Tipping point failure and robustness in single development projects

Tim Taylor* and David N. Ford**

Abstract

Tipping point feedback structures can push a series of product development projects into firefighting mode in which rework overwhelms progress. Similar dynamics also threaten the performance of individual development projects. The current work extends previous tipping point dynamics research to single projects and demonstrates how a simple, common feedback structure can cause complex tipping point dynamics, trap projects in deteriorating modes of behavior, and cause projects to fail. Basic tipping point dynamics in single projects are described, demonstrated, and analyzed with a model. The concept of applying robustness to project design is preliminarily tested and system robustness to tipping point-induced failure is quantified for a simple project and analyzed with sensitivity analysis. Impacts of tipping points on project performance and future research opportunities concerning tipping point and robustness in project management are discussed. Copyright © 2006 John Wiley & Sons, Ltd.


Introduction

Although development projects are pursued to create value for their developers or users, many projects fail (Evans 2005; Matta and Ashkenas 2003; Wells 1999). Project failure can take many forms, including schedule and cost overruns and unacceptable quality. Project failure is relatively easy to identify if the final product grossly fails to meet performance targets (e.g., some of NASA’s Mars probes) or if development stops before a product is completed (e.g., the US Department of Energy’s Super collider project). But some projects that are completed should also be considered failures. An example is the Channel Tunnel (the “Chunnel”) that connects England and France. While the Chunnel is arguably one of the great engineering achievements of the last century, its final cost of $17.5 billion was more than double the original estimate of $7.2 billion (Kharbanda and Pinto 1996). Chunnel usage is below the level estimated in the project’s feasibility study and even the most optimistic estimates predict that the Chunnel will not be profitable in the next 10–20 years (Kharbanda and Pinto 1996). Although a technical marvel, the Chunnel failed...
to meet two of its fundamental goals: finish within budgeted funds and produce a financially viable product. Failure of these large projects can have dire consequences for all parties associated with the project.

Project management research has identified many factors that can lead to project failure, including overestimation of benefits (Evans 2005), poor stakeholder analysis (Paul 2005), and errors (Busby and Hughes 2004). Despite considerable research into these factors, clearly identifying project failure is difficult. Comparing differences between project performance and targets is a standard means of measuring project success or failure. But variations of final project performance from targets can be poor measures if targets are flexible. For example, U.S. Department of Energy projects are not allowed to exceed Congressionally approved cost targets. So targets are revised based on final performance, even in cases of gross cost overruns. If performance relative to original targets is a measure of project success or failure, some Department of Energy projects that meet final targets should be considered failures (USGAO 1996, 1997). Some organizations explicitly label such projects as failures. For example, as part of development improvement efforts one organization known to the authors labeled a set of completed projects that exceeded their cost or schedule targets by 20 percent or more as “wrecks” (as in “train wrecks”). A clear, inclusive definition of project failure is needed to study the performance of projects. Changes over time in the work remaining to be completed can provide an improved metric. Although these project backlogs are intended generally to decrease over time, they can stagnate or grow. Projects with backlogs that increase continuously over significant periods of time ultimately lead to failures to meet original project targets and may be terminated. The current work defines a project as a failure when its backlog grows continuously over an extended period of time.¹

