System dynamics to understand and improve the performance of complex projects

Burak Gozluklu[a], John Sterman[a,*]

Abstract

Projects large and small are chronically late, over-budget, and fail to meet quality standards and customer requirements. These problems persist even though project managers and organizations have never had more frameworks, tools, and data to help them manage projects more effectively. Here we explore the causes of poor project outcomes using system dynamics. System dynamics has been used successfully in dispute resolution and, more importantly, in proactively improving project performance for more than forty years. We describe the feedback processes that lead to poor outcomes, including productivity and quality decline from failure to account for, and delays in, the discovery of errors and rework, rapid unplanned hiring, fatigue and employee turnover from excessive overtime, work done out of sequence, and corner cutting in project work, quality assurance, and testing. We illustrate the dynamics with the MIT Sloan Project Management simulation, a system dynamics model integrating the factors above in a realistic model that can be parameterized to represent projects in product development, software, construction, and other domains. The dynamic systems perspective explains how well-intentioned, experienced project owners, contractors, and managers can inadvertently cause cost and schedule overruns, low quality, and other problems by taking actions intended to improve schedule, cost, or quality in the short run, but worsen performance later and thwart learning at the organizational level. We close with recommendations to improve the management of complex projects.

Projects: Late, Expensive, and Wrong

Complex projects are chronically LEW: Late, Expensive, and Wrong (fail to meet quality standards and customer needs). In a 2019 survey (Gartner, 2019), only 11% of organizations reported they met 100% of initially defined launch targets, while approximately 45% were reported to be delayed. Projects often appear to be going smoothly until near the end, when errors made earlier are discovered, necessitating costly rework, expediting, overtime, unplanned hiring, schedule slippage, or reductions in project scope or quality. The consequences of these difficulties include poor profitability, loss of market share and reputation, increased turnover of management and work force, and, all too often, divisive and costly litigation between customers and contractors.

Project failures are nothing new. In 1625 the king of Sweden, Gustavus Adolphus II, commissioned several new warships. One, the Vasa, would be the largest and most powerful in the world. The keel was laid in 1626, but the king then demanded that a second gun deck be added—something we recognize today as a late customer specification change. The second gun deck doubled the Vasa's cannon to 64, but the cascading impact of this design change forced rework from stem to stern, seriously disrupting the project. The second gun deck also raised the ship's center of gravity. More ballast was needed, but with construction underway, the amount that could be added was limited by the space below decks. The partially completed ship was put in the water in the spring of 1627, while "hundreds of craftsmen work through the summer to finish the hull and rigging."[1] Concerned about its stability, the "captain supervising the construction of Vasa, Söfring Hansson, calls Vice Admiral Klas Fleming down to the ship…because he is worried. He has thirty men run back and forth across the deck and the ship rolls alarmingly. The Admiral has the demonstration stopped, afraid the ship will sink at the quay." Under

[a] System Dynamics Group, MIT Sloan School of Management, Cambridge MA 02142 USA.
[*] Corresponding author. jsterman@mit.edu.

[1] Quotes from https://www.vasamuseet.se/en/vasa-history/timeline.

The content of this paper, divided into two chapters, is forthcoming as Gozluklu, B. and Sterman, J. "System dynamics to understand and improve the performance of complex projects" and Gozluklu, B. and Sterman, J. "A management flight simulator to catalyze learning about complex projects" in Winch, G., Brunet, M., Cao, D., ed., (2022) *Research Handbook on Complex Project Organizing*. Cheltenham: Edward Elgar.

severe schedule pressure, the ship was launched on 10 August 1628 despite the clear evidence it was top-heavy. "Still within sight of the shipyard where it was built, Vasa heels to port under a gust and water gushes in through the open gun-ports." The Vasa sank just minutes into its maiden voyage, killing 30 to 50, including women, children, and dignitaries on board to celebrate the king's great achievement.

Of course, the Vasa disaster was four centuries ago. The builders lacked computational fluid dynamics models to assess stability, modern construction methods, IT systems, and project management tools we have today. Yet projects large and small continue to be LEW. Examples abound.

> BP's Thunderhorse, a deep-water oil production platform, nearly capsized in 2005 due to errors in design and construction (Minerals Management Service, 2010). "First oil"—the first barrel produced—originally scheduled for January 2005, wasn't pumped until June 2008. The lost revenue from that delay exceeded $32 billion.
>
> In 2003 the Finnish nuclear power authority signed a fixed-price, turn-key contract with Areva and Siemens for the Olkiluoto 3 nuclear power plant. Construction began in 2005, with completion scheduled for 2009. As of February 2022 the plant had yet to deliver a single kilowatt hour of power (World Nuclear News 2018, 2022), while costs ballooned from the initial estimate of about €3 billion to more than €8.5 billion, and by some estimates, €11 billion (Vanttinen, 2020), triggering a costly dispute between the customer and suppliers.
>
> The Samsung Galaxy Note 7 smartphone was released mid-August 2016 with design defects that could cause the battery to catch fire. Social media soon exploded with photos of burning Note 7s, quickly leading the US Consumer Product Safety Commission to mandate a recall while the US Federal Aviation Administration banned the Note 7 from commercial aircraft. Samsung cancelled the product on 11 October, with costs estimated at up to $17 billion plus reputational damage (Lee 2016).
>
> Intense pressure to accelerate development of the Boeing 737 Max 8 led to corner cutting in design and testing. Stability problems led to the late addition of a software system, MCAS, that automatically forced the nose of the plane down if sensors reported an excessive angle of attack. Defects in that system caused two fatal crashes in 2018 and 2019, killing 346 (House Committee on Transportation and Infrastructure, 2020).

Most projects don't end in fatal disasters, yet cost and schedule overruns, low quality, product recalls, lawsuits, and financial losses remain all too common. Figures 1-3 provide typical examples. Figure 1 shows planned and actual progress for a project to develop an ASIC (Application-Specific Integrated Circuit). Progress is far slower than planned: at the original deadline of 30 weeks detailed design is just starting. Progress accelerates rapidly, but the chips fail prototype testing, requiring an unplanned iteration back to detailed design. The reworked design fails again, requiring another unplanned iteration. This time the chips pass prototype testing but fail on the production equipment, suggesting the prototype test was not stringent enough. After a third unplanned iteration the project is finally completed—three times later than planned. Project costs were far higher than forecast. More importantly, the long delay in delivering the chip to the customer meant the customer's product was late to market and did not deliver the sales expected. Note that the original schedule provided for no iteration—that is, it assumed no errors requiring rework by prior stages would be revealed after prototype fabrication, assembly, or testing. In fact, none of the three revised schedules allow for any iteration. Furthermore, each new schedule is more aggressive than the last, despite the evidence that the design team is less and less productive, and commits more errors requiring rework, than projected.

Figure 1. Planned and actual progress of an integrated circuit development project.

Figure 2 shows progress and labor hours for the construction of an oil pipeline. Slow progress led to rapid unplanned hiring, especially after month 6. But the rapid unplanned expansion of the workforce and excessive overtime caused skill dilution, fatigue, worksite congestion, coordination problems, and work done out of sequence, reducing productivity and increasing errors requiring rework. Even though labor hours were more than double the peak planned level from months 7 through 19, the rate of progress after month 7 is no higher than in the first 6 months. Ultimately, cumulative labor hours exceeded the plan by more than 400% and the project took 160% of the planned duration.

