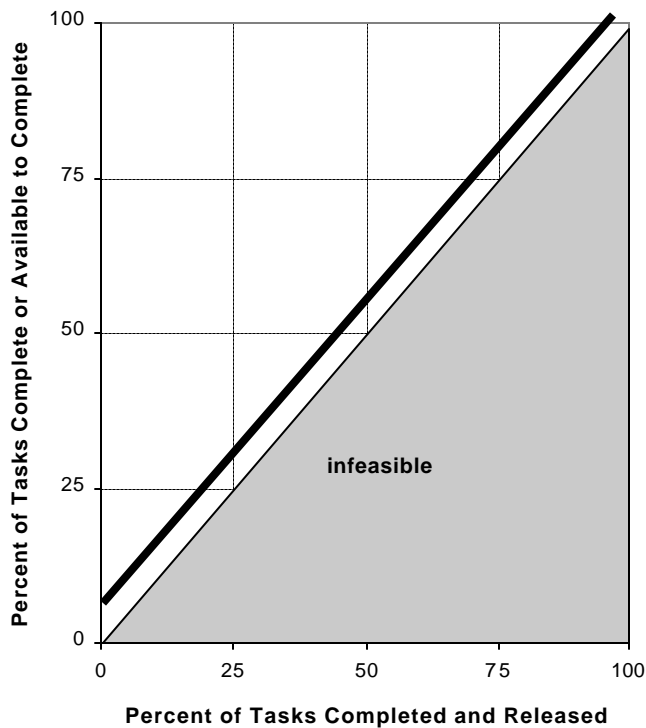
- Sequential constraints imply tasks cannot be partitioned.
 - Applying more people has no effect on schedule
- Men and months are interchangeable only when tasks can be partitioned with no communication among them.



Internal Process Concurrency

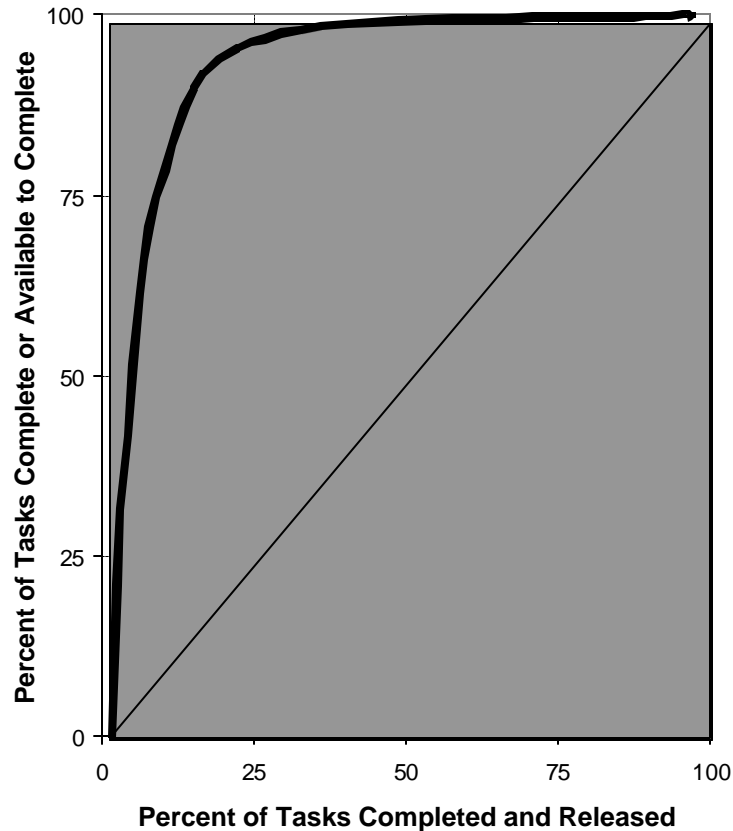
- Internal process concurrency relationship shows how much work can be done based on the percent of work already done.
- The relationships represent the degree of sequentiality or concurrency of the tasks aggregated within a phase.

