References

Abdel-Hamid, Tarek (1984) The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective. Doctoral thesis. Massachusetts Institute of Technology. Cambridge, MA.

Abdel-Hamid, Tarek and Madnick, Stuart (1991) **Software Project Dynamics, An Integrated Approach.** Prentice-Hall. Englewood Cliffs, NJ.

Adler, et al. (1995) From Project to Process Management: An Empirically-Based Framework for Analyzing Product Development Time. **Management Science.** 41:3:458-484.

Bacon, Glenn et al. (1994) Managing Product Definition in High-Technology Industries: A Pilot Study. unpublished draft. Haas Business School. University of California. Berkeley, CA.

Barrie, Donald S. and Paulson, Boyd C. (1984) **Professional Construction Management.** McGraw-Hill. New York.

Brooks, F. P. (1978) **The Mythical Man-Month**. Addison-Wesley Publishing Co. Reading, MA.

Burchill, Gary (1992) Concept Engineering. Doctoral thesis. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Burchill, Gary and Walden, David (1994) Mutual Learning: Industry/Academia Collaboration for Improved Product Development. **The Center for Quality Management Journal**. 3:2.

Burrus, C. Sidney et al. (1994) **Computer-Based Exercises for Signal Processing Using MATLAB®**. Prentice-Hall. Englewood Cliffs, NJ.

Chao, Linda (1993) Improving Quality and Time to Market in the VLSI Product Life Cycle. Master's thesis. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Clark, Kim B and Fujimoto, Takahiro (1991a) The Power of Product Integrity. **Managing Product Life Cycles From Start to Finish**. Harvard Business Review Paperback. Cambridge, MA.

Clark, Kim B. and Fujimoto, Takahiro (1991b) **Product Development Performance, Strategy, Organization, and Management in the World Auto Industry**. Harvard Business School Press. Cambridge, MA.

Clark, Kim B. and Wheelwright, Steven (1993) **Managing New Product and Process Development**. Harvard Business School Press. Cambridge, MA.

Clemson, Barry et al. (1995) Efficient Methods for Sensitivity Analysis. **System Dynamics Review.** 11:1:31-49.

Cooper, Kenneth, G. (1980) Naval Ship Production: A Claim Settled and a Framework Built. **Interfaces**. 10:6. The Institute of Management Sciences.

Cooper, Kenneth, G. (1994) The $2,000 hour: How Managers Influence Project Performance Through the Rework Cycle. **Project Management Journal**. 25:1.

Cooper, Kenneth, G. (1993a) The Rework Cycle: Benchmarks for the Project Manager. **Project Management Journal**. 24:1.

Cooper, Kenneth, G. (1993b) The Rework Cycle: How It *Really* Works...And Reworks.... **Project Management Journal**. 24:1.

Cooper, Kenneth, G. (1993c) The Rework Cycle: How Projects are Mismanaged. **Project Management Journal**. 24:1.

Cooper, Kenneth, G. and Mullen, Thomas W. (1993) Swords & Plowshares, The Rework Cycles of Defense & Commercial Software Development Projects. **American Programmer.** 6:1.

Cooper, Robert G. (1994) Third-Generation New Product Processes. **The Journal of Product Innovation Management.** 11:1:3-14.

Cooper, Robert G. (1990) Stage-Gate Systems: A New Tool for Managing New Products. **Business Horizons.** May-June:44-54.

Cooper, Robert G. and Kleinschmidt, Elko (1994) Determinants of Timeliness in Product Development. **The Journal of Product Innovation Management.** 11:5:381-96.

Cusumano, Michael A. and Nobeoka, Kentaro (1991) Strategy, Structure, and Performance in Product Development: Observations from the Auto Industry. Report MITJP 91-02. The MIT Japan Program. Center for International Studies. Massachusetts Institute of Technology. Cambridge, MA.

Dean, James W. Jr. and Susman, Gerald I. (1991) Organizing for Manufacturable Design. **Managing Product Life Cycles From Start to Finish**. Harvard Business Review Paperback. Cambridge, MA.

Diehl, Ernst and Sterman, John (1995) Effects of Feedback Complexity on Dynamic Decision Making. forthcoming in **Organizational Behavior and Human Decision Processes**. August.

Dorf, Richard C. and Bishop, Robert H. (1995) **Modern Control Systems.** Addison-Wesley Publishing Company. MA.

Eppinger, Steven D. et al. (1990) Organizing the Tasks in complex Design Projects. ASME Conference on Design Theory and Methodology. Chicago, IL. Sept:39-46.

Fiddaman, Thomas, Oliva, Rogelio and Aranda, R. Rembert (1993) Modeling the Impact of Quality Initiatives Over the Software Product Life Cycle. **Proceedings of the 1993 System Dynamics Conference**. Cancun, Mexico.

Ford, David, Hou, Alex, and Seville, Donald (1993) An Exploration of Systems Product Development at Gadget, Inc.. Working Paper D-4460. System Dynamics Group. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Ford, David and Paynting, Richard (1995) Linking Academic Theory and Industry Practice through Interactive Student Projects: A Case Study of System Dynamics and Product Development. forthcoming in **The Center for Qulaity Management Journal**. Cambridge, MA.

Forrester, Jay W. (1992) Policies, Decisions and Information Sources for Modeling. **European Journal of Operational Reseasrch.** 59:42-63.

Forrester, Jay W. (1961) I**ndustrial Dynamics.** Productivity Press. Cambridge, MA.

Forrester, Jay and Senge, Peter (1980) Tests for Building Confidence in System Dynamics Models. **TIMS Studies in the Management Sciences.** 14:209-228.

Forrester, Nathan B. (1982) A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis. Doctoral thesis. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Fujimoto, Takahiro, Iansiti, Marco and Clark, Kim B. (1992) External Integration in Product Development. Working Paper 92-025. Division of Research. Harvard Business School. Cambridge, MA.

Gomory, Ralph E. (1989) From the 'Ladder of Science' to the Product Development Cycle. **Harvard Business Review.** November.

Goodman, Michael R. (1988) **Study Notes in System Dynamics**. The MIT Press. Cambridge, MA.

Grahan, Alan K. (1980) Parameter Estimation in System Dynamics Modleing. in Randerts Jorgen (Ed.). **Elements of the Ststem Dynamics Method.** Productivity Press. Cambridge, MA.

Griffin, Abbie and Page, Albert (1993) An Interim Report on Measuring Product Development Success and Failure. **Journal of Product Innovation Management.** 10:291-308.

Halpin, Daniel W. and Woodhead, Ronald W. (1980) **Construction Management**. John Wiley & Sons. New York.

Hannon, Bruce and Ruth, Mathias. (1994) **Dynamic Modeling**. Springer-Verlag. New York.

Hauser, John R. and Clausing, Don (1988) The House of Quality. **Harvard Business Review**. May-June. Cambridge, MA.

Hayes, Robert H., Wheelwright, Steven C. and Clark, Kim B. (1988) **Dynamic Manufacturing, Creating the Learning Organization**. The Free Press. New York.

Hoedemaker, Geert M., Blackburn, Joesph D. and Van Wassenhove, Luk N. (1994) Limits to Concurrent Engineering. ORSA/TIMS Semi-annual Meeting. April 25, 1994. Boston, MA.

Homer, Jack (1983) Partial-Model Testing as a Validation Tool for System Dynamics. **Proceedings of 1983 International System Dynamics Conference.** Boston, MA.

Homer, Jack (1983) A Dynamic Model for Analyzing the Emergence of New Medical Technologies. Doctoral thesis. Massachusetts Institute of Technology. Cambridge, MA.

Homer, Jack (1985) Worker Burnout: a dynamic model with implications for prevention and control. **System Dynamics Review.** 1:1.

Homer, Jack, Sterman, John, Greenwood, Brian and Perkola, Markku (1993) Delivery Time Reduction in Pulp and Paper Mill Construction Projects: A Dynamic Analysis of Alternatives. **Proceedings of the 1993 International Syste m Dynamics Conference**. Cancun, Mexico. Monterey Institute of Technology.

Iansiti, Marco (1993a) Real-World R&D: Jumping the Product Generation Gap. **Harvard Business Review**. May-June:138-47.

Iansiti, Marco (1993b) Technology Development and Integration: An Empirical Study of the Interaction Between Applied Science and Product Development. Working Paper 93-024. Division of Research. Harvard Business School. Cambridge, MA.

Iansiti, Marco (1993c) Technology Integration: Managing Technological Evolution in a Complex Environment. Working Paper 93-057. Division of Research. Harvard Business School. Cambridge, MA.

Iansiti, Marco (1992a) Science-Based Product Development: An Empirical Study of the Mainframe Computer Industry. Working Paper 92-083. Division of Research. Harvard Business School. Cambridge, MA.

Iansiti, Marco (1992b) Technology Integration: Exploring the Interaction between Applied Science and Product Development. Working Paper 92-026. Production and Operations Management. Harvard Business School. Cambridge, MA.

Iansiti, M. and Clark, K. B. (1993) Integration and Dynamic Capability: Evidence from Product Development in Automobiles and Mainframe Computers. Working Paper 93-047. Production and Operations Management. Harvard Business School. Cambridge, MA.

Iansiti, M. and Khanna, Tarun (1993) Technological Evolution, System Architecture and the Obsolescence of Firm Capabilities. Working Paper 93-005. Division of Research. Harvard Business School. Cambridge, MA.

Irving, Clive (1993) **Wide-Body, The Triumph of the 747.** William Morrow & Co., Inc. New York.

Jessen, Svein Arne (1988) Can Project Dynamics be Modeled?. in **Proceedings of the 1988 International Conference of the System Dynamics Society**. La Jolla, CA.

Jessen, Svein Arne (1992) **The Nature of Project Leadership.** Scandinavian University Press. Oslo, Norway.

Jessen, Svein Arne (1990) The Motivation of Project Managers, A Study of Variation in Norwegian Project Managers' Motivation and Demotivation by Triangulation of Methods. Doctoral thesis. The Henley Management College and Brunel University.

Kampmann, Christian P. (1992) Feedback Complexity and Market Adjustment: An Experimental Approach. Doctoral thesis. Massachusetts Institute of Technology. Cambridge, MA.

Karu, Zoher Z. (1995). **Signals and Systems Made Ridiculously Simple.** ZiZi Press. Cambridge, MA.

Kim, Daniel H. (1988). Sun Microsystems, Sun3 Product Development/Release Model. Technical Report D-4113, System Dynamics Group. Massachusetts Institute of Technology. Cambridge, MA.

Kleinmuntz, Don N. (1993) Information Processing and Misperceptions of the Implications of Feedback in Dynamic Decision Making. **System Dynamics Review.** 9:3.

Malone, Thomas W. and Crowston, Kevin (1990) What is Coordination Theory and How Can It Help Design Cooperative Work Systems. **Proceedings of the Conference on Computer Supported Cooperative Work.** Los Angeles, CA.

Malone, T. W. and Smith, S. A. (1988) Modeling the Performance of Organizational Structures. **Operations Research**. 36:3:421-436.

Maxam, F. A. (1978) Design Development of the 727-100. Case Study in Aircraft Design, The Boeing 727. American Institute of Aeronautics & Astronautics Professional Study Series. September 14. Washington, DC.

McCord, Kent R., and Eppinger, Steven D. (1993) Managing the Integration Problem in Concurrent Engineering. Working Paper #3594-93. Sloan School of Management. Massachusetts Institue of Technology. Cambridge, MA.

Merrills, Roy (1991) How Northern Telecom Competes on Time. **Managing Product Life Cycles From Start to Finish**. Harvard Business Review Paperback. Cambridge, MA.

Mintzberg, Henry (1979) **The Structuring of Organizations.** Prentice-Hall, Inc. Englewood Cliffs, NJ.

Moder, Joesph J, Phillips, Cecil R. and Davis, Edward W. (1983) **Project Management with CPM, PERT and Precedence Diagramming**. Van Nostrand Reinhold Co. New York.

Montoya-Weiss, Mitzi M. and Calantone, Roger. (1994) Determinants of New Product Performance: A Review and Meta-Analysis. **The Journal of Product Innovation Management**. 11:5:397-417.

Morelli, Mark D. and Eppinger, Steven, D. (1993) Evaluating Communications in Product Development Organizations. Working Paper #3602-93. Sloan School of Management. Massachusetts Institue of Technology. Cambridge, MA.

Murmann, Philipp A. (1994) Expected Development Time Reductions in the German Mechanical Engineering Industry. **The Journal of Product Innovation Management**. 11:3:236-252.

Nevens, T. Michael, Summe, Gregory L. and Uttal, Bro (1991) Commercializing Technology: What the Best Companies Do. **Managing Product Life Cycles From Start to Finish**. Harvard Business Review Paperback. Cambridge, MA.

Nevins, James L. and Whitney, Daniel (1989) **Concurrent Design of Products & Processes, A Strategy for the Next Generation in Manufacturing**. McGraw-Hill. New York.

O'Connor, Paul (1994) Implementing a Stage-Gate Process: A Multi-Company Perspective. **The Journal of Product Innovation Management**. 11:3:183-200.

Ogata, Katsuhiko (1989) **Modern Control Engineering.** Prentice-Hall. Englewood Cliffs, NJ.

Osborne, Sean M. (1993) Product Development Cycle Time Characterization through Modeling of Process Iteration. Master's thesis. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Oppenheim, Alan V. and Schafer, Ronald W. (1989) **Discrete-Time Signal Processing.** Prentice-Hall. Englewood Cliffs, NJ.

Page, Albert L. (1993) Assessing New Product Development Practices and Performance: Establishing Crucial Norms. **The Journal of Product Innovation Management**. 10:4:273-290.

Paich, Mark and Sterman, John D. (1993) Boom, Bust, and Failures to Learn in Experimental Markets. **Management Science**. 39:12:1439-1458.

Patterson, Marvin L. (1993) **Accelerating Innovation, Improving the Process of Product Development**. Van Nostrand Reinhold. New York.

Peterson, Donald E. and Hillkirk, John (1991) **A Better Idea, Redefining the Way Americans Work.** Houghton Mifflin Co. Boston, MA.

Peterson, T. J. and Sutcliffe, P. L. (1992) Systems Engineering as Applied to the Boeing 777. 1992 Aerospace Design Conference. February 3-6, 1992. Irvine, CA. AIAA 92-1010. American Institute of Aeronautics and Astronautics. Washington, DC.

Rechtin, Eberhardt (1991) **Systems Architecting, Creating and Building Complex Systems**. Prentice-Hall. Englewood Cliffs, NJ.

Reichelt, Kimberley S. (1990) Halter Marine: A Case Study in the Dangers of Litigation. Master's thesis. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Richardson, George P. (1995) Loop Polarity, Loop Dominance, and the Concept of Dominant Polarity (1984). **System Dynamics Review**. 11:1:67-88.

Richardson, George P. (1982) Sources of Rising Product Development Times. Technical Report D-3321-1. System Dynamics Group. Sloan School of Management. Massachusetts Instititute of Technology. Cambridge, MA.

Richardson, George P. and Paich, Mark (1980) Analyzing Rising Development Times in a Rapidly Growing, High Technology Firm. Technical Report D-3228-1. System Dynamics Group. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Richardson, George P. and Paich, Mark (1981) Understanding the Behavior of Product Development Times at Codex. Technical Report D-3311. System Dynamics Group. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Richardson, George P. and Pugh III, Alexander L. (1981) **Introduction to System Dynamics Modeling with Dynamo.** MIT Press. Cambridge, MA.

Roberts, Edward B. (1964) Research and Development Policy-Making. **Technology Review**. 66:8:3-7.

Roberts, Edward B. (1974) A Simple Model of R&D Project Dynamics. **R&D Management**. 5:1.

Rosenau, Milton D. and Moran, John J. (1993) **Managing the Development of New Products, Achieving Speed and Quality Simultaneously Through Multifunctional Teamwork**. Van Nostrand Reinhold. New Yo rk.

Rosenthal, Stephen R. (1992) **Effective Product Design and Development, How to Cut Lead Time and Increase Customer Satisfaction.** Business One Irwin. Homewood, IL.

Senge, Peter (1990) **The Fifth Discipline.** Currency Doubleday. New York.

Seville, Donald A. and Kim, Daniel H. (1993) New Product Development Management, Flight Simulator Facilitator's Guide v2.08. unpublished report. Organizational Learning Center. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Smith, Robert P. and Eppinger, Steven D. (1991) Identifying Controlling features of Engineering Design Iteration. Working Paper Working Paper #3348-91-MS. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Sterman, John (1994) Learning in and about Complex Systems. **System Dynamics Review.** 10:2-3:291-330.

Sterman, John (1992) System Dynamics Modeling for Project Management. unpublished working paper. System Dynamics Group. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Sterman, John (1989a) Misperceptions of Feedback in Dynamic Decision Making. **Organizational Behavior and Human Decision Processes**. 43:301-35.

Sterman, John (1989b) Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. **Management Science.** 35:3.

Steward, Donald V. (1981) The Design Structure System: A Method for Managing the Design of Complex Systems. **IEEE Transactions of Engineering Management**.

Stix, Gary. (1991) Plane Geometry, Boeing uses CAD to design 130,000 parts for its new 777. **Scientific American**. March:110-111.

Suh, Nam P. (1990) **The Principles of Design**. Oxford University Press. New York.

Takeuchi, Hirotaka and Nonaka, Ikujiro. (1991) The new new product development game. **Managing Product Life Cycles From Start to Finish**. Harvard Business Review Paperback. Cambridge, MA.

Tank-Nielsen, Carsten. (1980) Sensativity Analysis in System Dynamics. in Randers, Jorgen (Ed.). **Elements of the System Dynamics Method.** Productivity Press. Cambridge, MA.

Thomas, H. Randolph and Napolitan, Carmen L. (1994) **The Effects of Changes on Labor Productivity: Why and How Much.** Construction Industry Institute. Austin, TX.

Thomsett, Michael C. (1990) **The Little Black Book of Project Management.** AMACOM. New York.