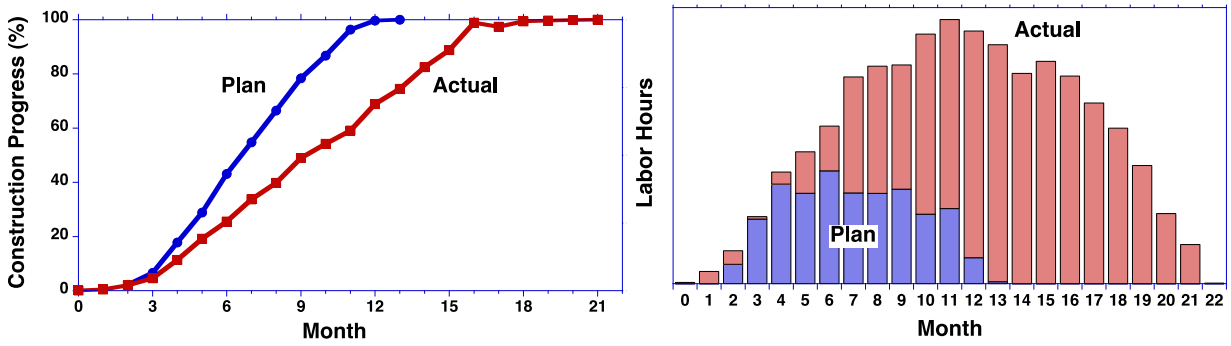

Figure 2. Planned and actual progress and labor hours for construction of an oil pipeline.

Figure 3 shows the progress of a software project. The project, a web application of modest complexity, was developed by an experienced team, building on an existing code base, and using modern agile software development processes. Drawing on the existing code base, the team made extensive use of concurrent engineering, with coding and testing starting in parallel with customer requirement specification and product design. Design and coding were not complete at the original deadline of five months, while Quality Assurance (QA) and testing lagged. Schedule pressure led to a soft launch just two weeks after the first beta test, and before all known rework requirements were resolved. The schedule overrun at soft launch was about a factor of two, and the cost overrun on the fixed-price contract at soft launch was 53%.

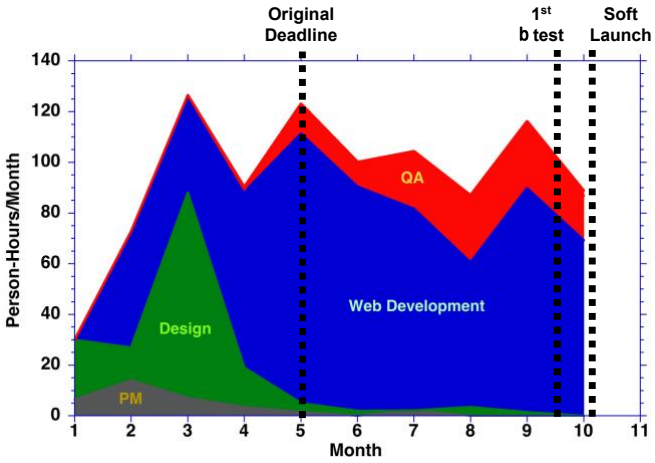

Figure 3. Labor hours per month for a software project.
PM = Project Management; QA = Quality Assurance.

These examples illustrate common project pathologies that contribute to delays, cost overruns, and quality problems. Many suffer from the "90% syndrome" in which the project appears to be on schedule through the first 90% of the scope, but the late discovery of errors and rework causes progress to stall; completing the last 10% often takes about as long as the first 90% (Ford and Sterman 2003a). Inadequate initial resources, aggressive delivery dates, and late customer specification changes often lead to excessive overtime, rapid unplanned hiring, and work done out of sequence, causing worker fatigue and burnout, skill dilution and worksite congestion, and increased error rates that worsen schedule and cost pressure in a vicious cycle. Schedule and cost pressure commonly lead to corner cutting, quality erosion, and inadequate testing, increasing errors while delaying their discovery, all too often allowing defects to be released to the customer. From semiconductor design to software to construction, from the Vasa to the Boeing Max 8, the consequences range from financial losses to failure in the marketplace to fatalities.

Why do these project pathologies occur, and what can be done? A typical reaction to project failure is to blame individual managers or workers. As a senior executive in a large company the second author studied said, "Around here we don't ask the '5 why's' [when there's a problem]… we ask the '5 who's.'" Asked what should be done to get a poorly performing project back on track, "fire the project manager" is a common suggestion among executives in our project management workshops. But the prevalence and persistence of these pathologies suggests it is not a matter of "just a few bad apples." The system dynamics perspective, consistent with decades of research in the social sciences, emphasizes that the structure of the systems in which we are embedded powerfully conditions our behavior. We take the position that the vast majority of people are honest, capable, and hard-working. But the complexity of the systems in which we are embedded far exceeds our ability to understand the many feedbacks and interactions that create their dynamics. Instead, people facing intense pressure to hit targets for cost, schedule, and quality make decisions in good faith that, while perhaps helpful in the short run, worsen performance later, further intensifying the pressures they face.

Here we describe the structure of the system governing project performance. That structure consists of the stocks of work to be done, tested, and released; the resource stocks available to manage the project and do the work; and the decision processes of project participants that control the flows of work and resources. Together these create multiple reinforcing and balancing feedback loops whose interactions generate project dynamics.

System Dynamics in Project Management

System dynamics has been used successfully in project management for over forty years, both for dispute resolution (e.g., Cooper and Reichelt 2004) and, more importantly, to prevent the problems that can lead to disputes (e.g., Cooper and Lee 2009, Godlewski, Lee, and Cooper 2012). Lyneis and Ford (2007), updated in Ford and Lyneis (2020), provide an extensive bibliography. A key concept in this work is the *rework cycle*, first described by Cooper (1980); see also Abdel-Hamid and Madnick (1991), Sterman (2000, Ch. 2).

The Rework Cycle

Figure 4 presents a simplified representation of the core structure for a project phase.[2] Project scope determines the initial stock of base work to be completed. Depending on the phase, base work could be defining customer requirements, creating construction drawings, pouring concrete, coding software, or other activity. The stock of base work remaining is drawn down as work is completed. Work completion depends on the workforce, the average workweek, and labor productivity, along with complementary resources not shown, such as plant and equipment, materials, and the work product of other phases the focal phase relies on. For example, the rate of work completion for the construction phase for a building depends not only on the on workforce, workweek, and labor productivity, but construction drawings provided by the design phase, building permits, materials such as concrete and steel, and other resources.

Not all work is done correctly. The gross flow of work completed splits into the work completed correctly and work completed incorrectly, according to the probability of defect creation (the error rate).

"Defects" include outright errors—a bug in a line of code, a bad weld, check valves specified backwards in construction drawings—a major cause of the near-capsize of Thunderhorse, and any attribute of the work that does not meet customer requirements even if it is not technically flawed (e.g., software that runs, but too slowly).

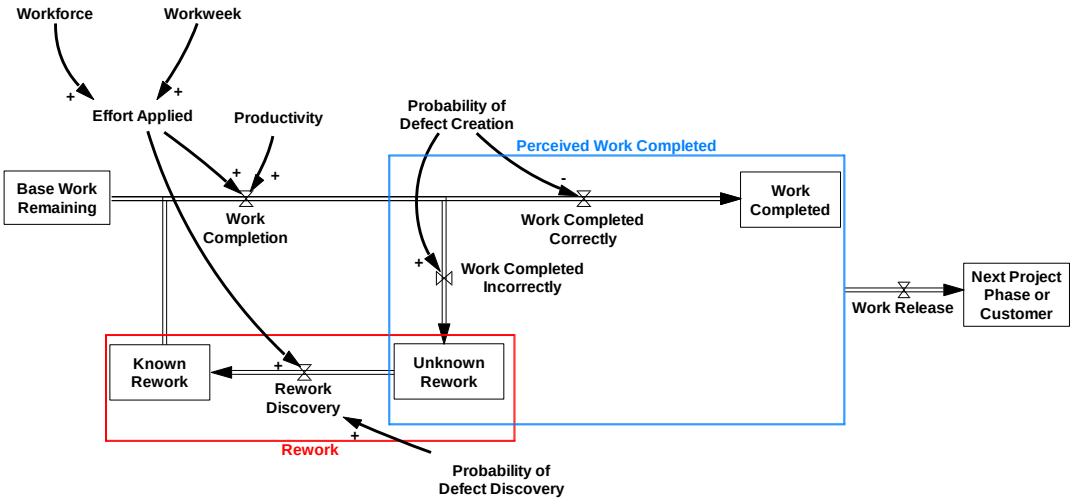

Figure 4. The Rework Cycle.
Stocks (rectangles) are connected by flows of work ("pipes" and valves).
Arrows with polarities show the causal factors that control the flows. See Sterman (2000).