The continuous growth of project backlogs over time can be attributed to many different dynamic factors. Dynamic causes identified through system dynamics include a lack of knowledge transfer between projects (Cooper et al. 2002), rework (Cooper 1993a, 1993b, 1993c) and concealing rework (Ford and Sterman 2003b), schedule pressure (Cooper 1994, Ford and Sterman 2003a), and “fire-fighting” (Repenning 2001). A complete dynamic hypothesis of development project failure would include unrealistic performance targets and how negative feedback loops that describe responses to schedule, budget, and other pressures can trigger fatal reinforcing loops through productivity losses, overstaffing, inadequate training, and other project behaviors. Other exogenous changes that slow progress, degrade performance, and can lead to failure (e.g., increased regulation, scope changes, temporary work stoppage) would provide the bases for additional hypotheses. The dynamic structure would also include the amplification of impacts due to delays in discovering rework that allow problems to be passed among development phases. These and other causes of project failure have been used in system dynamics practice and several have been addressed in the literature.
The current work focuses on how a particular dynamic structure, tipping points, can cause a common project feature, ripple effects, to generate rework and project failure. A development project's ripple effects are the secondary or tertiary impacts of a change. Thomas and Napolitan (1994) identify indirect changes due to ripple effects in construction projects as an important cause of project failure. They estimate the impacts on labor efficiency in some projects to be seven times larger than the impacts of direct changes (p. 26). Ripple effects can be triggered by many unplanned events or conditions, including the exogenous factors described above. Likewise, ripple effects can have multiple types of impacts, including creating more work, requiring rework in previously correct work, and reducing productivity. We focus on the work effort created by ripple effects and disaggregate that effort into two forms: contamination and adding new tasks. Contamination is work required in part of the existing project scope that is created due to rework being discovered in a different portion of the project. For example, if, after a reinforced concrete column was poured the inspectors discovered that the reinforcing steel used was too small, part of the beams above and below the column might have to be demolished in order to replace the column. Replacing the column (rework) requires reworking the beams even though the beams were not otherwise defective. The column rework contaminated the adjacent beams, but did not add any new activities to the project. In contrast to contamination, adding new tasks, as used here, creates development activities beyond the project scope due to rework required on portions of the existing project scope. In the column example temporary shoring required to support the upper beams while the column is replaced would be new tasks. Rework on previously created new tasks can also contaminate and add more new tasks. For example, inadequate temporary shoring of the beams in the column example could damage adjacent floors (contamination) and require more shoring for floor repairs (more new tasks). The critical difference between contamination and adding new tasks is that contamination creates more rework within the existing project scope or previously added tasks, while adding new tasks creates development activities that were not previously a part of the project. The current research focuses on adding new tasks because it can be difficult to identify during the course of a project when created by rework and, as will be shown, can cause challenging project behavior and failure.

Tipping points are one explanation of bifurcated system behavior such as project backlogs that diminish and lead to success or grow and lead to failure. A tipping point is a threshold condition that, when crossed, shifts the dominance of the feedback loops that control a process (Sterman 2000). Systems tend to remain stable as long as conditions remain “below” the tipping point and controlling feedback is dominant (Sterman 2000, p. 306). But when conditions cross the tipping point behavior can become (temporarily) unstable and, in the case of projects, lead to failure. Social physiologists have used tipping points to describe an unexpected spread of disease, a dramatic change
in the crime rate in a city, and an increase in the number of teenage smokers despite a campaign of increased awareness (Gladwell 2000). System dynamics can be used to elucidate tipping points and their impacts on systems in several ways: (1) by specifying, formalizing, and explaining structures that create tipping points; (2) by describing behaviors resulting from tipping points; and (3) by developing policies for managing systems with tipping points. Here we investigate whether a combination of a tipping point structure and ripple effects can explain the failure of some large complex development projects.

The current work addresses this question by examining the generation of tipping point dynamics due to adding new tasks in single development projects. More specifically, we use a tipping point structure to model projects that initially make progress but unexpectedly change their behavior for the worse and eventually fail. In doing so, we present tipping point dynamics as a new, potentially important factor in explaining the failure of some large development projects. The model provides a means of describing a project’s robustness to failure by measuring the distance of project conditions from the tipping point. Challenges posed by tipping points in single projects are discussed next. Then a model of a single product development project is described and used to generate complex tipping point failure modes, followed by model analysis and a measure of project robustness. The conclusions discuss implications and research opportunities.

**Project management challenges near tipping points**

Complex development projects are difficult to manage partially because of the dynamic nature of project systems (Lyneis et al. 2001). Iterative processes, resources, and management interact to generate puzzling behaviors. For example progress can slow, stop, and even turn negative for no apparent reason. The behavior modes of increasing and then decreasing percent of work complete and increasing project backlog may be characteristic of projects that add many new tasks in a tipping point structure. As will be shown, managers can lose control of the addition of new tasks and their projects if projects move beyond tipping point conditions.

Multi-project system behavior and management has been described using tipping points. Repenning (2001) and Black and Repenning (2001) investigated fire-fighting in an overlapping series of product development projects. Repenning defined fire-fighting as “the unplanned allocation of engineers and other resources to fix problems discovered late in a project’s development cycle.” Tipping point conditions were an unstable equilibrium where small shocks pushed the system toward one of two stable equilibriums. Similar structures and conditions may drive some individual development projects. Many development projects are managed largely in isolation from other projects.
and can fail due to dynamics solely within or near the project. Therefore, the
explanation of tipping point impacts on project performance needs to be
expanded to single project design and management.