Ulrich, Karl T. and Eppinger, Steven D. (1994) **Methodologies for Product Design and Development**. prepublication draft. McGraw-Hill. New York.

Vogel, Brian L. (1993) Concept Development for High-Technology and Medical Products. **Design Management Journal**. Fall: 43-54.

Ward, Allen et al. (1995) The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster. **Sloan Management Review.** Spring.

Watton, Harry B. (1969) **New-Product Planning, A Practical Guide for Diversification**. Prentice-Hall, Inc. Englewood Cliffs, NJ.

Weil, Henry B. et al. (1973) The Dynamics of R&D Strategy. in Roberts, Edward (ed.) **Managerial Applications of System Dynamics**. Productivity Press. Norwalk, CN

Wetherbe, James C. (1995) Principles of Cycle Time Reduction: You Can Have Your Cake and Eat It Too. **Cycle Time Research.** 1:1:1-24. FedEx Center for cycle Time Research. Memphis, TN.

Wheelwright, Steven C. and Clark, Kim B. (1992) **Revolutionizing Product Development, Quantum Leaps in Speed, Efficiency, and Quality**. The Free Press. New York.

Wheelwright, Steven C. and Sasser W. Earl Jr. (1991) The New Product Development Map. **Managing Product Life Cycles From Start to Finish**. Harvard Business Review Paperback. Cambridge, MA.

Womack, James P., Jones, Daniel T. and Roos, Daniel (1990) **The Machine that Changed the World, The Story of Lean Production**. Rawson Associates. New York.

Zaccai, Gianfranco. (1991). How to Make the Client/Consultant Relationship More Like a Basketball Game than a Relay Race. **Design Management Journal**. 2:2.

Appendix 3.1

Model Equations

Model Equation Notes

1. Model equations are in a DYNAMO format. Letters before an equation indicate the type of parameter as follows:
   L - level
   N - initial value of a level
   R - rate
   A - auxiliary
   T - table function
   C - constant
2. To conserve space the phase-arrayed equations have generally been left with passing arguments which represent individual phases as "Phase". Assigning a value to a parameter in this form assigns that value to the parameter for all five phases. Expanding a parameter to vary the values for different phases is done by using five copies of the assignment equation, replacing "Phase" in each with the appropriate phase number. Examples of this customization procedure are illustrated at the External and Internal Precedence Relationship parameters.
3. See Appendix 3.2 Model Parameters for the meanings of specific parameters.

*** PARAMETER VALUE ASSIGNMENT EQUATIONS**
***==========**
*
*** PROCESS**
*** ========**

A Release_Trigger_Sensativity(Phase)=0.6
A Release_Hold_Avg_Time(Phase)=1.06
A Complexity(Phase)=10
A Ref_Complexity(Phase)=100
A BW_Min_Task_duration(Phase)=2
A RW_Min_Task_Duration(Phase)=1
A QA_Min_Task_Duration(Phase)=1
A Coord_Min_duration(Phase)=1

*** SCOPE**
*** ======**
A Task_List(Phase)=1000

*** TARGETS**
* =========
*** Schedule**
A Deadline_Switch=1
A Initial_Proj_Deadline=125
A Time_to_Avg_Exp_Compl_Time(Phase)=1
A Resistance_to_Sched_Slip=2
A Max_DL_Change(Phase)=100
A Max_Proj_DL_Change=100

*** Quality**
A Quality_Goal_Adjust_Time(Phase)=12
A Initial_Quality_Goal(Phase)=0.9
A Project_Quality_Goal=0.9
A Basic_prob_flawed_Task(Phase)=0.5

*** Cost**
A Percent_hourly_Labor(Phase)=0.00
A Avg_Straight_Pay(Phase)=25
A Cost_Markup(Phase)=2
A Overtime_Premium(Phase)=0.50
A Proj_Budget=500000

*** RESOURCES**
* ===========
*** Gross Labor**
A Inital_Headcout(Phase)=0.50
A Headcount_Adjustment_Time(Phase)=8

A Max_Headcount(Phase)=1
A Min_Headcount(Phase)=0.001
A HdctJumpSwitch(Phase)=0 *0 = off and 1 = on
A HdctJumpStartTime(Phase)=1
A HdctJumpStopTime(Phase)=30

*** Labor Allocation**
A alpha(Phase)=100
A Coord_Priority(Phase)=1
A BW_Priority(Phase)=3
A RW_Priority(Phase)=1
A QA_Priority(Phase)=1
A BW_Labor_Delay(Phase)=2
A RW_Labor_Delay(Phase)=1
A QA_Labor_Delay(Phase)=8
A Coord_Labor_Delay(Phase)=4

*** Workweek**
A Max_Workweek(Phase)=140
A Normal_Workweek(Phase)=40
A Wrkwk_Avg_Time(Phase)=4

*** Experience**
A Exper_Assim_Time(Phase)=1
A Avg_New_member_Exper(Phase)=6
A Ref_Exper(Phase)=50

*** Productivity**
A Ref_Coord_Prdctvty(Phase)=1
A Ref_BW_Prdctvty(Phase)=2
A Ref_RW_Prdctvty(Phase)=1
A Ref_QA_Prdctvty(Phase)=1.75
A BW_Prdctvty_Avg_Time(Phase)=1
A RW_Prdctvty_Avg_Time(Phase)=1
A QA_Prdctvty_Avg_Time(Phase)=1
A Coord_Prdctvty_Avg_Time(Phase)=1
A Min_Exp_Coord_Prdy(Phase)=0.1
A Min_Exp_QA_Prdy(Phase)=0.1
A Min_Exp_BW_Prdy(Phase)=0.1
A Min_Exp_RW_Prdy(Phase)=0.1
A Adjust_Expect_BW_Prdctvty_Time(Phase)=1
A Change_Expected_QA_Prdctvty_Time(Phase)=1

A Ch_Expect_Coord_Prdctvty_time(Phase)=1
A QA_Prdctvty_Report_Time(Phase)=1
A Report_Coord_Prdctvty_time(Phase)=1
A Report_BW_Prdctvty_Time(Phase)=1
A Wt_to_Current_BW_Prdctvty(Phase)=1.00
A BW_Prdctvty_Influences_Time(Phase)=1
A Avg_Act_BW_Prdctvty_Time(Phase)=1
A Avg_Act_RW_Prdctvty_Time(Phase)=1
A Avg_Act_QA_Prdctvty_Time(Phase)=1
A Avg_Act_Coord_Prdctvty_Time(Phase)=1
A Ref_Qual_of_Practice(Phase)=5

*** MODEL OPERATION**
* ================
C LastPhase=5
C PhaseNo(1)=1
C PhaseNo(2)=2
C PhaseNo(3)=3
C PhaseNo(4)=4
C PhaseNo(5)=5
A CloseEnough=0.01
A Test_Input_1(Phase)=1 * ITERATION SWITCH
A Test_Input_3(Phase)=1 * NOT USED
A Test_Input_2(Phase)=0 * RELEASE"DUMP" SWITCH
A QA_STATUS(Phase)=1 * switches off when Resourses included
A COORD_STATUStest(Phase)=0.5 * switches off when Resourses included
A ResourseSwitch(Phase)=1
A Budget_Switch=1
A Quality_Goal_Switch=1

***ARRAY STRUCTURES**
*==================
*USE A 2D MATRIX for each inter-phase variable
* (i.e. each inter-phase relationship)
* DIFFERENTIATE MATRIX "COLUMNS" FROM "ROWS" BY CALLING THEM
*"UP" (upstream phase) and "DOWN" (downstream phase)

*** THE PROJECT NETWORK (PHASE DEPENDENCY) MARIX**
*----------
*

* value of 1 means the "down" phase depends on the "up" phase,
* i.e. Up feeds down
* value of 0 means the "down" phase does not depend on the "up" phase
*
* arrange network (i.E. Number the phases) so that smaller numbered
* phases feed higher numbered phases
*
* Delcare Dependency Matrices Variables
A Dep(up,down)=Dependency(up,down)

*** DIAGONAL OF MATRIX = 0 TO PREVENT "DEATH GRIP" SELF-DEPENDENCIES**
A Dependency(1,1)=0
A Dependency(2,2)=0
A Dependency(3,3)=0
A Dependency(4,4)=0
A Dependency(5,5)=0

*** UPPER RIGHT HALF OF MATRIX = 0 TO PREVENT "DEATH GRIP" INTERLOCKING**
A Dependency(2,1)=0
A Dependency(3,1)=0
A Dependency(4,1)=0
A Dependency(5,1)=0
A Dependency(3,2)=0
A Dependency(4,2)=0
A Dependency(5,2)=0
A Dependency(4,3)=0
A Dependency(5,3)=0
A Dependency(5,4)=0

*** LOWER LEFT HALF OF MATRIX DESCRIBES THE PROJECT'S PHASE DEPENDENCY NETWORK**
A Dependency(1,2)=1
A Dependency(1,3)=1
A Dependency(1,4)=0
A Dependency(1,5)=0
A Dependency(2,3)=1
A Dependency(2,4)=0
A Dependency(2,5)=0
A Dependency(3,4)=0
A Dependency(3,5)=0
A Dependency(4,5)=0

A TaskListScale(up,down)=Task_List(up)/Task_List(down)

*** AVAILABLE-WORK - INTERNAL PRECEDENCE RELATIONSHIPS**

*===

A Fraction_Avail_due_to__Int_gate(Phase)=TABHL(T5(*,Phase),Fract_Compl_and_Rel(Phase),0,1,0.10)

*** INTERNAL PRECEDENCE RELATIONSHIPS**

T T5(*,1)=0.2/0.40/0.60/0.650/0.70/0.750/0.750/0.80/0.95/0.98/1.00 * hyperbolic

T T5(*,2)=0.2/0.40/0.60/0.650/0.70/0.750/0.750/0.80/0.95/0.98/1.00 * hyperbolic

T T5(*,3)=0.2/0.40/0.60/0.650/0.70/0.750/0.750/0.80/0.95/0.98/1.00 * hyperbolic

T T5(*,4)=0/0/0/0/0/0/0/0/0/0 *Closed Gate

T T5(*,5)=0/0/0/0/0/0/0/0/0/0 *Closed Gate

*** LIBRARY OF INTERNAL PRECEDENCE RELATIONSHIP TABLES**

* CAN DELETE PHASE BY ALLOWING NO BASEWORK

* T T5(*,1)=0/0/0/0/0/0/0/0/0/0 *Closed Gate

* NO INTERNAL GATE

* T T5(*,1)=1/1/1/1/1/1/1/1/1/1.00 *Completely Open Gate

* LOCKSTEP INTERNAL GATE

* T T5(*,1)=0.1/0.2/0.3/0.4/0.5/0.6/0.7/0.8/0.9/1.00/1.00 * lockstep

* TIGHT LOCKSTEP

* T T5(*,1)=0.05/0.15/0.25/0.35/.45/.55/.65/.75/.85/.95/1.00 * tight lockstep

* FAST LOCKSTEP

* T T5(*,1)=0.2/0.4/0.6/0.8/1/1/1/1/1/1.00 * Fast lockstep

* 20%-THEN-FREE INTERNAL GATE TO RELEASE CONTROL TO LABOR SECTORS

* T T5(*,1)=0.2/0.2/1.0/1.0/1.0/1.0/1.0/1.0/1.00/1.00 * 20% then free

* 20%-THEN-FREE INTERNAL GATE TO RELEASE CONTROL TO LABOR SECTORS

* T T5=0.2/0.2/1.0/1.0/1.0/1.0/1.0/1.0/1.00/1.00

* COBRA DESIGN PHASE

* T T5(*,1)=0.2/0.40/0.60/0.650/0.70/0.750/0.750/0.80/0.95/0.98/1.00

*** AVAILABLE-WORK - EXTERNAL PRECEDENCE RELATIONSHIPS**

* ===

*** SET PRECENDENCE RELATIONSHIPS BETWEEN PHASES**
*** SET INSIDE AN "IF THEN" TO SET INDEPENDENT UPSTREAM**
*** CONCURRENCE TO 1.00 (disables independent phase precedences)**

A
Concurrence(up,down)=FIFZE(1.00,TABHL(T6(*,up,down),Fraction_Released(up),0,1,0.10),Dependency (up,down))

*** THESE TABLES NOT USED, THEY FILL THE UPPER RIGHT HALF OF THE MATRIX**
T T6(*,1,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,2,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/ 0.00/0.00/0.00/1.00
T T6(*,2,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,3,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,3,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,3,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,4,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,4,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,4,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,4,4)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,5,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,5,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,5,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00
T T6(*,5,4)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0. 00/1.00
T T6(*,5,5)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00

*** THESE TABLES DEFINE THE EXTERNAL PRECEDENCE RELATIONSHIPS**
T T6(*,1,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * Almost Sequential
T T6(*,1,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * Almost Sequential
T T6(*,2,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * Almost Sequential
T T6(*,3,4)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * PPS to RelQual
T T6(*,1,4)=0.00/0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20, JUMP to 100
T T6(*,1,5)=0.00/0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20, JUMP to 100
T T6(*,2,4)=0.00/0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20, JUMP to 100
T T6(*,2,5)=0.00/ 0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20, JUMP to 100
T T6(*,3,5)=0.00/0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20, JUMP to 100
T T6(*,4,5)=0.00/0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20, JUMP to 100

*** LIBRARY OF EXTERNAL PRECEDENCE RELATIONSHIPS**
* ----------
* NO GATE

* T T6=1/1/1/1/1/1.0000000000/1.0/1.0/1.0/1.0/1.0 * No Gate

* PARALLEL
* T T6=0.00/1.00/1.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Parrallel

* LOCKSTEP
* T T6=0.00/.1/.2/.3/.4/.5/.6/.7/.8/.9/1.0 * Lockstep

* DELAY TO 20% THEN OPEN TO 100% IMMEDIATELY
* T T6=0.00/0.00/0.0/1/1/1/1.0/1.0/1.0/1.0/1.0 * Delay to 20%, JUMP to 100%

* DELAY TO 10% THEN RAMP TO 100% AT 60%
* T T6=0.00/0.00/0.2/0.4/0.6/0.8/1.0/1.0/1.0/1.0/1.0 * Delay to 10, ramp to 100

* DELAY TO 60% THEN OPEN FULLY TO 100%
* T T6=0.00/0.0/0.0/0.0/0.0/0.0/0.0/1.0/1.0/1.0/1.0 *Delay to 10% then JUMP to 100%

* DELAY TO 50% THEN OPEN 20% per 10%
* T T6=0.00/0.00/0.0/0.0/0.0/0.20/0.40/0.80/0.90/1.00/1.00 * Delay to 50%, ramp

* SEQUENTIAL PHASES
* T T6=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * Almost Sequential

* S-CURVE
* T T6=0.00/0.05/0.15/0.30/0.45/0.525/0.60/0.65/0.85/0.95/1.00 * S Curve

A
Fraction_Avail_due_to__Ext_gates(Phase)=MIN(FIFZE(1.00,Concurrence(1,Phase),Dependency(1,Phase)),FIFZE(1.00,Concurrence(2,Phase),Dependency(2,Phase)),FIFZE(1.00,Concurrence(3,Phase),Dependency(3,Phase)),FIFZE(1.00,Concurrence(4,Phase),Dependency(4,Phase)),FIFZE(1.00,Concurrence(5,Phase),Dependency(5,Phase)))

*** CORE DEVELOPMENT ACTIVITIES**
* ==========
L
Known_Rework(Phase)=Known_Rework(Phase)+dt*(RW_due_to_InPhase_QA(Phase)+RW_due_to_Corrupted_tasks(Phase)+RW_due_to__Dwnstrm_QA(Phase)-Rework(Phase))
N Known_Rework(Phase)=0

R RW_due_to_InPhase_QA(Phase)=QA_inspection_rate(Phase)*prob_Task_Flawed_and_Found(Phase)

R RW_due_to_Corrupted_tasks(Phase)=(Net_Corrupted_and_Found_Tasks(Phase)-(QA_inspection_rate(Phase)*((Net_Corrupted_and_Found_Tasks(Phase)/(QA_inspection_rate(Phase)+1e-9))*prob_Task_Flawed_and_Found(Phase))))

L Tasks_Completed(Phase)=Tasks_Completed(Phase)+dt*(Basework(Phase)+Rework(Phase)-Release_Tasks(Phase)-RW_due_to_InPhase_QA(Phase)-RW_due_to_Corrupted_tasks(Phase)-Tasks_to_Release_Hold(Phase))
N Tasks_Completed(Phase)=0

R RW_due_to__Dwnstrm_QA(Phase)=Total_Err_disc_by_Dn(Phase)

A QA_inspection_rate(Phase)=MIN(QA_Process_Limit(Phase),QA_Labor_Limit(Phase))
R Rework(Phase)=MIN(RW_Process_Limit(Phase),RW_Labor_Limit(Phase))
R Basework(Phase)=MIN(BW_Process_Limit(Phase),BW_Labor_Limit(Phase))

A Coord_Limit(Phase)=MIN(Coord_Process_Limit(Phase),Coord_Labor_Limit(Phase))

R Release_Tasks(Phase)=FIFZE(QA_inspection_rate(Phase)-RW_due_to_InPhase_QA(Phase)-RW_due_to_Corrupted_tasks(Phase),0,Test_Input_2(Phase))

L Tasks_Released(Phase)=Tasks_Released(Phase)+dt*(Release_Tasks(Phase)+Release_Tasks_from_Hold(Phase)-RW_due_to__Dwnstrm_QA(Phase))
N Tasks_Released(Phase)=0

A BW_Process_Limit(Phase)=BW_Task_Avail_Gap(Phase)/BW_Min_Task_duration(Phase)

A RW_Process_Limit(Phase)=Known_Rework(Phase)/RW_Min_Task_Duration(Phase)
A QA_Process_Limit(Phase)=Tasks_Completed(Phase)/QA_Min_Task_Duration(Phase)
A Coord_Process_Limit(Phase)=Coord_Backlog(Phase)/Coord_Min_duration(Phase)