[2] The diagrams are simplified compared to the simulation model described here. For readability, variables and relationships added in later diagrams are shown with bold fonts in thick arrows.

Work done correctly accumulates in the stock of Work Completed. Critically, the work completed incorrectly accumulates in a stock of Unknown Rework. The stock of undiscovered rework is drained as errors are discovered. But errors are not detected immediately. Rework discovery requires effort in the form of inspection and testing. Testing is not perfect. The probability of defect discovery depends on the resources applied to and the accuracy and thoroughness of inspection and testing. Defect discovery adds to the stock of known rework, where it remains until those tasks are reworked. Reworked tasks can be done correctly or incorrectly. Consequently, tasks can flow around the rework cycle multiple times before they are ultimately done correctly, as illustrated by the multiple unplanned iterations in the ASIC project in Figure 1. Work containing undiscovered defects can be released to the next project phase, or, worse, to the customer, as in the Boeing Max 8 and Samsung Note 7.

The rework cycle has critical implications for project dynamics. First, by definition, unknown rework is work believed to have been done correctly. The sum of the work completed and unknown rework is the work perceived to be completed. Similarly, the total rework required is the sum of the known and undiscovered rework. Unless all work is done *perfetcly* the first time, true progress will be overestimated and rework requirements underestimated, often leading to insufficient time and resources allocated to the project. The stock of unknown rework increases with a higher probability of defect creation, lower probability of defect detection, and longer the delay in defect discovery, all worsening the overestimation of actual progress.

Second, unknown defects are discovered only after a delay. That delay depends on the nature of the project and the effort applied to inspection and testing. An engineer might check the drawings created that day before leaving work, perhaps finding the reversed check valves. The probability of finding errors, however, depends on the resources applied to inspection and testing, the quality of testing protocols, and the care taken to follow those protocols. The engineer may not complete the drawings until late in the evening, not spend enough time checking, rely on CAD software that makes it difficult to see the problem, or, due to fatigue from prior long hours, not catch the error.

Third, the probability of error discovery depends on the nature of the product and defect. Some defects can be found quickly. Others cannot be detected despite careful testing until a full prototype is built or the components of the full system are integrated and tested under realistic field conditions. In one infamous example, a subcontractor on NASA's Mars Climate Orbiter built their software using English units while NASA used metric units. Each tested their software in isolation, finding no problems. The incompatibility only manifested when the systems were integrated. Even then, the problem was not detected despite evidence that accumulated after launch, leading to large errors in the orbiter's calculated position. The $125 million mission was lost when NASA executed the orbital insertion maneuver.

Scope Creep

Project scope often changes during the project. Late scope changes can be proposed by customers, the marketing team, or project team members who learn of new technologies or recognize new possibilities as the work proceeds. Figure 5 adds scope changes to the core rework cycle. Demands for new features accrue in a backlog of pending feature requests. Project managers or organization executives decide whether to add those late changes to the project scope. Similarly, under cost or schedule pressure, the team may decide to reduce the scope, eliminating features deemed less essential, or postponing them to a later version of the product.

The decision to accept a late feature request directly increases the stock of base work remaining, but often makes some previously completed work obsolete and requires reworking other completed tasks. Similarly, the decision to cut scope reduces the stock of base work remaining, but can also reduce the stock of work perceived to be completed and the stock of known rework, depending on how much of the work involving the cancelled features has already been completed. However, scope reductions can also

increase the stock of known rework. For example, the decision to eliminate a planned feature from a software project may require recoding to eliminate calls to and data returned by the deleted feature, redesign of the user interface, and revisions to documentation and training materials.

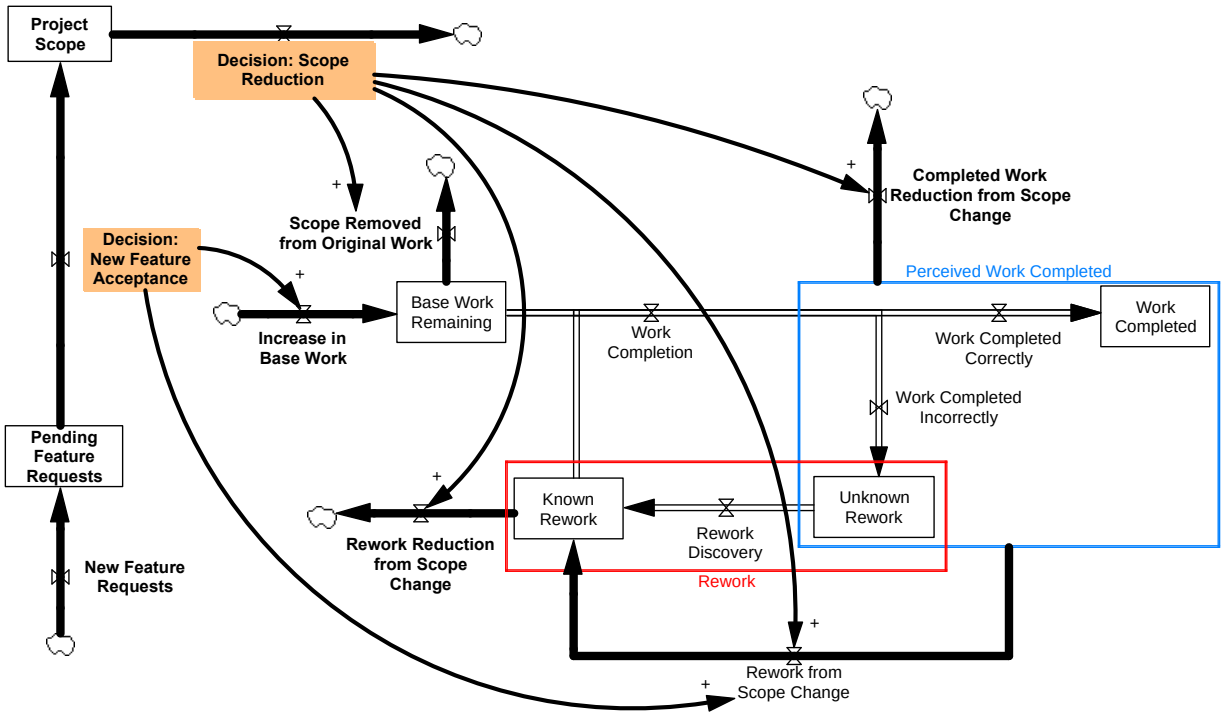

Figure 5. Impact of late specification changes and scope cuts.

For simplicity, Figure 5 shows only the focal phase, but scope changes also force rework in downstream phases when they occur after some the work of the focal phase has been released to later phases. For example, the owner of a large commercial real estate development decided a building originally specified to be office space be converted into a hotel. The change was made after detailed design was under way and, due to the use of fast-tracking, after construction had begun. The change not only led to substantial new design work, but obsolescence and rework of previously completed construction drawings, plans for subcontractors, material acquisition, and significant construction rework including changes to building foundations and utility infrastructure arising from the significant difference in water and sewer loads of hotels compared to office buildings. The resulting "avalanche of changes" through project phases, also known as "ripple effects" or "snowball effects" (Ullah, Tang, Yin 2016, Terwiesch and Loch 1999) significantly increased project costs far beyond the direct costs of reissuing the drawings. The cascading impacts of late changes have been at the heart of many project management applications of system dynamics, in both dispute resolution and proactive project improvement (e.g., Cooper 1980, Williams et al. 1995, Eden et al. 2000, Stephens, Graham and Lyneis, 2005, Lyneis and Ford 2007, Godlewski, Lee, Cooper 2012).