Linear and Non-linear Internal Process Concurrency

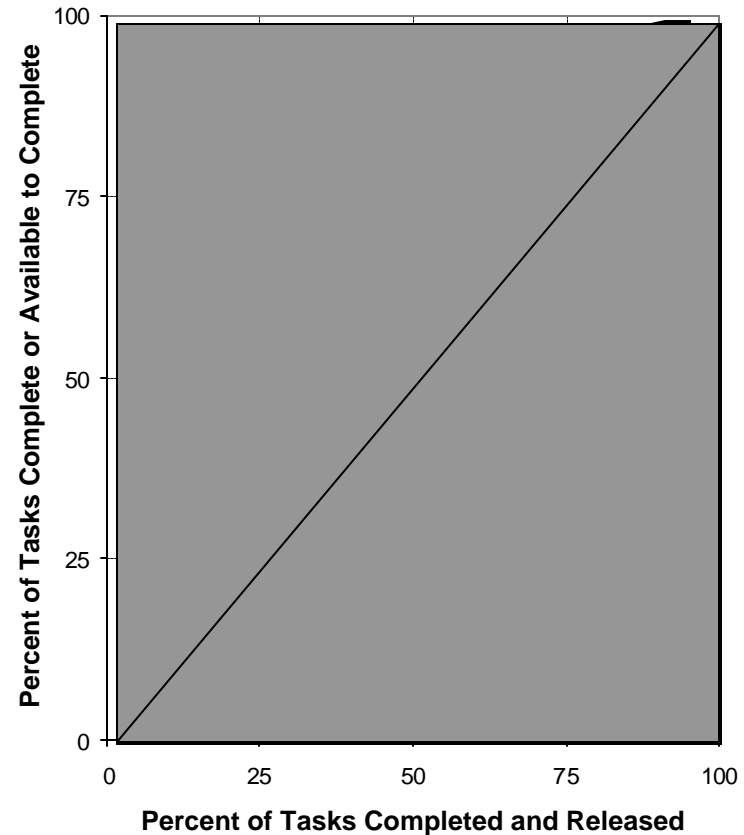


The overarching segments of software must be completed before other parts can begin.

Internal Concurrency Examples



Simple conversion task where tasks can be partitioned with no communication



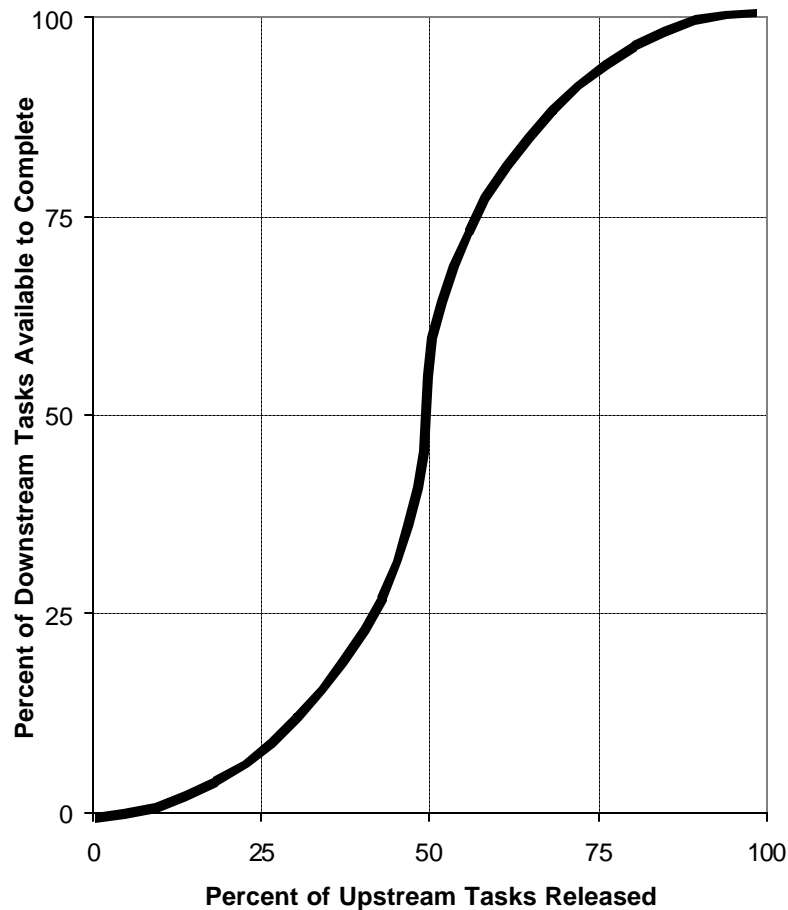
Complex system development where men and months are not interchangeable



External Process Concurrency

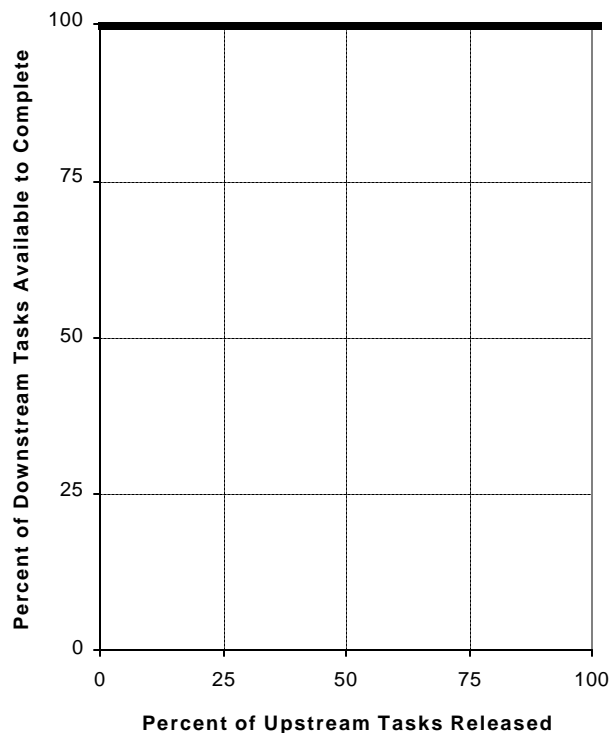
- External process concurrency relationships describe constraints on amount of work that can be done in a downstream phase based on the percent of work released by an upstream phase.
- See examples on next slide
 - More concurrent processes have curves near the upper left axes, and less concurrent processes have curves near the lower and right axes.