L Coord_Backlog(Phase)=Coord_Backlog(Phase)+dt*(Current_Coord_added(Phase)-Coord_Limit(Phase))
N Coord_Backlog(Phase)=0

A BW_Tasks_Remaining(Phase)=Task_List(Phase)-(Known_Rework(Phase)+Tasks_Completed(Phase)+Tasks_Released(Phase)+Hold_for_Release(Phase))

A BW_Task_Avail_Gap(Phase)=MAX(0,Tot_Tasks_Avail(Phase)-Tasks_Compl_and_Rel(Phase)-Known_Rework(Phase))

A
Tot_Tasks_Avail(Phase)=Task_List(Phase)*MIN(Fraction_Avail_due_to__Int_gate(Phase),Fraction_Avail_due_to__Ext_gates(Phase))

A Fraction_Released(Phase)=Tasks_Released(Phase)/Task_List(Phase)

A
Tasks_Compl_and_Rel(Phase)=Tasks_Completed(Phase)+Tasks_Released(Phase)+Hold_for_Release(Phase)
A Fract_Compl_and_Rel(Phase)=Tasks_Compl_and_Rel(Phase)/Task_List(Phase)

* ---

*** RELEASE TASKS AS A GROUP EQUATIONS**

L Hold_for_Release(Phase)=Hold_for_Release(Phase)+dt*(Tasks_to_Release_Hold(Phase)-Release_Tasks_from_Hold(Phase))
N Hold_for_Release(Phase)=0

R Tasks_to_Release_Hold(Phase)=Test_Input_2(Phase)*(QA_inspection_rate(Phase)-RW_due_to_InPhase_QA(Phase)-RW_due_to_Corrupted_tasks(Phase))

R
Release_Tasks_from_Hold(Phase)=Test_Input_2(Phase)*Release_Trigger(Phase)*((Hold_for_Release(Phase)/dt)+Tasks_to_Release_Hold(Phase))

A Release_Trigger_Task_Gate(Phase)=FIFGE(1,0,TIME,65)

A Release_Trigger(Phase)=FIFGE(0,1,MAX(Release_Hold_Avg_Stability(Phase),(-1)*Release_Hold_Avg_Stability(Phase)),Release_Trigger_Sensativity(Phase))*Release_Trigger_Task_Gate(Phase)

A Release_Hold_Avg_Stability(Phase)=Release_Hold_Avg(Phase)-Hold_for_Release(Phase)

L Release_Hold_Avg(Phase)=Release_Hold_Avg(Phase)+dt*(Change_in_Release_Hold_Avg(Phase)-Empty_Release_Hold_Avg(Phase))
N Release_Hold_Avg(Phase)=0

R
Change_in_Release_Hold_Avg(Phase)=Test_Input_2(Phase)*((Hold_for_Release(Phase)+Tasks_to_Release_Hold(Phase)-Release_Hold_Avg(Phase))/Release_Hold_Avg_Time(Phase))

R Empty_Release_Hold_Avg(Phase)=FIFZE(0,Release_Hold_Avg(Phase)/dt,Release_Trigger(Phase))

A Tasks_Compl_and_Holding(Phase)=Tasks_Completed(Phase)+Hold_for_Release(Phase)

*** INTERNAL ERRORS**

* ==========

L
Our_Discd_Errors(Phase)=Our_Discd_Errors(Phase)+dt*(Receive_Our_Errors_fr_Dn(Phase)+Disc_Our_Errors(Phase)-Correct_Our_Errors(Phase))
N Our_Discd_Errors(Phase)=0

R Receive_Our_Errors_fr_Dn(Phase)=Total_Err_disc_by_Dn(Phase)

R Disc_Our_Errors(Phase)=RW_due_to_InPhase_QA(Phase)

R Correct_Our_Errors(Phase)=Rework(Phase)*Our_Discd_Error_density(Phase)

L Our_Errors_Released(Phase)=Our_Errors_Released(Phase)+dt*(Release_Errors(Phase)-Receive_Our_Errors_fr_Dn(Phase)+Release_Errors_from_Hold(Phase))
N Our_Errors_Released(Phase)=0

R Release_Errors(Phase)=(Release_Tasks(Phase)*Clean_Task_error_density(Phase))*(1-Test_Input_2(Phase))

L Our_Undiscd_Errors(Phase)=Our_Undiscd_Errors(Phase)+dt*(Generate_Errors(Phase)-Release_Errors(Phase)-Disc_Our_Errors(Phase)-Errors_lost_in_Corrupted_Tasks(Phase)-Errors_to_Release_Hold(Phase))
N Our_Undiscd_Errors(Phase)=0

L
Hold_Errors_for_Release(Phase)=Hold_Errors_for_Release(Phase)+dt*(Errors_to_Release_Hold(Phase)-Release_Errors_from_Hold(Phase))
N Hold_Errors_for_Release(Phase)=0

R
Errors_to_Release_Hold(Phase)=Test_Input_2(Phase)*Clean_Task_error_density(Phase)*Tasks_to_Release_Hold(Phase)

R
Release_Errors_from_Hold(Phase)=Release_Trigger(Phase)*Test_Input_2(Phase)*(Hold_Errors_for_Release(Phase)/dt)

R Generate_Errors(Phase)=(Basework(Phase)+Rework(Phase))*prob_Task_flawed(Phase)

R
Errors_lost_in_Corrupted_Tasks(Phase)=Compl_Task_error_density(Phase)*RW_due_to_Corrupted_tasks(Phase)

A Compl_Task_error_density(Phase)=Our_Undiscd_Errors(Phase)/(Tasks_Completed(Phase)+1e-9)

A Our_Discd_Error_density(Phase)=Our_Discd_Errors(Phase)/(Known_Rework(Phase)+1e-9)

A prob_Task_flawed(Phase)=1-((1-Basic_prob_flawed_Task(Phase))*(1-prob_err_gen_fr_Task_Complexity(Phase))*(1-prob_of_err_gen_by_QofP(Phase)))

A
prob_err_gen_fr_Task_Complexity(Phase)=TABHL(T8,FIFZE(0,Complexity(Phase)/Ref_Complexity(Phase),Test_Input_1(Phase)),0,2,0.20)
T T8=0.00/0.00/0.05/0.25/0.40/0.5000000000/0.57/0.69/0.78/0.85/0.90

A
prob_of_err_gen_by_QofP(Phase)=TABHL(T3,FIFZE(2,Quality_of_Practice(Phase)/Ref_Qual_of_Practice(Phase),Test_Input_1(Phase)),0,1,0.10)
T T3=0.55/0.45/0.36/0.28/0.21/0.15/0.10/0.06/0.03/0.01/0.00

*** FIND INTERNAL ERRORS**

A
Prob_finding_our_error_if_exists(Phase)=MIN(1,TABHL(T2,QA_for_Find_Error(Phase),0,5,0.50)*Task_Complex_effect_on_Err_disc(Phase)*QofP_Error_Disc_effect(Phase))
T T2=0.00/0.335/0.535/0.685/0.81/0.88/0.925/0.96/0.985/1.00/1.00

A
Task_Complex_effect_on_Err_disc(Phase)=TABHL(T9,Complexity(Phase)/Ref_Complexity(Phase),0,2,0.20)
T T9=1.50/1.20/0.95/0.75/0.60/0.50000000/0.45/0.35/0.25/0.20/0.20

A
QofP_Error_Disc_effect(Phase)=TABHL(T4,Quality_of_Practice(Phase)/Ref_Qual_of_Practice(Phase),0,2,0.20)
T T4=0.70/0.80/0.88/0.94/0.98/1.00000000/1.3/1.45/1.52/1.55/1.57

***ERROR DENSITIES**

* ==========
A Rel_Task__error_density(Phase)=Our_Errors_Released(Phase)/(Tasks_Released(Phase)+1e-9)

A Clean_Task_error_density(Phase)=(Compl_Task_error_density(Phase)*(1-Prob_finding_our_error_if_exists(Phase)))/((1-Compl_Task_error_density(Phase))+(Compl_Task_error_density(Phase)*(1-Prob_finding_our_error_if_exists(Phase)))+1e-9)

A prob_Task_flawed_and_Found(Phase)=Prob_finding_our_error_if_exists(Phase)*Compl_Task_error_density(Phase)

*** INHERITED ERRORS**
* ==========

A prob_Task_Corrupted_and_found(up,Phase)=prob_find_Up_error__if_exists(Phase)*Rel_Task__error_density(up)*Dependency(up,Phase)

A Tasks_Corrupted_and_Found(up,Phase)=QA_inspection_rate(Phase)*prob_Task_Corrupted_and_found(up,Phase)

A Corrupted_task_discoveries(Phase)=Tasks_Corrupted_and_Found(1,Phase)+Tasks_Corrupted_and_Found(2,Phase)+Tasks_Corrupted_and_Found(3,Phase)+Tasks_Corrupted_and_Found(4,Phase)+Tasks_Corrupted_and_Found(5,Phase)

A percent_Multiple_Corruption_discoveries(Phase)=(prob_Task_Corrupted_and_found(1,Phase)*prob_Task_Corrupted_and_found(2,Phase))+(prob_Task_Corrupted_and_found(1,Phase)*prob_Task_Corrupted_and_found(3,Phase))+(prob_Task_Corrupted_and_found(1,Phase)*prob_Task_Corrupted_and_found(4,Phase))+(prob_Task_Corrupted_and_found(1,Phase)*prob_Task_Corrupted_and_found(5,Phase))+(prob_Task_Corrupted_and_found(2,Phase)*prob_Task_Corrupted_and_found(3,Phase))+(prob_Task_Corrupted_and_found(2,Phase)*prob_Task_Corrupted_and_found(4,Phase))+(prob_Task_Corrupted_and_found(2,Phase)*prob_Task_Corrupted_and_found(5,Phase))+(prob_Task_Corrupted_and_found(3,Phase)*prob_Task_Corrupted_and_found(4,Phase))+(prob_Task_Corrupted_and_found(3,Phase)*prob_Task_Corrupted_and_found(5,Phase))+(prob_Task_Corrupted_and_found(4,Phase)*prob_Task_Corrupted_and_found(5,Phase))

A
Multiple_Corruption_discoveries(Phase)=QA_inspection_rate(Phase)*percent_Multiple_Corruption_discoveries(Phase)

A Net_Corrupted_and_Found_Tasks(Phase)=Corrupted_task_discoveries(Phase)-Multiple_Corruption_discoveries(Phase)

A Up_Flawed_Tasks_found(up,Phase)=Tasks_Corrupted_and_Found(up,Phase)*TaskListScale(up,Phase)

A
Total_Err_disc_by_Dn(Phase)=Up_Flawed_Tasks_found(Phase,1)+Up_Flawed_Tasks_found(Phase,2)+Up_Flawed_Tasks_found(Phase,3)+Up_Flawed_Tasks_found(Phase,4)+Up_Flawed_Tasks_found(Phase,5)

A
prob_find_Up_error__if_exists(Phase)=MIN(1,Coord_effect_on_Find_Up_Errors(Phase)*QofP_Error_Disc_effect(Phase))

A Coord_effect_on_Find_Up_Errors(Phase)=TABHL(T1,Coord_Status(Phase),0,1,0.10)
T T1=0.00/0.015/0.045/0.115/0.265/0.54/0.715/0.84/0.92/0.975/1.00

*** PROJECT SCHEDULE**
*** ==========**

L Project_Deadline=Project_Deadline+dt*(Proj_Sched_Slip*Deadline_Switch)
N Project_Deadline=Initial_Proj_Deadline

R Proj_Sched_Slip=FIFGE(MIN(Max_Proj_DL_Change,MAX((-1)*Max_Proj_DL_Change,Expected_Proj_Completion_Time - Project_Deadline)),0,Proj_Sched_Press,Resistance_to_Sched_Slip)

A Proj_Sched_Press=FIFGE(MAX((Exp_Compl_Time(LastPhase)-TIME)/(Project_Deadline-TIME),(Project_Deadline-TIME)/(Exp_Co mpl_Time(LastPhase)-TIME)),1,Project_Deadline-TIME,CloseEnough)

A Expected_Proj_Completion_Time=Exp_Compl_Time(LastPhase)

L DL(Phase)=DL(Phase)+dt*Change_DL(Phase)
N DL(Phase)=(PhaseNo(Phase)*Initial_Proj_Deadline)/LastPhase

R Change_DL(Phase)=FIFGE(MIN(Max_DL_Change(Phase),MAX((-1)*Max_DL_Change(Phase),Exp_Compl_Time(Phase)-DL(Phase))),0,Sched_Pressure(Phase),Resistance_to_Sched_Slip)

A Max_Duration(5)=Path5
A Path5=0

A Max_Duration(4)=MAX(Path4,Path45)
A Path4=0
A Path45=Dependency(4,5)*Expected_Duration(5)

A Max_Duration(3)=MAX(Path3,Path34,Path35,Path345)
A Path3=0
A Path34=Dependency(3,4)*Expected_Duration(4)
A Path35=Dependency(3,5)*Expected_Duration(5)
A Path345=(Dependency(3,4)*Dependency(4,5))*(Expected_Duration(4)+Expected_Duration(5))

A Max_Duration(2)=MAX(Path2,Path23,Path24,Path25,Path234,Path235,Path245,Path2345)
A Path2=0
A Path23=Dependency(2,3)*Expected_Duration(3)
A Path24=Dependency(2,4)*Expected_Duration(4)
A Path25=Dependency(2,5)*Expected_Duration(5)
A Path234=(Dependency(2,3)*Dependency(3,4))*(Expected_Duration(3)+Expected_Duration(4))
A Path235=(Dependency(2,3)*Dependency(3,5))*(Expected_Duration(3)+Expected_Duration(5))
A Path245=(Dependency(2,4)*Dependency(4,5))*(Expected_Duration(4)+Expected_Duration(5))
A Path2345=(Dependency(2,3)*Dependency(3,4)*Dependency(4,5))*(Expected_Duration(3)+Expected_Duration(4)+Expected_Duration(5))

A Max_Duration(1)=MAX(Path1,Path12,Path13,Path14,Path15,Path123,Path124,Path125,Path134,Path135,Path145,Path1234,Path1235,Path1245,Path1345,Path12345)
A Path1=0
A Path12=Dependency(1,2)*Expected_Duration(2)
A Path13=Dependency(1,3)*Expected_Duration(3)
A Path14=Dependency(1,4)*Expected_Duration(4)
A Path15=Dependency(1,5)*Expected_Duration(5)
A Path123=(Dependency(1,2)*Dependency(2,3))*(Expected_Duration(2)+Expected_Duration(3))
A Path124=(Dependency(1,2)*Dependency(1,4))*(Expected_Duration(2)+Expected_Duration(4))
A Path125=(Dependency(1,2)*Dependency(2,5))*(Expected_Duration(2)+Expected_Duration(5))

A Path134=(Dependency(1,3)*Dependency(3,4))*(Expected_Duration(3)+Expected_Duration(4))
A Path135=(Dependency(1,3)*Dependency(3,5))*(Expected_Duration(3)+Expected_Duration(5))
A Path145=(Dependency(1,4)*Dependency(4,5))*(Expected_Duration(4)+Expected_Duration(5))
A
Path1234=(Dependency(1,2)*Dependency(2,3)*Dependency(3,4))*(Expected_Duration(2)+Expected_Dur
ation(3)+Expected_Duration(4))
A
Path1235=(Dependency(1,2)*Dependency(2,3)*Dependency(3,5))*(Expected_Duration(2)+Expected_Dur
ation(3)+Expected_Duration(5))
A
Path1245=(Dependency(1,2)*Dependency(2,4)*Dependency(4,5))*(Expected_Duration(2)+Expected_Dur
ation(4)+Expected_Duration(5))
A
Path1345=(Dependency(1,3)*Dependency(3,4)*Dependency(4,5))*(Expected_Duration(3)+Expected_Dur
ation(4)+Expected_Duration(5))
A
Path12345=(Dependency(1,2)*Dependency(2,3)*Dependency(3,4)*Dependency(4,5))*(Expected_Duratio
n(2)+Expected_Duration(3)+Expected_Duration(4)+Expected_Duration(5))

*** SCHEDULE EFFECTS**
*** ==========**
L Time_spent_to_Date(Phase)=Time_spent_to_Date(Phase)+dt*(Time__Phase_Active(Phase))
N Time_spent_to_Date(Phase)=0

R Time__Phase_Active(Phase)=FIFZE(0,1,StartFlag(Phase)*Not_Done(Phase))

A Not_Done(Phase)=FIFGE(1,0,Task_List(Phase)-Tasks_Released(Phase),0.01)

A StartFlag(Phase)=FIFGE(1,0,Task_List(Phase)-0.01,BW_Tasks_Remaining(Phase))

A Expected_Duration(Phase)=Time_spent_to_Date(Phase)+Time_Required(Phase)

***--------Expected Start Times for Phase Deadlines before Phase has started----------**

L Exp_Compl_Time(Phase)=Exp_Compl_Time(Phase)+dt*Change_Exp_Compl_Time(Phase)
N Exp_Compl_Time(Phase)=Expected_Duration(Phase)+Start_Time(Phase)

R Change_Exp_Compl_Time(Phase)=FIFGE(((Expected_Duration(Phase)+Start_Time(Phase))-
Exp_Compl_Time(Phase))/Time_to_Avg_Exp_Compl_Time(Phase),DL(Phase),Time_Required(Phase),1e
-9)

A Start_Time(Phase)=FIFZE(Exp_Start_Time(Phase),Start_Time1(Phase),StartFlag(Phase))

A Exp_Start_Time(1)=0

A Exp_Start_Time(2)=Dependency(1,2)*Expected_Duration(1)

A Exp_Start_Time(3)=MAX(ExpStart3,ExpStart13,ExpStart23,ExpStart123)
A ExpStart3=0
A ExpStart13=Dependency(1,3)*Expected_Duration(1)
A ExpStart23=Dependency(2,3)*Expected_Duration(2)
A ExpStart123=Dependency(1,2)*Expected_Duration(1)+Dependency(2,3)*Expected_Duration(2)