The later a scope change is made, the more prior work is made obsolete and requires rework, in both the focal phase and downstream phases. This rather obvious fact suggests that scope changes should be few, and decline quickly, approaching zero before detailed design and other phases get underway. Often, however, the opposite occurs. Figure 6 expands the stock-and-flow structure described so far to show some of the feedback processes affecting decisions to add or cut scope.

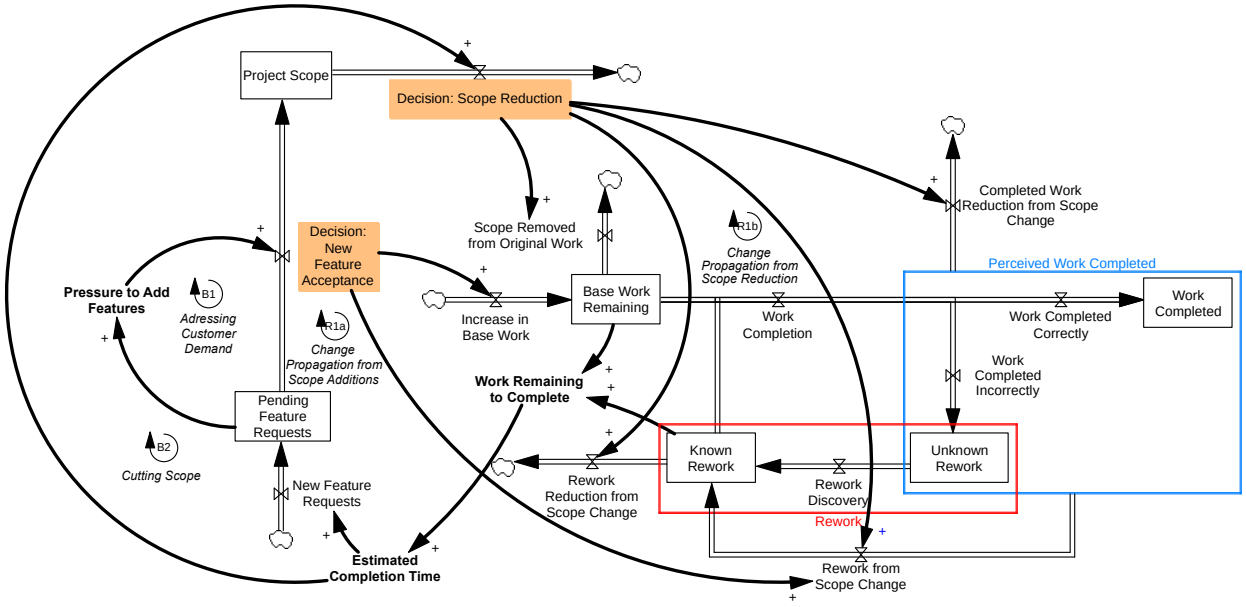

Figure 6. Feedbacks generating scope creep.

The larger the backlog of pending requests for new features, the greater the pressure on management and the project team to add those changes. Doing so reduces the backlog of pending changes, easing the pressure, creating the balancing feedback B1, *Addressing Customer Demand*.

However, while accepting new features satisfies the customer and other stakeholders, the increase in base work and in reworking tasks previously completed increase the total work remaining to be done. The estimated completion time for the project lengthens. The longer the project lasts, the greater the likelihood that still more new features will be proposed. As those changes are accepted the work remaining grows still further, and the project becomes even later, in a vicious cycle, the reinforcing *Change Propagation from Scope Additions* feedback R1a.

Scope reductions to get a late project back on schedule can also trigger unanticipated problems. Scope reductions directly reduce the work to be completed, speeding completion (the balancing *Cutting Scope* feedback, B2), but can also increase the work to be done because some work previously completed must now be reworked. The later scope reductions are made, the more work will have been completed, and the greater the resulting rework. Scope cuts are usually proposed after projects are clearly so late or over budget that project managers can make a strong case to customers or marketing that scope cuts are necessary. However, the later scope reductions occur, the greater the rework, potentially increasing estimated project duration and the likelihood of new late scope additions, another a vicious cycle, the reinforcing *Change Propagation from Scope Reduction* feedback R1b.

Why would customers, marketing, or other stakeholders continue to request new features even when doing so causes further delay and disruption? Consider commercial product development. Every week brings the possibility of changes in technology, consumer preferences, and the features of competitor products. Similar changes arise in other domains, for example military systems: every week brings the possibility of changes in technology, adversary capabilities, and mission. The longer a project takes, the more new features are likely to be suggested. The Project Management Institute (PMI) recognizes this dynamic: “The longer a project, the greater the chance for scope creep to occur” (Larsen & Larsen, 2009). However, PMI’s material does not recognize the reinforcing feedbacks that create the possibility that late scope changes can cascade to delay projects and increase costs far beyond the direct time and costs needed make the change in specifications and reissue drawings. Rahmandad and Weiss (2009)

develop a formal system dynamics model that includes those feedbacks, showing how they undermined the capability of a large firm to develop new software to remain competitive.

Christensen's (1997) case study on medical device maker, Medtronic, provides an excellent example of the reinforcing feedbacks creating scope creep trap. At the time, Medtronic was losing market share due to late product launches. A Medtronic executive explained,

> "The problem then fed on itself.…the development people would tell me that they could never get anything to market because marketing kept changing the product description in the middle of the projects. And the marketing people would say that it took so long for engineering to get anything done, that by the time they got around to completing something, the market demands would have changed. When customer requirements evolve faster than you can develop products, it becomes a vicious spiral" (Christensen 1997, p. 3).

Responses to Schedule Pressure: Overtime and corner cutting

Overtime and long hours are perhaps the most commonly used method to boost progress when a project falls behind. Figure 7 shows the key feedbacks governing these factors.

