

# Modeling rework cycle: comparing alternative formulations

## **Abstract**

Rework cycle is at the heart of modeling projects, one of the major research and application areas in system dynamics. Alternative formulations have been used for modeling rework cycle but little is known about the comparative benefits and shortcomings of these formulations. In this paper we introduce a new formulation for rework cycle in which multiple defects may exist in a task. We compare this formulation with three others, two from system dynamics literature and one agent based model, on multiple project performance metrics including finish time, quality, and overall project trajectory. Extensive analysis shows some important differences in performance metrics across different models and informs the impact of assumptions about the nature of defects and the homogeneity of tasks on the behavior of alternative models. We discuss the implications for selecting robust formulations in modeling project dynamics.

## **Keywords:**

project dynamics, rework cycle, model comparison, formulation, agent-based

## **Acknowledgements:**

We would like to thank Jim Lyneis and three anonymous reviewers at 2008 system dynamics conference for their valuable feedback. We also thank Thanujan Ratnarajah for his excellent research support in this project.

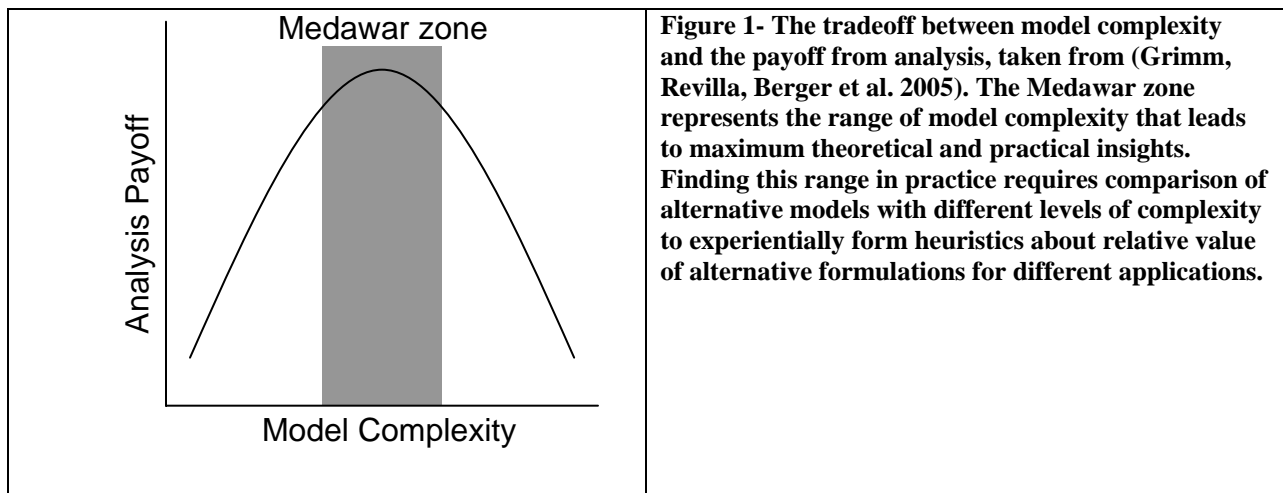
## **1- Introduction and motivation**

Projects are central to how work is done in organizations. From construction to software development, systems engineering, and product development, projects are at the heart of organization of work in modern societies. Project performance, in terms of schedule, cost, and quality, evolves through time, thus it is a dynamic concept that lends itself to application of system dynamics modeling. The vast variation in quality, timeliness, and cost performance of projects across different fields has motivated many consultants and academics to study dynamics of projects. In fact, project dynamics has been one of the core areas of study in system dynamics which has produced much academic research and ample consulting fees. Lyneis and Ford provide a comprehensive review of the literature on applications of system dynamics to project management (Lyneis and Ford 2007).

Majority of system dynamics studies that focus on project dynamics include a simulation model of project evolution. These models vary in their level of complexity and the feedback effects they capture. However, a core feature of all these models, building on the ground breaking consulting project by Pugh Roberts Associates in the 70s, is the rework cycle (Cooper 1980). The basic insight in rework cycle formulation is the realization that tasks that are completed as part of a project may be flawed and may need rework. Given the widespread application of system dynamics in project modeling, the basic rework cycle idea is among the most widely used formulation concepts in the field. Not surprisingly, multiple alternative formulations have been used to capture the rework cycle with different levels of complexity and different conceptions of defects (e.g. See Cooper 1980; AbdelHamid and Madnick 1991; Ford and Sterman 1998; Taylor and Ford 2006; Lee and Pena-Mora 2007).

Despite the large number of alternative rework cycle formulations, the literature lacks insights on their comparative advantages and shortcomings. Such understanding is needed for practitioners and researchers who seek to use the most appropriate formulation for the

application at hand. Figure 1 represents the typical tradeoff between model complexity and the analytical payoff of different models (Grimm, Revilla et al. 2005). Where as very simple models do not go beyond general intuition, models that are very complex can be ineffective as they are costly to build, hard to analyze, and challenging to communicate. A sweet spot in the level of complexity, the Medawar zone, should be found for each application area where modeling leads to maximum return. Studies that compare alternative formulations, their complexity, and their insights, under controlled experiments, inform the research community about the Medawar zone in their area of research. Such studies have been conducted in other areas of inquiry such as modeling diffusion, epidemics, and supply chains (e.g. Parunak, Savit et al. 1998; Rahmandad and Sterman 2008), yet research is lacking on relative advantages and disadvantages of alternative formulations to model project dynamics. If different formulations of projects lead to similar results for the purpose at hand, the analyst is well-advised to use the simplest possible formulation which maximizes transparency and analysis speed. If different formulations significantly differ, however, one needs to know and recognize those differences and select the formulation that strikes an appropriate balance between precision and simplicity. Without controlled comparative studies, however, such decisions will be based more on intuition than data, and accumulation of knowledge is harder to achieve.



Moreover, a comparative study will inform the mapping of different parameters from one formulation to another, and thus enables better accumulation of knowledge about project parameters across different studies. For example, error rate in some formulations reports the probability of a task being defective, where as in other contexts, may inform the number of errors generated per task completed. Thus a parameter mapping transformation is needed to use parameter values from the first formulation to inform the value of parameters in the second.

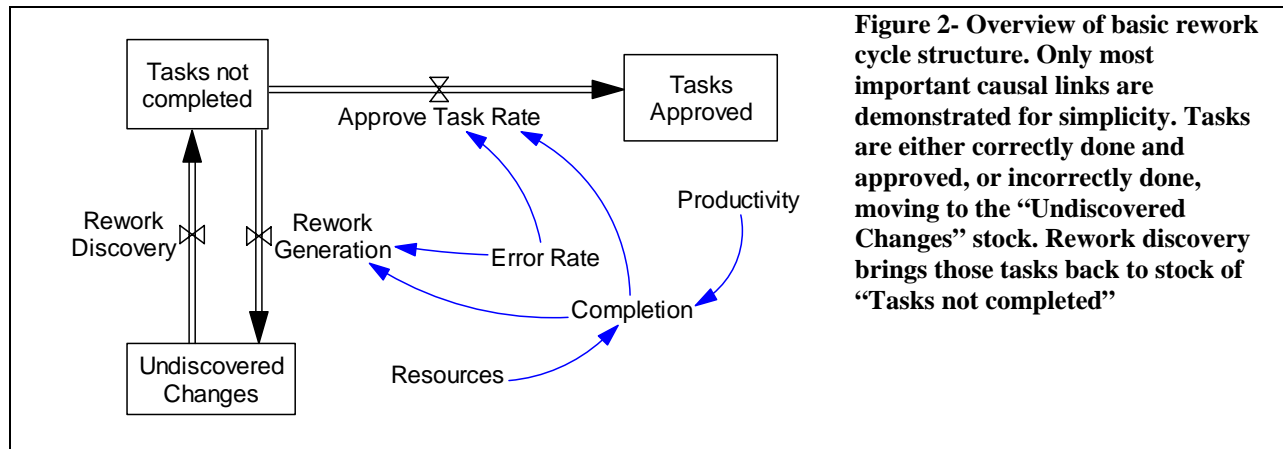
This paper tackles these two questions by providing a comparison of four formulations of rework cycle on different points of the complexity scale. Moreover, we introduce one of these formulations for the first time in the literature and show that it strikes a good balance between complexity and precision for some applications.

### **1-1- Rework Cycle at the Heart of Project Dynamics**

Rework cycle may be the most important feature of system dynamics project models (Lyneis and Ford 2007). It recognizes that completion of a project task may be flawed, resulting

in a need for rework. Rework can itself be flawed, requiring further rework in a recursive cycle that can extend project duration and work load far beyond what is originally conceived. In the absence of rework cycle project completion is a function of the number and scope of tasks and the resources available. By considering defects, quality and testing find their salient role in generating the path-dependent reinforcing loops (e.g. burnout, error building on error, cutting corner) that have been shown to be critical in project dynamics. Thus rework cycle also allows for capturing some of the main mechanisms leading to projects going overtime and budget and compromising quality and functionality.

Figure 2 provides a very simple conception of rework cycle, adopted from one of the early models of projects (Chapter 4, Richardson and Pugh 1981). Here tasks, upon completion, may flow into “Tasks Approved” (if they are correctly completed and accepted), or “Undiscovered Changes”, those tasks that have to be reworked, but not yet recognized as such. The completion of tasks depends on available resources and their productivity, while quality of the work is captured by defect rate. The schematic below is chosen to represent the simplest conception of rework cycle, a second order system<sup>1</sup>. However, even this very simple model can be used as the building block for capturing much more complex project dynamics. For example effects of schedule pressure, morale, communication congestion, overtime, and experience on error (defect) rate and productivity can be directly applied here (Cooper 1980; Sengupta and Abdelhamid 1993). The changes in resource availability and speed of rework discovery can also be easily accommodated. Multiple simple blocks, with similar structures, can be connected in cascades that represent multiple phases of bigger projects and different feedback effects between up-stream and down-stream projects (Repenning 2000). Other modelers have expanded this basic formulation to include an explicit testing procedure (AbdelHamid and Madnick 1991), and to keep track of changes (or defects) in a separate stock and flow structures (Ford and Sterman 1998).



This framework can be used with exceedingly more details, e.g. to represent small homogeneous chunks of a project. In the extreme, a detailed agent-based model of a project includes separate entities for each task, as well as each unit of resource (typically individuals who will be assigned to tasks, but can also include machinery, money etc) (Levitt, Thomsen et al. 1999). In such formulation each task can go through multiple “states”, e.g. “not worked on”,

<sup>1</sup> The total number of tasks in the project, which is the sum of three stocks, is constant, and thus one of the stocks can always be calculated based on the other two, leading to two independent stocks.

“worked on and pending testing”, “approved”, or “pending rework”. Moreover each task entity can include properties such as the number of defects that have accumulated in the process of working on that task. Growing computational power and software availability have encouraged the application of disaggregated models. Detailed representation and including stochasticity improves external validity of the model (likeness to the real world) and thus potential precision of the analysis. However, the additional disaggregation often also complicates a comprehensive understanding of which model structures and processes are generating the observed behavior; model development and bug fixing is harder in these models; and communication of the insights to real-world decision-makers may prove more demanding. Moreover, typically detailed agent based<sup>2</sup> models are stochastic in nature and require more computational resources than simpler differential equation representations, both to get a reliable view of the range of stochastic results they generate, as well as for conducting a single simulation. These computational costs can lead to reduced sensitivity analysis, longer-turn around time for analysis, and smaller model boundary (i.e. the range of feedback effects and sub-systems included in the analysis) (Rahmandad and Sterman 2008).

In the context of these tradeoffs, two questions are relevant to researchers and practitioners who model projects: 1) What kind of tradeoffs exist in terms of analysis payoff vs. complexity, when choosing different formulations for rework cycle? 2) How do different formulations connect to each other? For example, if task level data suggests that 20% of tasks have at least one defect in their first implementation, should we use a defect rate of 0.2, or another number is more appropriate, in an aggregate formulation. Our study of these questions sheds light on the relationships between different formulations for rework cycle and the tradeoffs among complexity, performance, precision, and flexibility.