A
Exp_Start_Time(4)=MAX(ExpStart4,ExpStart14,ExpStart24,ExpStart34,ExpStart124,ExpStart134,ExpStart1234)
A ExpStart4=0
A ExpStart14=Dependency(1,4)*Expected_Duration(1)
A ExpStart24=Dependency(2,4)*Expected_Duration(2)
A ExpStart34=Dependency(3,4)*Expected_Duration(3)
A ExpStart124=Dependency(1,2)*Expected_Duration(1)+Dependency(2,4)*Expected_Duration(2)
A ExpStart134=Dependency(1,3)*Expected_Duration(3)+Dependency(3,4)*Expected_Duration(3)
A
ExpStart1234=Dependency(1,2)*Expected_Duration(1)+Dependency(2,3)*Expected_Duration(2)+Dependency(3,4)*Expected_Duration(3)

A
Exp_Start_Time(5)=MAX(ExpStart5,ExpStart15,ExpStart25,ExpStart35,ExpStart45,ExpStart125,ExpStart135,ExpStart145,ExpStart1235,ExpStart1245,ExpStart1345,ExpStart12345)
A ExpStart5=0
A ExpStart15=Dependency(1,5)*Expected_Duration(1)
A ExpStart25=Dependency(2,5)*Expected_Duration(2)
A ExpStart35=Dependency(3,5)*Expected_Duration(3)
A ExpStart45=Dependency(4,5)*Expected_Duration(4)
A ExpStart125=Dependency(1,2)*Expected_Duration(1)+Dependency(2,5)*Expected_Duration(2)
A ExpStart135=Dependency(1,3)*Expected_Duration(1)+Dependency(3,5)*Expected_Duration(3)
A ExpStart145=Dependency(1,4)*Expected_Duration(1)+Dependency(4,5)*Expected_Duration(4)
A
ExpStart1235=Dependency(1,2)*Expected_Duration(1)+Dependency(2,3)*Expected_Duration(2)+Dependency(3,5)*Expected_Duration(3)

A
ExpStart1245=Dependency(1,2)*Expected_Duration(1)+Dependency(2,4)*Expected_Duration(2)+Dependency(4,5)*Expected_Duration(4)
A
ExpStart1345=Dependency(1,3)*Expected_Duration(1)+Dependency(3,4)*Expected_Duration(3)+Dependency(4,5)*Expected_Duration(4)
A
ExpStart12345=Dependency(1,2)*Expected_Duration(1)+Dependency(2,3)*Expected_Duration(2)+Dependency(3,4)*Expected_Duration(3)+Dependency(4,5)*Expected_Duration(4)
***----------**

L Start_Time1(Phase)=Start_Time1(Phase)+dt*Wait_Time(Phase)
N Start_Time1(Phase)=0

R Wait_Time(Phase)=FIFZE(1,0,StartFlag(Phase))

*** For testing: sched press is set = 1.00 for "no effect"**

A
Sched_Pressure(Phase)=FIFZE(1,MIN(10,FIFGE(MAX(Time_Required(Phase)/Time_to_Deadline(Phase),Time_to_Deadline(Phase)/Time_Required(Phase)),1,Time_Required(Phase),CloseEnough)),ResourseSwitch(Phase)*Deadline_Switch)

A Time_to_Deadline(Phase)=FIFGE(0.01,(DL(Phase)-TIME),0,(DL(Phase)-TIME))

A Sched_Error_Disc_effect(Phase)=TABHL(T10,Sched_Pressure(Phase),0,5,0.50)
T T10=0.7/0.9/1.00/1.20/1.60/2.20/3.00/4.00/5.20/6.60/8.20

A
Time_Required(Phase)=FIFZE(Initial_Proj_Deadline/LastPhase,Required_Personweeks(Phase)/(Headcount(Phase)+1e-9),StartFlag(Phase))

A Sched_BW_Press_effect(Phase)=TABHL(TL7,Sched_Pressure(Phase),0,5,0.50)
T TL7=1.00/1.01/1.04/1.07/1.12/1.17/1.25/1.32/1.44/1.58/1.73

A
Sched_Workweek_effect(Phase)=TABHL(TL10,Time_Required(Phase)/Time_to_Deadline(Phase),0,5,0.5)
T TL10=0.97/0.99/1.0000000000000000/1.05/1.15/1.30/1.50/1.80/2.20/2.70/3.30

*** REPORTING AND TESTING EQUATIONS**

*** ==========**

L Cycle_Time(Phase)=Cycle_Time(Phase)+DT*(Cycle_Time_Change(Phase))
N Cycle_Time(Phase)=0

R Cycle_Time_Change(Phase)=FIFZE(0,1,(StartFlag(Phase)*NotStoppedFlag(Phase)))

A NotStoppedFlag(Phase)=FIFGE(0,1,Tasks_Released(Phase),(Task_List(Phase)-CloseEnough))

A StoppedFlag(Phase)=FIFGE(1,0,Tasks_Released(Phase),(Task_List(Phase)-CloseEnough))

A Percent_Tasks_in_Rework(Phase)=Known_Rework(Phase)/Task_List(Phase)

A
Sum_of_Tasks(Phase)=Tasks_Completed(Phase)+Tasks_Released(Phase)+Known_Rework(Phase)+Hold_for_Release(Phase)

*** LABOR**
*** =======**
A Total_Hdct=SUM(Headcount)

L Headcount(Phase)=Headcount(Phase)+dt*(Change_Headcount(Phase))
N Headcount(Phase)=Inital_Headcout(Phase)

*** ----------Heacdcount Jump Equations---------**

R Change_Headcount(Phase)=(HdctJump(Phase)+HdctJumpBack(Phase))+MIN(Max_Headcount(Phase)-Headcount(Phase),(Required_Headcount(Phase)-Headcount(Phase))/Headcount_Adjustment_Time(Phase))

A HdctJump(Phase)=HdctJumpSwitch(Phase)*FIFZE(0,(Min_Headcount(Phase)-Headcount(Phase))/dt,HdctJumpStartFlag(Phase)*HdctJumpNotStoppedFlag(Phase))

L HdctJumpStore(Phase)=HdctJumpStore(Phase)+dt*ChangeHdctJumpStore(Phase)
N HdctJumpStore(Phase)=0

A ChangeHdctJumpStore(Phase)=FIFZE(Min_Headcount(Phase)-Headcount(Phase)/dt,0,HdctJumpStartTime(Phase)-TIME)

A HdctJumpStartFlag(Phase)=FIFGE(1,0,TIME,HdctJumpStartTime(Phase))

A HdctJumpNotStoppedFlag(Phase)=FIFGE(0,1,TIME,HdctJumpStopTime(Phase))

A HdctJumpBack(Phase)=HdctJumpSwitch(Phase)*FIFZE(-
1*HdctJumpStore(Phase)/dt,0,HdctJumpStopTime(Phase)-TIME)

*** ----------**
A Gross_Labor(Phase)=Headcount(Phase)*Workweek(Phase)

A BW_Labor(Phase)=Labor_Fraction_to_BW(Phase)*Gross_Labor(Phase)
A RW_Labor(Phase)=Labor_Fraction_to_RW(Phase)*Gross_Labor(Phase)

A Coord_Labor(Phase)=Labor_Fraction_to_Coord(Phase)*Gross_Labor(Phase)

A QA_Labor(Phase)=Labor_Fraction_to_QA(Phase)*Gross_Labor(Phase)
A
Required_Headcount(Phase)=FIFZE(0,(Required_Personweeks(Phase)*Budget_Rqrd_Headct_effect)/(Time_to_Deadline(Phase)+0.1),NotStoppedFlag(Phase))

A Required_Personweeks(Phase)=Total_Labor_Required(Phase)/Normal_Workweek(Phase)

A
Total_Labor_Required(Phase)=QA_Labor_Required(Phase)+Coord_Labor_Required(Phase)+BW_Labor_Required(Phase)+RW_Labor_Required(Phase)

*** LABOR ALLOCATION**
*** ==========**
A Coord_Labor_Required(Phase)=Coord_Process_Limit(Phase)/Expect_Coord_Prdctvty(Phase)
A BW_Labor_Required(Phase)=BW_Process_Limit(Phase)/Expect_BW_Prdctvty(Phase)
A RW_Labor_Required(Phase)=RW_Process_Limit(Phase)/Expect_RW_Prdctvty(Phase)
A QA_Labor_Required(Phase)=QA_Process_Limit(Phase)/Expected_QA_Prdctvty(Phase)

A Labor_Fraction_to_Coord(Phase)=Press_for_Coord(Phase)/Total_Pressure_for_Activites(Phase)
A Labor_Fraction_to_BW(Phase)=Press_for_BW(Phase)/Total_Pressure_for_Activites(Phase)
A Labor_Fraction_to_RW(Phase)=Press_for_RW(Phase)/Total_Pressure_for_Activites(Phase)
A Labor_Fraction_to_QA(Phase)=Press_for_QA(Phase)/Total_Pressure_for_Activites(Phase)

A
Press_for_BW(Phase)=EXP(((BW_Labor_Required(Phase)*BW_Priority(Phase)*Sched_BW_Press_effect(Phase)*Cost_effect_on_BW_Import)/alpha(Phase))+0.01)

A
Press_for_RW(Phase)=Quality_Goal_Switch*(EXP((RW_Labor_Required(Phase)*RW_Priority(Phase)*Qual_Gap_effect_on_QARW_priority)/alpha(Phase))+0.01)

A
Press_for_Coord(Phase)=Quality_Goal_Switch*((EXP((Coord_Labor_Required(Phase)*Coord_Priority(Phase)*Qual_Gap_effect_on_Coord_Import)/alpha(Phase)))+0.01)

A
Press_for_QA(Phase)=Quality_Goal_Switch*(EXP((QA_Labor_Required(Phase)*QA_Priority(Phase)*Qual_Gap_effect_on_QARW_priority)/alpha(Phase))+0.01)

A
Total_Pressure_for_Activites(Phase)=Press_for_QA(Phase)+Press_for_Coord(Phase)+Press_for_BW(Phase)+Press_for_RW(Phase)+1e-9

*** WORKWEEK**
*** ============**

L Avg_Wrkwk(Phase)=Avg_Wrkwk(Phase)+dt*(Avg_Wrkwk_Change(Phase))
N Avg_Wrkwk(Phase)=Normal_Workweek(Phase)

R Avg_Wrkwk_Change(Phase)=(Workweek(Phase)-Avg_Wrkwk(Phase))/Wrkwk_Avg_Time(Phase)

A
Workweek(Phase)=MIN(Normal_Workweek(Phase)*Sched_Workweek_effect(Phase),Max_Workweek(Phase))

*** EXPERIENCE**
*** ============**
L Cumm_Exper(Phase)=Cumm_Exper(Phase)+dt*(Change_Cum_Exper(Phase))
N Cumm_Exper(Phase)=Avg_New_member_Exper(Phase)*Inital_Headcout(Phase)

R
Change_Cum_Exper(Phase)=FIFGE(Net_Exper_Gain(Phase)/Exper_Assim_Time(Phase),0,Cumm_Exper(Phase),0)

A
Avg_memb_Exper(Phase)=FIFGE(Cumm_Exper(Phase)/Headcount(Phase),Cumm_Exper(Phase),Headcount(Phase),1.0)

A Exper_Lost(Phase)=(-1)*MIN(0,Change_Headcount(Phase))*Avg_memb_Exper(Phase)

A Exper_index(Phase)=(0.80)**(LOGN(Cumm_Exper(Phase)/Ref_Exper(Phase))/LOGN(2))

A Net_Exper_Gain(Phase)=Basework(Phase)+New_Memb_Exper_Gain(Phase)-Exper_Lost(Phase)

A
New_Memb_Exper_Gain(Phase)=Avg_New_member_Exper(Phase)*MAX(0,Change_Headcount(Phase))

A Exper_on_Prdctvty_effect(Phase)=TABHL(TL13,Exper_index(Phase),0,5,0.50)
T TL13=1.33/1.30/1.24/1.18/1.1/1.00/.9/0.82/0.76/0.72/0.70

*** BASEWORK PRODUCTIVITY**
*** ==========**
L Actual_BW_Prdctvty(Phase)=Actual_BW_Prdctvty(Phase)+dt*Change_Actual_BW_Prdctvty(Phase)
N Actual_BW_Prdctvty(Phase)=Ref_BW_Prdctvty(Phase)

R
Change_Actual_BW_Prdctvty(Phase)=(Current_BW_Prdctvty(Phase)/Avg_Act_BW_Prdctvty_Time(Phase))*Exper_on_Prdctvty_effect(Phase)*Coord_effect_on_Prdy(Phase)

A
BW_Labor_Limit(Phase)=(FIFZE(1e10,Actual_BW_Prdctvty(Phase)*BW_Labor(Phase),ResourseSwitch(Phase)))/BW_Labor_Delay(Phase)

L Expect_BW_Prdctvty(Phase)=Expect_BW_Prdctvty(Phase)+dt*(Change_Expect_BW_Prdctvty(Phase))
N Expect_BW_Prdctvty(Phase)=Ref_BW_Prdctvty(Phase)

R
Change_Expect_BW_Prdctvty(Phase)=MAX(Change_Avg_BW_Prdctvty(Phase)+BW_Prdctvty_Influences(Phase),Min_Exp_BW_Prdy(Phase)-Expect_BW_Prdctvty(Phase))

A Change_Avg_BW_Prdctvty(Phase)=(Current_BW_Prdctvty(Phase)-
Expect_BW_Prdctvty(Phase))/BW_Prdctvty_Avg_Time(Phase)

A
BW_Prdctvty_Influences(Phase)=((Wt_to_Current_BW_Prdctvty(Phase)*Current_BW_Prdctvty(Phase))+((1-Wt_to_Current_BW_Prdctvty(Phase))*Historical_BW_Prdctvty_Belief(Phase))-Expect_BW_Prdctvty(Phase))/BW_Prdctvty_Influences_Time(Phase)

A Current_BW_Prdctvty(Phase)=Basework(Phase)/(BW_Labor(Phase)+1e-9)

A Historical_BW_Prdctvty_Belief(Phase)=Ref_BW_Prdctvty(Phase)

*** REWORK PRODUCTIVITY**
*** ======================**
L Actual_RW_Prdctvty(Phase)=Actual_RW_Prdctvty(Phase)+dt*Change_Actual_RW_Prdctvty(Phase)
N Actual_RW_Prdctvty(Phase)=Ref_RW_Prdctvty(Phase)

R
Change_Actual_RW_Prdctvty(Phase)=(Current_RW_Prdctvty(Phase)/Avg_Act_RW_Prdctvty_Time(Phase))*Exper_on_Prdctvty_effect(Phase)*Coord_effect_on_Prdy(Phase)

A
RW_Labor_Limit(Phase)=(FIFZE(1e50,Actual_RW_Prdctvty(Phase)*RW_Labor(Phase),ResourseSwitch(Phase)))/RW_Labor_Delay(Phase)

L Expect_RW_Prdctvty(Phase)=Expect_RW_Prdctvty(Phase)+dt*(Change_Expect_RW_Prdctvty(Phase))
N Expect_RW_Prdctvty(Phase)=Ref_RW_Prdctvty(Phase)

R
Change_Expect_RW_Prdctvty(Phase)=MAX(Change_Avg_RW_Prdctvty(Phase),Min_Exp_RW_Prdy(Phase)-Expect_RW_Prdctvty(Phase))

A Change_Avg_RW_Prdctvty(Phase)=(Current_RW_Prdctvty(Phase)-Expect_RW_Prdctvty(Phase))/RW_Prdctvty_Avg_Time(Phase)

A Current_RW_Prdctvty(Phase)=Rework(Phase)/(RW_Labor(Phase)+1e-9)

*** QA PRODUCTIVITY**
*** =================**
L Actual_QA_Prdctvty(Phase)=Actual_QA_Prdctvty(Phase)+dt*Change_Actual_QA_Prdctvty(Phase)
N Actual_QA_Prdctvty(Phase)=Ref_QA_Prdctvty(Phase)

R
Change_Actual_QA_Prdctvty(Phase)=(Current_QA_Prdctvty(Phase)/Avg_Act_QA_Prdctvty_Time(Phase))*Exper_on_Prdctvty_effect(Phase)*Coord_effect_on_Prdy(Phase)

A
QA_Labor_Limit(Phase)=(FIFZE(1e50,QA_Labor(Phase)*Actual_QA_Prdctvty(Phase),ResourseSwitch(Phase)))/QA_Labor_Delay(Phase)

L
Expected_QA_Prdctvty(Phase)=Expected_QA_Prdctvty(Phase)+dt*(Change_Expect_QA_Prdctvty(Phase))
N Expected_QA_Prdctvty(Phase)=Ref_QA_Prdctvty(Phase)

R
Change_Expect_QA_Prdctvty(Phase)=MAX(Change_Avg_QA_Prdctvty(Phase),Min_Exp_QA_Prdy(Phase)-Expected_QA_Prdctvty(Phase))

A Change_Avg_QA_Prdctvty(Phase)=(Current_QA_Prdctvty(Phase)-Expected_QA_Prdctvty(Phase))/QA_Prdctvty_Avg_Time(Phase)

A Current_QA_Prdctvty(Phase)=QA_inspection_rate(Phase)/(QA_Labor(Phase)+1e-9)

*** COORDINATION PRODUCTIVITY**
***** ==========
L
Actual_Coord_Prdctvty(Phase)=Actual_Coord_Prdctvty(Phase)+dt*Change_Actual_Coord_Prdctvty(Phase)
N Actual_Coord_Prdctvty(Phase)=Ref_Coord_Prdctvty(Phase)

R
Change_Actual_Coord_Prdctvty(Phase)=(Current_Coord_Prdctvty(Phase)/Avg_Act_Coord_Prdctvty_Time(Phase))*Exper_on_Prdctvty_effect(Phase)