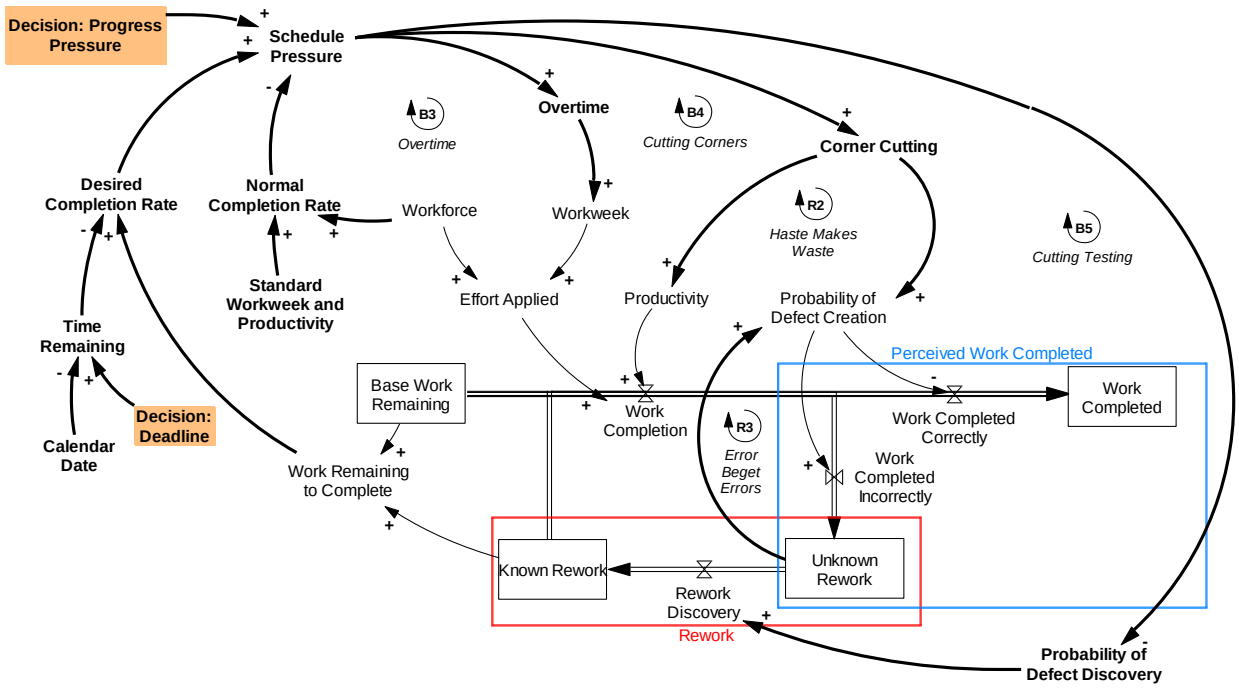

Figure 7. Overtime, cutting corners, and cutting testing.

Overtime and corner cutting respond to *schedule pressure*. Schedule pressure is the ratio of the rate at which work must be completed to finish the project by the deadline to the rate work can be completed given the workforce, standard workweek, and current estimate of productivity. Schedule pressure can also be affected by management (the Decision to impose more or less pressure for progress). High schedule pressure leads to overtime, either mandated by management or voluntarily as individuals try to hit their deadlines. Overtime speeds work completion, draining the stock of base work remaining faster and reducing schedule pressure, creating the balancing *Overtime* feedback, B4.

Individual workers can also speed progress by cutting corners. They can work faster by taking less care

on each task, skipping steps, and by failing to document their work, coordinate with others on the team, or train and mentor less experienced team members. All carpenters know they should "measure twice, cut once" but under schedule pressure, they can speed progress by measuring once. The balancing *Cutting Corners* feedback, B4, can quickly boost progress and reduce schedule pressure. Individuals are often tempted to cut corners as it is often difficult for to detect, and because it enables people to hit their deadlines when they are already working long hours and further overtime is not possible.

Similarly, high schedule pressure can lead people to cut testing. By testing less thoroughly, or skipping some tests, the probability of discovering defects falls. The stock of known rework will be less than it would have been, reducing the perceived work remaining and easing schedule pressure, the balancing *Cutting Testing* feedback, B5.

All three of these responses to schedule pressure operate swiftly to help get a late project back on track. But all three create the possibility of unintended harms to the project.

Cutting corners increases productivity but also increases the chance of error: Carpenters who measure once often cut the board to the wrong length. More work is done incorrectly. When those defects are discovered, known rework rises, increasing schedule pressure and potentially leading to still more corner cutting, a vicious cycle shown as the reinforcing *Haste Makes Waste* feedback, R2.

Further, while cutting testing slows the growth of the known rework, the defects not found accumulate in the stock of undiscovered rework, where they become the basis for subsequent work, which leads to still more defects. The result is another vicious cycle, the reinforcing *Error Begets Error* feedback, R3, as when the check valves in the Thunderhorse self-ballasting system were installed backwards because of an error in the construction drawings. The backwards valves led to the near capsize and additional damage requiring still more rework.

Finally, while overtime immediately boosts progress, extended overtime soon leads to fatigue and a host of harms (Figure 8). Long hours, whether in manual labor or knowledge work, lead to inadequate sleep, personal and family time, and poor diet. Fatigue cuts productivity, slowing progress and increasing schedule pressure, creating even more pressure for overtime (*Fatigue: Productivity*, R4). Fatigue also increases the error rate, boosting rework and further increasing schedule pressure (*Fatigue: Errors*, R5). Fatigued workers are also less able to detect errors. In the short run, lower quality testing induced by fatigue limits rework discovery (the Balancing *Fatigue: Error Detection* feedback, B6), but, by accumulating more undiscovered rework, leads to still more errors through the reinforcing *Haste Makes Waste* feedback that, when detected, increase work remaining and schedule pressure further (*Fatigue: Less Testing Now, More Rework Later*, R6).

Fatigue eventually leads to burnout and increases employee attrition, creating two more reinforcing feedbacks that operate as vicious cycles when a project falls behind: a decline in the workforce increases schedule pressure by slowing progress and by cutting the normal rate at which work can be done (the reinforcing *Attrition Erodes Progress* and *Attrition Erodes Capability* feedbacks, R7and R8, respectively.

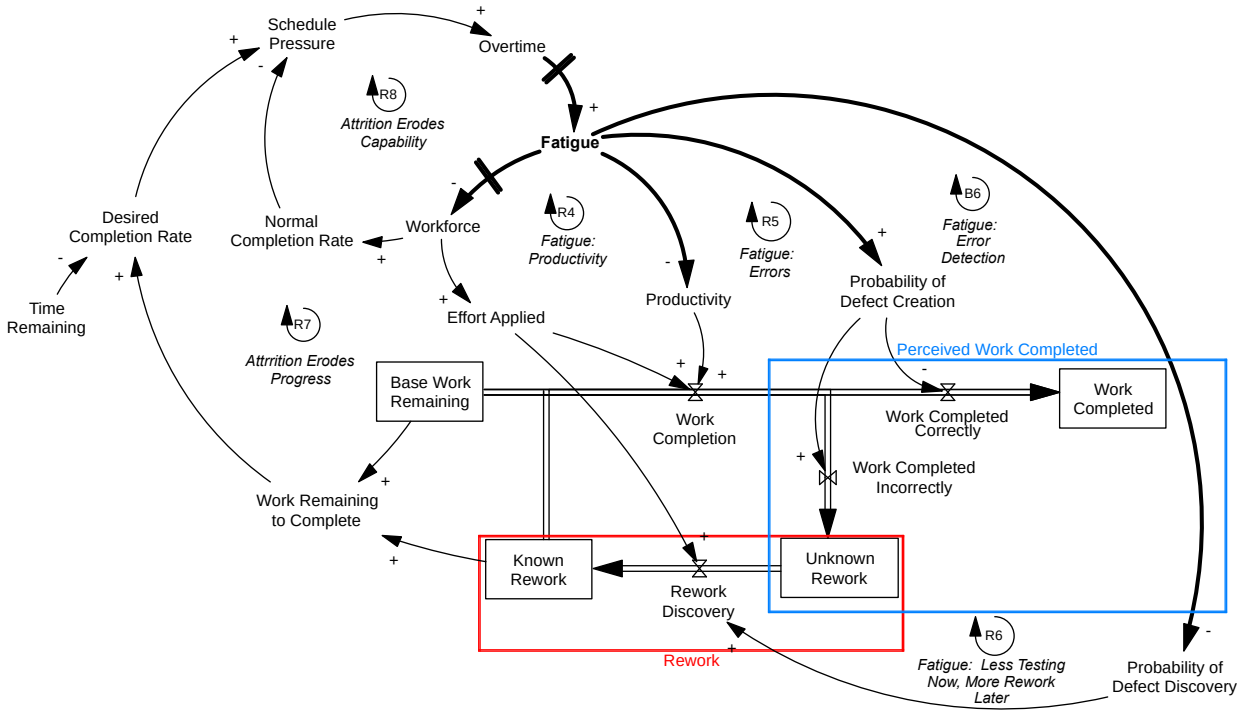

Figure 8: Fatigue.