## **2- Research design**

In this study we compare and analyze alternative formulations for the core rework cycle process that is central to different projects. For this purpose we focus on analyzing a very simple project that consists of a pre-determined number of tasks. Tasks are completed by a group of dedicated resources with fixed productivity. Completed tasks are then tested, and accepted or sent for rework. Same resources are in charge of rework. Testing is only limited by the time it takes to conduct the tests (no resource bottleneck for testing phase). We keep the study manageable in scope and controlled in design by including no feedback on productivity, quality of completion and rework, testing quality, amount of resources, or scheduled finish date in the core model. Nevertheless, we briefly address the impact of additional feedback processes in the analysis section.

Four different model formulations capture and simulate this simple project. Three of these formulations are selected from the literature to inform different points on the model complexity dimension (Figure 1) and another model is introduced into the literature proposing a potentially better balance between complexity and analytic payoff for some application areas. These models are built in parallel so that similar concepts, variables, and parameters are comparable across the models. Parallel model development is achieved by starting from the most complex model, and deriving the parameters for the successively more aggregate models from

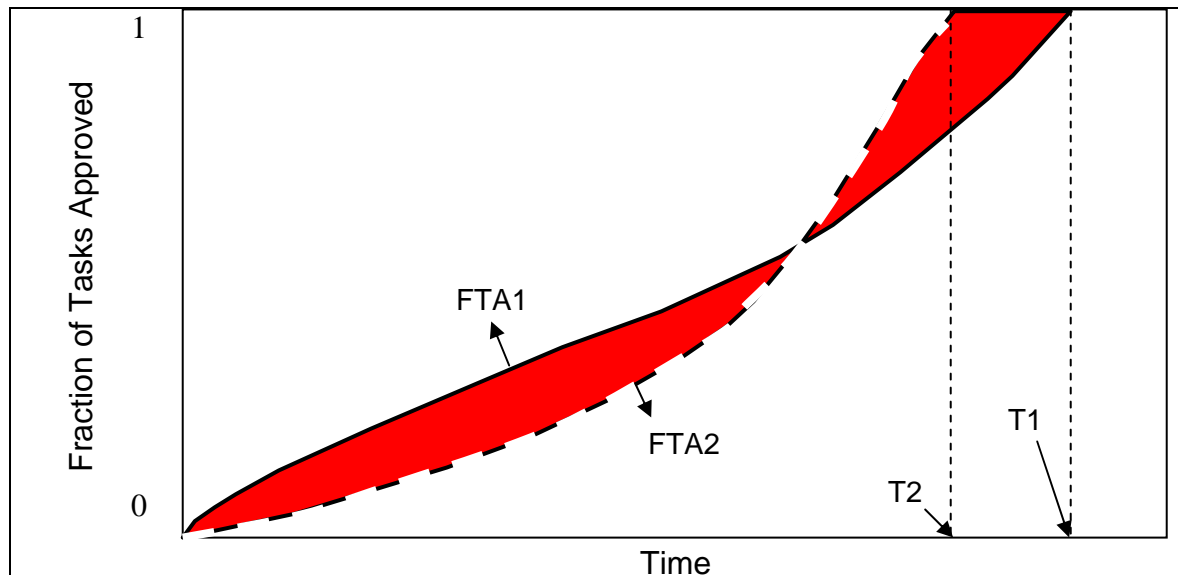
---

<sup>2</sup> Here we use the term agent-based to refer to a wide category of models that represent individual tasks and resources as distinct objects. Depending on the complexity of such representation these models could fall on a wide continuum from traditional discrete event models to more detailed agent-based ones.

this version. This process informs the relationship between parameter values at different levels of aggregation thus answering one of our research questions.

We address the other question through comparing the alternative models on costs and benefits of disaggregation. On the costs of additional detail we observe model complexity in terms of the number of stock variables in each formulation. That correlates strongly with total number of variables, and computational costs. It also moderately correlates with the costs of building and maintaining a model. We measure the benefits of disaggregation by comparing the flexibility and performance of different formulations. Flexibility measures the different parameter settings and assumptions about the nature of work that can be changed in each formulation. It therefore informs the range of applications that a formulation can support. Performance metrics compare the behavior of the four alternative models on the quality and schedule dimension. We also measure the difference between alternative formulations in terms of project evolution and costs.

We measure and compare three performance metrics: “Finish Time”, “Average Defects in Approved Task”, and “Average Absolute Error”. Finish Time, the time it takes to finish 99% of the tasks in the project, is reported for different formulations. Given the constant resources allocated to each project, and constant initial project size, Finish Time is a good metric for overall project cost as well. Average defects in approved task is reported for the three more complex models (the concept is not captured in the simplest formulation). Finally, “Average absolute error” (AAE) compares the four formulations head-to-head in terms of the fraction of tasks approved at different points of time in the project, and reports their difference. Figure 3 gives an overview of this performance metric. Note that AAE is normalized for project size and duration and therefore can be used across projects of different sizes.



**Figure 3- Finish time and average absolute error metrics. The results of hypothetical simulations with two alternative formulations (M1 in solid line and M2 in dashed line) are reported for a project. The two lines represent Fraction of Tasks Approved for M1 and M2 at different points in time (FTA1 and FTA2). The Finish Times for two formulations (T1 and T2) are reported in months. The average absolute error between M1 and M2 (AAE12) is calculated as:**

$$AAE12=AAE21=(\int_0^{Max(T1,T2)} |FTA1 - FTA2|) / Max(T1,T2)$$

After describing the model formulations and deriving the parameter values from the most detailed model (Section 3) we present the base case results where the performances of the four models are compared using different metrics discussed above (Section 4). Next we will discuss the sensitivity of the results to different project settings including alternative defect rates, project sizes, testing accuracy and speed, and different probability distributions for the stochastic parameters of the most detailed model (Section 4-2). The conclusions and implications for modeling project dynamics are provided in Section 5.

### 3- Four Formulations for Rework Cycle

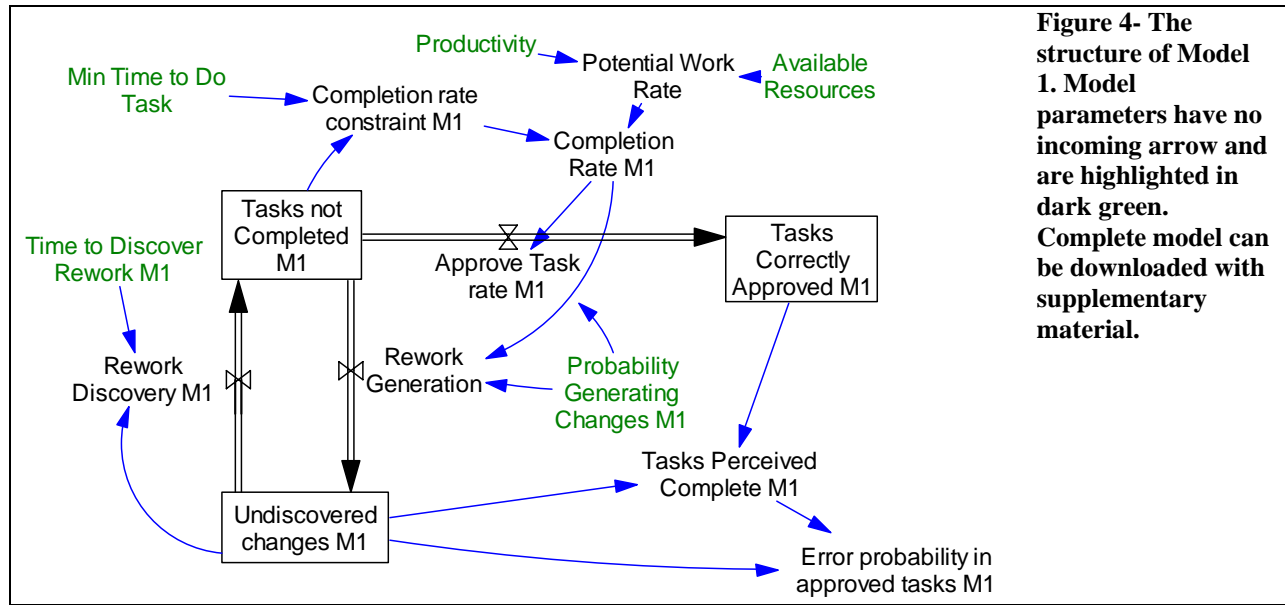
The four different models are introduced in this section. We name these models according to their level of complexity, so the simplest model is “Model 1” or M1, etc. We first briefly describe each model and then discuss the derivation of the parameters of the more aggregate models (M1-M3) based on the most detailed one (M4).

**Model 1-** This formulation is adopted from an early model of project dynamics (Richardson and Pugh 1981) because it includes the minimum number of independent stocks required to capture the core idea of the rework cycle. Figure 4 presents an overview of the stock and flow structure for this model. Basically the tasks are completed based on current *Available Resources* and *Productivity*. The *completion Rate* is also bounded by minimum time to complete a task. A fraction of tasks are approved (either because they are done correctly, or because the testing process has not flagged them as requiring change). The rest of the tasks are sent to the “Undiscovered changes M1” where they remain until they are discovered to be in need for rework, and thus sent back to the stock of tasks not completed.

It is important to note that testing process is not explicitly captured in this model. Therefore rework discovery relates both to the defect generation and the quality of testing. Even with high real defect rates, a lousy testing procedure can reduce the speed of discovering changes. Moreover, the formulation does not distinguish between rework and initial work because it sends all the discovered rework into the tasks not completed. This assumption implies that M1 formulation does not have enough degrees of freedom to capture significant differences in the nature of work between the first time work and the rework. These considerations therefore require us to derive the parameters for “*Probability Generating Changes M1*” and “*Time to Discover Rework M1*” from the parameter values in the more detailed models. Finally, this model does not directly keep track of potential problematic tasks that are approved, rather it lumps together approved errors with undiscovered changes that could be discovered later. Therefore in using this formulation modelers typically consider a project finished when the sum of *Tasks Approved* and *Undiscovered Changes* reach a threshold, and then the fraction of tasks considered finished that are in the undiscovered changes stock will inform the quality (“*Error Probability in Finished Task M1*”). Figure 4 graphs the stock and flow structure for this model. Standard formulations are used in M1 and the full model is available as an online appendix<sup>3</sup>.

---

<sup>3</sup> The online appendix with models is currently available from: <http://www.savefile.com/files/1870115>. Given the limitations of the Journal’s online submission system the material was uploaded anonymously to this public space. Upon acceptance, all the simulation models will be made available for interested researchers on Journal website and author’s personal website.



**Figure 4- The structure of Model 1. Model parameters have no incoming arrow and are highlighted in dark green. Complete model can be downloaded with supplementary material.**

**Model 2-** Ford and Sterman (1998) introduced a more detailed formulation for the rework cycle. Their model includes a co-flow structure to track defective tasks along with the tasks in the core rework cycle formulation. We adopted, and slightly modified, their rework cycle formulation for the study at hand, calling it model 2 or M2. Figure 5 provides an overview of this model. Full model is available in the online appendix.

This formulation includes six independent stock variables<sup>4</sup>. Available resources are allocated between “*Completion Rate M2*” and “*Change Task Rate M2*” proportional to the desired completion rates. Desired rates depend on stocks of “*Tasks Not Completed M2*” and “*Tasks to be Changed M2*”, as well as time to do the tasks or the rework. “*Completion Defect Probability M2*” determines what fraction of completed tasks potentially requires a change. Not all such tasks are however discovered through the testing process. “*Probability of Missing a Defect*” informs what fraction of problematic tasks is discovered and sent to “*Known changes M2*” stock and what fraction flows into “*Tasks Approved M2*” along with their “*Changes approved M2*.” For tasks with known changes, rework is not always successful in removing the problem. While some of the problems are fixed in the rework process, some defects remain and flow back into “*Undiscovered Changes M2*.”