A
Coord_Labor_Limit(Phase)=(Coord_Labor(Phase)*Actual_Coord_Prdctvty(Phase))/Coord_Labor_Delay(Phase)

A Coord_effect_on_Prdy(Phase)=TABHL(TL2,Coord_Status(Phase),0,1,0.10)
T TL2=0.195/0.41/0.575/0.725/0.825/0.89/0.945/0.96/0.975/0.985/1.00

A Coord_effect_on_QofP(Phase)=TABHL(TL3,Coord_Status(Phase),0,2,0.20)
T TL3=0.00/0.06/0.18/0.36/0.6/0.9/1.28/1.66/1.84/1.96/2.00

A
Coord_Status(Phase)=FIFZE(COORD_STATUStest(Phase),Coord_Labor(Phase)/(Coord_Labor_Required(Phase)+1e-9),ResourseSwitch(Phase))

L
Expect_Coord_Prdctvty(Phase)=Expect_Coord_Prdctvty(Phase)+dt*(Change_Expect_Coord_Prdctvty(Phase))
N Expect_Coord_Prdctvty(Phase)=Ref_Coord_Prdctvty(Phase)

R
Change_Expect_Coord_Prdctvty(Phase)=MAX(Change_Avg_QA_Prdctvty(Phase),Min_Exp_Coord_Prdy(Phase)-Expect_Coord_Prdctvty(Phase))

A Change_Avg_Coord_Prdctvty(Phase)=(Current_Coord_Prdctvty(Phase)-Expect_Coord_Prdctvty(Phase))/Coord_Prdctvty_Avg_Time(Phase)

A Current_Coord_Prdctvty(Phase)=Coord_Limit(Phase)/(Coord_Labor(Phase)+1e-9)

A
Current_Coord_added(Phase)=RW_due_to__Dwnstrm_QA(Phase)+RW_due_to_Corrupted_tasks(Phase)

*** COST**
*** =====**
L Phase_Cost_to_Date(Phase)=Phase_Cost_to_Date(Phase)+dt*Cumm_Cost(Phase)
N Phase_Cost_to_Date(Phase)=0

R
OT_Cost(Phase)=(Avg_Straight_Pay(Phase)*Overtime_Premium(Phase))*Cost_Markup(Phase)*Over_Time(Phase)

R Straight_Cost(Phase)=Avg_Straight_Pay(Phase)*Straight_Time(Phase)*Cost_Markup(Phase)

A Cumm_Cost(Phase)=OT_Cost(Phase)+Straight_Cost(Phase)

A Over_Time(Phase)=(Gross_Labor(Phase)-Straight_Time(Phase))*Percent_hourly_Labor(Phase)

A Straight_Time(Phase)=Headcount(Phase)*Normal_Workweek(Phase)

*** QUALITY**
*** ========**

L Quality_Goal(Phase)=Quality_Goal(Phase)+dt*(Change_Quality_Goal(Phase))
N Quality_Goal(Phase)=Initial_Quality_Goal(Phase)

R Change_Quality_Goal(Phase)=(Current_Known_Quality(Phase)-
Quality_Goal(Phase))/Quality_Goal_Adjust_Time(Phase)

A Current_Known_Quality(Phase)=MAX(0,1-Our_Discd_Error_density(Phase))

A Project_Errors_Released=SUM(Our_Errors_Released)

*** COST CONTROL**
*** ==========**
A Forecasted_Costs=Avg_Cost*(MAX(Project_Deadline-TIME,0))

A Avg_Cost=Project_Cost_to_Date/(TIME+1e-9)

A Project_Cost_to_Date=SUM(Phase_Cost_to_Date)

A Tot_Exp_Costs=Forecasted_Costs+Project_Cost_to_Date

A Budget_Surplus=(Proj_Budget-Tot_Exp_Costs)*Budget_Switch

A Budget_Status=Budget_Surplus/Proj_Budget

A Budget_Rqrd_Headct_effect=TABHL(TC1,Budget_Status*Budget_Switch,-1.00,0,0.10)
T TC1=0.00/0.05/0.10/0.20/0.25/0.35/0.40/0.50/0.7/0.90/1.00

A Cost_effect_on_BW_Import=TABHL(TC2,Budget_Status*Budget_Switch,-0.0,2.0,0.20)
T TC2=1.87/1.58/1.35/1.17/1.06/1.00/0.98/0.95/0.89/0.81/0.65

*** QUALITY OF PRACTICE**
*** ==========**
A
QA_for_Find_Error(Phase)=FIFZE(QA_STATUS(Phase),QA_Labor(Phase)/(QA_Labor_Required(Phase)
+1e-9),ResourseSwitch(Phase))

A
Quality_of_Practice(Phase)=FIFZE(Ref_Qual_of_Practice(Phase),Ref_Qual_of_Practice(Phase)*Exper_ef

fect_on_QofP(Phase)*Sched_Qual_of_Prac_effect(Phase)*Fatigue_Qual_of_Prac_effect(Phase)*Coord_effect_on_QofP(Phase),ResourseSwitch(Phase))

A Exper_effect_on_QofP(Phase)=TABHL(T7,Exper_index(Phase),0,5,0.50)
T T7=2.50/2.4/2.2/1.9/1.5/1.00/.8/.6/5/.4/.35

A Sched_Qual_of_Prac_effect(Phase)=TABHL(T11,Sched_Pressure(Phase),1,10,0.90)
T T11=1/0.99/0.97/0.94/0.90/0.85/0.79/0.72/0.64/0.55/0.45

A
Fatigue_Qual_of_Prac_effect(Phase)=TABHL(TL12,Avg_Wrkwk(Phase)/Normal_Workweek(Phase),0,5,0.50)
T TL12=1.05/1.05/1.0000000000/0.98/0.94/0.88/0.80/0.70/0.58/0.44/0.44

*** QUALITY CONTROL**
* ==========
A Percent_RW_goal=1.00-Project_Quality_Goal

A
Proj_Tasks_Compl_and_Rel=SUM(Tasks_Completed)+SUM(Tasks_Released)+SUM(Hold_for_Release)

A Current_Project_Rework_Percent=SUM(Known_Rework)/(Proj_Tasks_Compl_and_Rel+1e-9+SUM(Known_Rework))

A Proj_Quality_Gap=Percent_RW_goal-Current_Project_Rework_Percent*Quality_Goal_Switch

A Qual_Gap_effect_on_Coord_Import=TABHL(TQ1,Proj_Quality_Gap*Quality_Goal_Switch,-1.00,0.00,0.10)
T TQ1=2.10/1.90/1.72/1.56/1.42/1.30/1.20/1.12/1.06/1.02/1.00

A Qual_Gap_effect_on_QARW_priority=TABHL(TQ2,Proj_Quality_Gap*Quality_Goal_Switch,-1.00,0.00,0.10)
T TQ2=2.20/2.14/2.07/1.99/1.90/1.80/1.68/1.54/1.38/1.20/1.00

Appendix 3.2

Model Variables

Actual_BW_Prdctvty(Phase): the tasks initially completed per developer-hour. tasks per developer-hour.

Actual_Coord_Prdctvty(Phase): the tasks coordinated per developer-hour. tasks per developer-hour.

Actual_QA_Prdctvty(Phase): the tasks inspected for flaws per developer-hour. tasks per developer-hour.

Actual_RW_Prdctvty(Phase): the tasks reworked per developer-hour. tasks per developer-hour.

Adjust_Expect_BW_Prdctvty_Time(Phase): the time for team members to adjust their expected productivity to the current actual productivity of basework development activities. weeks.

alpha(Phase): a variable to keep the pressure values in the exponential functions from growing out of range. dimensionless.

Avg_Act_BW_Prdctvty_Time(Phase): time to smooth instantaneous basework productivity to prevent unrealistic high frequency fluctuations. weeks.

Avg_Act_Coord_Prdctvty_Time(Phase): time to smooth instantaneous coordination productivity to prevent unrealistic high frequency fluctuations. weeks.

Avg_Act_QA_Prdctvty_Time(Phase): time to smooth instantaneous quality assurance productivity to prevent unrealistic low frequency fluctuations. weeks.

Avg_Act_RW_Prdctvty_Time(Phase): time to smooth instantaneous rework productivity to prevent unrealistic low frequency fluctuations. weeks.

Avg_Cost: the average cost of the project to date for forecasting total project cost. dollars per week.

Avg_memb_Exper(Phase): the average experience level of team members. experience units.

Avg_New_member_Exper(Phase): the amount of useful experience which each new team member brings to the Phase. Experience is measured in units similar to tasks, with the performance of each task generating one more unit of experience. experiences.

Avg_Straight_Pay(Phase): the average hourly cost of non-overtime work by a developer without markup for overhead costs. dollars per developer per hour.

Avg_Wrkwk(Phase): the average number of hours worked in a week by the team of developers. hours per week.

Avg_Wrkwk_Change(Phase): the change in the average workweek. hours per week.

Basework(Phase): the initial completion of development tasks in a phase. tasks per week.

Basic_prob_flawed_Task(Phase): the probability that a task will be flawed when performed due to the inherent difficulty of the task. dimensionless.

Budget_Rqrd_Headct_effect: the reduction in the required (target) headcount to reduce project costs in response to project budget overruns. dimensionless.

Budget_Status: the relationship of the project budget and the currently forecasted project costs. dimensionless.

Budget_Surplus: the amount which the budget exceeds or is exceeded by the current cost forecast. dollars.

Budget_Switch: a model control variable engaging the cost and budget feedback loops.

BW_Labor(Phase): the amount of labor applied to basework development activities in a focal phase. person-hours.

BW_Labor_Delay(Phase): the time between the need for basework labor as reflected in the pressure for development activities and when labor allocation shifts in response to that pressure. weeks.

BW_Labor_Limit(Phase): the maximum basework rate allowed by the availability and effectiveness of resources for basework. tasks per week.

BW_Labor_Required(Phase): the number of full time experienced, rested developers required to complete the currently available basework based on the perceived productivity. developers.

BW_Min_Task_duration(Phase): the minimum time required to complete a single task when the task is done the first time, assuming no resource or available-work constraints. weeks.

BW_Prdctvty_Avg_Time(Phase): the smoothing time for perceived basework productivity based on the assumption that developers average instantaneous productivity to predict labor needs. weeks.

BW_Prdctvty_Influences(Phase): a model control variable used to combine the effects of the historical basework productivity and the expected productivity based on recent work. dimensionless.

BW_Prdctvty_Influences_Time(Phase): the smoothing time for expected basework productivity for combination with the effects of historical basework productivity. weeks.

BW_Priority(Phase): the importance of applying available labor to basework development activities in relation to the importance of quality assurance or coordination activities. dimensionless.

BW_Process_Limit(Phase): the maximum rate of completing tasks the first time they are worked on based upon the process. tasks per hour.

BW_Tasks_Remaining(Phase): the number of tasks which have not been worked on for the first time. tasks.

BW_Task_Avail_Gap(Phase): the number of tasks which are available to be completed for the first time but have not been completed for the first time. tasks.

ChangeHdctJumpStore(Phase): the storage or dumping of the amount which the headcount changed due to an exogenous action. developers.

Change_Actual_BW_Prdctvty(Phase): the adjustment of the basework productivity due to the current work. tasks per hour.

Change_Actual_Coord_Prdctvty(Phase): the adjustment of the coordination productivity due to the current work. tasks per hour.

Change_Actual_QA_Prdctvty(Phase): the adjustment of the quality assurance productivity due to the current work. tasks per hour.

Change_Actual_RW_Prdctvty(Phase): the adjustment of the rework productivity due to the current work. tasks per hour.

Change_Avg_BW_Prdctvty(Phase): the adjustment of the average basework productivity due to the current actual basework productivity. tasks per hour.

Change_Avg_Coord_Prdctvty(Phase): the adjustment of the average coordination productivity due to the current actual coordination productivity. tasks per hour.

Change_Avg_QA_Prdctvty(Phase): the adjustment of the average quality control productivity due to the current actual quality control productivity. tasks per hour.

Change_Avg_RW_Prdctvty(Phase): the adjustment of the average rework productivity due to the current actual rework productivity. tasks per hour.

Change_Cum_Exper(Phase): the net increase or decrease in the total experience of the developers working on a phase of the project. experiences.

Change_DL(Phase): the change in the deadline of a phase. weeks.

Change_Expected_QA_Prdctvty_Time(Phase): the time for team members to adjust their expected productivity to the current productivity for quality assurance activities. weeks.

Change_Expect_BW_Prdctvty(Phase): the increase or decrease in the expected basework productivity based on the current reported basework productivity. tasks per hour per developer.

Change_Expect_Coord_Prdctvty(Phase): the increase or decrease in the expected coordination productivity based on the current reported coordination productivity. tasks per hour per developer.

Change_Expect_QA_Prdctvty(Phase): the increase or decrease in the expected quality assurance productivity based on the current reported quality assurance productivity. tasks per hour per developer.

Change_Expect_RW_Prdctvty(Phase): the increase or decrease in the expected rework productivity based on the current reported rework productivity. tasks per hour per developer.

Change_Exp_Compl_Time(Phase): the increase or decrease in the forecasted completion time of a phase. weeks.

Change_in_Headcount(Phase): the moving of full time, rested (no fatigue), experienced persons on or off the development team for a single Phase. developers per week.

Change_in_Release_Hold_Avg(Phase): the difference between the tasks being held for group release and the average tasks being held for group release. This variable measures the sensitivity of the release trigger. tasks.

Change_Quality_Goal(Phase): the movement of the quality goal for the project toward the actual quality experienced in the project. percent defects.

Ch_Expect_Coord_Prdctvty_time(Phase): the time for team members to adjust their expected productivity to the current productivity for coordination activities. Set at 4. weeks.

Clean_Task_error_density(Phase): the percent of tasks which have been released or are being held for release which contain errors generated by the focal development Phase. This is the number of flawed tasks which were not found by the quality assurance process divided by the sum of those tasks and the unflawed tasks. dimensionless.

CloseEnough: a model control parameter to prevent extreme and unrealistic behavior when denominators of division portions of model equations approach zero. dimensionless.

Complete_BW_Tasks(Phase): The rate at which tasks are completed the first time they are worked on (basework). tasks per week.

Complexity(Phase): the inherent difficulty of performing the tasks in the Phase when compared to other Phases. dimensionless.

Compl_Task_error_density(Phase): the percent of tasks which contain flaws and have been completed but not checked for errors. dimensionless.

Concurrence(up,down): the percent of the task list of the Phase which could be completed without resource constraints based upon the percent of the Task List which has been released by an upstream Phase (up). dimensionless.

Coordination_Status(Phase): the measure of the adequacy of coordination efforts in a Phase, defined by the ratio of the required to actual coordination labor. dimensionless.

Coord_effect_on_Find_Up_Errors(Phase): a table function which relates the effect of how well coordination needs are being met on the probability of finding errors inherited from preceding Phases. dimensionless.

Coord_Labor(Phase): the amount of labor applied coordination activities in a focal Phase. person-hours.

Coord_Labor_Delay(Phase): the time between the need for coordination labor as reflected in the pressure for development activities and when labor allocation shifts in response to that pressure. weeks.

Coord_Labor_Limit(Phase): the maximum coordination rate allowed by the availability and effectiveness of resources for coordination. tasks per week.

Coord_Labor_Required(Phase): the number of full time experienced, rested developers required to complete the currently available coordination based on the perceived productivity. developers.

Coord_Limit(Phase): the rate of coordination work as limited by the development process and coordination labor limits. tasks per week.

Coord_Min_duration(Phase): the minimum time required to coordinate efforts due to finding an inherited error or having a downstream Phase find a flaw inherited from the focal Phase, assuming no resource or available-work constraints. Set at 13. weeks.

Coord_Prdctvty_Avg_Time(Phase): the smoothing time for perceived coordination productivity based on the assumption that developers average instantaneous productivity to predict labor needs. weeks.

Coord_Priority(Phase): the importance of applying available labor to coordination development activities in relation to the importance of quality assurance or basework development activities. dimensionless.

Coord_Process_Limit(Phase): the maximum coordination rate allowed by the development process. tasks per week.

Coord_Status(Phase): the measure of the adequacy of coordination efforts by comparing the coordination needed and the coordination provided. dimensionless.

COORD_STATUStest(Phase): a model control parameter which sets the measure of the adequacy of coordination efforts in a Phase. Switches off when resources are included in model. dimensionless.

Coord_effect_on_Prdy(Phase): a table function which relates the adequacy of coordination efforts to the productivity of work in the Phase. dimensionless.

Coord_effect_on_QofP(Phase): a table function which relates the adequacy of coordination efforts to the Quality of Practice. dimensionless.

Correct_Our_Errors(Phase): the rate at which errors which were generated within the focal development Phase are eliminated by reworking tasks. tasks per week.

Corrupted_task_discoveries(Phase): the number of times that a task was found to be flawed due to an inherited error. This variable includes repeated findings of the same flawed task when the flaw was induced by more than one error inherited from more than one upstream development task. tasks.

Cost_effect_on_BW_Import: a table function relating the budget status to the importance of performing basework. Basework priority increases as the budget status worsens. dimensionless.

Cost_Markup(Phase): the multiplier of direct labor costs to reflect overhead and other development costs. dimensionless.

Cumm_ Exper_index(Phase): a measure of the growth in the cumulative experience of the development team in the focal Phase, as measured by the ratio of the cumulative experience to the reference experience. this variable is used to find the improvement in productivity due to increased experience. This formulation is based upon the learning curve concept. The learning curve effect is modeled by increasing the current productivity 20% for every doubling of cumulative experience of the product development team in the focal Phase. The model takes current experience, experience lost, and experience gained as input and tracks cumulative experience over the lifetime of the simulation.

Cumm_Cost(Phase): the sum of the straight-time and over-time development costs in a phase. dollars.

Cumm_Exper(Phase): the total accumulated experience of the developers in a phase. experiences.