Unplanned hiring

Unplanned hiring is another common response to high schedule pressure (Figure 9). Expanding the project team increases work completion, easing schedule pressure (the balancing *Hiring* feedback, B7). But new hires, shown as the stock of "rookies," are less effective than experienced workers, and only gradually gain experience and build relationships with others on the team. Thus, the first impact of hiring is an increase in the rookie fraction—that is, a reduction in average experience and skill. The result is lower productivity, more defects, and lower chances of defect discovery. The result is another set of reinforcing feedbacks (*Rookie Fraction: Productivity, Error Creation, Error Detection*, R9, R10, and R11, respectively) that undermine the benefits of hiring, analogous to the impacts of fatigue on productivity, defect creation, and test quality.

High schedule pressure increases the quit rate, as overworked and overstressed team members transfer to other projects or leave the organization altogether. Additional hiring can replace these losses, but the new hires are rookies, leading to further erosion of productivity, work quality, and testing, creating another set of reinforcing feedbacks, not shown, analogous to the impacts of inexperience on productivity, error creation, and error detection. Additional harms from unplanned hiring, not shown, include increased worksite congestion and team coordination problems, further harming productivity, quality, and defect detection.

These effects are most severe when new hires are truly inexperienced, but also arise when consultants or veterans of the organization are transferred in to help a project in distress. Obviously, poaching experienced people from other projects causes those projects to suffer from the feedbacks described here. But even if experienced people with good technical skills could be hired without harming other projects (an unlikely prospect: what organizations today have skilled people sitting on the bench?), they don't know the details of the project they are joining and lack the personal relationships that build the trust required for effective collaboration.

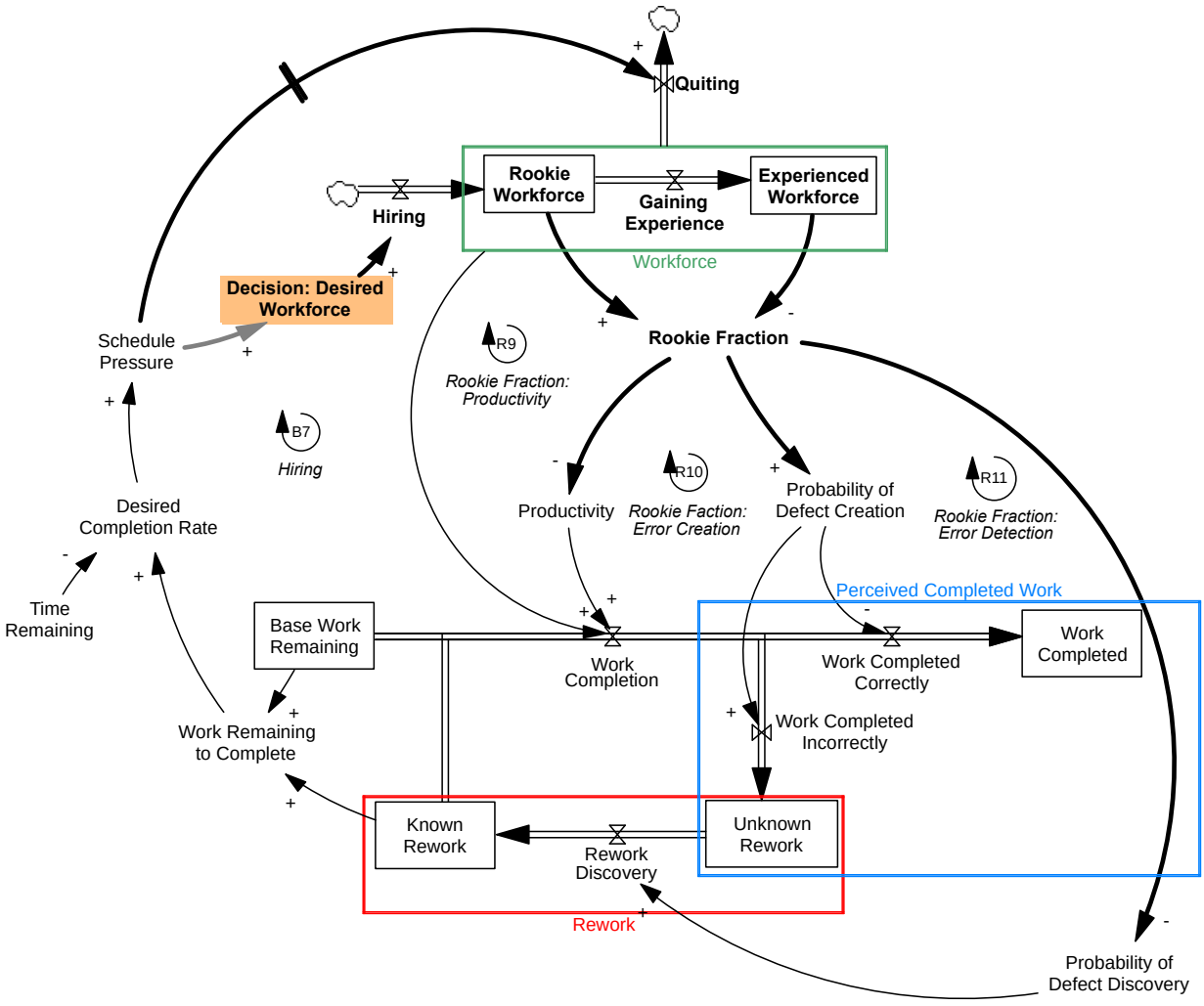

Figure 9. Unplanned hiring.

Increasing concurrency: Working out of sequence

Another common response to get a late project back on track is doing more work in parallel. Increasing the overlap of tasks planned to be done sequentially is designed to speed progress by enabling people to work on different tasks at the same time. Denoted "concurrent project management", "concurrent engineering", or "fast tracking", these practices can be effective—if the organization redesigns the development process and builds stronger capabilities for cross-functional collaboration and coordination. Doing so, however, takes time. All too often, managers of a project behind schedule will increase concurrency without undertaking the needed capability development for it to be effective. The result is an increase in the amount of work done out of sequence (Figure 10). Increased concurrency immediately boosts work completion, easing schedule pressure (the balancing *Concurrency: Higher Throughput* feedback, B8). But lacking the project structure and capabilities for enhanced concurrency, doing more work out of sequence increases defect creation as team members must make assumptions about the work product of others they need to do their own work correctly but that isn't yet available. The result is a reinforcing feedback that increases schedule pressure further as lower work quality eventually increases rework (*Concurrency: Increased Errors,* R12).