External Process Concurrency Example



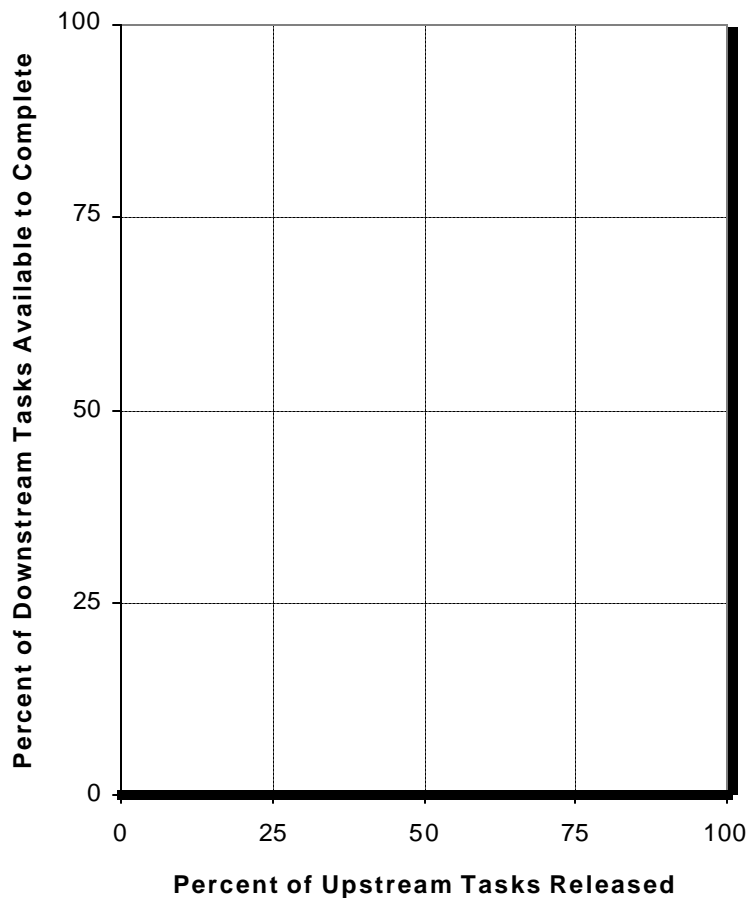
- Typical non-linear external process concurrency relationship

No Inter-phase Relationship



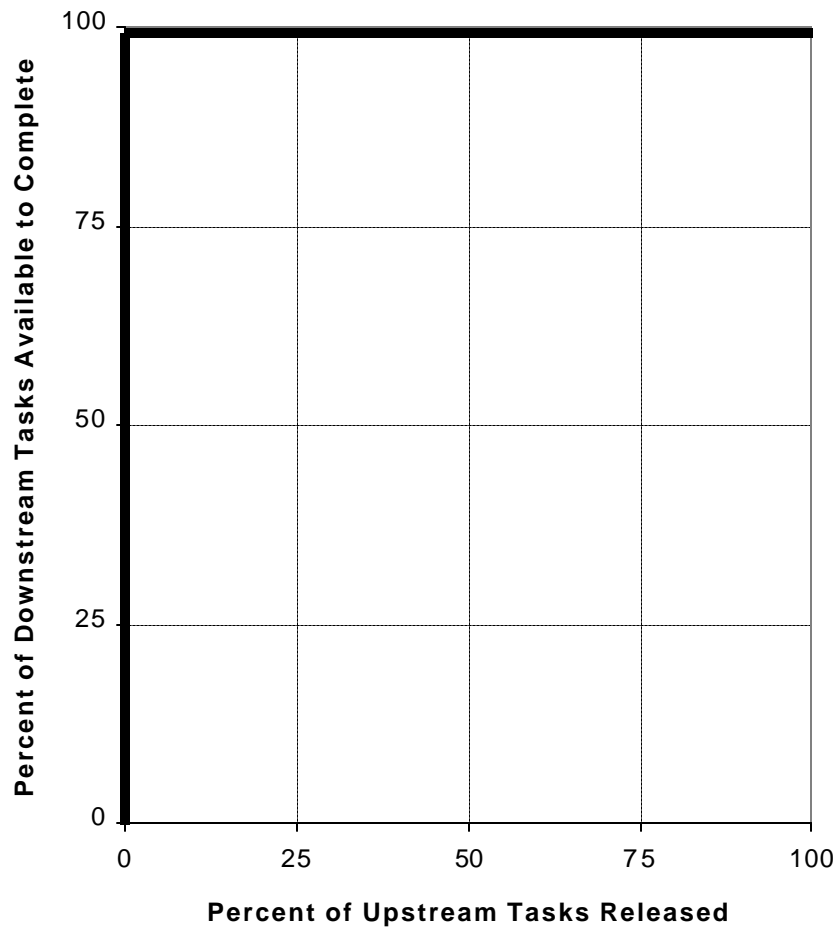
- No dependencies between the phases.
- The downstream phase can progress independently of the upstream phase.
- The entire downstream work is available to be completed with none of the upstream work released.