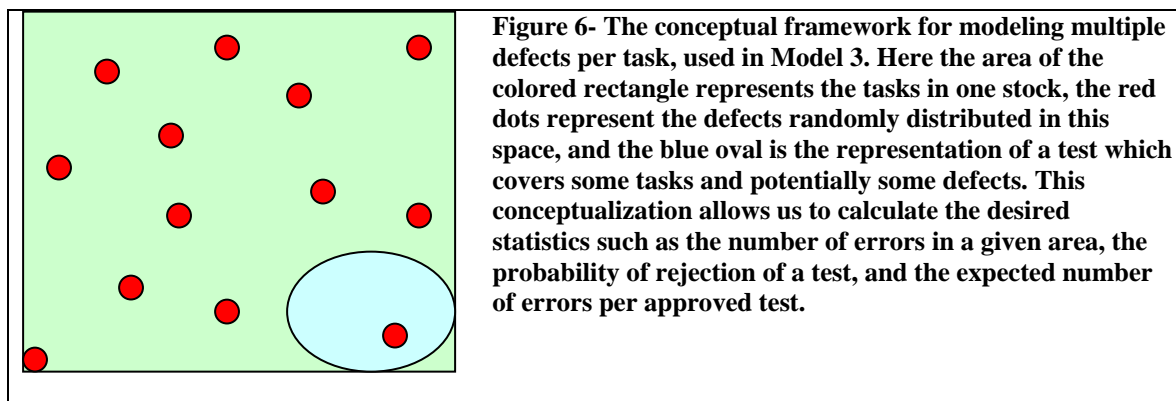
<sup>4</sup> Sum of all tasks remain constant, reducing the number of independent stocks by one, from those visible in the figure.



what fraction are sent for rework, we need to know what fraction of tasks are defect-free. Finding that fraction requires some additional distributional assumptions as for how the defects are distributed among the tasks. Furthermore, if we consider the imperfections of testing process, we should take into account that not only defect-free tasks, but also some of the tasks with one, two, or more defects could, by chance, be accepted. Thus a flexible formulation should be able to determine what the defect density is in both approved and rejected tasks, depending on the effectiveness of testing.

Moreover, the testing process often includes tests that cover more than a single task. For example consider integration testing in software development (vs. unit testing). Where unit testing makes sure a specific piece of code (a small module or function) is working properly given a set of inputs, the integration testing examines interactions between multiple modules, thus increasing the chance that a test fails because any of the components have a defect. These larger tests could in fact be rejected more easily, for example a failure in one of five tasks included in a test leads to sending all those tasks for further rework.

Figure 6 presents the conceptual framework we use to incorporate these considerations. Here we represent the tasks as the area of the rectangle: the larger the number of tasks, the bigger the rectangle. The number of tasks at each stage is represented in a stock variable (See Figure 7 for a partial stock and flow diagram for Model 3). Defects are shown as the dots distributed in the task area. We track the number of these defects in the stock variable “*Undiscovered changes M3*.”<sup>5</sup> Perfect mixing nature of material in a stock requires these defects to be randomly distributed in the area. That is, the location of each defect is completely independent of the other defects, and all points in the area have the same chance of including a defect. Finally, a test is represented as an oval that covers some area, i.e. a number of tasks. A test is then rejected if at least one of the defects inside the test area is identified in the testing process. Otherwise the test is approved, accepting the part of the area covered by test into the stock of “*Tasks Approved M3*” (See Figure 7). The defects/changes that have been missed in that test flow in parallel into the “*Changes Approved M3*” stock.



The perfect mixing requirement proves to be analytically helpful in this setting. It leads to the independence of placement of defects in the task area which results in the Poisson distribution of the number of defects per any area (e.g. the area covered by a test). This is due to the fact that our setting satisfies the Poisson distribution’s underlying assumptions: that

<sup>5</sup> We use defect and change interchangeably for Model 3. While defect is conceptually more appropriate in this context, we want to keep the parallelism with the Model 2 and therefore use similar names for the stock and flow variables as used in Model 2.

probability of observing a defect over an area does not change over different areas, and that probability of observing a defect over an area is independent of the observation of defects in the rest of the space. We can therefore find the probability of finding  $k$  defects in a test area of size  $a$  (unit: task), where the defect density is  $d$  (unit: defect/task) based on the Poisson probability distribution:

$$P(\#errors = k) = \frac{e^{-da} (da)^k}{k!} \quad (1)$$

For example, the probability that no defect is present in the test area, and thus the test is accepted (releasing the tasks in the test area into the stock of approved tasks) is:  $e^{-da}$ .

Furthermore, we can calculate the impact of imperfect testing. A test is passed under this setting if no defects are present in the test area, or a single defect is present and it is missed (with probability  $\varepsilon$ ), or two defects are present and both are missed (which assuming independence between different defects will have a probability of  $\varepsilon^2$ ), and so on. Therefore the probability of a test with area  $a$  passing can be calculated as:

$$P(\text{TestPass}) = \sum_{k=0}^{\infty} \frac{e^{-da} (da)^k}{k!} \varepsilon^k = e^{-da(1-\varepsilon)} \quad (2)$$

The expected number of defects in a test that has passed is another variable we need to calculate in order to determine the flow of defects from stock of “*Undiscovered Changes M3*” to stocks of “*Changes Approved M3*” and “*Known Changes M3*”. This expected value can also be calculated:

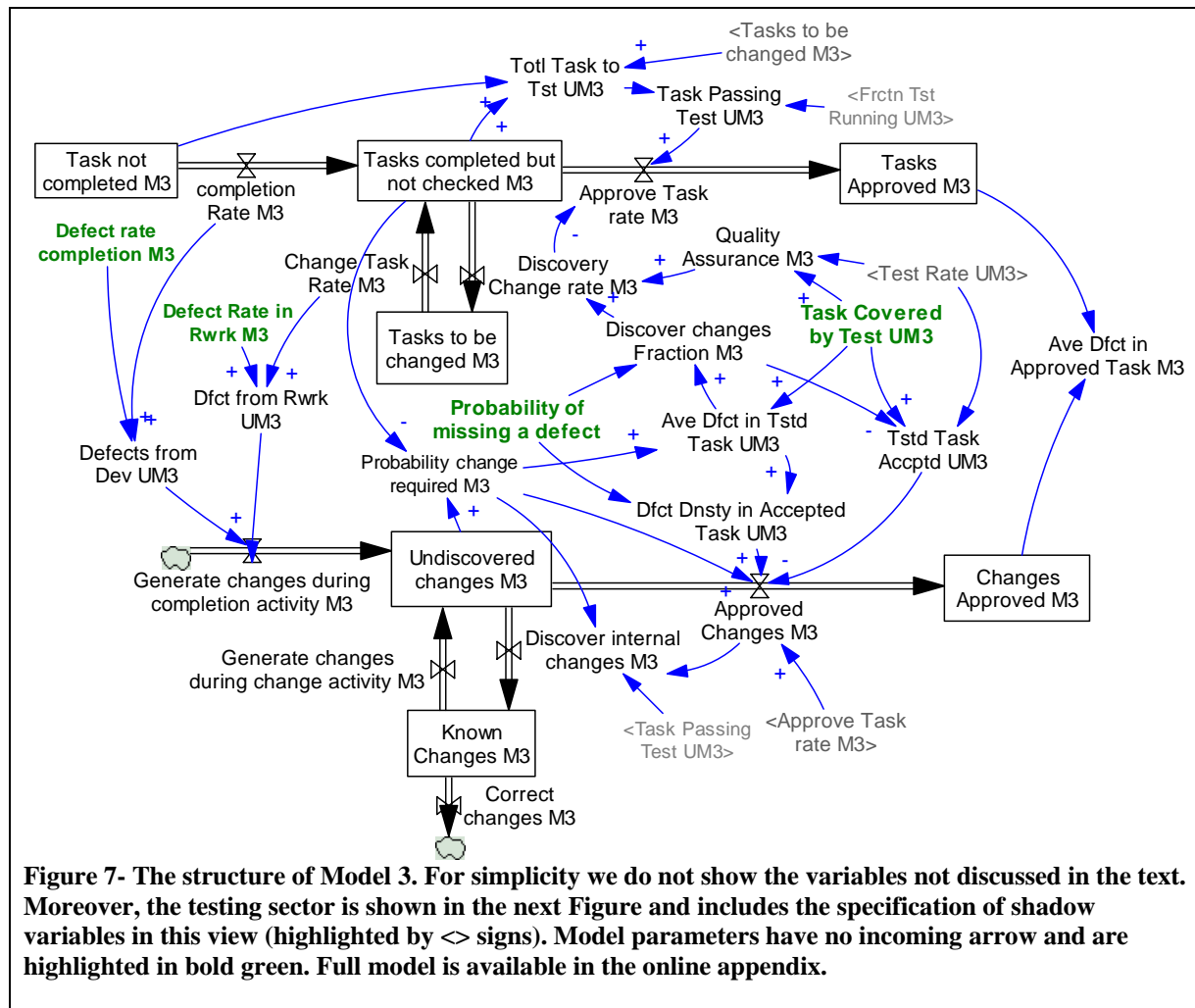
$$E(\text{DefectPerPassedTest}) = \sum_{k=0}^{\infty} k \frac{e^{-da} (da)^k}{k!} \varepsilon^k = da \varepsilon e^{-da(1-\varepsilon)} \quad (3)$$

Finally, this conceptualization allows us to consider lower-than-complete test coverage. Testing can be imperfect not only because we may miss some defect in the task area that is being tested, but also because we may not test all the tasks in the system. Under low test coverage, a part of task area is released without testing, along with the area covered by each test. The defects in the untested area are also released into the approved tasks with the average defect density for the task area in general. In order to consider such imperfect testing scenarios, it is helpful to introduce explicit stocks and flows for the tests that are being run<sup>6</sup>.

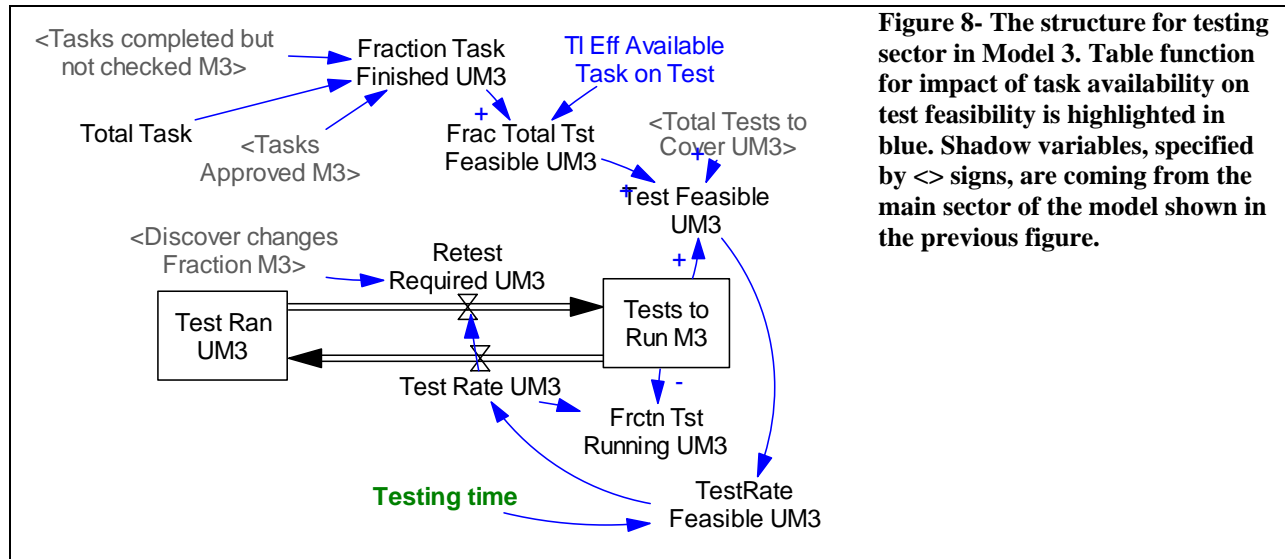
In model 3 we use “*Discover changes Fraction M3*” to measure what fraction of tests fail. This is one minus the value found according to equation 2. We calculate the number of tasks that are sent to be changed using this fraction and the total amount of tasks covered by testing (“*Quality Assurance M3*”). On the other hand, the tasks approved are potentially more, if test coverage is lower than 100%. Therefore we calculate the “*Approve Task Rate M3*” by subtracting the rejected tasks from the total tasks that either go through testing or are accepted without test. The number of these tasks is calculated by using the fraction of total tests that are conducted at any point in time (“*Frctn Tst Running UM3*”) and extending that fraction to all the tasks pending testing (“*Totl Task to Tst UM3*”)<sup>7</sup>. For example if testing rate is conducted at 0.15 of tests per month, then 0.15 of tasks to be tested are flowing monthly out of the stock of “*Tasks Completed but not Checked M3*”, out of which some are caught in test and set to “*Tasks to be Changed M3*” and the rest are approved.