The current work expands an existing generic project model to include
tipping point structures. In this way the work extends the multi-project work
by Repenning (2001) and Black and Repenning (2001) to single development
projects. After calibration the model is used to test whether tipping point
structures can cause project failure modes like those experienced by the nu-
clear power plant construction industry. The model is then used to initially
test the potential of robustness as a measure of project vulnerability to tipping
point failure.

A model of project tipping point dynamics

Most traditional project management models, such as the critical path method,
are linear and cannot adequately predict the effects that increased rework, con-
tamination, and the addition of new tasks have on projects. In contrast,
system dynamics is well suited to modeling development dynamics. System
dynamics has a strong and established history of modeling development projects
and has been successfully applied to a variety of project management issues,
including failures in fast track implementation (Ford and Sterman 1998), poor
schedule performance (Abdel-Hamid 1984), and the impacts on project per-
formance of changes (Rodrigues and Williams 1997; Cooper 1980, 1993a,
1993b, 1993c) and concealing rework requirements (Ford and Sterman 2003a).

The model is purposefully simple relative to actual practice to expose the
relationships between tipping point structures, project behavior modes, and
management. Therefore, although many development processes and the fea-
tures of project participants and resources interact to determine project per-
formance, only those features that describe a particular tipping point structure,
project management policies, and the fundamental processes they impact are
included. Simulated performances using different policies are, therefore, con-
sidered relative and useful for improving understanding and developing
insights, but not sufficient for final policy design. Complete model equations
and documentation are available from the authors or at http://ceprofs.tamu.edu/
dford/.

The model consists of three sectors: a workflow sector (Figure 1), a resource
allocation sector, and a schedule sector. The workflow sector is based on Ford
and Sterman’s (1998) structure of a development value chain with a rework
cycle. Work is initially completed and moves from the initial completion
backlog” (IC backlog) to the backlog of work requiring quality assurance (QA
backlog). A fraction of the work checked by quality assurance is discovered
to require change and moves into the rework backlog. Completed rework is
returned to the QA backlog for checking again because rework can reveal
previously hidden rework or create new change requirements. The complement of the checked work found to require rework passes quality assurance, is approved, released, and adds to the stock of work approved and released (Work Released). Flows between the stocks of IC backlog, QA backlog, RW backlog, and Work Released can be constrained by either process rates or resources. Process rates assume infinite resources and are the amounts of work available divided by the minimum times required to perform a work package. Resource rates are the products of the quantities of resources allocated to each activity and resource productivity. See Ford and Sterman (1998) for a more detailed work flow model description and model equations.

A unique expansion of this model in the current work is the explicit modeling of adding new tasks in a tipping point structure. Adding new tasks creates work that is added to the IC backlog during a project. We assume that the amount of work created is proportional to the work discovered to require rework:

\[ R_{nt} = (D_{rw})(s_{nt}) \]  

where

- \( R_{nt} \) = rate of adding new tasks due to ripple effects (work packages/week)
- \( D_{rw} \) = discover rework rate (work packages/week)
- \( s_{nt} \) = add new tasks strength (work packages created/work packages discovered, or dimensionless)
The add new task strength is a project characteristic that describes the strength of ripple effects, the amount of impact that reworked portions of the project have on the total work required to complete the project. This variable can be used to describe different project types. It is related to the amount of interdependence between project subsystems. For example, the strength between the foundation and superstructure components of a building would be high compared to the strength between the foundation and the heating system.

Resources are allocated among the initial completion, quality assurance, and rework activities proportionally based on the current demand for each of these activities. The desired fraction of resources for each activity is the size of the backlog compared to the project backlog (ICbacklog + QAbacklog + RWbacklog). For example, if resource productivities are equal and the current RW backlog is 40 percent of the current project backlog the desired portion of the available resources to be allocated to the rework activity is 40 percent. Applied resource fractions are delayed with a first-order exponential adjustment toward the desired fractions to reflect reallocation delays.