Sequential Inter-phase Relationship



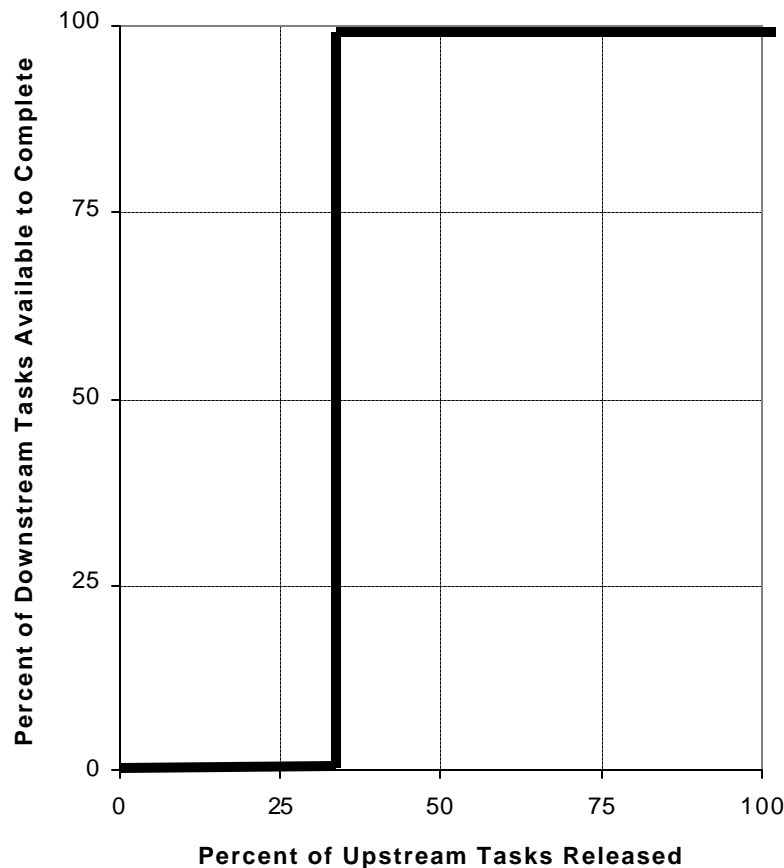
- None of the downstream phase can occur until the upstream phase is totally complete.
- Like a theoretical waterfall development process where no phase can start until the previous phase is completed and verified.
- Same as a finish-stop relationship in a critical path network.

Parallel Inter-phase Relationship



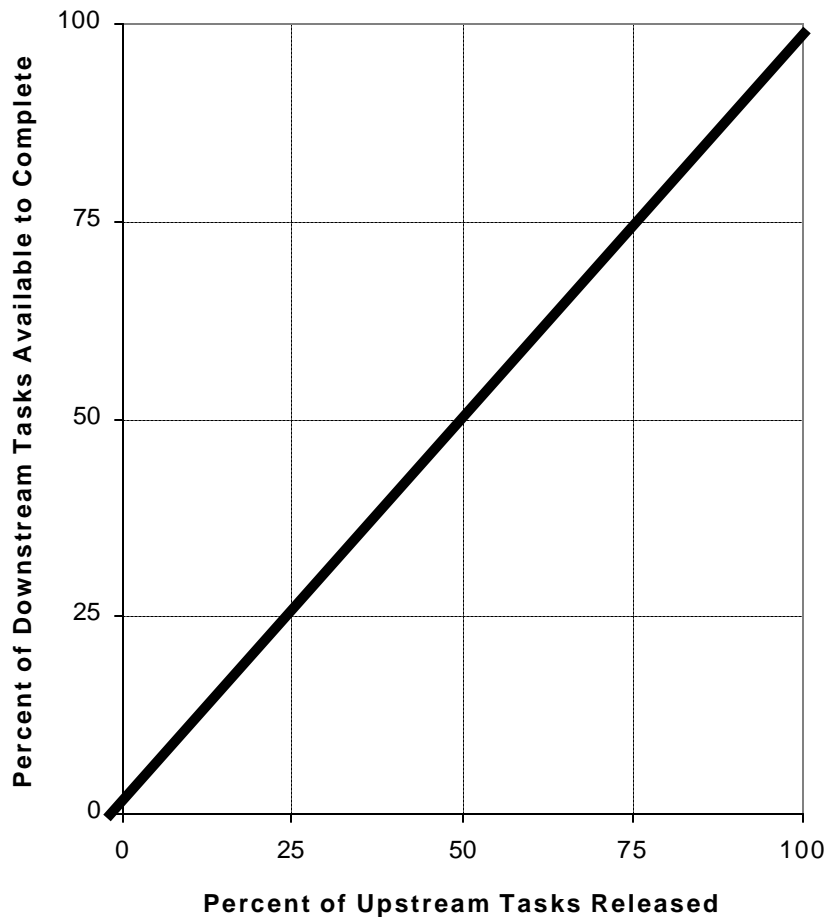
- The two phases can be implemented completely in parallel.
- The downstream phase can be completed as soon as the upstream phase is started.