<sup>6</sup> While the additional stocks are not required if the test coverage is unchanged throughout a simulation, we discuss the additional structure for increased clarity and flexibility of formulation to be applied to dynamic test coverage.

<sup>7</sup> UM3 is used to distinguish a concept unique to model 3, where as M3 specifies a concept that has parallels in other models.



We formulate the rate “*Approved Changes M3*” (see Figure 7) in the defect co-flow by adding up the two sources of defects accepted. First, those defects missed in testing are captured in “*Dfct Dnsty in Accepted Task UM3*” (See equation 3), multiplied by “*Approve Task rate M3*.” Second the tasks approved without testing carry defects with the density of “*Probability change required M3*.” Once the rate for approval of defects is determined, it is easy to calculate the rate of discovery of defects: from all the defects that are passing through testing (*Task Passing Test UM3* \* *Probability Change Required M3*) we subtract the approved ones (“*Approved Changes M3*”) and let the rest flow into “*Known Changes M3*.” The Model 3 formulation also allows us to consider the introduction of additional defects during the rework process (“*Dfct from Rwrk UM3*”) as well as the possibility that some defects are not corrected in the rework process (“*Generate Changes During Change Activity M3*”). All these defects flow into “*Undiscovered Changes M3*”.



**Figure 8-** The structure for testing sector in Model 3. Table function for impact of task availability on test feasibility is highlighted in blue. Shadow variables, specified by <> signs, are coming from the main sector of the model shown in the previous figure.

Finally, the testing process starts with a fixed set of tests (in the stock “*Tests to Run*” (see Figure 8)). The fraction of tasks that are finished determines, based on the nature of work, how much testing can be done (through “*TI Eff Available Task on Test*”). At one extreme all tasks completed can be tested because different pieces of project are tested independent from each other. Under these conditions the table function represents the line  $x=y$ . At another extreme one may need all the pieces of the project to come together before anything can be tested. Such project has a table function that remains at zero for all values but for input value of one, which has an output of one. Realistic projects are often somewhere between the two extremes, but we use the first case in this study because Model 1 does not capture testing explicitly. Once tasks are run, they flow into the “*Test Ran UM3*” (see Figure 8) stock. On the other hand, the fraction of tasks that are rejected due to discovered defects are cycled back to “*Tests to Run UM3*” so that they can be administered again later.

Model 3 allows us to consistently account for multiple defects per task. It also ties the testing process with alternative accuracy and coverage levels to the co-flows of tasks and defects. It achieves these by including one more independent stock than Model 2 (total of 7 independent stock variables). However, by its nature, this structure continues to assume perfect mixing of tasks and defects and thus misses on heterogeneity in tasks, defect probability per task, stochastic variations in task completion, or interactions between different tasks. To account for some of these considerations an agent-based model is required.

**Model 4-** An agent based model of rework cycle is the last formulation we study. This model can relax several of assumptions necessary in the first three formulations and thus can lead to more accurate simulations. A project structure similar to the previous ones is used here. However, the agent-based model requires tracking of each task, rather than stock variables for different groups of tasks with similar characteristics. For example Model 2 included only six independent stock variables. In Model 4 each task can be in one of four alternative states (Not completed; completed but not tested; to be reworked; and approved). Moreover, each task includes dynamic variable such as the number of defects in the task, as well as, static (potentially stochastic) parameters including the completion (“*TaskSize*”) or rework time (“*ReworkSize*”) for the task.

Figure 9 offers an overview of this model. Here the different states of a task (and thus rework cycle) along with important parameters of the model are presented. The state transition rates (solid arrows) are events that happen instantly based on some rules, e.g. transition from “Not completed” to “Completed but not tested” happens after “TaskSize” time has elapsed since a resource is assigned to working on the task. At the beginning of simulation all the tasks (of which there are “initialTasksToComplete”) are in the state *Not completed*. A task waits until a resource is free to work on that task (from a resource pool of size “NumResource”). For the base case analysis we assume the *TaskSize* is equal to “meanTaskSize” for all of the tasks. This concept directly relates to the productivity concept in the first three models (See Section 3-1). During the completion process defects are generated due to different reasons with a Poisson process with mean rate of “errorRateCompletion”. Completed tasks are ready for testing. Testing resources are unlimited (for simplicity and keeping in parallel with other models) thus testing finishes after “TestingTime”, which we assume to equal “meanTestTime” in the base case for all tasks. During this time each of the defects is missed with the probability of “probMissingDefect”. In the absence of defects, or if all defects are missed, the task changes to *Approved* state, otherwise, it goes to the “To be reworked” state. Here the task is reworked for “Rework Size” once assigned to an available resource (in the base case *Rework Size* is assumed to be same across all tasks, equaling “meanReworkSize”). We assumed in previous models that productivity of personnel in doing the work initially or doing rework is the same, thus  $meanTaskSize = meanReworkSize$ . Defects can be both added, and removed, during the rework process. The two Poisson processes have rates of “errorCorrectionRate” and “errorRateRework”. While generally *errorCorrectionRate* is bigger than *errorRateRework* (or otherwise the project is expected to finish only in infinity), by chance some tasks may end up with more defects at the end of rework process than how they started. After rework tasks are back into *Completed but not tested* state. Resources are shared for completion and rework stages and a first-come first-served allocation policy is put into effect.

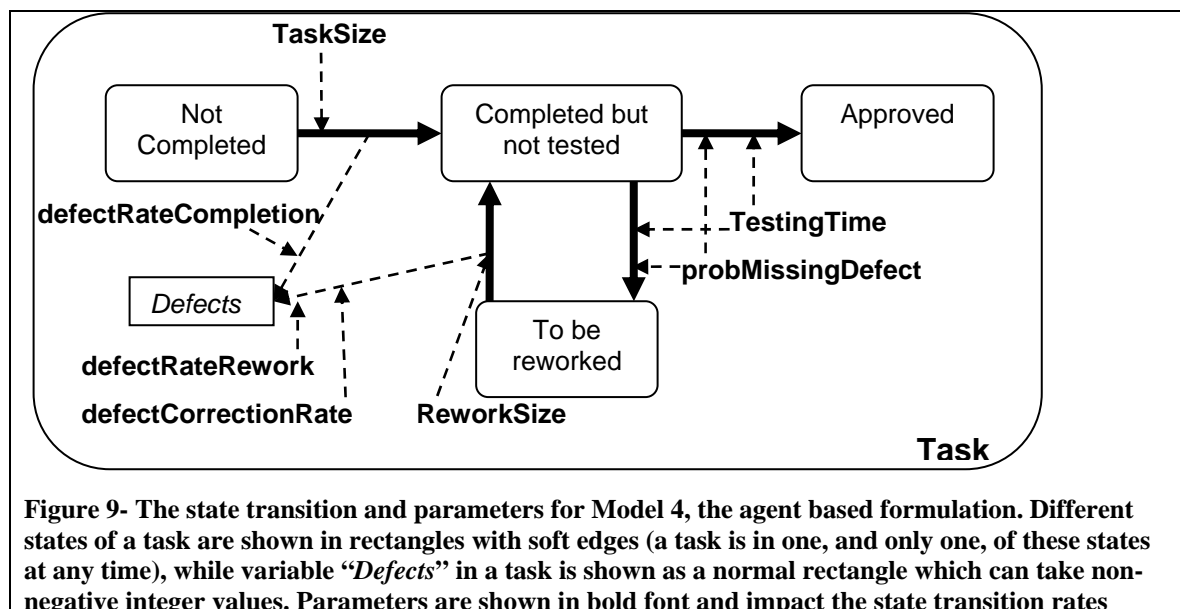


Figure 9- The state transition and parameters for Model 4, the agent based formulation. Different states of a task are shown in rectangles with soft edges (a task is in one, and only one, of these states at any time), while variable “Defects” in a task is shown as a normal rectangle which can take non-negative integer values. Parameters are shown in bold font and impact the state transition rates

(solid lines) or changes in variables. The same structure is replicated for every task in the project, i.e. a total of “*initialTasksToComplete*”. The work on completion and rework activities is conducted by resource agents, of which “*NumResource*” exist.

The complexity of Model 4 relates directly to the number of tasks tracked. Each additional task requires keeping track of its active state and number of defects and planning its related events, linearly increasing the model complexity with project size (and resources). Model 4 includes 9 parameters that govern its performance. In the next section we discuss how we derive the parameters for other models from these nine parameters. Before that however, we summarize the main characteristics and capabilities of the four models discussed above. Table 1 provides this comparison. For each model we discuss the level of complexity of the model and what it adds, in terms of capability, to the previous model. We limit this summary to the capabilities of the models discussed here and not their potential capability which may be achieved by altering these structures.

**Table 1- The overall comparison of complexity and capabilities between the different formulations discussed in this paper. For each formulation its complexity (in terms of number of stocks or state variables) and the capabilities included in that formulation are discussed. The capabilities are cumulative, so each model includes what is available in the previous models, plus the items discussed in its row.**

<b>Model</b>	<b>Complexity</b>	<b>Capabilities (cumulative)</b>
M1 (Richardson & Pugh)	2 independent stocks	<ul style="list-style-type: none"> <li>• Basic rework cycle</li> <li>• Can include feedbacks to productivity, quality, and resources.</li> <li>• Captures a proxy of quality</li> </ul>
M2 (Ford & Sterman)	6 independent stocks	<ul style="list-style-type: none"> <li>• Captures the testing process, includes testing precision</li> <li>• Allows for different rework and initial work sizes</li> <li>• Calculates quality directly</li> </ul>
M3 (Rahmandad)	7 independent stocks	<ul style="list-style-type: none"> <li>• Can include multiple defects per task</li> <li>• Can include different test coverage levels</li> <li>• Can include different sizes for tests</li> </ul>
M4 (Agent-based)	Linearly increases with number of tasks	<ul style="list-style-type: none"> <li>• Includes task heterogeneity</li> <li>• Includes stochastic behavior in completion and rework</li> <li>• Can include different priorities for tasks or precedence relationships</li> <li>• Can include interactions in task quality (e.g. a defect in task 3 increases the probability of defect in task 4)</li> </ul>

All models are available in supplementary material for independent simulation and analysis<sup>8</sup>.

### **3-1- Parameter Relationships**

In this section we derive the parameters of the M1-M3 based on the micro level parameters we defined for M4. The derivation of parameters allows us to have a better understanding of the relationships between different formulations. It also enables building on parameter values from previous work to inform new models with alternative formulations.

<sup>8</sup> All models are implemented both in Vensim™ and Anylogic™. Vensim provides better readability for M1-M3, while Anylogic offers more robust implementation and simulation platform for M4. All reported simulations are completed in Anylogic. We supply both of these models for interested researchers.

Several concepts remain unchanged across all models. Namely *Initial Tasks to Complete*, *Available Resource*, *Testing Time*, *Probability of Missing a Defect*, and *Error Correction Rate* are conceptually equivalent across M1-M3 to their counter parts in the agent based model (See Table 2). Many of these concepts are shared, without any change, across all models (thus no suffix included in parameter names). *Productivity* in M3 (and other models) is the number of tasks a resource can complete in one unit of time (e.g. month). This is therefore the reverse of the number of time units it takes to finish one task (or *meanTaskSize* in M4). Similarly rework productivity would be the reverse of *meanReworkSize*. In this study we assumed the productivity for rework and completion are equal, and thus the mean sizes for a task and its rework are equal (in expectation, since M4 can have a distribution for task and rework sizes). Minimum time to do a task or to do rework also equals the *meanTaskSize* and *meanReworkSize* parameters because those parameters determine how long a single task will take to finish, which is the minimum time needed to finish the tasks.

Defect rates in rework and completion for M3 count the number of defects introduced per task completed (or reworked). Therefore they should take into account both the average defects created per unit of time (*defectRateCompletion/Rework*) and the length of time spent on each task (*meanTaskSize* or *meanReworkSize*). Similarly “*Defect Correction Rate M3*” is equal to *defectCorrectionRate* in M4 multiplied by the *meanReworkSize*. Finally, we introduced the parameter “*Task Covered by Test UM3*” which is not used in any other model, because the other models do not include the possibility of having different types of tests with different numbers of tasks covered in each test. Such considerations are possible to implement in Model 4, however, for simplicity we do not include them and assume that a single task is covered by each test, making all models compatible. We also assume all the tasks are included in the testing process (complete test coverage), thus “*Tests to Run UM3*” are initially equal to the number of tasks.