Current_BW_Prdctvty(Phase): the instantaneous basework productivity in a phase. tasks per week per developer.

Current_Coord_Prdctvty(Phase): the instantaneous coordination productivity in a phase. tasks per week per developer.

Coord_Backlog(Phase): the accumulated demand for coordination work. tasks.

Current_Coord_added(Phase): the current demand for coordination defined as the number of tasks found to be corrupted by inherited errors plus the number of flawed and released tasks which are returned after being discovered by downstream phases. tasks.

Current_Known_Quality(Phase): the percent of total tasks completed initially which are believed to not be flawed. percent.

Current_Project_Rework_Percent: the percent of total tasks completed initially which are known to be flawed. percent.

Current_QA_Prdctvty(Phase): the instantaneous quality assurance productivity in a phase. tasks per week per developer.

Current_RW_Prdctvty(Phase): the instantaneous rework productivity in a phase. tasks per week per developer.

Cycle_Time(Phase): the time between the start of the Phase and the completion of the Phase. weeks.

Cycle_Time_Change(Phase): the indicator that a Phase's cycle time is still accruing, i.e. that the Phase has started but has not yet been completed. weeks.

DeadlineFlag(Phase): an indicator that a the project has passed the deadline of a Phase. dimensionless.

Deadline_Switch: a model control parameter which engages the schedule feedback loops. dimensionless.

Dep(up,down): A dummy variable used to declare the variable Dependency(up,down). dimensionless.

Dependency(up,down): The level of dependency between each pair of Phases. A value of 1 means the "down" Phase depends on the "up" Phase, i.e. Up feeds down. A value of 0 means the "down" Phase does not depend on the "up" Phase. The network is arranged (i.e. numbering of the Phases) so that smaller numbered Phases feed higher numbered Phases. Therefore the diagonal of the matrix = 0 to prevent "death grip" self-dependencies, the upper right half of the matrix = 0 to prevent "death grip" interlocking, and the lower left half of matrix describes dependency network. dimensionless

Disc_Our_Errors(Phase): the rate at which quality assurance efforts find tasks with errors which were generated within the focal development Phase. tasks per week.

DL(Phase): the deadline of a phase as set by the project deadline, the critical path from the phase to the project deadline, and the duration of that critical path. weeks from start.

Empty_Release_Hold_Avg(Phase): the emptying of the averaging stock when the held tasks are released as a group. tasks per week.

Errors_lost_in_Corrupted_Tasks(Phase): the rate at which errors generated within the focal development Phase become "lost" because the task which was flawed must be reworked because of an error inherited from an upstream Phase. tasks per week.

Errors_to_Release_Hold(Phase): the flow of tasks with errors which are inspected but are being held with a collection of tasks for release as a group. tasks.

Expected_Duration(Phase): the predicted time between the start and completion of a Phase based upon the number of tasks remaining, the headcount, and the productivity. weeks.

Expected_Proj_Completion_Time: the forecasted completion time is the current time plus the longest duration estimate through all possible critical paths from active phases to the completion of the last phase. weeks from start.

Expected_QA_Prdctvty(Phase): the quality assurance productivity which developers expect to be experienced based on the reported quality assurance productivity. tasks per hour per developer.

Expect_BW_Prdctvty(Phase): the basework productivity which developers expect to be experienced based on the reported basework productivity. tasks per hour per developer.

Expect_Coord_Prdctvty(Phase): the coordination productivity which developers expect to be experienced based on the reported coordination productivity. tasks per hour per developer.

Expect_RW_Prdctvty(Phase): the rework productivity which developers expect to be experienced based on the reported rework productivity. tasks per hour per developer.

Exper_Assim_Time(Phase): the time required to assimilate experience and make it useful to improve productivity. This variable can represent the speed of disseminating new knowledge or learning within the Phase's development team. weeks.

Exper_effect_on_QofP(Phase): a table function describing the impact of the experience of the developers on the quality of their practice. dimensionless.

Exper_index(Phase): the improvement in productivity factor due to increased experience. This formulation is based upon the learning curve concept. The learning curve effect is modeled by increasing

the current productivity 20% for every doubling of cumulative experience of the product development team in the focal Phase. The model takes current experience, experience lost, and experience gained as input and tracks cumulative experience over the lifetime of the simulation.

Exper_Lost(Phase): the experience lost to the phase due to the departure of developers. experiences.

Exper_on_Prdctvty_effect(Phase): a table function describing the impact of the experience of the developers on their productivity. dimensionless.

ExpStart123: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart1234: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart12345: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart1235: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart124: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart1245: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart125: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart13: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart134: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart1345: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart135: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart14: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart145: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart15: the expected start date of phase 1 based on the project deadline and the durations through a critical path. weeks.

ExpStart23: the expected start date of phase 2 based on the project deadline and the durations through a critical path. weeks.

ExpStart24: the expected start date of phase 2 based on the project deadline and the durations through a critical path. weeks.

ExpStart25: the expected start date of phase 2 based on the project deadline and the durations through a critical path. weeks.

ExpStart3: the expected start date of phase 3 based on the project deadline and the durations through a critical path. weeks.

ExpStart34: the expected start date of phase 3 based on the project deadline and the durations through a critical path. weeks.

ExpStart35: the expected start date of phase 3 based on the project deadline and the durations through a critical path. weeks.

ExpStart4: the expected start date of phase 4 based on the project deadline and the durations through a critical path. weeks.

ExpStart45: the expected start date of phase 4 based on the project deadline and the durations through a critical path. weeks.

ExpStart5: the expected start date of phase 5 based on the project deadline and the durations through a critical path. weeks.

Exp_Compl_Time(Phase): the time that a Phase is predicted to complete its work based upon the time the current time, the tasks remaining, the headcount, and productivity. weeks from start of project.

Exp_Start_Time(1): the expected start time of phase 1 based on the critical path duration and the project dependency network. weeks.

Exp_Start_Time(2): the expected start time of phase 2 based on the critical path duration and the project dependency network. weeks.

Exp_Start_Time(3): the expected start time of phase 3 based on the critical path duration and the project dependency network. weeks.

Exp_Start_Time(4): the expected start time of phase 4 based on the critical path duration and the project dependency network. weeks.

Exp_Start_Time(5): the expected start time of phase 5 based on the critical path duration and the project dependency network. weeks.

Fatigue_Qual_of_Prac_effect(Phase): a table function which relates the level of Fatigue to the Quality of Practice. 1/hours per week per person.

Find_Any_Errors_Flag(Phase): an indicator if any internally generated errors were found in a Phase. weeks that an error was found.

Find_Any_Errors_Flow(Phase): the rate for the indicator showing if any internally generated errors were found in a Phase. dimensionless.

Flawed_and_Found_Error_density(Phase): the percent of tasks which contain flaws and have been found to contain errors. The quality assurance effort is assumed to never consider an unflawed task to be flawed. Therefore this variable is always assumed to be 1.00. dimensionless.

Forecasted_Costs: total forcasted costs of all phases. dollars.

Fraction_Avail_due_to_Ext_gate(up, Phase): the minimum percent of the focal development Phase's task list which could be completed without resource, rework, or speed of processing constraints based upon the percentages of the Task Lists of all upstream development Phases upon which the focal Phase depends and which has been released by those upstream Phases (up).

Fraction_Avail_due_to_Int_gate(Phase): the percent of the task list which could be completed without resource, rework, or speed of processing constraints based upon the percent of the Task List which is completed and released. dimensionless.

Fraction_Released(Phase): the percent of tasks which a development Phase has released to downstream Phases. dimensionless.

Fract_Compl_and_Rel(Phase): the percent of tasks which have been completed and not yet checked for errors or have been completed, checked for errors, and released to downstream development Phases. dimensionless.

Frac_Labor_in_BW(Phase): the percent of basework labor used on basework in the Phase, based upon the relative amount of rework and basework waiting (available) to be done. dimensionless.

Generate_Errors(Phase): the rate at which tasks are completed which contain errors. tasks per week.

Gross_Labor(Phase): the total labor available to a Phase, based upon the current headcount and workweek. person-hours.

HdctJump(Phase): size of instantaneous headcount change due to an exogenous force. developers.

HdctJumpBack(Phase): size of return to initial headcount prior to instantaneous jump in headcount. developers.

HdctJumpNotStoppedFlag(Phase): a model control parameter to indicate when the headcount jump is active. dimensionless.

HdctJumpStartFlag(Phase): a model control parameter to indicate when the headcount jump has started. dimensionless.

HdctJumpStopTime(Phase): the time the headcount returns to the initial value. weeks.

HdctJumpStore(Phase): size of headcount jump. developers.

HdctJumpSwitch(Phase): a model control parameter to engage the headcount jump feature. dimensionless.

Headcount(Phase): the number of full time developers active in the focal development Phase. persons.

Headcount_Adjustment_Time(Phase): the average time required to move a person onto or off of the development team in a focal Phase. weeks.

Historical_BW_Prdctvty_Belief(Phase): the historical belief of developers concerning their productivity in performing basework. tasks per week per developer.

Hold_Errors_for_Release(Phase): the number of flawed tasks which have been completed and checked for errors and are being accumulated for release as a group. tasks.

Hold_for_Release(Phase): the number of tasks which have been completed and checked for errors and are being accumulated for release as a group. tasks.

Inital_Headcout(Phase): headcount at start of Phase. Set at 1.00. persons.

Initial_Proj_Deadline: the project deadline at the beginning of the project. weeks.

Initial_Quality_Goal(Phase): desired percent flawless tasks passed downstream by each Phase. Set at 1.00 (no errors passed). dimensionless.

Known_Rework(Phase): the number of tasks in a development Phase which are known to be flawed and require rework. tasks.

Labor_Fraction_to_BW(Phase): the percent of the labor at any time which is used for basework. dimensionless.

Labor_Fraction_to_Coord(Phase): the percent of the labor at any time which is used for coordination. dimensionless.

Labor_Fraction_to_QA(Phase): the percent of the labor at any time which is used for quality assurance. dimensionless.

Labor_Fraction_to_RW(Phase): the percent of the labor at any time which is used for rework. dimensionless.

LastPhase: identifier of the final Phase in the Phase Dependency Network. dimensionless.

Max_DL_Change(Phase): a model control parameter to prevent extreme and unrealistic schedule pressures as the phase approaches its deadline. weeks.

Max_Duration(1): the duration of the critical path from the last phase to phase 1. weeks.

Max_Duration(2): the duration of the critical path from the last phase to phase 2. weeks.

Max_Duration(3): the duration of the critical path from the last phase to phase 3. weeks.

Max_Duration(4): the duration of the critical path from the last phase to phase 4. weeks.

Max_Duration(5): the duration of the critical path from the last phase to phase 5. weeks.

Max_Headcount(Phase): the most full time developers possible on a single Phase. persons.

Max_Proj_DL_Change: a model control parameter to prevent extreme and unrealistic schedule pressures as the project approaches its deadline. weeks.

Max_Workweek(Phase): Highest possible hours per week each person can spend on project. Set at 140. hours per week per person.

Min_Exp_BW_Prdy(Phase): minimum expected productivity of basework development activities. Primarily used to prevent division by zero or unrealistically high required headcount due to division by a very small number. tasks per person per hour.

Min_Exp_Coord_Prdy(Phase): minimum expected productivity of coordination activities. Primarily used to prevent division by zero or unrealistically high required headcount due to division by a very small number. tasks per person per hour.

Min_Exp_QA_Prdy(Phase): minimum expected productivity of quality assurance activities. Primarily used to prevent division by zero or unrealistically high required headcount due to division by a very small number. tasks per person per hour.

Min_Exp_RW_Prdy(Phase): minimum expected productivity of rework activities. Primarily used to prevent division by zero or unrealistically high required headcount due to division by a very small number. tasks per person per hour.

Min_Headcount(Phase): the fewest number of full time developers possible on a single Phase. persons.

Multiple_Corruption_discoveries(Phase): the number of corrupted tasks having been flawed due to more than one inherited upstream error. This variable is used to correct the number of corrupted tasks for multiple findings of the same corrupted task. dimensionless.

Net_Corrupted_and_Found_Tasks(Phase): the corrected (for multiple findings of flawed tasks) number of tasks found to be corrupted by upstream inherited errors. tasks.

Net_Exper_Gain(Phase): the sum of the experience gained by performing basework and adding new developers to the phase. experiences.

New_Memb_Exper_Gain(Phase): the increase in experience gained by adding new developers to the phase. experiences.

Normal_Workweek(Phase): reference workweek length. Set at 40. hours per week per person.

NotStoppedFlag(Phase): an indicator that a Phase has not been completed. dimensionless.

Not_Done(Phase): a model control parameter to indicate when a phase is active. dimensionless.

OT_Cost(Phase): the overtime costs of a phase. dollars.

Our_Discd_Errors(Phase): the number of tasks which are known to be flawed. tasks.

Our_Discd_Error_density(Phase): the percent of tasks known to require rework which contain errors generated by the focal development Phase. dimensionless.

Our_Errors_Released(Phase): the number of tasks with errors generated within the focal development Phase which have been released to downstream development Phases. tasks.

Our_Undiscd_Errors(Phase): the number of tasks with errors generated within the focal development Phase which have not been checked for errors. tasks.

Overtime_Premium(Phase): the percent added to straight time hourly cost to estimate overtime labor cost. Set at 50%. dimensionless.

Over_Time(Phase): the number of hours spent over 40 hours per week per developer in a phase. hours.

Path1: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path12: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path123: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path1234: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path12345: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path1235: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path124: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path1245: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path125: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path13: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path134: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path1345: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path135: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path14: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path145: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path15: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path2: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path23: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path234: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path2345: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path235: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path24: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path245: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path25: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path3: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path34: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path345: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path35: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path4: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path45: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Path5: a switch indicating whether a path through the project network is active. Set to 0 if not used and 1 if used, this switch is used to nullify any inactive paths in schedule and deadline calculations. dimensionless.

Percent_hourly_Labor(Phase): the portion of the total labor hours which are subject to overtime pay. dimensionless.

percent_Multiple_Corruption_discoveries(Phase): the probability of a corrupted task having been flawed due to more than one inherited upstream error. This variable is used to correct the number of corrupted tasks for multiple findings of the same corrupted task. dimensionless.

vthe project quality goal described in terms of the amount of rework. dimensionless.

Percent_Tasks_in_Rework_Que(Phase): the percent of tasks in a Phase which are known to require rework. This variable is used to find the perceived quality for comparison with the quality goal.

PhaseNo(1): a model control parameter to identify a phase of the project. dimensionless.

PhaseNo(2): a model control parameter to identify a phase of the project. dimensionless.

PhaseNo(3): a model control parameter to identify a phase of the project. dimensionless.

PhaseNo(4): a model control parameter to identify a phase of the project. dimensionless.

PhaseNo(5): a model control parameter to identify a phase of the project. dimensionless.

Phase_Cost_to_Date(Phase): the accumulation of all costs incurred by a phase up to the current simulation time. This value is used with the forecast of costs to estimate total costs of the phase. dollars.

Phase_Deadline(Phase): the date by which the Phase must be completed (all tasks correct and released) to have the project completed by the current Project Deadline. This time is found by subtracting the longest duration (determined by the paths of Phase dependencies from the Phase to the project completion) from the project deadline. weeks from start.

Press_for_BW(Phase): the pressure on developers to allocate available labor to the basework activity. dimensionless.

Press_for_Coord(Phase): the pressure on developers to allocate available labor to the coordination activity. dimensionless.

Press_for_QA(Phase): the pressure on developers to allocate available labor to the quality assurance activity. dimensionless.

Press_for_RW(Phase): the pressure on developers to allocate available labor to the rework activity. dimensionless.

prob_err_gen_fr_Task_Complexity(Phase)= a table function which relates the complexity of the task to the probability of generating an error in a task being performed. dimensionless.

Prob_finding_our_error_if_exists(Phase): a table function which relates levels of quality assurance, quality of practice, and complexity to the probability of finding internally generated errors. dimensionless.

prob_find_Up_error__if_exists(Phase): the probability of finding an error in a focal Phase task which was caused by an error inherited from an upstream development Phase if the error exists in the task. dimensionless.

prob_of_err_gen_by_QofP(Phase): a table function which relates how well the quality of practice needs are being met to the probability of generating an error in a task being performed. dimensionless.

prob_of_no_err_gen_from_effects(Phase): the chance that a task is flawed due to the effects of Task Complexity and Quality of Practice. dimensionless.

prob_Task_Corrupted_and_found(up,Phase): the probability that a task contains an error due to an inherited error and that the induced error was found by the quality assurance effort. dimensionless.

prob_Task_flawed(Phase): the chance that a task is flawed due the effects of Task Complexity, Quality of Practice, and the basic difficulty of performing the task. dimensionless.

prob_Task_flawed_and_Found(Phase): the chances that a task both contains an error and the quality assurance process found the error. dimensionless.

Project_Cost_to_Date: the sum of the costs of all active phases incurred to the current simulation time. dollars.

Project_Deadline: the current planned completion date of all Phases. weeks from start.

Project_Errors_Released: the sum of the flawed tasks released by all active phases. tasks.

Project_Quality_Goal: the percent of tasks which the developers want to be released without flaws. dimensionless.

Proj_Budget: the cost allocated for the project. This is compared with forcasted costs to establish the budget status. dollars.

Proj_Quality_Gap: the difference between the objective and actual rework percentages. dimensionless.

Proj_Sched_Press: the ratio of the time estimated to be required to complete the project to the time available to complete the project. dimensionless.

Proj_Sched_Slip: the change in the project deadline in response to project schedule pressure. weeks.

Proj_Tasks_Compl_and_Rel: a record keeping parameter which is the sum of the Completed, not Checked and Tasks Released stocks for all phases. tasks.

QA_for_Find_Error(Phase): the measure of the adequacy of quality assurance efforts in a Phase, defined by the ratio of the required to actual quality assurance labor. dimensionless.

QA_inspection_rate(Phase): the rate at which completed tasks are inspected for errors. tasks per week.