Delayed Start Inter-phase Relationship



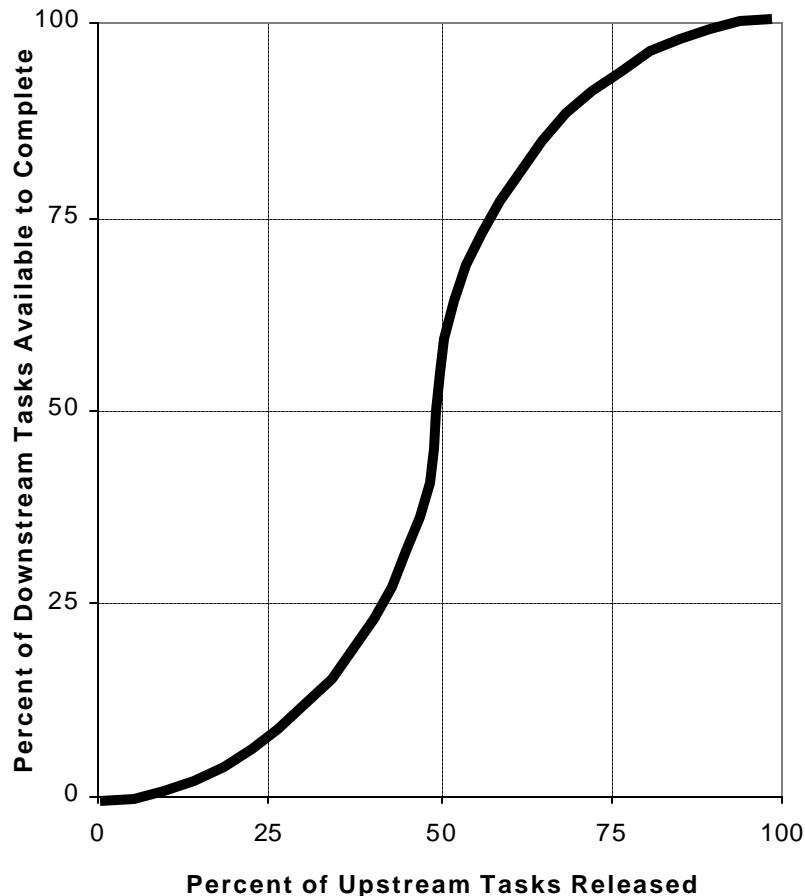
- The downstream phase must wait until a major portion of the upstream phase is completed, then it can be completed in its entirety.
- Like a start-start relationship in a critical path network.

Lockstep Inter-phase Relationship



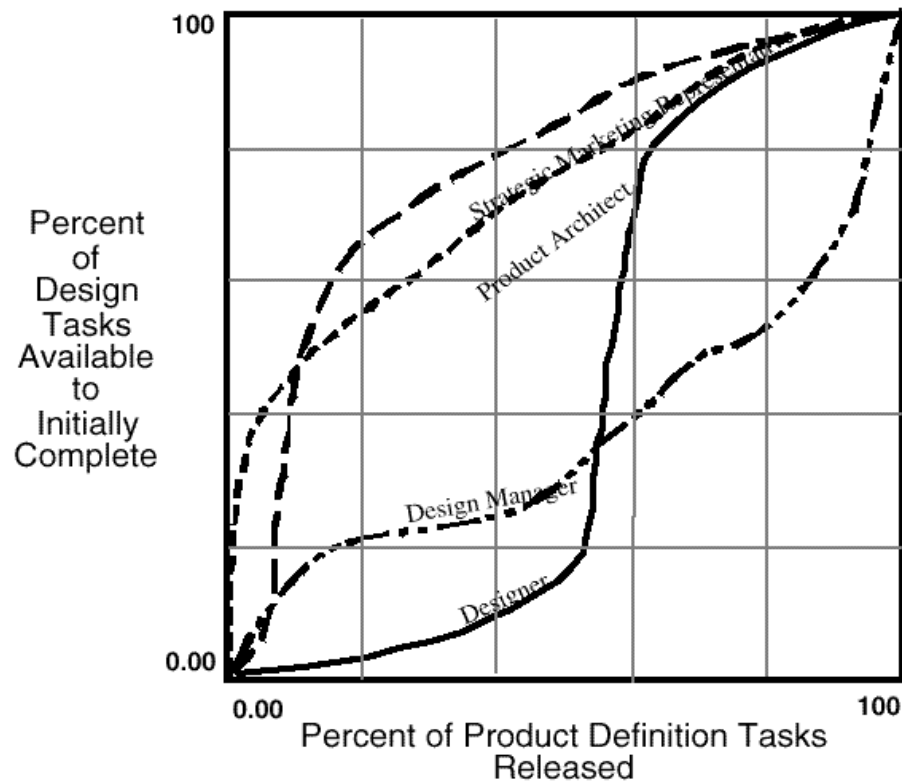
- The downstream phase can progress at the same speed as the upstream phase; thus they are in lockstep with each other.
- Like stovepipe components stacked on top of each other, similar to building the stories of a skyscraper.
- This relationship is not available in PERT/CPM methods.