Models 1 and 2 have several parameters similar to Model 3, which are similarly derived from Model 4. The main conceptual difference between models 2 and 3 is the different formulation and understanding of defect (or change). In model 2 tasks can either be defective (have a corresponding change in the co-flow) or not, a binary choice. Where as in Models 3 and 4 tasks can be defective to different degrees, because they can have multiple defects. This difference in conceptualization requires us to re-define the probability of introducing changes in models 1 and 2, as well as the defect correction and discovery processes. Let us first consider “*Completion Defect Probability M2*”. This parameter specifies the probability a completed task is defective (in the binary conceptualization of task quality). We can relate this probability to the conceptualization in the Models 3 and 4 by finding what the probability that a task has no defect is, given the distribution of defects per task. That probability, calculated as one minus the probability of having no defects in a Poisson distribution, is<sup>9</sup>:

$$\text{Completion Defect Probability M2} = 1 - e^{-(\text{Defect rate completion M3})} \quad (4)$$

Similarly, *Probability Generating Changes M1* can be calculated as:

$$\text{Probability Generating Changes M1} = 1 - e^{-(\text{Defect rate completion M3} * (1 - \text{Probability of Missing a Defect}))} \quad (5)$$

The same consideration comes into play for calculating the *Defect Correction Probability M2*. Here, however, both fixing of defective tasks and making mistakes that can make the task further defective come into play. We estimate this concept based on parallel parameters in M3 and M4 by assuming that a defective task is corrected if a) At least one defect is fixed in that task (one minus probability of fixing no error), AND b) No error is added to the task. This formulation is only an approximation. In fact the different concepts of task and error are not completely

<sup>9</sup> For a Poisson distribution with mean M defects per task, the probability that no defect exists is  $e^{-M}$ .

consistent across Models 3 and 2, and thus a one to one relationship does not exist between the two models. While in the Model 3 (and 4) a project can accumulate more defects in rework if “*Defect Rate in Rwrk M3*” exceeds “*Defect Correction Rate M3*”, Model 2 does not include such possibility. If tasks are always either correct or incorrect, we can not make them more incorrect, and therefore the rework cycle ultimately finishes.

In light of the conceptual difference between Models 2 and 3, one of the equations in Model 2 requires further elaboration. As Figure 5 shows, “*Discover Changes Fraction M2*” is a function of “*Probability of missing a defect*” and “*Probability change required M2*”. A seemingly straight forward, but incorrect, formulation includes multiplying “*Probability change required M2*” by one minus “*Probability of missing a defect*” (call  $PM=1-\text{Probability of missing a defect}$ ). However, note that the testing process misses one of the (possibility many) defects with the latter probability, but needs to catch one of defects to reject the task. If tasks are either correct or incorrect (binary view of M2), then probability of missing a defective task is lower than the same concept for a “defect” among many that can spoil a task in Model 3. Therefore we first find what the expected number of defects embedded in a defective task (call this  $ED$ ) is, based on the “*Probability change required M2*” (shorthand:  $PC$ ). We then find what the probability of rejecting a task ( $DC$ ) is, given the expected number of defects in that task ( $ED$ ):

$$PC=1-\text{Probability No Error Present}=1-e^{-ED} \text{ (from the Poisson distribution)}$$

$$\rightarrow ED=-\ln(1-PC) \quad (6)$$

$$DC=1-\text{Probability No Error Detected}=1-e^{-(ED.PM)}$$

$$\rightarrow DC=1-e^{-(\ln(1-PC)PM)} \quad (7)$$

Note that  $PC$  is a dynamic variable and therefore  $DC$  can also change dynamically in a simulation. We use this latter formulation instead of the original formulation (Ford and Sterman 1998) to reduce the discrepancy between M2 and M3 performances due to different conceptions of task and defect. The original formulation is appropriate when a binary conception of defective tasks is better fitting the problem at hand.

Finally, Model 1 does not include several of the parameters used in Models 2-4 because it lacks formulations for testing, or separate stocks for rework and new work. The only parameter introduced by Model 1 is “*Time to Discover Rework M1*” which represents the average time for discovery of an unknown error. Conceptually, this parameter captures elements from both the testing process, and the chances of accepting a task given its quality. If testing is very rapid, undiscovered rework will be relatively quickly discovered. On the other hand, if testing is poorly done, no matter the speed, the undiscovered rework will remain high. We can not capture the exact parallel concept in M1 of the testing and rework in M2-M4. However, as an approximation, we attempt to capture both effects by scaling “*Testing Time*” based on the fraction of tested tasks (in M3) that are cycled back into rework process (rather than being accepted). If this fraction is low, little rework can be discovered in model one, thus *Time to Discover Rework M1* will be higher, if that fraction is high, then rework can be discovered (faster) and thus *Time to Discover Rework M1* is larger:

$$\text{Time to Discover Rework M1} = \text{Testing time} / (1 - e^{-(\text{Defect rate completion M3} * (1 - \text{Probability of missing a defect}))}) \quad (8)$$

Table 2 summarizes the relationship between different parameters across different models.

**Table 2- The summary of parameters in different models and their relationships. For models 1-3 parameter values in terms of Model 4 parameters is included. Detailed discussion is available in the text.**

<b>Model 4</b>	<b>Model 3</b>	<b>Model 2</b>	<b>Model 1</b>
<i>initialTasksToComplete</i>	<i>Initial Tasks to Complete= initialTasksToComplete</i>	<i>Initial Tasks to Complete= initialTasksToComplete</i>	<i>Initial Tasks to Complete= initialTasksToComplete</i>
<i>numResource</i>	<i>Available Resource= numResource</i>	<i>Available Resource= numResource</i>	<i>Available Resource= numResource</i>
<i>meanTaskSize</i>	<i>Productivity=1/meanTaskSize</i>	<i>Productivity=1/meanTaskSize</i>	<i>Productivity= 1/meanTaskSize</i>
	<i>Min Time to Do Task= meanTaskSize</i>	<i>Min Time to Do Task= meanTaskSize</i>	<i>Min Time to Do Task= meanTaskSize</i>
<i>defectRateCompletion</i>	<i>Defect rate Completion M3= defectRateCompletion* meanTaskSize</i>	<i>Completion Defect Probability M2= 1-e<sup>-</sup>(Defect rate completion M3)</i>	<i>Probability Generating Changes M1= 1-e<sup>-</sup>(Defect rate completion M3*(1-Probability of Missing a Defect))</i>
<i>meanTestTime</i>	<i>Testing Time= meanTestTime</i>	<i>Testing Time= meanTestTime</i>	NA
NA	NA	NA	<i>Time to Discover Rework M1= Testing time/(1-e<sup>-</sup>(Defect rate completion M3*(1-Probability of missing a defect)))</i>
<i>probMissingDefect</i>	<i>Probability of missing a defect= probMissingDefect</i>	<i>Probability of missing a defect= probMissingDefect</i>	NA
<b>Each test is assumed to cover a single task (no parameter)</b>	<i>Task Covered by Test UM3</i>	<b>Each test is assumed to cover a single task</b>	<b>No explicit testing</b>
<i>meanReworkSize</i>	<i>Rework productivity assumed to be equal to completion productivity (=1/meanTaskSize)</i>		
NA	<i>Min Time to Do Rework= meanReworkSize= meanTaskSize</i>	<i>Min Time to Do Rework= meanReworkSize= meanTaskSize</i>	<i>Min Time to Do Rework= meanReworkSize= meanTaskSize</i>
<i>errorCorrectionRate</i>	<i>Defect Correction Rate M3= defectCorrectionRate* meanReworkSize</i>	<i>Defect Correction Probability M2= (1-e<sup>-</sup>(Defect Correction Rate M3))*e<sup>-</sup>(Error Rate in Rwrk M3)</i>	NA
<i>errorRateRework</i>	<i>Defect Rate in Rwrk M3= defectRateRework* meanTaskSize</i>		NA

#### 4- Simulation Analysis

In this section we report the simulation results comparing the four alternative models of rework cycle. We will first report the results of the base case analysis in which the models are simulated using the parameter values reported in Table 3. Given the stochastic nature of M4, 100 different simulations with different streams of random numbers are generated in this case and the statistics from this sample is reported. We will then conduct a comprehensive sensitivity analysis to understand how the alternative models compare under different parameter settings and what underlying structures create their differences.

**Table 3- Parameters values used in the analysis for the base case and sensitivity analysis, with their initial, low and high values, units and description. Low and high values are used in the sensitivity analysis.**

Parameter	Base (Low,	Unit	Description
-----------	------------	------	-------------

	<b>High)</b>		
<i>probMissingDefect</i>	0.3 (0.1, 0.9)	Dmnl	This parameter represents the quality of testing, as captured by the probability of missing during the testing process an error that exists in the task.
<i>meanTaskSize</i>	0.2 (0.1, 0.4)	Month.Person/Task	This parameter defines the average size of tasks, as well as, the productivity of initial work and rework of the workers in M1-M3.
<i>meanTestTime</i>	0.1 (0.05, 0.2)	Month.Person/Task	This parameter defines the average time for doing the testing.
<i>defectRateCompletion</i>	2 (0.5, 8)	Defect/Month	The rate of creation of defects in initial work
<i>errorRateRework</i>	2 (0.5, 3.5)	Defect/Month	The rate of creation of defects while doing rework
<i>errorCorrectionRate</i>	4 (2.5, 8)	Defect/Month	The rate of correction of defects in the rework
<i>initialTasksToComplete</i>	100 (20, 400)	Task	Total number of tasks included in the project
<b>Distribution of Testing Time (M4)</b>	<i>Fixed, Exponential(1)</i>	The base case includes no distribution, exponential distribution is used for the sensitivity analysis, where it is multiplied by mean value of the parameter	
<b>Distribution of Task Size (M4)</b>	<i>Fixed, Exponential(1)</i>	The base case includes no distribution, exponential distribution is used for the sensitivity analysis, where it is multiplied by mean value of the parameter	
<b>Distribution of Rework Size (M4)</b>	<i>Fixed, Exponential(1)</i>	The base case includes no distribution, exponential distribution is used for the sensitivity analysis, where it is multiplied by mean value of the parameter	
<i>numResource</i>	5	Person	No sensitivity analysis on the number of people

#### 4-1- Base case

In the base case we investigate a project with 100 tasks which is assigned to five resources who can each complete five tasks per month. In absence of any errors, therefore, this project could be finished in four months. However, task completion is accompanied by introduction of defects, which can activate the rework cycle and lead to longer times for completion of the project. In fact models from M1 to M4 take exceedingly longer times to finish the project under these settings (see Table 4). Whereas M1 is finished in 6.27 months, M4 takes in average 8.91 months to finish, and includes a relatively high standard deviation in this measure (2.6 months). Overall, the behavior modes are familiar: a smooth increase in the tasks approved, slowed down towards the end by the last iterations inside the rework cycle (and/or the maximum finish rate feasible). Figure 10 presents the base case simulations for M1-M3 and a single random simulation for M4.

As Figure 10 and Table 4 show, M1-M3 are close in their base case behavior. M3 tends to better resemble the M4, specially in its longer finish time. The reason for significantly longer finish times for M3 and M4 is in the conception of errors in these models where a few tasks include multiple defects. These remain in the rework cycle and go through multiple cycles of rework before they are fully corrected, or incorrectly accepted. M2, with its binary conception of defective tasks, does not allow for such outliers.

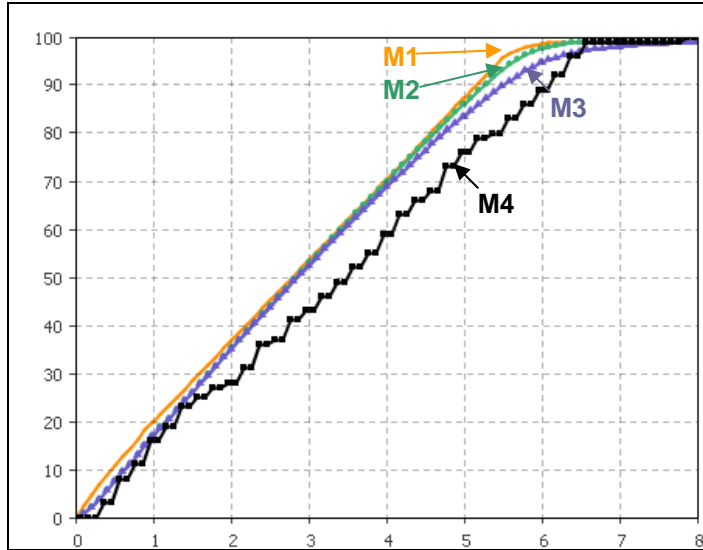


Figure 10- The time line of task approval for M1-M4. A single random run is shown for M4. The finish time for different formulations in this case are: M1:6.27, M2: 6.47, M3: 8.04, M4: 6.51.