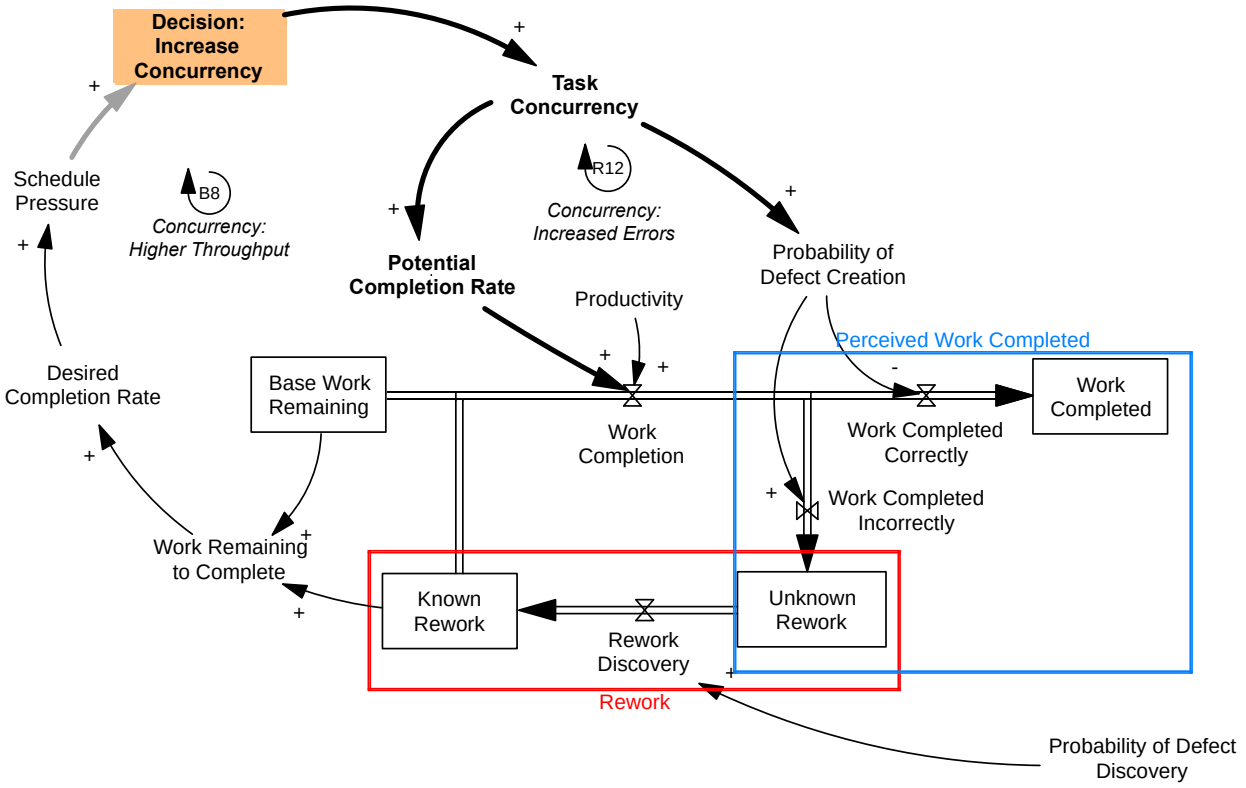

Figure 10. Increasing concurrency (work done out of the optimal sequence)

The diagrams and discussion here illustrate only a few of the many feedbacks created by common managerial responses to projects in distress. The reader is encouraged to map the causal structure of others, e.g., slipping the deadline, soft launch, pressuring workers for faster progress, and others. For each, first seek the intended impact of an action—how it might improve project performance, then ask what unintended consequences might arise and what feedback processes these create.

Summary: Multiple Interacting Feedbacks Create Delay and Disruption

Two lessons emerge from the dynamic feedback perspective on project performance outlined here.

First, each common response to a project in trouble, whether overtime, corner cutting in the work and in testing, unplanned hiring, increasing concurrency, or accepting late customer changes, is intendedly rational. Each creates a balancing feedback that attempts to speed progress, control costs, or better meet customer and market needs. In almost all cases, these interventions are undertaken in good faith by well-intentioned people who sincerely seek to improve project performance.

Second, each of these actions creates a network of unintended consequences including scope creep, stress, fatigue, skill dilution, work done out of sequence, and others. These unintended impacts cut productivity, increase errors, and erode testing quality, slowing progress and triggering multiple reinforcing feedbacks that act as vicious cycles, leading to cumulative delay and disruption far larger than expected. Although for clarity these feedbacks are presented separately in the diagrams here, they operate simultaneously. Each reinforcing feedback may be relatively weak, but their joint impact can be far larger than people anticipate. The increase in errors caused by fatigue, by itself, may not outweigh the benefits of longer hours, but the joint impact of fatigue on errors, productivity, test quality, worker attrition, and other

impacts can overwhelm the benefits of longer hours.

Third, these effects are nonlinear and often delayed. Increasing the workweek from 40 to 48 hours immediately boosts output and may not harm productivity or boost errors and attrition much, but further increases sharply increase these harms. Those harms manifest after delays, that, when unrecognized, can lead people who sincerely seek to improve performance to inadvertently worsen it.

Finally, these feedbacks are not captured in most project management tools. Gantt charts, critical path methods, and project management software typically represent work activities in great detail, but the planned duration, resource requirements, and costs of each activity are specified exogenously. They do not capture the feedback processes identified above. As a result, the unintended impact of, say, a late customer specification change on overtime, fatigue, productivity, error rates, error detection, unplanned hiring, and so on are not captured. Unless a project manager or scheduling analyst anticipates all these interactions and feedbacks and manually inserts the changes, project management software will underestimate the impact of the changes, often dramatically.

System dynamics is complementary to traditional scheduling and project management tools: where traditional tools are useful to deal with the *combinatorial complexity* of complex projects with multiple parallel and sequential activities, system dynamics is useful to deal with the *dynamic complexity* created by the interdependencies, feedbacks, time delays, and nonlinearities in large scale projects.

Rigorously accounting for these interactions requires a simulation model that captures these interactions with sufficient granularity for the purpose, and with parameters and relationships grounded in evidence. The system dynamics literature describes how such models can be developed, estimated, and tested (Ford and Sterman 1998a, 1998b, Sterman 2000, Rahmandad and Hu 2010, Parvan et al. 2015, Lyneis and Ford 2007).

Simulating the Dynamics of Complex Project

To illustrate how the dynamics unfold and provide insight into superior policies, we present here results from the MIT Sloan Project Management Simulator. This interactive management flight simulator enables individuals to play the role of managers for a complex project. Users can select projects calibrated for a new consumer product, software, or construction project, and the simulation can be customized to other settings (https://mitsloan.mit.edu/teaching-resources-library/management-simulations; scroll down and select "Project Management Simulation"). Each simulated week participants make decisions including project schedule and staffing, whether to accept late customer feature requests or reduce scope, whether to alter the degree of concurrent development, and whether to pressure the workforce for faster progress or higher quality (Figure 11 shows the main interface). Users receive extensive feedback on progress, current and projected costs, productivity, rework detection, a host of other key performance indicators, and emails from the (simulated) engineering, quality assurance, human resources, marketing departments—and CEO.

The simulations here assume project scope and deadline parameters approximating the development of a new consumer product. The simulation represents two main phases, design and prototyping/manufacturing. Table 1 summarizes the key parameters. The project deadline is 48 weeks. Design and prototype/build are each initially scheduled to last 24 weeks; users can change these during the course of the simulated project. The initial scope is 24,000 design tasks and 96,000 prototype/build tasks, where a "task" is defined as the amount of work an experienced worker can do in one hour. The required levelized work completion rates for each phase are then 1000 and 4000 tasks/week. At the normal workweek of 40 hours, 25 design staff and 100 prototype/build staff are needed to complete the initial scope on time (assuming no rework). To capture late specification changes, we assume marketing proposes new features stochastically at rate averaging 20% of the initial project scope per year (and the budget rises for each new feature accepted).

The commercial success of the product depends on time to market, the scope delivered, and product quality: releasing the product late, without the latest features, or with many defects erodes revenue; cost overruns raise development costs, and low quality drives up warranty costs.

Figure 11. Interface for the MIT Sloan Project Management Simulator.

Parameter	Design	Prototype/Build	Sources
Initial Deadline (week)	24	48	Scenario Assumption
Initial Planned Start Time (week)	0	24	Scenario Assumption
Initial Scope (tasks)	24,000	96,000	Scenario Assumption
Normal Workweek (hours/week)	40	40	Scenario Assumption
Productivity: Work Completion (tasks/person-hour)	1	1	Definition
Relative Productivity of Inexperienced Staff (d'less)	0.4	0.5	Boehm et al., 2000
Time required to gain experience (weeks)	24	18	Boehm et al., 2000
Average time to hire staff (weeks)	8	6	Society for Human Resource Management (SHRM 2018)
Normal employee tenure (weeks)	300	300	US average 2010-2020 for "Computers and Electronic Products" (US BLS 2020)

Table 1: Key assumptions for simulations

A Tale of Two Managers

To illustrate, we simulate two project managers: Reactive and Proactive. Both use the same information to manage their projects. The simulated managers cannot use information about the future, such as when

marketing will demand late specification changes, or how many team members will quit each week. They do not know how large the stock of unknown defects is until testing discovers them. Neither simulated manager is assumed to behave optimally. Instead, they use behavioral decision rules (Sterman 2000) to make decisions.