QA_Labor(Phase): the amount of labor applied to quality assurance activities in a focal Phase. person-hours.

QA_Labor_Delay(Phase): the time between the need for quality assurance labor as reflected in the pressure for development activities and when labor allocation shifts in response to that pressure. weeks.

QA_Labor_Limit(Phase): the maximum quality assurance rate allowed by the availability and effectiveness of resources for quality assurance. tasks per week.

QA_Labor_Required(Phase): the number of full time experienced, rested developers required to complete the currently available quality assurance work based on the perceived productivity. developers.

QA_Min_Task_Duration(Phase): the minimum time required to check a single task for errors, assuming no resource or available-work constraints. weeks.

QA_Prdctvty_Avg_Time(Phase): the smoothing time for perceived quality assurance productivity based on the assumption that developers average instantaneous productivity to predict labor needs. weeks.

QA_Prdctvty_Report_Time(Phase): the delay in reporting the actual productivity of quality assurance activities. weeks.

QA_Priority(Phase): the importance of applying available labor to quality assurance activities in relation to the importance of basework, rework, or coordination activities. dimensionless.

QA_Process_Limit(Phase): the maximum rate of checking tasks for errors possible based upon the quality assurance process. tasks per hour.

QA_Prdctvty_Report_Time(Phase): the time delay between actual and use of reported productivity. This variable is fairly long since formal means of productivity reporting are not used and team members effectively report productivity through experiencing the progress and effort of the project. weeks.

QA_STATUS: the measure of the adequacy of quality assurance efforts in a Phase, defined by the ratio of the required to actual quality assurance labor. dimensionless.

QofP_Error_Disc_effect(Phase): a table function which relates the quality of practice to the probability **of finding an internally generated error. dimensionless.**

Quality_Goal(Phase): the percent of tasks without defects which is the objective of a phase. percent defects.

Quality_Goal_Adjust_Time(Phase): the time over which the development team of a Phase allows its quality goal to move to the current quality performance. This is rather long, reflecting the strong history and current priority on producing error-free products to customers and downstream Phases. weeks.

Quality_Goal_Switch: a model control parameter which engages the quality feedback loops. dimensionless.

Quality_of_Practice(Phase): the relative performance of the work of developers without error, as influenced by project conditions such as schedule pressure and fatigue. dimensionless

Qual_Gap_effect_on_Coord_Import: a table function which describes the impact of the project's quality gap on the priority of doing coordination. Coordination priority generally increases with increasing quality gap. dimensionless.

Qual_Gap_effect_on_QARW_priority: a table function which describes the impact of the project's quality gap on the priority of doing quality assurance and rework. These priorities generally increase with increasing quality gap. dimensionless.

Receive_Our_Errors_fr_Dn(Phase): the rate at which tasks which are flawed are returned to the focal development Phase for rework from downstream development Phases. tasks per week.

Ref_BW_Prdctvty(Phase): benchmark productivity of labor at basework development activities. tasks per person per hour.

Ref_BW_Tasks_per_Hour(Phase): benchmark productivity of process at basework development activities. tasks per person per hour.

Ref_Complexity(Phase): benchmark complexity of project for comparison of different projects. Impacts error generation and discovery. dimensionless.

Ref_Coord_Prdctvty(Phase): benchmark productivity of labor at coordination activities. tasks per person per hour.

Ref_Exper(Phase): benchmark amount of experience for finding effect of cumulative experience on productivity. Experience is measured in units similar to tasks, with the performance of each task generating one more unit of experience. experiences.

Ref_QA_Prdctvty(Phase): benchmark productivity of labor at quality assurance activities. tasks per person per hour.

Ref_Qual_of_Practice(Phase): a benchmark quality of practice for Phase work. dimensionless.

Ref_RW_Prdctvty(Phase): benchmark productivity of labor at quality assurance activities. tasks per person per hour.

Release_Any_Errors_Flag(Phase): an indicator if any internally generated errors were released from a Phase. weeks that an error was released.

Release_Any_Errors_Flow(Phase): the rate for the indicator which shows if any internally generated errors were released from a Phase. dimensionless.

Release_Any_Tasks_Flag(Phase): an indicator if any tasks were released from a Phase. weeks that a task was released.

Release_Any_Tasks_Flow(Phase): the rate for the indicator if any tasks were released from a Phase. dimensionless.

Release_Errors(Phase): the rate at which errors generated within the focal development Phase are released to downstream Phases. tasks per week.

Release_Errors_from_Hold(Phase): the rate at which errors generated within a phase are released to downstream as a group. tasks per week.

Release_Hold_Avg(Phase): the rolling average of the number of tasks being held for group release. tasks.

Release_Hold_Avg_Stability: a measure of the change in the holding stock before release as a group. This is used to simulate the developers waiting until task completion rates become small (stable average), indicting that they have fixed errors. dimensionless.

Release_Hold_Avg_Time(Phase): the time over which the tasks being held for group release are averaged. weeks

Release_Tasks(Phase): The rate at which tasks which have been checked for errors are released by the focal development Phase to downstream development Phases. tasks per week.

Release_Tasks_from_Hold(Phase): the rate of group release of tasks which have been accumulated, based upon the stable average of the number of tasks waiting for release. tasks per week.

Release_Trigger(Phase): the indicator of the adequate accumulation of tasks for group release, based upon the stable average of the number of tasks waiting for release. dimensionless.

Release_Trigger_Sensativity(Phase): the level of stability required before the held tasks are released. dimensionless.

Release_Trigger_Task_Gate(Phase): the release trigger switch. dimensionless.

Rel_Task__error_density(Phase): the percent of tasks released to downstream Phases which contain errors. dimensionless.

Report_BW_Prdctvty_Time(Phase): the time delay between actual and use of reported productivity. This variable is fairly long since formal means of productivity reporting are not used and team members effectively report productivity through experiencing the progress and effort of the project. weeks.

Report_Coord_Prdctvty_time(Phase): the time delay between actual and use of reported productivity. This variable is fairly long since formal means of productivity reporting are not used and team members effectively report productivity through experiencing the progress and effort of the project. weeks.

Required_Headcount(Phase): the number of persons required to complete the Phase by the Phase deadline based upon the estimated hours required to finish and current headcount and normal workweek. persons.

Required_Personweeks(Phase): the number of labor hours required to complete the phase. developers.

Resistance_to_Sched_Slip: the ratio of required to available time at which the project deadline is slipped. dimensionless.

ResourseSwitch(Phase): switch enabling and disabling the labor portions of the model. dimensionless.

Rework_Any_Flag(Phase): an indicator if any tasks were reworked in a Phase. weeks that a task was reworked.

Rework_Any_Flow(Phase): the rate for the indicator if any tasks were reworked in a Phase. dimensionless.

Rework(Phase): The rate at which tasks which are known to require rework are corrected. New errors may be generated in these tasks while they are being reworked. tasks per week.

RW_due_to_Corrupted_tasks(Phase): tasks known to require rework which were flawed by an upstream development Phase and in which the flaws were discovered by quality assurance efforts within the focal development Phase. tasks per week.

RW_due_to_InPhase_QA(Phase): tasks known to require rework which were flawed by the focal development Phase and in which the flaws were discovered by quality assurance efforts within the focal development Phase. tasks per week.

RW_due_to__Dwnstrm_QA(Phase): tasks known to require rework which were flawed by the focal development Phase and in which the flaws were discovered by quality assurance efforts within a downstream development Phase. tasks per week.

RW_Labor(Phase): the time between the need for rework labor as reflected in the pressure for development activities and when labor allocation shifts in response to that pressure. weeks.

RW_Labor_Delay(Phase): the time between the need for rework labor as reflected in the pressure for development activities and when labor allocation shifts in response to that pressure. weeks.

RW_Labor_Limit(Phase): the maximum rework rate allowed by the availability and effectiveness of resources for rework. tasks per week.

RW_Labor_Required(Phase): the number of full time experienced, rested developers required to complete the currently available rework based on the perceived productivity. developers.

RW_Min_Task_Duration(Phase): the minimum time required to complete a single task when the task is done subsequent to the first time, assuming no resource or available-work constraints. weeks.

RW_Prdctvty_Avg_Time(Phase): the smoothing time for perceived rework productivity based on the assumption that developers average instantaneous productivity to predict labor needs. weeks.

RW_Priority(Phase): the importance of applying available labor to rework in relation to the importance of basework, quality assurance or coordination activities. dimensionless.

RW_Process_Limit(Phase): the maximum rate of reworking tasks which are known to be flawed possible based upon the rework process. tasks per hour.

Sched_BW_Press_effect(Phase): a table function which relates the amount of schedule pressure to the priority given to spending available time on basework development activities. dimensionless.

Sched_Error_Disc_effect(Phase): a table function which relates the Schedule Pressure to the probability of discovering an error if the error exists. dimensionless.

Sched_Pressure(Phase): The ratio of the time required to complete a Phase to the time to the Phase deadline. If schedule pressure is < 1.00 (e.g.. 0), Quality of Practice drops and not all errors will be caught. Set schedule pressure to 1.00 for "no schedule pressure" testing. dimensionless.

Sched_Qual_of_Prac_effect(Phase): a table function which relates Schedule Pressure to the Quality of Practice. dimensionless.

Sched_Workweek_effect(Phase): a table function which relates the level of schedule pressure to the number of hours worked per week. dimensionless.

StartFlag(Phase): a indicator that a Phase has begun. dimensionless.

Start_Time(Phase): time Phase starts, defined as when basework begins. weeks from start of project.

StoppedFlag(Phase): a model control parameter to indicate when a phase has ended. dimensionless.

Straight_Cost(Phase): the cost incurred due to non overtime work. dollars.

Straight_Time(Phase): the hours worked within the limit of 40 hours per week per developer. hours.

Sum_of_Tasks(Phase): the total number of tasks that have been completed at least once in a Phase, i.e. the completed, known rework, released, and hold for release tasks. tasks.

TaskListScale(up,down): the scaling factor which relates the relative sizes of two Phases. dimensionless

Tasks_Completed(Phase): the number of tasks in a development Phase which have been initially finished but have not been checked for errors. tasks.

Tasks_Compl_and_Holding(Phase): the number of tasks which have been completed and not yet checked for errors or have been completed, checked for errors, and are being held for group release to downstream development Phases. tasks.

Tasks_Compl_and_Rel(Phase): the number of tasks which have been completed and not yet checked for errors or have been completed, checked for errors, and released to downstream development Phases. tasks.

Tasks_Corrupted_and_Found(up,Phase): the number of tasks that contain an error due to an inherited error and the induced error was found by the quality assurance effort. tasks.

Tasks_Released(Phase): the number of tasks in a development Phase which have been completed, checked for errors, and released to downstream development Phases. tasks.

Tasks_to_Release_Hold(Phase): the rate of tasks checked and believed to have no errors to accumulate for release as a group. tasks per week.

Task_Complex_effect_on_Err_disc(Phase)=a table function which relates the complexity of the task to the probability of finding an error generated by the focal development Phase. dimensionless.

Task_List(Phase): the number of "tasks" or pieces of atomic work which must be completed correctly for a Phase to be finished. A task is small enough that it is completely correct or completely incorrect. number of tasks.

Test_Input_1(Phase): the switch enabling and disabling errors and rework in the focal Phase. dimensionless.

Test_Input_2(Phase): the switch which selects release direct from completed tasks or holding of completed tasks for clustered release of tasks. This is used to model the aggregation of design products and their release together to make masks. dimensionless.

Test_Input_3(Phase): a model control parameter which is not used. dimensionless.

Time_Required(Phase): the time needed to complete a Phase, based upon the required person weeks and current headcount. weeks.

Time_spent_to_Date(Phase): the number of weeks between the current time (or the Phase deadline if Phase is completed) and the week when the Phase began. weeks.

Time_to_Avg_Exp_Compl_Time(Phase): a short smoothing parameter to prevent extreme fluctuations in expected completion times and thereby extreme movement of the project deadline. weeks.

Time_to_Deadline(Phase): the number of weeks from the current time to the Phase Deadline. weeks.

Time__Phase_Active(Phase): the flag signaling that a Phase in active (begun but not completed). dimensionless.

Total_Err_disc_by_Dn(Phase): the sum of errors passed to downstream Phases by the focal Phase which are returned to the focal Phase for rework. tasks.

Total_Hdct: the sum of the active headcount of all phases. developers.

Total_Labor_Required(Phase): the sum of the labor required by all four development activities. developer hours.

Total_Pressure_for_Activites(Phase): the sum of the pressures on developers to allocate labor to the four development activities. dimensionless.

Tot_Exp_Costs: the sum of the total expected cost of all phases in the project. dollars.

Tot_Tasks_Avail(Phase): the number of tasks which are available to be completed the first time (basework). tasks.

Up_Flawed_Tasks_found(up,Phase): the number of tasks in each upstream Phase which were responsible for the errors in the focal Phase which were caused by inherited errors. This variable is used to pass error information back to upstream Phases. tasks.

Wait_Time(Phase): time before Phase starts. weeks

Workweek(Phase): the average number of hours worked in each week by each developer in a phase. hours per week.

Wrkwk_Avg_Time(Phase): the time over which the average workweek is calculated. This variable is used to find fatigue levels. It represents the time it takes for extended overtime work to impact workers. Set at 4. weeks.

Wt_to_Reported_BW_Prdctvty(Phase): the percent of influence given to the reported productivity of basework development activities. This variable is balanced by the historical productivity assumptions held by team members. Set at 50%. dimensionless.

Appendix 4.1

Signal Processing Model

```
//  SIGNAL PROCESSING MODEL

//  x is the vector of state variables
//  u is the input signal (a function of t)
//  X1 = COLUMN 2 IS THE STOCK OF COMPLETED, NOT CHECKED TASKS
//  X2 = COLUMN 3 IS THE STOCK OF KNOWN REWORK
//  X3 = COLUMN 4 IS THE STOCK OF TASKS RELEASED
tstart=1;                                   //  Start simulation time
tend=100;                                   //  End simulation time
tincrement=0.1;                     // Time increment (dt)

//  FUNCTION CALCULATING CHANGE IN OUTPUT = XDOT
xdot=function(t,x)
{
//  Declare variables
global (Frequency);
global (AmpIn);
local (Output);
global (Tau1);
global (Tau2);
global (IFR);
global (Par);
Output=zeros(3,1);
Freq=Frequency;
```

```
// INPUT SIGNAL
AmpIn=3;                                    // Amplitude of Input Signal
theta=0;                                    // Phase shift of frequency input signal
Beta=(t*Freq*2*3.1415962) + theta;
u=AmpIn+ AmpIn*sin(Beta);                   //SINE WAVE INPUT

Tau1=Par;                                   // Inspection Time Constant, to be vaired for 3d plot
Tau2=8;
IFR=0.25;                                   // Inspection Failure Fraction
A=[-1/Tau1, 1/Tau2 ,0; IFR/Tau1, -1/Tau2, 0; (1-IFR)/Tau1, 0, 0];
B=[1;0;0];
Output=A*x + B*u;
return Output;
};

// LOOP SIMULATION THROUGH PARAMETER VALUES
MinPar=1;                                   // Starting parameter value
MaxPar=10;                                  // Ending parameter value
FrequencyIterations=21;                     // Number of paratemeter values simulated
Data=zeros(FrequencyIterations,5);
MaxFrequency=10;
Y=zeros(MaxPar-MinPar+1,1);
X=zeros(FrequencyIterations,1);
Z=zeros(size(X)[1],size(Y)[1]);

for(i in MinPar:MaxPar)                     // Set up output matrix
{
Y[i]=i;
printf("Par=%2.0f\t",i);
Par=i;

for(f in 1:FrequencyIterations)
{
printf("i = %2.2f\t",i);
Frequency=(f/FrequencyIterations)*MaxFrequency;
X[f]=Frequency;
printf("Frequency= %4.2f\t",Frequency);
ystart=[0,0,0];
y=ode(xdot, tstart, tend, ystart, tincrement, , );      // integration function call

LargerY=zeros(size(y)[1],5);
LargerY[;1]=y[;1];
```

```
LargerY[;2]=y[;2]*(1-IFR)*(1/Tau1);       //ReleaseRate vector
LargerY[;3]=y[;2]*(IFR/Tau1);               //  REWORK DUE TO QA FLOW
LargerY[;4]=y[;3]/Tau2;                     //  REWORK FLOW
LargerY[;5]=LargerY[;3]-LargerY[;4];        //  NetKnownReworkFlow
```

// OPTIONAL CALUCULATION OF SYSTEM GAIN FOR EACH SIMULATION

```
MaxReleaseRate=max(LargerY[;2]);
BeginSteadyState=70;                        //End of Transient response
MinReleaseRate=min(LargerY[BeginSteadyState,size(LargerY)[1];2]);
AmpOut=MaxReleaseRate-MinReleaseRate;
Gain=AmpOut/AmpIn;
printf("Gain = %3.2f\n",Gain);

Data[f;1]=i;
Data[f;2]=Frequency;
Data[f;3]=Gain;
//  printf("\tInto Data");
}
//  ================
for (no in 1:size(X)[1])
{Z[no;i]=Data[no;3];}
}
```

// PLOT OUTPUT

// 3D PLOT OF SYSTEM GAIN

```
pstart(1,1,"Mac");
ptitle("System Gain ");
xlabel ("Frequency");
ylabel ("T1");
zlabel ("System Gain");
plmesh(<<x=X;y=Y ;z=Z>>);

pstart(1,1,"lj_hpgl");
ptitle("System Gain");
xlabel ("Frequency");
ylabel ("T1");
zlabel ("System Gain");
plmesh(<<x=X;y=Y ;z=Z>>);
```

```
// PLOT SYSTEM GAIN ON SCREEN
pstart(1,1,"Mac");
ptitle("Frequency versus System Gain");
xlabel ("Frequency");
ylabel ("Gain");
plot([Data[;2],Data[;3]]);
// PLOT SYSTEM GAIN ON PRINTER
pstart(1,1,"lj_hpgl");
ptitle("Frequency versus System Gain");
xlabel ("Frequency");
ylabel ("Gain");
plot([Data[;2],Data[;3]]);


// PLOT EXAMPLE SIMULATION ON SCREEN
pstart(1,1,"Mac");
ptitle("Highest Frequency Task Release Rate");
xlabel ("Time");
ylabel ("Development Tasks per Time Unit");
plot([LargerY[;1],LargerY[;2]]);
// PLOT EXAMPLE SIMULATION ON PRINTER
pstart(1,1,"lj_hpgl");
ptitle("Highest Frequency Task Release Rate");
xlabel ("Time");
ylabel ("Development Tasks per Time Unit");
plot([LargerY[;1],LargerY[;2]]);

// PLOT EXAMPLE SIMULATION ON SCREEN
pstart(1,1,"Mac");
ptitle("Highest Frequency Rework due to QA");
xlabel ("Time");
ylabel ("Development Tasks per Time Unit");
plot([LargerY[;1],LargerY[;3]]);
// PLOT EXAMPLE SIMULATION ON PRINTER
pstart(1,1,"lj_hpgl");
ptitle("High Frequency Rework due to QA");
xlabel ("Time");
ylabel ("Development Tasks per Time Unit");
plot([LargerY[;1],LargerY[;3]]);
```

```
// PLOT EXAMPLE SIMULATION ON SCREEN
pstart(1,1,"Mac");
ptitle("Highest Frequency Rework Flow");
xlabel ("Time");
ylabel ("Development Tasks per Time Unit");
plot([LargerY[;1],LargerY[;4]]);
// PLOT EXAMPLE SIMULATION ON PRINTER
pstart(1,1,"lj_hpgl");
ptitle("High Frequency Rework Flow");
xlabel ("Time");
ylabel ("Development Tasks per Time Unit");
plot([LargerY[;1],LargerY[;4]]);
```

Appendix 5.1

Parameter Estimates for Model Calibration

See Appendix 3.1 Model Equations for a complete listing of the model.
See Appendix 3.2 Model Variables for definitions of all model variables.