In terms of final quality, the average defect in approved tasks can be calculated directly for M4 and M3, in terms of defects per approved task. For M1 and M2 a transformation is needed because it conceptualizes defect a binary fashion. Here the *Error Probability in Approved Tasks* is telling us what fraction of tasks is defective, but not the average number of defects in finished/approved tasks. The latter quantity can however be calculated. Given Poisson distribution for the number of errors, the probability that a task is not defective can be calculated based on the density of defects in the tasks:  $Probability\ of\ Having\ No\ Defect = 1 - e^{-Ave\ Dfct\ in\ Approved\ Task}$ , thus:

$$Ave\ Dfct\ in\ Approved\ Task = -\ln(1 - Error\ Probability\ in\ Approved\ Task) \text{ for M1 and M2} \quad (8)$$

Having calculated this metric parallel to the similar concepts in M3 and M4, we can now compare the quality of resulting projects across different models. The second row in Table 4 reports these results. In this case, somewhat surprisingly, M2 does a better job in replicating the results of the agent based model than M3 and M1. Model 3 under-counts the defect density in the approved work (0.12 in M3 vs. 0.16 in M2 and 0.18 defects per task in M4). M1's performance is quite poor, as with the current parameter settings it shows a very small number of remaining defective tasks (undiscovered errors) at the end of project. This fraction depends on *Time to Discover Rework*, which as we discussed, can not be fully derived from other models.

The reason behind M3's underestimation of defects is that M3, based on perfect mixing assumption, takes the distribution of errors to remain Poisson in the tasks cycling through rework process. In other words, it assumes the errors in the tasks can freely move around and get re-distributed among all of the tasks in a stock. This assumption is in fact consequential in this case. Tasks with no errors are accepted in the testing process and go through without any problem. The remaining tasks therefore do not include any task with no defect, whereas M3 assumes many of these tasks are indeed flawless (by imposing the Poisson distribution). Given the same average number of defects per tasks cycled through rework process, M3 will include more tasks with no defects and thus (balancing the average defects per task) more with multiple defects, compared to M4. In short, M3 increases the heterogeneity in the number of defects per task by assuming a Poisson distribution. The increased heterogeneity leads to a lower number of problems that pass through the testing process in the next round of testing. This is because the tasks assumed

flawless have no defect as they pass through, while the more defective tasks are likely not to pass the testing.

An example illustrates this mechanism better. Consider two sets of two tasks with the one average defects per task. First set includes a task with no defects and a task with two defects (higher heterogeneity, more similar to M3). Second set includes two tasks with one defect each (lower heterogeneity, more similar to M4). If these two sets go through a testing process with probability of  $p$  for missing a defect, then first set will have an expected number of errors equal to  $p^2$ , while this number is equal to  $2p$  for the second set. For all feasible values of  $p$  (between 0 and 1), the defect density will be lower for the first set, which resembles M3.

This comparison also suggests the main reason why M2 does better in capturing the quality metric: the binary definition of task correctness in M2 makes a clear distinction between tasks with no defects and those defective, and follows that distinction throughout the testing and acceptance processes. It therefore does not suffer from the problem discussed for the M3.

**Table 4- Values of Finish Time and Average Defect in Approved Tasks for all 4 models in the Base case. Results for 100 simulations are summarized for M4.**

	<b>M1</b>	<b>M2</b>	<b>M3</b>	<b>M4 (Stdev)</b>
<b>Finish Time</b>	6.269	6.467	8.044	8.909 (2.601)
<b>Average Defect in Approved Tasks</b>	0.015	0.16	0.12	0.18 (0.04)

The differences in the time evolution between the four models are reported in the first row of Figure 11. This graph reports the average absolute error (AAE) metric we defined before (Figure 3) for comparison of different models (six different comparisons). Overall, M2 and M3 have the closest trajectory over time. After that M1 and M3 are most similar, followed by M1-M2, M3-M4, M1-M4 and M2-M4. These results are consistent with the process of building the models, where M1 was the simplest, M2 and M3 followed, and M4 was the most detailed model. The differences between the models are not so significant and range between 1% (M2-M3) and 6% (M2-M4), even though Finish Times vary much more (over 40% from M1 to M4). In selecting among different aggregate formulations, M1 is the most aggregate, and the least precise model. It is most useful if a quick and simple insight model is needed to capture the basic idea of rework cycle. Selecting between M2 and M3 is less clear cut. While M2 offers more accurate quality estimates of the final project, M3 provides a more accurate evolutionary pathway and project finish time.

The standard deviations for M4 metrics are relatively large e.g. 20-30 percent of the mean values. This suggests that variation in project evolution created through rework cycle, even in the absence of any other feedback structure, could still be significant. Given that spread in expected project performance may be as important as the expected performance itself, having more detailed models may be warranted for both practical problems that include risk assessment as part of the problem definition, and for theoretical studies that focus on heterogeneity in performance.

#### **4-2- Model comparison under alternative project settings**

Different projects may differ significantly in their structure and parameter ranges. We therefore conduct sensitivity analysis to understand the scope and robustness of initial results and the level of difference between different models under alternative project specifications. Moreover, such sensitivity analysis informs the mechanisms that lead to differences and

similarities across different models and thus to a better understanding of underlying structural factors a modeler should take into account for deciding on the best formulation for the problem at hand. We change the model parameters from the base case values as reported in Table 3. Parameters are varied on a large interval to see how the models compare under a wide range of assumptions.

We also test the impact of including a reinforcing loop that is typically raised as an important contributor to problematic project performance, i.e. the reinforcing loop created from the impact of schedule pressure on error rate. Increased schedule pressure is often observed to lead to increased error rate, thus reducing the net task approval rate and further contributing to increased pressure. This dynamic, widely recognized and analyzed in system dynamics models (e.g. AbdelHamid and Madnick 1991; Eden, Williams et al. 2000; Taylor and Ford 2006), may also play an important role in distinguishing M4 from other models. One may hypothesize that this reinforcing loop could be activated for some instances of M4, but not for the other luckier projects, which by chance face limited schedule pressure early on. As a result the loop may derive a wedge between different M4 simulations and increase the heterogeneity of results, thus further highlighting the importance of including random variations. We model schedule pressure as the ratio of expected time remaining from a project (calculated based on remaining tasks and productivity) to scheduled remaining time (floored at a minimum of one month). Schedule pressure then impacts both completion and rework error rates. The table function relating schedule pressure to error rate is shown in Figure 11, last row.

In each sensitivity analysis case we repeat 100 simulations for M4 and calculate the performance metrics similar to how we conducted the base case analysis. We also measure if M1, M2, and M3 behave relatively similar to M4. We capture this by reporting not only the standard deviation of M4, but also whether M1-M3 fall in the confidence bands of metrics from M4 simulations. The detailed results of this analysis are reported in Table 5 (Final time and Quality) and Figure 11 (Average Absolute Error). The following conclusions emerge as we compare the alternative formulations under different project parameterizations.

Increasing (decreasing) probability of missing an error decreases (increases) the finish time and has the reverse effect on average defect in accepted work. By reducing project finish time and the stochasticity induced by the rework cycle, poor testing procedure also brings the four models closer to each other. In fact in the extreme when no testing is done and every task is accepted, rework cycle is no more active, and thus it can not introduce randomness into M4. Consequently as the quality of testing process improves, the additional value of M4 increases in terms of capturing stochastic behavior not included in M1-M3. An improved testing process also increases the relative value of M3 over M2 because it leads to longer projects and higher quality, where as with poor testing, the two models become almost identical.

Besides the obvious impact on project length, increasing the task size has the effect of increasing the defect hazard for each task, leading to more significant impact of rework cycle (e.g. higher standard deviation for M4). Moreover, M3's quality metric further diverges from M4 given the higher average numbers of defects per task. Finally, M1 includes no explicit testing, thus all undiscovered error is cycled into undiscovered rework. Higher error rate thus creates more rework for M1 because it does not accept defective pieces by chance. Furthermore, given the formulation of *Time to Discover Rework* in M1, which depends on the average number of defects per task, most defects are quickly discovered with higher task size, not allowing the project to be finished earlier with many undiscovered errors. These two effects join to push the finish time of M1 closer to M4, while the quality further diverges.

Increasing testing time increases the length of the rework cycle and thus its impact on the stochastic behavior of M4, which now takes longer in average with higher uncertainty. Limited sensitivity of M1 to this parameter (given that M1 includes no explicit testing process) leads to further departure of M1 from other models when testing time is long. The other comparative results are not changed significantly by testing time.

Completion defect rate has a significant impact on the behavior of the models. Increasing this parameter activates rework cycle significantly and leads to longer project times, increased discrepancy of M4 and M2, and increased quality difference between M4 and M3. Interestingly, the M3 remains very close in its evolutionary time line to M4, despite the fact that it underestimates the defects that pass through the testing process and are in approved work. Similar to the case with larger task size, M1 tends to finish much later with higher error rates, though its quality metric becomes deceptively optimistic.

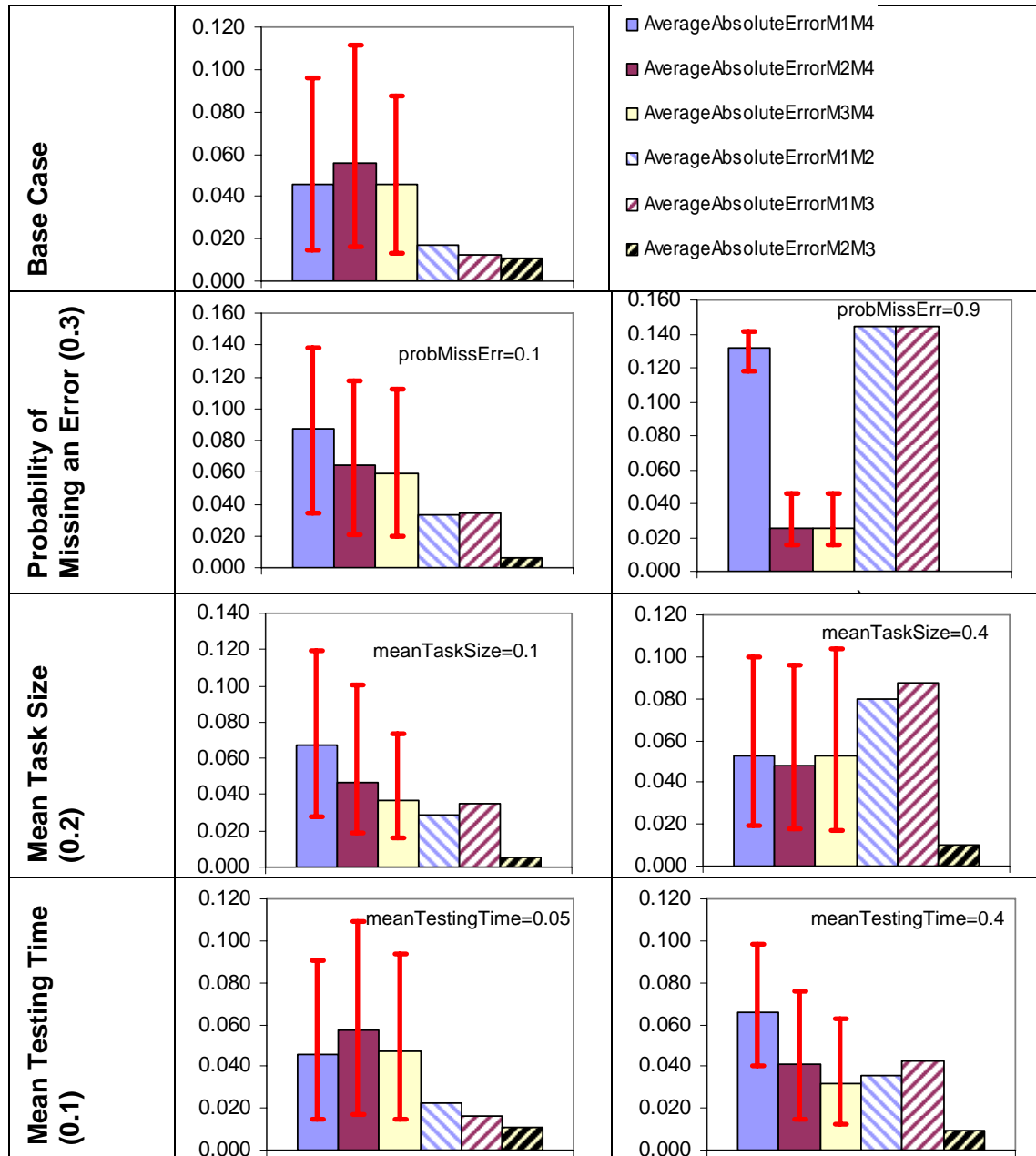
The impact of rework defect rate is somewhat more subtle. It has very limited impact on M2 and no impact on M1. Yet an increase in this parameter significantly increases the finish times for M3 and M4 because by introduction of more defects during rework these models will often include a few tasks which will accumulate multiple defects and will lengthen the project for a long time. Therefore the change in finish time is quite significant. Nevertheless, this process does not increase the gap (AAE) between M4 and M2 or M1 significantly, because for the majority of project life the models remain close to each other, and only the work on final few defective tasks are extended. By this final stage all projects are largely completed and therefore the AAE measure remains small. In fact the impact of error correction rate is also very similar to the impact of defect rate in rework, but only in the opposite direction. This should come as no surprise since both processes of rework defect introduction and defect correction impact the stock of tasks with known changes.