Delay with Partially Concurrent Inter-phase Relationship



- The downstream phase has to wait until a certain percentage of upstream tasks have been released, and then can proceed at varying degrees of concurrence per the graph.
- Representative of much software development work.
- This relationship is not available in PERT/CPM.

Roles Have Different Mental Models

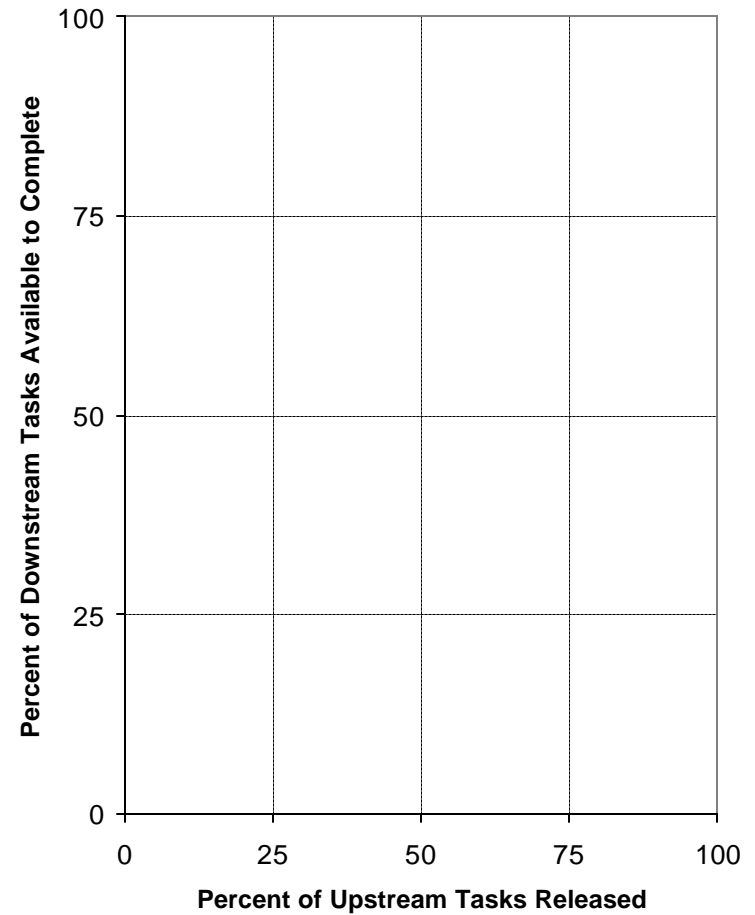
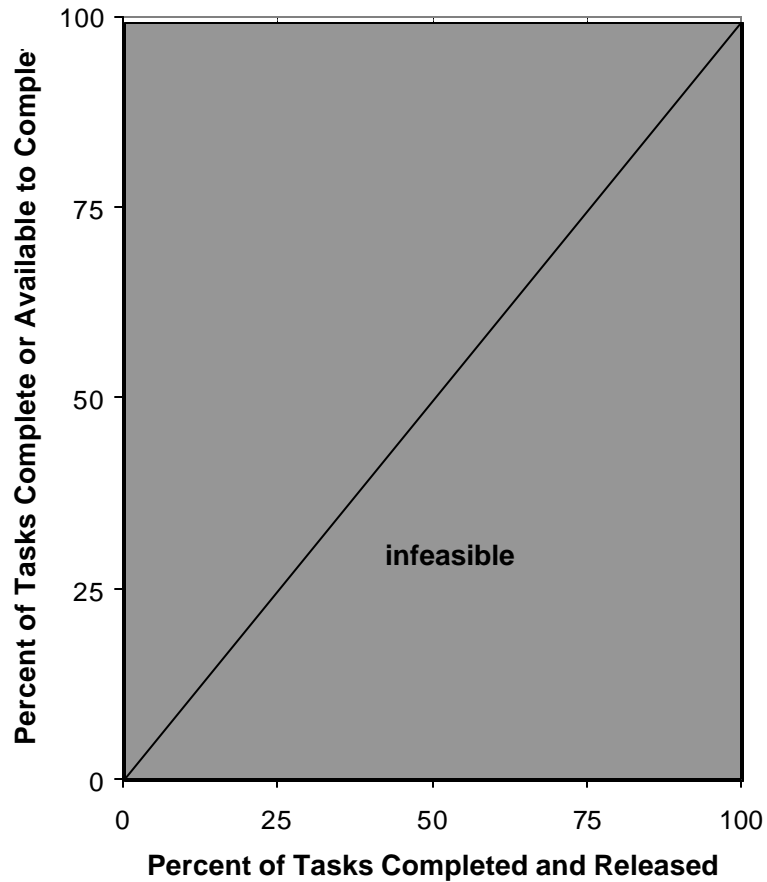