Single Phase Model Parameter Estimates

PROCESS PARAMETERS

BW_Min_Task_duration(1)=2 hours
RW_Min_Task_Duration(1)=1 hours
QA_Min_Task_Duration(1)=0.75 hours
Internal Precedence Relationship: (dimensionless)
Values of (Percent Completed and Released, Percent Available for Completion) are: (0,0.01) (0.1,0.21) 90.2,0.31) (0.3,0.41) (0.4,0.51) (0.5,0.61) (0.6,0.71) (0.7,0.81) (0.8,0.91) (0.9,1.00) (1.0,1.00)
Release_Trigger_Sensativity(Phase)=0.1 (dimensionless)
Release_Hold_Avg_Time(Phase)=1.05 weeks

SCOPE PARAMETERS

Task_List(1)=445

TARGET PARAMETERS

Schedule

Initial_Proj_Deadline=25 (weeks from phase start)
Time_to_Avg_Exp_Compl_Time(Phase)=1 weeks
Resistance_to_Sched_Slip=2 (dimensionless)

Quality

Quality_Goal_Adjust_Time(Phase)=24 weeks

Project_Quality_Goal=1.00 (dimensionless percent defects)
Initial_Quality_Goal(Phase)=1.00 (dimensionless percent defects)
Basic_prob_flawed_Task(Phase)=0.85 (dimensionless)
Complexity(Phase)=10 (dimensionless)

Cost
Budget_Switch=0 (dimensionless)

RESOURCE PARAMETERS

Gross Labor
Inital_Headcout(Phase)=0.50 (persons)
Max_Headcount(Phase)=2 (persons)
Headcount_Adjustment_Time(Phase)=8 (1/weeks)
HdctJumpStartTime(Phase)=11 (weeks from start)
HdctJumpStopTime(Phase)=14 (weeks from start)
Max_Workweek(Phase)=140 (hours per week)
Normal_Workweek(Phase)=40 (hours per week)
Wrkwk_Avg_Time(Phase)=4 (1/weeks)

Labor Allocation
BW_Priority(Phase)=3 (dimensionless)
RW_Priority(Phase)=1 (dimensionless)
QA_Priority(Phase)=1 (dimensionless)
BW_Labor_Delay(Phase)=1.5 (1/weeks)
RW_Labor_Delay(Phase)=1 (1/weeks)
QA_Labor_Delay(Phase)=7.75 (1/weeks)

Experience
Exper_Assim_Time(Phase)=1 (1/weeks)
Avg_New_member_Exper(Phase)=6 (experience units)

Productivity
Ref_BW_Prdctvty(Phase)=2 (tasks per person per hour)
Ref_RW_Prdctvty(Phase)=1 (tasks per person per hour)
Ref_QA_Prdctvty(Phase)=1.75 (tasks per person per hour)
BW_Prdctvty_Avg_Time(Phase)=1 (1/weeks)
RW_Prdctvty_Avg_Time(Phase)=1 (1/weeks)
QA_Prdctvty_Avg_Time(Phase)=1 (1/weeks)
Adjust_Expect_BW_Prdctvty_Time(Phase)=1 (1/weeks)
Change_Expected_QA_Prdctvty_Time(Phase)=1 (1/weeks)
BW_Prdctvty_Influences_Time(Phase)=1 (1/weeks)
Ch_Expect_Coord_Prdctvty_time(Phase)=1 (1/weeks)
QA_Prdctvty_Report_Time(Phase)=1 (1/weeks)
Report_BW_Prdctvty_Time(Phase)=1 (1/weeks)
Wt_to_Current_BW_Prdctvty(Phase)=1.00 (dimensionless)

Model Control Parameters

LastPhase=1
PhaseNo(1)=1
ResourseSwitch(Phase)=1
Test_Input_1(Phase)=1
Test_Input_2(1)=1
Test_Input_3(Phase)=1
Quality_Goal_Switch=1
Deadline_Switch=1
alpha(Phase)=100
QA_STATUS(Phase)=1
CloseEnough=0.01
Min_Exp_RW_Prdy(Phase)=0.1
Min_Exp_BW_Prdy(Phase)=0.1
Min_Exp_QA_Prdy(Phase)=0.1
Min_Headcount(Phase)=0.001
Max_Proj_DL_Change=100
Max_DL_Change(Phase)=100
COORD_STATUStest(Phase)=0.5
Coord_Min_duration(1)=1
Coord_Labor_Delay(Phase)=1
Ref_Coord_Prdctvty(Phase)=1
Min_Exp_Coord_Prdy(Phase)=0.1
Avg_Act_Coord_Prdctvty_Time(Phase)=1
Report_Coord_Prdctvty_time(Phase)=1 (1/weeks)
Coord_Priority(Phase)=1
Coord_Prdctvty_Avg_Time(Phase)=1
Proj_Budget=500000
Overtime_Premium(Phase)=0.50
Cost_Markup(Phase)=2
Avg_Straight_Pay(Phase)=15
Percent_hourly_Labor(Phase)=0.00
Ref_Qual_of_Practice(Phase)=5
Ref_Exper(Phase)=50
Ref_Complexity(Phase)=100
Avg_Act_BW_Prdctvty_Time(Phase)=1 (1/weeks)
Avg_Act_RW_Prdctvty_Time(Phase)=1 (1/weeks)
Avg_Act_QA_Prdctvty_Time(Phase)=1 (1/weeks)

Multiple Phase Model Paramter Es timates

Notes:

Phases are identified by the integer in parenthesis at the end of the parameter name.
Phase 1 is Product Defintion.
Phase 2 is Design.
Phase 3 is Prototype Testing.
Phase 4 is Reliability/Quality Control.
Phase 5 is not used.
"Phase" instead of a phase number or no phase indentifier indicates that all phases use the value given.

PROCESS PARAMETERS

Basic_prob_flawed_Task(1)=0.5
Basic_prob_flawed_Task(2)=0.3
Basic_prob_flawed_Task(3)=0.05
Basic_prob_flawed_Task(4)=0.05
Basic_prob_flawed_Task(5)=0
BW_Min_Task_duration(1)=5
BW_Min_Task_duration(2)=2
BW_Min_Task_duration(3)=6
BW_Min_Task_duration(4)=2
BW_Min_Task_duration(5)=2
Coord_Min_duration(1)=1
Coord_Min_duration(2)=1
Coord_Min_duration(3)=1
Coord_Min_duration(4)=1
Coord_Min_duration(5)=1
Dependency(1,1)=0
Dependency(1,2)=1
Dependency(1,3)=1
Dependency(1,4)=0
Dependency(1,5)=0
Dependency(2,1)=0
Dependency(2,2)=0
Dependency(2,3)=1
Dependency(2,4)=1
Dependency(2,5)=0
Dependency(3,1)=0
Dependency(3,2)=0
Dependency(3,3)=0
Dependency(3,4)=1
Dependency(3,5)=0
Dependency(4,1)=0
Dependency(4,2)=0
Dependency(4,3)=0
Dependency(4,4)=0
Dependency(4,5)=0
Dependency(5,1)=0

Dependency(5,2)=0
Dependency(5,3)=0
Dependency(5,4)=0
Dependency(5,5)=0
QA_Min_Task_Duration(1)=2
QA_Min_Task_Duration(2)=2
QA_Min_Task_Duration(3)=0.5
QA_Min_Task_Duration(4)=1
QA_Min_Task_Duration(5)=1
Release_Hold_Avg_Time(Phase)=1.06
Release_Trigger_Sensativity(1)=0.6
Release_Trigger_Sensativity(2)=0.05
Release_Trigger_Sensativity(3)=0.6
Release_Trigger_Sensativity(4)=0.6
Release_Trigger_Sensativity(5)=0.6
RW_Min_Task_Duration(1)=3
RW_Min_Task_Duration(2)=0.5
RW_Min_Task_Duration(3)=0.5
RW_Min_Task_Duration(4)=1
RW_Min_Task_Duration(5)=1
Internal Precedence Relationships:
T5(*,1)=0.01/0.15/0.3/0.60/0.80/0.90/0.95/1/1/1/1.00 * Product Definition
T5(*,2)=0.01/0.15/0.40/0.5/0.65/0.75/0.85/0.95/0.97/1.00/1.00 * Design
T5(*,3)=0.4/0.5/0.6/0.7/0.8/0.9/0.95/1/1/1/1.00 * Prototype Testing
T5(*,4)=1/1/1/1/1/1/1/1/1/1.00/1.00 * Reliability/Quality Control
T5(*,5)=0/0/0/0/0/0/0/0/0/0 *Closed Gate for Phase 5
External Precedence Relationships:
T6(*,1,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,1,2)=0.00/0.1/0.25/0.5/.65/0.80/0.90/0.95/0.97/1.00/1.0 * Product Definition to Design
T6(*,1,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0/1.00/1.00 * Product Definition to Prototype Testing
T6(*,2,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,2,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,2,3)=0.01/ 0.05/0.2/0.4/0.6/0.80/1/1/1/1/1.00 * Design to Prototype Testing
T6(*,2,4)=0.00/0.00/0.0/0/0/0/0/0/0/1/1.0 * Design to Reliability/Quality Control
T6(*,3,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,3,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,3,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,3,4)=0.00/0.00/0.0/0.0/0.0/0.0/0.0/0.0/0.0/1.0/1.00 * Prototype Testing to Reliability/Quality Control
T6(*,4,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,4,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,4,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,4,4)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,5,1)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,5,2)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,5,3)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
T6(*,5,4)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA

T6(*,5,5)=0.00/0.00/0.0/0.0/0.0/0.0/0.00/0.00/0.00/0.00/1.00 * NA
Test_Input_2(1)=0 * RELEASE"DUMP" SWITCH
Test_Input_2(2)=1 * RELEASE"DUMP" SWITCH
Test_Input_2(3)=0 * RELEASE"DUMP" SWITCH
Test_Input_2(4)=0 * RELEASE"DUMP" SWITCH
Test_Input_2(5)=0 * RELEASE"DUMP" SWITCH

SCOPE PARAMETERS

Task_List(1)=466
Task_List(2)=1219
Task_List(3)=1219
Task_List(4)=1219
Task_List(5)=1

TARGET PARAMETERS

Schedule

Initial_Proj_Deadline=94
Resistance_to_Sched_Slip=2

Quality

Project_Quality_Goal=1
Quality_Goal_Adjust_Time(1)=24
Quality_Goal_Adjust_Time(2)=24
Quality_Goal_Adjust_Time(3)=24
Quality_Goal_Adjust_Time(4)=24
Quality_Goal_Adjust_Time(5)=24

Cost

Avg_Straight_Pay(Phase)=25
Budget_Switch=1
Cost_Markup(Phase)=2
Overtime_Premium(Phase)=0.50
Percent_hourly_Labor(Phase)=0.00
Proj_Budget=500000

RESOURSE PARAMETERS

Gross Labor

HdctJumpStartTime(1)=1
HdctJumpStartTime(2)=11
HdctJumpStartTime(3)=1
HdctJumpStartTime(4)=1
HdctJumpStartTime(5)=1
HdctJumpStopTime(1)=1

HdctJumpStopTime(2)=14
HdctJumpStopTime(3)=1
HdctJumpStopTime(4)=1
HdctJumpStopTime(5)=1
HdctJumpSwitch(1)=0
HdctJumpSwitch(2)=1
HdctJumpSwitch(3)=0
HdctJumpSwitch(4)=0
HdctJumpSwitch(5)=0
Headcount_Adjustment_Time(1)=12
Headcount_Adjustment_Time(2)=8
Headcount_Adjustment_Time(3)=8
Headcount_Adjustment_Time(4)=8
Headcount_Adjustment_Time(5)=8
Inital_Headcout(1)=0.5
Inital_Headcout(2)=0.5
Inital_Headcout(3)=0.5
Inital_Headcout(4)=0.5
Inital_Headcout(5)=0.5
Initial_Quality_Goal(Phase)=0.9
Max_Headcount(1)=2
Max_Headcount(2)=2
Max_Headcount(3)=2
Max_Headcount(4)=2
Max_Headcount(5)=2
Max_Workweek(Phase)=140
Normal_Workweek(Phase)=40

Labor Allocation
BW_Labor_Delay(1)=1.5
BW_Labor_Delay(2)=1.5
BW_Labor_Delay(3)=1.5
BW_Labor_Delay(4)=1.5
BW_Labor_Delay(5)=1.5
BW_Prdctvty_Avg_Time(Phase)=1
BW_Prdctvty_Influences_Time(Phase)=1
BW_Priority(1)=3
BW_Priority(2)=3
BW_Priority(3)=5
BW_Priority(4)=3
BW_Priority(5)=3
Coord_Labor_Delay(1)=1
Coord_Labor_Delay(2)=1
Coord_Labor_Delay(3)=1
Coord_Labor_Delay(4)=1
Coord_Labor_Delay(5)=1

Coord_Priority(1)=1
Coord_Priority(2)=1
Coord_Priority(3)=1
Coord_Priority(4)=1
Coord_Priority(5)=1
QA_Labor_Delay(1)=12
QA_Labor_Delay(2)=3
QA_Labor_Delay(3)=0.5
QA_Labor_Delay(4)=3
QA_Labor_Delay(5)=7.75
RW_Labor_Delay(1)=1
RW_Labor_Delay(2)=1
RW_Labor_Delay(3)=1
RW_Labor_Delay(4)=1
RW_Labor_Delay(5)=1
RW_Priority(1)=1
RW_Priority(2)=2
RW_Priority(3)=1
RW_Priority(4)=1
RW_Priority(5)=1

Experience
Avg_New_member_Exper(Phase)=6
Exper_Assim_Time(Phase)=1

Productivity
Adjust_Expect_BW_Prdctvty_Time(Phase)=1
Avg_Act_BW_Prdctvty_Time(Phase)=1
Avg_Act_Coord_Prdctvty_Time(Phase)=1
Avg_Act_QA_Prdctvty_Time(Phase)=1
Avg_Act_RW_Prdctvty_Time(Phase)=1
Change_Expected_QA_Prdctvty_Time(Phase)=1
Ch_Expect_Coord_Prdctvty_time(Phase)=1
Coord_Prdctvty_Avg_Time(Phase)=1
Min_Exp_BW_Prdy(Phase)=0.1
Min_Exp_Coord_Prdy(Phase)=0.1
Min_Exp_QA_Prdy(Phase)=0.1
Min_Exp_RW_Prdy(Phase)=0.1
QA_Prdctvty_Avg_Time(Phase)=1
QA_Prdctvty_Report_Time(Phase)=1
QA_Priority(1)=1
QA_Priority(2)=2
QA_Priority(3)=1
QA_Priority(4)=1
QA_Priority(5)=1
Report_BW_Prdctvty_Time(Phase)=1

Report_Coord_Prdctvty_time(Phase)=1
RW_Prdctvty_Avg_Time(Phase)=1
Wt_to_Current_BW_Prdctvty(Phase)=1.00

MODEL CONTROL PARAMETERS

alpha(Phase)=100
CloseEnough 0.01
Complexity(Phase)=10
COORD_STATUStest(Phase)=0.5
Deadline_Switch=1
LastPhase=3
Max_DL_Change(Phase)=100
Max_Proj_DL_Change=100
Min_Headcount(Phase)=0.001
PhaseNo(1)=1
PhaseNo(2)=2
PhaseNo(3)=3
PhaseNo(4)=4
PhaseNo(5)=5
QA_STATUS(Phase)=1
Quality_Goal_Switch 1
Ref_BW_Prdctvty(Phase)=2
Ref_Complexity(Phase)=100
Ref_Coord_Prdctvty(Phase)=1
Ref_Exper(Phase)=50
Ref_QA_Prdctvty(Phase)=1.75
Ref_Qual_of_Practice(Phase)=5
Ref_RW_Prdctvty(Phase)=1
ResourseSwitch(Phase)=1

TaskListScale(up,down)=Task_List(up)/Task_List(down)