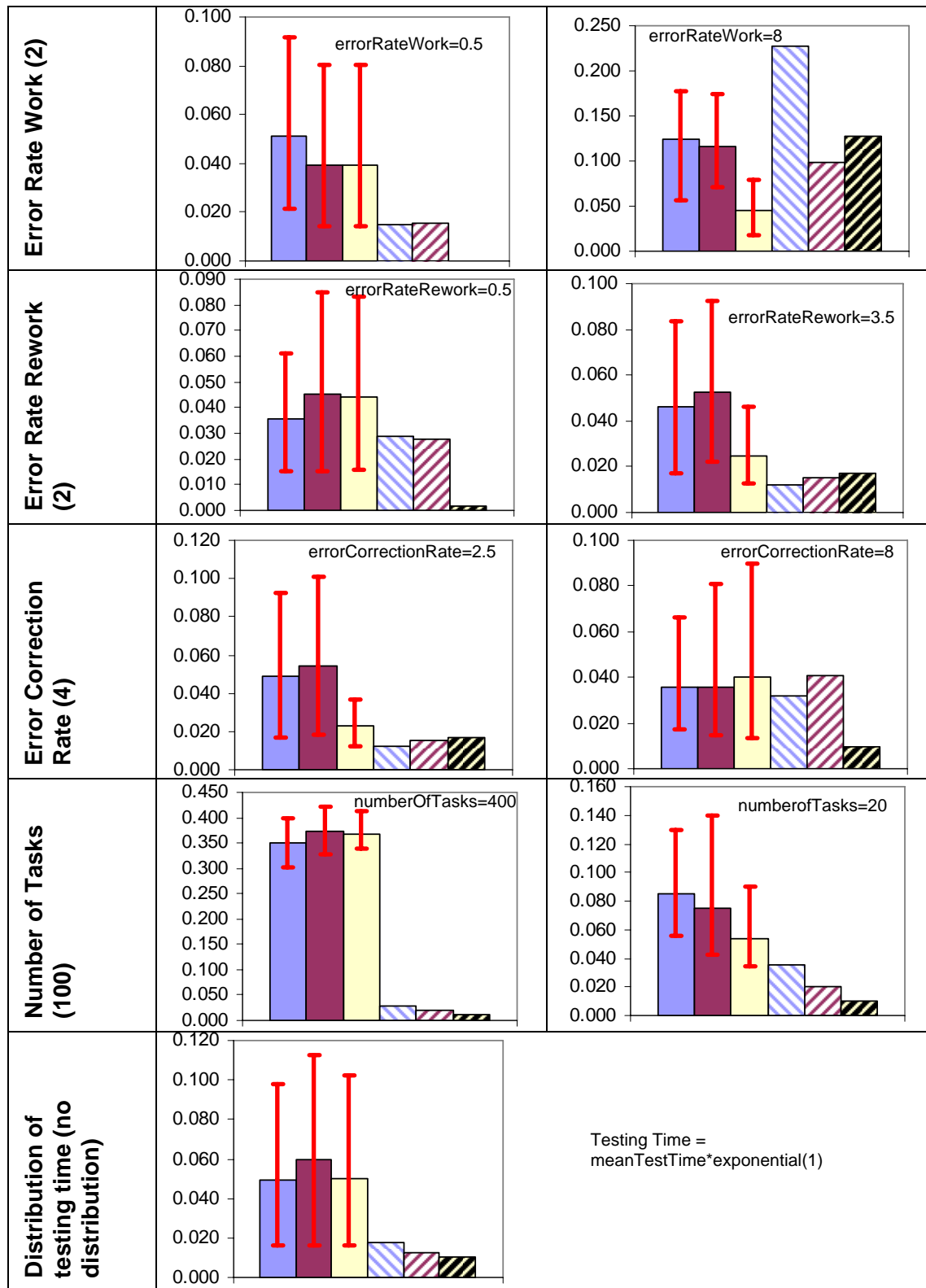
Changing the number of tasks in the project scales the projects without changing the strength of underlying processes and therefore does not impact the mean performances of the models significantly. However, it has a major impact on the variations across different simulations for M4. With increased project size, the law of large numbers washes the variations away and leads to standard deviation values not so far from base case for both finish time and average defects (in fact standard deviation for average defect reduces for larger projects). Therefore increased project size actually reduces the relative value of M4 in terms of capturing project heterogeneity.

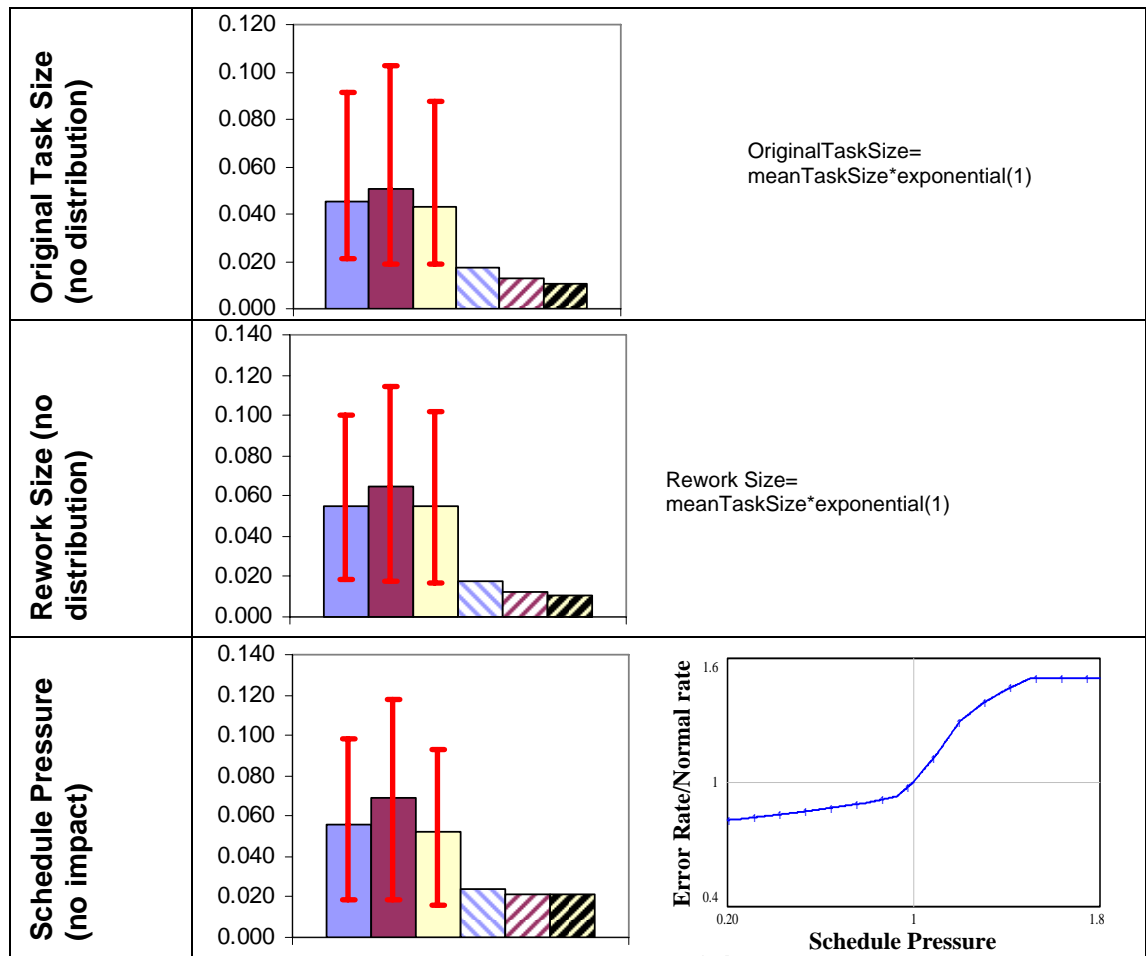
Having distributions, rather than fixed values, for testing time, task size, and rework size have very minor impacts on M4 performance. Overall, the spread in testing time, task size, and rework size can lead to a few outlier projects with longer finish time, thus a slight increase in finish time. The impact on quality is even less salient. These results are especially interesting given the high level of heterogeneity induced in parameters by the exponential probability distribution used for these experiments.

Finally, the impact of schedule pressure on error rate does not change the overall results much. All simulated projects take somewhat longer to finish, and the shift is slightly more significant in M3 and M4. This is expected as the error rate in rework, and schedule pressure's impact on that, is more precisely captured in these two models. However, contrary to our initial hypothesis, the reinforcing schedule pressure loop does not lead to a significant increase in the heterogeneity of M4 simulations. All simulations go through very similar cycles of increase and decrease in schedule pressure throughout a project life. The variations in these cycles are more a function of scheduled time left vs. expected time left; small random differences between projects

in terms of number of tasks finished have limited impact on schedule pressure patterns. Thus those random differences are not the important factor activating the reinforcing loop, and they can not induce any additional heterogeneity in project performance through the reinforcing loop of schedule pressure impact on quality.







**Figure 11- The average absolute error (AAE) for differences between different models in base case and alternative project settings. From left to right the AAEs are shown for: M1M4, M2M4, M3M4, M1M2, M1M3, M2M3. For sensitivity cases each condition's base case setting is identified in the left hand column, with the relevant parameter/impact in the base case in parenthesis. Sensitivity values are highlighted on the respective graphs. For AAE between M4 and other models (the first 3 bars on charts), 90% confidence intervals are also graphed based on empirical values from the sample of 100 simulations.**

Table 5- Detailed results of sensitivity analysis based on the change of each parameters/simulation setting listed in Table 3. Results for M4 are reported based on 100 simulations. Where M1, M2 or M3 metrics fall outside of 90/95% confidence band for simulation results for M4, they are highlighted with a \*/\*\*. The metrics for M4 are not normally distributed, so empirical confidence bands based on 100 simulations are used.