Schedule pressure is common in development projects. Increased rework is a side effect of schedule pressure that can degrade project performance (Cooper 1994; Graham 2000; Ford and Sterman 2003b). As a project approaches a fixed deadline schedule pressure increases and developers increase the pace of work to meet the deadline. This increases the risk of work being completed incorrectly. In the schedule sector pressure increases with the time required to complete the project backlog ($t_r$) and decreases with the time available to complete the project backlog ($t_a$). To explicitly model the impacts of schedule pressure on tipping point dynamics we disaggregate the rework fraction ($f_{rw}$) into the sum of a reference rework fraction ($f_{rw-r}$) and the schedule-induced rework fraction ($f_{rw-s}$). The reference rework fraction reflects basic project complexity. The schedule-induced rework fraction is the additional fraction of work requiring change due to schedule pressure. The schedule-induced rework fraction reflects mistakes made by developers due to pressures to meet the project deadline. This portion of the rework fraction is modeled as the product of schedule pressure and the sensitivity of the rework fraction to schedule pressure ($s_{rw-s}$). Forgoing the functions to limit values to 0–100 percent, the rework fraction becomes

$$f_{rw} = f_{rw-r} + f_{rw-s} = f_{rw-r} + \left[\left(\frac{t_r}{t_a} - 1\right) s_{rw-s}\right]$$

where

- $f_{rw}$ = rework fraction [dimensionless]
- $f_{rw-r}$ = reference rework fraction [dimensionless]
- $f_{rw-s}$ = rework fraction due to schedule pressure [dimensionless]
- $t_r$ = time required to complete project backlog [weeks]
- $t_a$ = time available to complete project backlog [weeks]
- $s_{rw-s}$ = sensitivity of rework fraction to schedule pressure [dimensionless]
Figure 2 shows the work flow structure (Figure 1) and tipping point feedback structure. Feedback loop B1 (Project Progress) withdraws work from the rework cycle. The QA backlog increases due to initial completion and rework, causing the QA rate to increase as resources are shifted to quality assurance. Increasing QA increases the rate at which work is approved and decreases the QA backlog. This balancing loop drives the project to completion as the backlogs decline to zero. If no new tasks are added loop B1 completes a project as quickly as processes and resources allow.

Loop R1 (Add New Tasks) adds to the total work required to complete the project through increases in the discovery of rework and ripple effects. Increasing initial completion and thereby the QA backlog increases the QA rate, increasing the rate at which work is discovered to require rework. This increases the rate at which new tasks are added, thereby adding more work to the IC backlog. In the absence of loop B1 (e.g., if the rework faction = 100 percent) loop R1 increases the rework and project backlog infinitely, thereby degrading project performance to eventual failure. Feedback loops B1 and R1 form a traditional tipping point structure that can dramatically change system behavior from being under control to being out of control due to a shift in feedback loop dominance from the balancing loop to the reinforcing loop. We show how,

![Diagram of a tipping point structure of a single development project](Fig. 2. A tipping point structure of a single development project)
through exogenous manipulation of loop dominance, managers can regain control of projects with tipping point structures. Loop R2 (Schedule Pressure) can increase the strength of the Add New Tasks loop (R1) by increasing the rework fraction as described above. The resulting increase in a project’s backlog increases the time required to complete the project, increasing schedule pressure. This increases the schedule-induced rework fraction and thereby the fraction discovered to require rework.7

Model testing and typical behavior

The model was tested using standard methods for system dynamics models (Sterman 2000). Basing the model on previously tested project models and the literature improves the model’s structural similarity to development processes and practices, as do unit consistency tests. Extreme condition tests were performed by setting model inputs, such as initial scope or total project staff, to extreme values and simulating project behavior. Model behavior remained reasonable. The model’s behavior for typical conditions is consistent with previous project models and practice (e.g., the common “S”-shaped increase in work released over time). As a successful project progresses, the backlog initially decreases slowly as the value chain and rework structures fill with work, the progress rate increases during stable production, and decreases to zero slowly as backlogs empty, indicating that the project is complete. Model behavior was also compared to actual project behavior as described by Ford and Sterman (1998, 2003b) and Lyneis et al. (2001) and found to closely match the behavior modes of actual projects.