Four Estimates of External Process Concurrency Relationship between the Product Definition and Design Phases

- Differing perceptions upstream and downstream

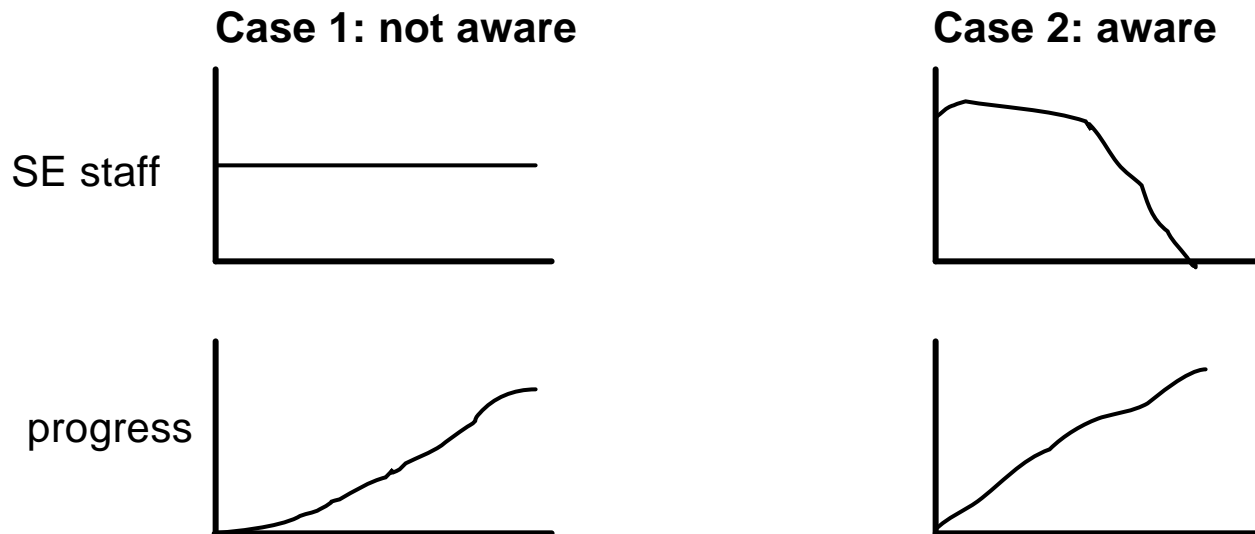


Let's Draw Our Own

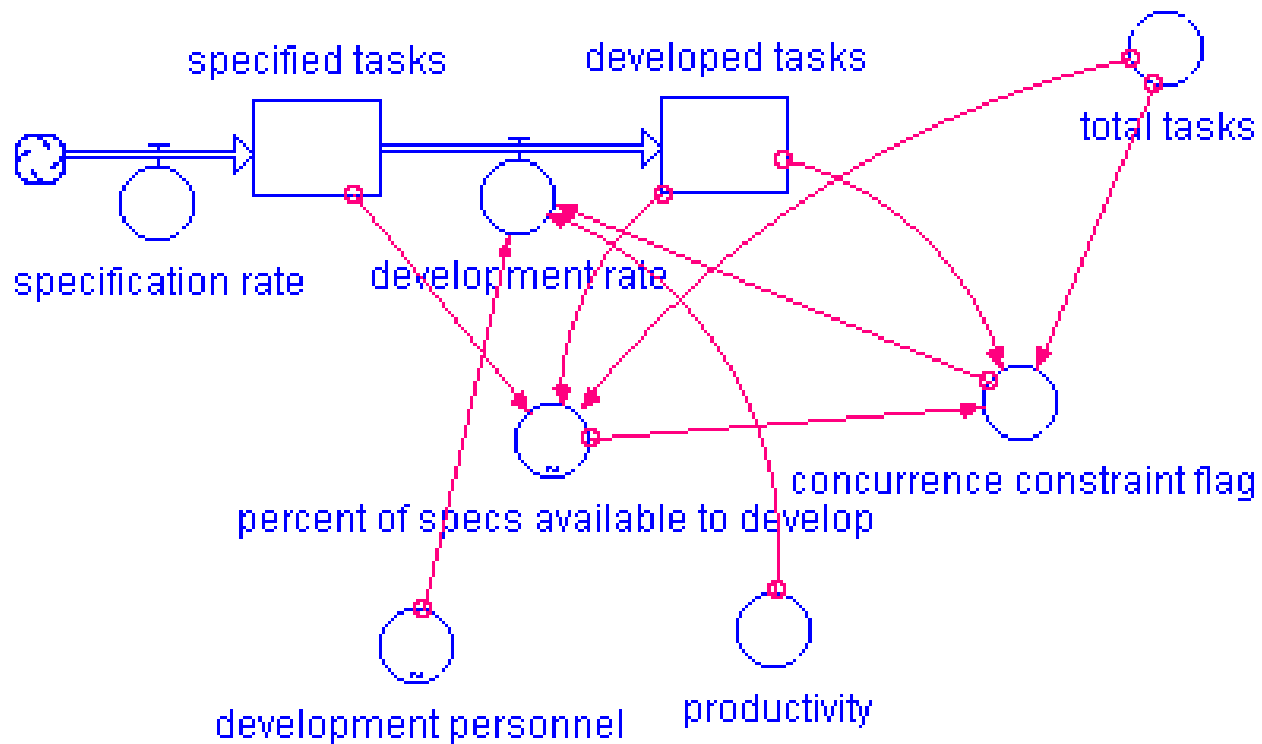


RAD Awareness

- Hypothesis: to optimize schedule on a complex project with partial inter-phase concurrency, the optimal systems engineering staffing is front-loaded vs. constant level-of-effort
 - downstream development is constrained by the specifications available



External Concurrency Model Diagram





Equations

- $developed_tasks(t) = developed_tasks(t - dt) + (development_rate) * dt$
INIT $developed_tasks = 0$

DOCUMENT: Tasks that have been developed.
INFLOWS:
 - $development_rate = \text{if} (concurrency_constraint_flag = 0) \text{ then } development_personnel * productivity \text{ else } 0$
DOCUMENT: If there are tasks that can be developed (i.e. not constrained by external process concurrence), then development rate = $personnel * productivity$.
- $specified_tasks(t) = specified_tasks(t - dt) + (specification_rate - development_rate) * dt$
INIT $specified_tasks = 0$

DOCUMENT: The tasks that have been specified but not developed yet.
INFLOWS:
 - $specification_rate = \text{if} (time < 5) \text{ then } 20 \text{ else } 0$
DOCUMENT: The rate of specifying tasks is 20 tasks/months for the first 5 months.OUTFLOWS:
 - $development_rate = \text{if} (concurrency_constraint_flag = 0) \text{ then } development_personnel * productivity \text{ else } 0$
DOCUMENT: If there are tasks that can be developed (i.e. not constrained by external process concurrence), then development rate = $personnel * productivity$.
- $concurrency_constraint_flag = \text{if} (developed_tasks < percent_of_specs_available_to_develop * total_tasks) \text{ then } 0 \text{ else } 1$
DOCUMENT: The constraint flag is set based on whether there are tasks that can currently be developed per the concurrence relationship (1 = constrained).