<b>Simulation Parameters</b>	<b>Finish Time M1</b>	<b>Finish Time M2</b>	<b>Finish Time M3</b>	<b>Finish Time M4, Mean (Stdev)</b>	<b>Ave Dfct in Approved Task M1</b>	<b>Ave Dfct in Approved Task M2</b>	<b>Ave Dfct in Approved Task M3</b>	<b>Ave Dfct in Approved Task M4, Mean (Stdev)</b>
<i>Base Case</i>	6.27	6.47	8.04	8.91 (2.6)	0.015**	0.16	0.12	0.18 (0.04)
<i>probMissingDefect=0.1</i>	6.25**	7.64*	8.92	11.83 (3.1)	0.011**	0.06	0.03	0.06 (0.02)
<i>probMissingDefect=0.9</i>	6.34**	4.44	4.45	4.43 (0.1)	0.130**	0.38	0.38	0.38 (0.06)
<i>meanTaskSize=0.1</i>	2.45**	3.13	4.32	4.77 (2.8)	0.056**	0.10	0.08	0.13 (0.04)
<i>meanTaskSize=0.4</i>	18.55	15.82	15.37	18.83 (3.2)	0.004**	0.27	0.14*	0.25 (0.06)
<i>meanTestTime=0.05</i>	6.22	6.30	7.64	8.52 (2.5)	0.006**	0.16	0.12**	0.19 (0.04)
<i>meanTestTime=0.4</i>	6.47**	7.90*	10.96	13.35 (4.7)	0.079**	0.16	0.11**	0.19 (0.04)
<i>defectRateCompletion=0.5</i>	4.46*	4.87	4.91	5.86 (1.8)	0.030	0.05	0.04	0.05 (0.03)
<i>defectRateCompletion=8</i>	20.80	9.94**	20.98	21.14 (4.5)	0.007**	0.36**	0.14**	0.52 (0.08)
<i>errorRateRework=0.5</i>	6.27	6.02	6.19	6.51 (0.7)	0.015**	0.14	0.10	0.16 (0.04)
<i>errorRateRework=3.5</i>	6.27**	7.01**	26.74	36.85 (46.0)	0.015**	0.19	0.12**	0.22 (0.05)
<i>errorCorrectionRate=2.5</i>	6.27**	7.08**	26.74	26.58 (21.1)	0.015**	0.19	0.12**	0.24 (0.05)
<i>errorCorrectionRate=8</i>	6.27	5.92	5.65	6.07 (0.6)	0.015**	0.14	0.09	0.13 (0.04)
<i>initialTasksToComplete=20</i>	23.44**	23.53**	26.53	29.76 (3.6)	0.008**	0.16	0.12**	0.19 (0.02)
<i>initialTasksToComplete=400</i>	2.64	2.49	4.01	3.13 (2.1)	0.015**	0.16	0.11	0.20 (0.11)
<i>Distribution of Testing Time (M4)</i>	6.27*	6.47*	8.04	9.55 (2.9)	0.015**	0.16	0.12	0.18 (0.05)
<i>Distribution of Task Size (M4)</i>	6.27*	6.47	8.04	8.92 (2.1)	0.015**	0.16	0.12	0.17 (0.05)
<i>Distribution of Rework Size (M4)</i>	6.27	6.47	8.04	9.31 (2.5)	0.015**	0.16	0.12*	0.20 (0.05)
<i>Schedule Pressure impacts Quality</i>	6.65**	6.90*	8.90	10.06 (2.3)	0.014**	0.20	0.13*	0.22 (0.05)

## 5- Discussion and conclusions

In this paper we develop and compare four alternative formulations for modeling rework cycle in projects. We show how these formulations relate to each other and how their overall behaviors compare. The following discussion summarizes our main findings and their applications for modelers. Finally we discuss some of potential avenues for further research in light of shortcomings of the current paper.

The basic mode of behavior for all models is similar. It consists of the typical growth in the number of tasks approved as resources work on tasks, until the project ends. The final stages of the project can however be significantly slower when we consider the possibility that some tasks may include more than one defect (i.e. in M3 and M4). Such tasks may cycle through rework process multiple times and for a long time. As a result M3 and M4 take longer times to finish and are closer to each other in overall fit. The difference in finish time between M4 and M2 or M1 may be of important theoretical and practical significance. Project finish time is often a very important factor in project success and profitability, as lost revenue or penalties accumulate with each additional day of delay. Moreover, previous research has attributed 90% syndrome, the lengthy continuation of projects that are almost finished, to managerial responses to schedule pressure that increase the error rate (Ford and Sterman 2003). Our results suggest that a more basic process, the iteration of a few tasks with multiple defects, may create similar symptoms, without requiring any changes in the defect rate, or involving other feedback mechanisms. The traditional formulations used in system dynamics for modeling projects hide this effect because they do not allow for multiple defects per task. Both M3 and M4 models introduced in this paper can capture these effects.

On the other hand, the measures of quality between M1, M2, and M3 show significant differences, with M2 giving the closest results to M4, and M3 and M1 to follow. Since M1 uses a proxy for quality (the ratio of undiscovered rework to tasks accepted), significant difference with other models could be expected. The mechanism that leads to higher quality measures in M3 is however more subtle. It is based on the heterogenizing effect of perfect mixing assumption in M3. In effect, the acceptance of tasks with no defect make the binary conception of defect used in M2 (tasks are either defective or not), closer to the real distribution of defects per task (in M4), than the Poisson distribution imposed by M3. Note that we have reformulated M2 from the original concept offered by Ford and Sterman (Ford and Sterman 1998), to make it consistent with the conception of tasks and defects that allow for multiple defects per task.

A detailed agent based model is also more informative in providing a range of behaviors not otherwise observable from deterministic models. Given the significant impact of extreme performances on expected costs and benefits of projects, a realistic cost-benefit analysis can benefit from the additional detail that allows for capturing stochastic variation endemic to project evolution. However, we show that the major modes of behavior and relationships between mean values of different metrics for different formulations remain largely similar across different parameter settings. Therefore basic insights and high-level conclusions can be derived from simpler, more aggregate formulations, without incurring additional cognitive and computational costs inherent to building more detailed models. In fact, for larger projects with many tasks, the stochastic effects are more likely to be washed out. Therefore, if a robust differential equation formulation is devised that captures both quality and schedule correctly, one can use it for most large projects without much loss compared to a detailed agent based model.

In this paper we assumed all tasks can be worked on, that is no precedence relationship exists among these tasks. Inclusion of precedence relationships is feasible both through introducing multiple phases for a project (in aggregate differential equation models) and by explicit inclusion in agent-based models. If such precedence relationships are elaborately connecting most of the tasks in a project, the complexity of agent-based models may not exceed the multi-staged differential equation ones required for capturing such effects.

Selection of appropriate rework cycle formulation depends both on the purpose of the modeling project and the project setting being modeled. Sensitivity analysis in this paper offers additional insights into the second consideration. In the absence of defects or with very poor testing, rework cycle is largely removed and all models behave in very similar patterns. Moreover, very effective rework process (high error correction rate) reduces the impact of rework cycle and the differences across models. As we increase the defect rate and testing precision, or reduce rework effectiveness, the rework cycle is strengthened, and the four models are more likely to diverge. The simplest model, M1, is often the furthest apart from the more realistic M4 model. The difference between M2 and M4 also increase, especially in comparison of their finish times. M3 and M4 remain close in finish time under different parameter settings, yet a persistent bias is observed in that M3 gives higher quality estimates for approved work, than M4 does. Overall we conclude that neither of M2 nor M3 dominates the other for practical purposes and both have their strength and weaknesses. If finish time is a more important metric or the relevant feedbacks cross through it in a given application, then M3 would be a better choice among the simpler formulations. If quality is more central to the problem at hand, M2 may be a better choice. For projects with a few tasks, those that need precise values for both the quality and the finish time, where performance stochasticity is part of the problem definition, or when elaborate precedence relationships between tasks are present, one may prefer to use a detailed agent based model such as M4.

Despite using the same or parallel parameters, the four models show differences in important metrics that typically range between 10 to 40 percent. Therefore where numerical precision is important, such as in model calibration to project histories, or prediction problems, selection of the most appropriate formulation remains challenging and non trivial. Where as calibration of simpler formulations (M1-M3) are more practical, field data about tasks and defect probabilities better inform a bottom-up model such as M4. A robust differential equation model that can tie those micro-level parameters to their aggregate dynamics in a simple formulation can bridge these two sources of data and model parameterization and enable more reliable analysis based on empirical data and multiple levels. An open research question is to find a simple differential equation formulation that captures both the quality and the finish time for a project more precisely than M2 or M3 do.

Many modeling projects do not aim at replicating numerical values of past projects or prediction of precise future paths. They may involve theoretical and managerial insights about the impact of alternative feedback effects on modes of behavior in projects. The four formulations in this paper provide slightly different opportunities for expansion and inclusion of such alternative feedbacks. M1 can include feedbacks to quality and productivity, as well as defect discovery rate. M2 has additional flexibility to capture feedbacks to testing quality. M3 allows for different test sizes (testing one or multiple tasks, e.g. unit testing vs. system testing) and changes in rework quality. Finally M4 is the most flexible model in terms of possible feedback and additional effects that it can capture, including precedence relationships, task priorities, heterogeneity of task size, resource productivity, and testing and rework size, among

others. Benefiting from these flexibilities in M4 however requires more detailed modeling and coding and may complicate communication of basic insights to stake holders.

In this research we focused on a central, but narrowly defined, piece of project dynamics and compared multiple formulations. Given this focus, we excluded many factors important to project dynamics such as schedule pressure, morale, experience, communication, and other feedback effects and interactions among different phases of a project, among others. Yet, we hope the controlled experiments enabled by such focus move us closer to a set of contingency rules of thumb for selection of appropriate formulations for specific project modeling problems. Depending on the structure and amount of available data, computational and modeling resources at hand, the level of quantitative precision required, and the client's main goals, we find that all four formulations could be appropriate in some settings. Future research can offer new formulations that build on this comparative analysis to address the short-comings of models M1, M2, and M3 without the need to disaggregate to the agent-based level. One can also study the effect of different precedence relationships and task priorities, as well as additional feedback effects, on the comparative performance of these and other formulations. Despite these shortcomings, we hope this research provides interested scholars with a better map of available formulations, their relationships, and most appropriate assumptions for modeling project dynamics, and thus contributes to faster accumulation of research results in this important area.

#### **References:**

- AbdelHamid, T. and S. Madnick (1991). Software project dynamics: An integrated approach. Englewood Cliffs, NJ, Prentice-Hall.
- Cooper, K. G. (1980). "Naval Ship Production: A Claim Settled and a Framework Built." Interfaces **10**(6).
- Eden, C., T. Williams, F. Ackermann and S. Howick (2000). "The role of feedback dynamics in disruption and delay on the nature of disruption and delay (D&D) in major projects." Journal of the Operational Research Society **51**(3): 291-300.
- Ford, D. N. and J. D. Sterman (1998). "Dynamic modeling of product development processes." System Dynamics Review **14**(1): 31-68.
- Ford, D. N. and J. D. Sterman (2003). "Overcoming the 90% syndrome: Iteration management in concurrent development projects." Concurrent Engineering-Research and Applications **11**(3): 177-186.
- Grimm, V., E. Revilla, U. Berger, F. Jeltsch, W. M. Mooij, S. F. Railsback, H. H. Thulke, J. Weiner, T. Wiegand and D. L. DeAngelis (2005). "Pattern-oriented modeling of agent-based complex systems: Lessons from ecology." Science **310**(5750): 987-991.
- Lee, S. and F. Pena-Mora (2007). "Understanding and managing iterative error and change cycles in construction." System Dynamics Review **23**(1): 35-60.
- Levitt, R. E., J. Thomsen, T. R. Christiansen, J. C. Kunz, Y. Jin and C. Nass (1999). "Simulating project work processes and organizations: Toward a micro-contingency theory of organizational design." Management Science **45**(11): 1479-1495.
- Lyneis, J. M. and D. N. Ford (2007). "System dynamics applied to project management: a survey, assessment, and directions for future research." System Dynamics Review **23**(2-3): 157-189.

- Parunak, H. V. D., R. R. Savit and R. L. Riolo (1998). Agent-Based modeling vs. equation based modeling: A case study and users' guide. Proceedings of Multi-agent systems and agent-based simulation, Springer.
- Rahmandad, H. (2005). Three essays on modeling dynamic organizational processes. Sloan School of Management. Cambridge, Massachusetts Institute of Technology. **Ph.D.**
- Rahmandad, H. and J. Sterman (2008). "Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models." Management Science **54**(5): 998-1014.
- Repenning, N. P. (2000). "A dynamic model of resource allocation in multi-project research and development systems." System Dynamics Review **16**(3): 173-212.
- Richardson, G. P. and A. L. Pugh (1981). Introduction to system dynamics modeling with DYNAMO. Cambridge, Mass., MIT Press.
- Sengupta, K. and T. K. Abdelhamid (1993). "Alternative Conceptions of Feedback in Dynamic Decision Environments - an Experimental Investigation." Management Science **39**(4): 411-428.
- Taylor, T. and D. N. Ford (2006). "Tipping point failure and robustness in single development projects." System Dynamics Review **22**(1): 51-71.