Limited project data prevented calibration to a specific project that experienced tipping point dynamics. Therefore, to test the ability of the model to replicate tipping point behavior modes the model was calibrated with reasonable values to reflect a hypothetical project in which the Add New Tasks and Schedule Pressure loops are active. The simulated behavior was compared to the behavior of the Tennessee Valley Authority (TVA) Watts Bar unit 2 project. Watts Bar unit 2 was a nuclear power plant whose construction was marred by design changes (rework) and the addition of work outside the original scope (addition of new tasks) (NRC 1982). See Taylor et al. (2005) for a more detailed description. Figure 3 shows the percent complete for the simulated project and the Watts Bar unit 2 project, as reported to the Nuclear Regulatory Commission. Degrading project performance is represented by the decreasing project percent complete after month 80. Construction progress reports from the Watts Bar unit 2 project identify the “redesign of containment to accommodate higher transient pressures” as the reason for a six-month increase in the expected completion date (NRC 1982). Progress reports on the Limerick nuclear power plant project, where a one-year completion delay was attributed to an “increase of scope due to design changes and new regulatory requirements” (NRC 1982), also support the tipping point hypothesis. The similarity between
the actual project and simulated behavior modes in Figure 3 supports the model's ability to reflect a failure mode in nuclear power plant construction that could be caused by tipping point dynamics. Based on these tests the model was assessed to be useful for investigating tipping point dynamics in single development projects.

**Simple model behavior with tipping point dynamics**

Similar to Repenning (2001), project progress is described with the project backlog as a fraction of the project’s initial scope. Figure 4 shows the evolution of two types of projects. The horizontal axis shows project backlog in the previous time period and the vertical axis shows project backlog in the current time period. As an example of reading project behavior from the graph, the horizontal and vertical dashed lines show that in one project the backlog was 80 percent of the scope in the previous time period and 72 percent in the current time period. All projects begin in the center of Figure 4, with backlog equals to their initial scope. *Improving* projects have decreasing backlogs and are reflected by conditions below the diagonal dashed line, when preceding project backlogs exceed the current project backlog. The behavior mode of the work released of the improving project in Figure 4 is the traditional “S-curve” common in project management literature. In contrast, *degrading* projects are reflected by conditions above the diagonal dashed line (when current project backlogs exceed previous project backlogs) and can theoretically have an ever-increasing backlog. The behavior mode of the project backlog of
the degrading project in Figure 4 is an ever-increasing backlog. Successful projects end near the origin, when there is no more project backlog. Failed projects approach the upper right corner of the graph, reflecting continuously increasing backlogs. A project that remains on any point along the diagonal line has a constant project backlog and is stagnant (in net progress terms). An upper limit describing project failure has been arbitrarily set at 2, when work remaining to be completed is twice the original scope. All simulations in the current work reaching this limit have continuously increasing total backlogs and are considered failures. However, this limit may need to be adjusted for some projects.

**Project tipping point conditions**

The tipping point is the condition between dominance by loop B1 (Figure 2) leading to shrinking backlogs and project success, and dominance by loop R1 leading to growing backlogs and failure. Adding new tasks adds work to the project backlog and approving and releasing work withdraws work from the project backlog. Therefore, the tipping point occurs when the new task addition rate ($R_{nt}$) is equal to the rate at which work is approved and
released. The rate at which work is approved and released is the complement of the QA rate that is discovered to require rework \( (D_{RW}) \). Therefore, at the tipping point:

\[
R_{nt} = R_{QA} - D_{RW} \tag{3}
\]

where

\[
R_{nt} = \text{rate of adding new tasks due to ripple effects} \{\text{work packages/week}\}
\]

\[
R_{QA} = \text{quality assurance rate} \{\text{work packages/week}\}
\]

\[
D_{RW} = \text{discover rework rate} \{\text{work packages/week}\}
\]

Temporarily using the aggregate rework fraction \( (f_{rw}) \), the rework discovery rate \( (D_{RW}) \) is the product the QA rate \( (R_{QA}) \) and the rework fraction. By substitution using Eq. (1), Eq. (3) becomes

\[
(s_{nt})(R_{QA})(f_{rw}) = R_{QA} - (R_{QA})(f_{rw}) \tag{4}
\]

Simplification yields a description of the conditions that define the tipping point.

\[
f_{rw}(s_{nt} + 1) = 1 \tag{5}
\]

When the left-hand side of Eq. (5) exceeds 1 the project is degrading, when less than 1 the project is improving, and when equal to 1 the project is stagnant. A project can only remain at a tipping point (i.e., \( f_{rw}(s_{nt} + 1) = 1 \)) if loop B1 completes work at exactly the rate that loop R1 adds work to the project backlog. The project behavior will bifurcate to failure if loop R1 dominates or to success if loop B1 dominates. Therefore the tipping point is an unstable equilibrium.