- $productivity = 1$
DOCUMENT: Individual development productivity of 1 task per person-month.
- $total_tasks = 100$
DOCUMENT: There are 100 tasks to be specified and developed.
- $development_personnel = GRAPH(time)$
(0.00, 0.05), (2.00, 0.05), (4.00, 0.15), (6.00, 20.0), (8.00, 19.9), (10.0, 19.8), (12.0, 19.9), (14.0, 19.6), (16.0, 9.95), (18.0, 0.00), (20.0, 0.00)
DOCUMENT: The personnel staffing profile.
- $percent_of_specs_available_to_develop = GRAPH((specified_tasks + developed_tasks) / total_tasks)$
(0.00, 0.02), (0.1, 0.045), (0.2, 0.045), (0.3, 0.045), (0.4, 0.07), (0.5, 0.125), (0.6, 0.185), (0.7, 0.44), (0.8, 0.745), (0.9, 0.91), (1, 1.00)
DOCUMENT: This external concurrence relationship describes the percent of tasks that are available to be developed as a function of the tasks specified to-date.



Homework

- Fix partitioning in your Brooks's Law model as appropriate (due next week)
 - validate your model before making runs
 - try to find optimal people addition at following times: 50, 100, 150, 200, 250 days
 - discuss overall optimal policy and Brooks's Law phenomena
- Read Abdel-Hamid model handout
- Prepare to discuss your project in class
- Remember short quiz again next week



Homework (cont.)

- Test the RAD hypothesis on slide 23 with an external process concurrence model (due in 2 weeks)
 - project parameters:
 - 8 person-years total project effort
 - SE effort (specification) = 25% of total = 2 person-years
 - compare level-of-effort systems engineering staffing with a front-loaded staffing profile
 - start out with a level-of-effort software staffing also, but vary this on later runs after the first comparison
 - is there an optimal staffing profile for development?
 - quantify cost and schedule tradeoffs and draw conclusions



Homework (cont.)

- You can read the Ford-Sterman paper on the class web site for further information on process concurrence
 - there is also a model available