**Project trajectory reversal**

When a project moves across the tipping point it experiences a change in project behavior from increasing to decreasing backlogs or vice versa. We refer to this fundamental change in project behavior as a trajectory reversal. A particular form of trajectory reversal of interest to managers is when a project initially improves but later degrades and eventually fails. Projects in which loop B1 or R1 dominates throughout the project experience monotonic trajectories that only improve or degrade (e.g., Figure 4). These projects cannot experience trajectory reversal because there is no shift in feedback loop dominance.

Permanent exogenous increases in the rework fraction, new tasks strength, or both can cause projects to fail (Taylor et al. 2005). But, barring structures that prevent a full and immediate recovery of those factors, when the exogenous change is removed the project crosses the tipping point again and can improve and be completed. In contrast, some projects may reverse trajectories and fail...
without experiencing major new problems. Other projects may fail even if the problems that initially generated trajectory reversal are temporary. Another dynamic structure, such as schedule pressure, is required to cause these projects to experience trajectory reversal and fail. For example, a temporary problem can activate the schedule pressure loop (R2) by increasing the project backlog and trap a project that would otherwise revert to an improving behavior mode and be completed above the tipping point.

As an example of a realistic scenario, consider a development project with an aggressive deadline that experiences an unexpected problem that temporarily increases the rework fraction. The backlog evolution of this project is shown in Figure 5. The time series plot of the percent complete for the same project is shown in Figure 6. The project begins below the tipping point (point 1) and improves. An exogenous temporary problem such as an unexpected conflict pushes the project over the tipping point \( f_{rw}(s_{nt} + 1) > 1 \), point 2). The project begins to build project backlog and degrades. When the problem is resolved and the temporary increase in the rework fraction is removed, the project dips below the tipping point and begins to improve again \( f_{rw}(s_{nt} + 1) < 1 \), point 3). But the temporary problem has increased the project backlog and

Fig. 5. Interactive impact of a temporary problem and schedule pressure on a project (graph truncated at project backlog as fraction of initial scope = 1.2 for clarity)
therefore the schedule pressure, rework, and addition of new tasks. This causes the project to remain closer to the tipping point conditions than before the problem. Although the project has returned to the improving side of the tipping point, the increased backlog has increased the strength of the schedule pressure loop and initiated an endogenous shift in feedback loop dominance. Eventually project conditions exceed the tipping point again ($f_{rw}(s_{nt} + 1) > 1$, point 5) due to schedule pressure, the project crosses the tipping point a second time, and fails (point 6).10

The behavior pattern in Figure 611 is similar to the behavior of the TVA’s Watts Bar unit 2 project shown in Figure 3. This shows that the combination of a tipping point structure and ripple effects can cause projects to fail. A project experiencing this type of behavior (ever-increasing backlogs and decreasing percent complete) would be faced with making major changes (i.e., increasing resources, scope reduction, revising project deadline) or terminating the project. Either way, the project would likely be considered a failure (e.g., increasing costs, lost revenues due to delays) and negatively impact all involved entities. This shows that managers need to measure the potential of development projects for tipping point-induced failure. We next investigate the use of robustness as such a measure.
Robustness in project design

Taguchi et al. (2000) defines robustness as “the state where the product/process design is minimally sensitive to factors causing variability.” The research of robustness in new product development has been largely limited to the robustness of the final product (Luo et al. 2005; Swan et al. 2005). The current work expands the concept of robustness to project design and measures the protection that the robustness of a project provides from tipping point failure. An inspection of Eq. (5) suggests that, if a project starts far enough away from its tipping point (i.e., $f_{rw}(s_n + 1) \ll 1$) and increases in the rework fraction and addition of new tasks strength are small, the project will not cross the tipping point and will monotonically improve. However, if the magnitude of the changes is large enough the project could be pushed past the tipping point. By modeling robustness ($r_{tp}$) as the distance between project conditions and the tipping point Eq. (5) can be rearranged to provide an intuitive meaning of robustness against tipping point-induced failure:

$$f_{rw} + (f_{rw} \times s_n) + r_{tp} = 1 \quad (6)$$

where

$r_{tp} = \text{robustness to tipping point-induced failure (dimensionless)}$.

The right side of Eq. (6) represents 100 percent of the project’s capacity to tolerate ripple effects. This capacity has been disaggregated into the three parts on the left side of Eq. (6): (1) capacity fraction absorbed by rework ($f_{rw}$); (2) capacity fraction absorbed by addition of new tasks ($f_{rw} \times s_n$); and (3) the unutilized capacity fraction that provides robustness ($r_{tp}$). When $r_{tp}$ is positive the project is below the tipping point (improving), when it is zero the project is at the tipping point (stagnant), and when it is negative the project is above the tipping point (degrading). For example, suppose a project has a fixed 20 percent reference rework fraction ($f_{rw}$ = 0.2), no schedule pressure-induced rework ($f_{rw-s} = 0$), and a fixed addition of new tasks strength ($s_n$) of 1. Applying Eq. (6), this project begins 0.6 from the tipping point (has an initial robustness of 60 percent). Given these conditions the project could tolerate schedule pressure-driven increases in the rework fraction of up to 30 percent (making $f_{rw} = 50$ percent) without crossing the tipping point.

Eq. (6) also provides a means of analyzing the effects of different variables on robustness. Robustness can vary significantly from initial conditions during a project. For example, schedule pressure can increase the fraction of work requiring change ($f_{rw}$) and thereby reduce robustness (Eq. 6). The minimum distance that project conditions come to the tipping point during the project represents a project’s most vulnerable conditions. Therefore, a project’s minimum distance from a tipping point is a better measure of a project’s robustness than the initial distance. Therefore we define project robustness as the minimum distance, as a percent of the project’s capacity to avoid failure, that a project
retains from failure conditions during the project life. Figure 7 shows the results of a sensitivity analysis of project robustness to five variables that impact tipping point dynamics in the model.

The horizontal axis of Figure 7 represents the percent change from base case values of the reference rework fraction, add new tasks strength, rework sensitivity to schedule pressure, and deadline without flexibility. The vertical axis represents the project robustness. For the base case the robustness at the beginning of the project (60 percent) is reduced by schedule pressure during the project to a project’s robustness of 51 percent. Values which “fall off” the bottom of the chart reflect negative project robustness, when the project has crossed the tipping point and failed. The sensitivity analysis reveals two important features of the relationships between the control variables and project robustness. First, with the exception of deadline flexibility, each variable has a threshold value, beyond which robustness quickly becomes negative. The threshold values for schedule pressure and adding new tasks strength are 250 percent and 120 percent of the base case conditions, respectively.
shown for clarity). In this analysis deadline flexibility does not have a threshold value because the base case project succeeds with no deadline flexibility. Therefore adding flexibility cannot degrade performance. Second, within the robust ranges, the control levers vary in their impacts on robustness. By inspection of Figure 7, project robustness is most sensitive to the reference rework fraction, then add new tasks strength, then rework sensitivity to schedule pressure, then deadline (inflexible), and is least sensitive to deadline flexibility.

In practice, avoiding tipping point conditions requires manipulating high leverage points in projects to keep the project away from a tipping point. Some current project management practices have this effect. For example, construction project managers often use relatively simple technologies, processes, and training programs to constrain underlying rework fractions. They also attempt to plan projects to keep operations separate and thereby constrain the addition of new tasks. Projects to develop tightly linked products with high rework fractions and ripple effects can potentially be designed to apply this strategy through methods such as modular design (Baldwin and Clark 2000).

Conclusions

The role of tipping points in project failure have been described with a characteristic behavior mode. Rework and ripple effects are used to model a common tipping point structure in single development projects. Robustness is defined as a project’s unused capacity to avoid tipping point-induced failure.

Results of the current work indicate that tipping point-induced project failure can be caused by delayed side effects of temporary problems as well as rework and strong ripple effects. Therefore the recognition and management of project tipping point structures may be important to successful project management. Shifts in feedback loop dominance can explain tipping point dynamics, supporting the importance of understanding feedback and loop dominance to successful project management. The underlying rework fraction was found to be a high leverage parameter for avoiding tipping point conditions.

This work has made contributions to the application of system dynamics to project management. The work proposed and initially tested a simple but common tipping point structure that can explain success and failure in single development projects. The use of shifts in feedback loop dominance to explain important project behavior illustrates and supports this unique system dynamics concept as a tool for understanding development projects as dynamic systems. We specified and quantified robustness for a specific system and challenge, and used it to measure the relative leverage of some project management tools. The work also advances the field of project management by further developing the concepts of tipping point dynamics, shifts in feedback loop dominance, and robustness as they relate to development projects. The results
suggest that practicing project managers should focus their efforts on features that control loop dominance (e.g., rework fractions and adding new tasks strength). By demonstrating the potential value added by adopting this approach, the work can help managers appreciate how perceiving and modeling feedback loops that drive behavior can improve project management. The results suggest that system dynamics researchers can improve project models and recommendations for improvement by developing other simple, generic, important project, structures, features and characteristics that both drive behavior and relate to practice.

The work also contributes a preliminary test of robustness as a measure of project performance. Our results show that robustness may be a good measure of a project’s protection from tipping point failure. Future research in this area should focus on operationalizing robustness for use across a wide range of project types. This future work could provide project managers with a method of evaluating the failure potential of projects.

The model used in this work can be improved in several ways. Explicitly, modeling work quality would allow the investigation of policies similar to Black and Repenning’s (2001), which releases lower-quality work. The model can be expanded to explicitly reflect contamination as well as the addition of new tasks and identify other tipping point structures. The model would also benefit from the investigation of the impacts of undiscovered rework on tipping point dynamics. Future research can improve model structure consistency with actual projects and calibrate the model to practice. Multi-variant sensitivity analysis can improve the task of the linking of structure and behavior and expand recommendations for practice. Tipping point dynamics can strongly influence the behavior and performance of individual development projects, and sometimes determine success or failure. Continued improvement in the understanding of tipping point dynamics can lead to better development project management and performance.

Notes

1. Active projects that stagnate, with no change in project backlog over time, are also considered failures but are less common. As will be shown, these conditions can be unstable and stagnant projects are likely to shift behavior modes into a mode with an increasing or decreasing project backlog.

2. As used here, scope refers to the tasks, measured in work packages, that, when approved and released, provide a specified performance, and work is an amount of development effort, also measured in work packages. Rework and adding new tasks cause the work required to complete the project to exceed its initial scope.

3. Development activity flows represent the completion of a development task. Therefore backlogs, as used here, include work in progress.
4. This creation of additional rework is not contamination because it represents additional rework required in the same piece of work, not additional rework required in a different piece of work.

5. Schedule pressure can have multiple beneficial and detrimental impacts on project performance which can be modeled with additional feedback loops (see Ford 1995 for examples). The current work models only the net effects of schedule pressure on rework and assumes the net effect is negative.

6. The loop dominance analysis discussed here is consistent with the results of a more rigorous analysis performed using behavioral analysis presented in Ford (1999). See Taylor et al. (2005) for details.

7. Third and fourth reinforcing loops exist in which the IC backlog and IC rate increase the QA Backlog and thereby the QA rate and Rework Backlog. These backlogs also increase the project backlog. These loops perform like loop R2, but instead of increasing the project backlog through the IC backlog, it is increased through the QA and Rework Backlogs.

8. The project simulation can reach the origin, when \((PB_{t-1}, PB_t) = (0,0)\), but actual projects stop when the backlog first reaches zero, when \((PB_{t-1}, PB_t) = (x,0)\) and \(x > 0\). This is represented in Figure 6 by a point on the horizontal axis close to the origin.

9. See Rahmandad (2005) for a similar project structure with constant addition of work and constrained work approval and release.

10. The simulation shown here does not account for a change in the project completion date. Therefore schedule pressure keeps the project above the tipping point indefinitely. If a new completion deadline was set when the project failed to meet the original deadline the project might fall below the tipping point and improve.

11. As defined for Figure 6 the Project Percent Complete is the work released as a fraction of the sum of the scope and added work. The Project Percent Complete can increase if projects are just slightly beyond the tipping point and if a large fraction of the new tasks added to the project backlog are simultaneously being approved and released. This can be shown by disaggregating the project backlog into the scope, total backlog added, and added backlog that is completed.

12. Deadline Flexibility is the fraction of the deadline gap corrected each time unit. Therefore, a completely inflexible project has Deadline Flexibility = 0 and a project with an infinitely flexible deadline has Deadline Flexibility = 1. For this variable the horizontal axis reflects the value of Deadline Flexibility.

Acknowledgement

The authors thank Prof. Kenneth Reinschmidt for assisting in the nuclear power plant investigation, Marsha Ward, NRC Research Librarian, for assisting in the acquisition of
NRC nuclear power plant construction data and the anonymous reviewers of a draft of this paper.

References