The Reactive manager staffs initially at the level indicated by the initial scope, deadline, and worker productivity, then continuously monitors the project, adjusting hiring to meet the deadline given the work remaining, time remaining, and reported worker productivity. The reactive manager expects the team will handle any rework that may be discovered with overtime, and does not account for the reinforcing feedbacks that can result from fatigue, corner-cutting, cuts in testing, increasing concurrency, and so on. The reactive manager accepts all new features proposed by marketing.

The proactive manager anticipates the possibility of late changes and unplanned rework and therefore staffs initially with more people, for both phases, than initially indicated. The proactive manager also monitors the project closely, including known rework, schedule pressure, and the workweek, and adjusts staffing to avoid pressure on workers to cut corners or testing. Finally, the proactive manager accepts new feature requests early in the design phase but, to avoid excessive rework, rejects them as design proceeds.

Figure 2 shows the dynamics. By accepting late change requests, the reactive manager finds the design staff is inadequate, and hires more people to stay on schedule. But the unplanned hiring increases the rookie fraction, lowering productivity and increasing errors. Progress slows, requiring still more hiring, further eroding experience. Schedule pressure builds. In response, the design team works longer hours, causing fatigue that eventually erodes productivity and quality, and increases worker attrition. Progress slows further, forcing still more hiring. High schedule pressure causes the team to cut corners and testing, speeding apparent progress but increasing defect creation and slowing defect detection. The eventual discovery of these defects forces even more late hiring. The design staff peaks many times above the initial level late in the design phase, quality is low, and cuts in testing cause many defects to be released to the prototype/build phase. The dynamics there are similar: with so many defects inherited from design, the prototype/build staff proves to be too small, forcing overtime and unplanned hiring. The reinforcing feedbacks described above work as vicious cycles, and the project finishes late, over budget, with poor quality and many defects released to the customer.

In contrast, the proactive manager immediately builds the design staff above the indicated level to create the capacity to handle rework and late feature requests without excessive overtime, corner cutting, or cuts in testing. Late feature requests are accepted only until most of the high-level design work is done, but none afterwards. The proactive manager also builds the staff for the prototype/build phase early and above the initially indicated level. The better-rested, more experienced team is more productive and does higher quality work. They also inherit fewer defects from design, boosting their productivity and speeding progress as fewer tasks are returned to the design team for rework. Consequently, the prototype/build staff peaks at about half the level of the reactive manager's project. The feedback structure in the two cases is the same, but the vicious cycles triggered by the reactive manager remain far weaker, or entirely dormant, for the proactive manager.

Comparing project outcomes (Figure 3), the reactive manager delivers the project 13 weeks late, a 27% schedule overrun, at a cost of $50 million, 40% over the final budget. Final product quality is low, with about 5.3% of the features in the project scope defective. Being so late to market with such poor quality causes the product to flop in the marketplace. The company loses approximately $38 million. In contrast, the proactive manager delivers the project more than a week early, about 1% under the budget, and with a defect rate of 1.67%, 69% lower than the reactive manager. As a result the product does well in the market and highly profitable, netting $45 million in lifetime profit.

The results are illustrative, and represent particular assumptions about the project, workforce, and the market's response to delivery times and quality. Extensive sensitivity analysis, not shown, shows these

patterns to be robust across a wide range of these assumptions.

Note that the proactive manager's results are not optimal. Rather they arise from realistic behavioral decision rules, based on information available to real project managers. The simulated proactive manager's results thus represent a realistic benchmark for performance. Optimal performance is even higher.

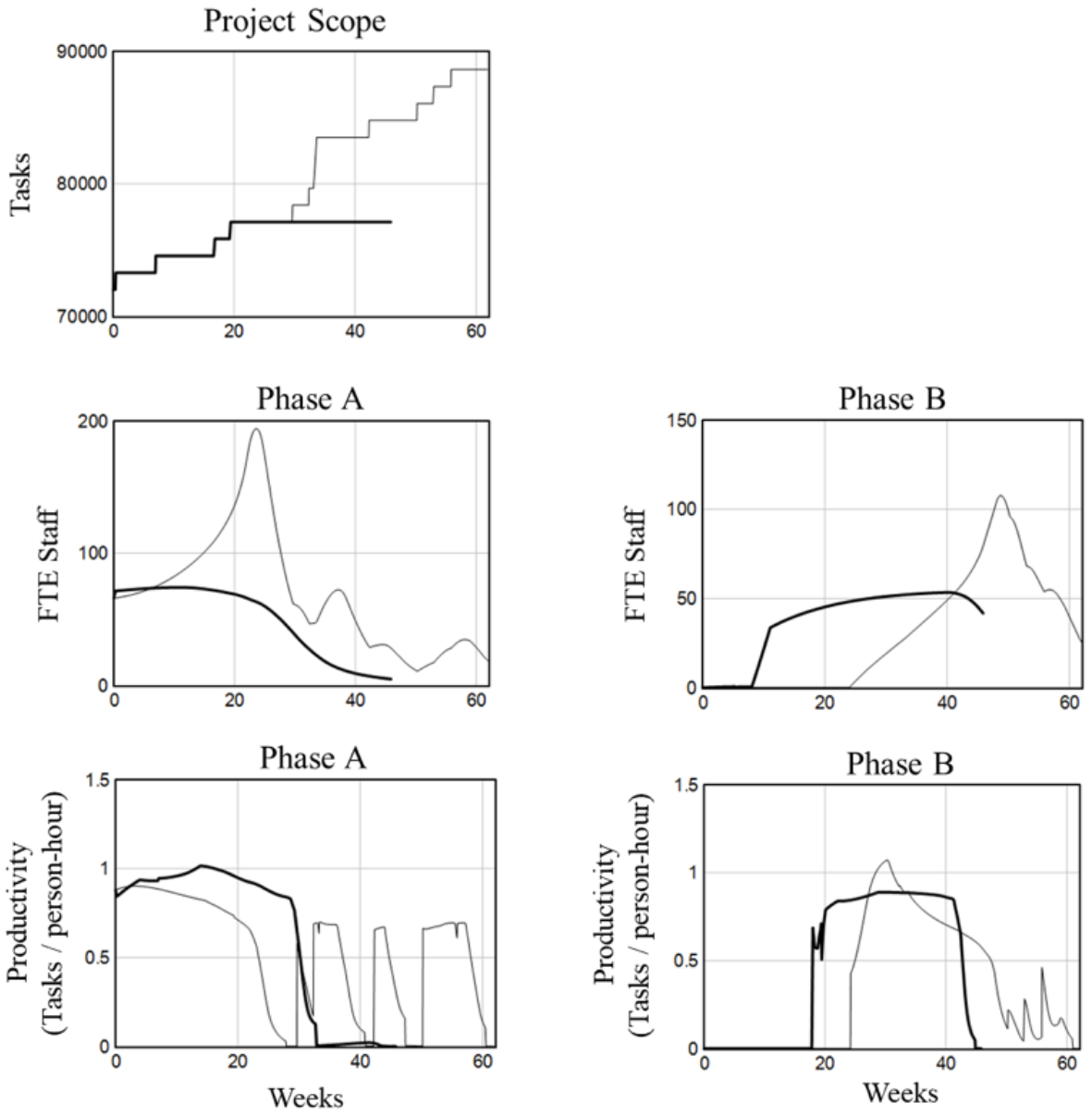

Figure 2. Project evolution for Simulated Proactive (black) and Reactive (grey) managers. Phase A: Design; Phase B: Prototype/Build. FTE = Full Time Equivalent.

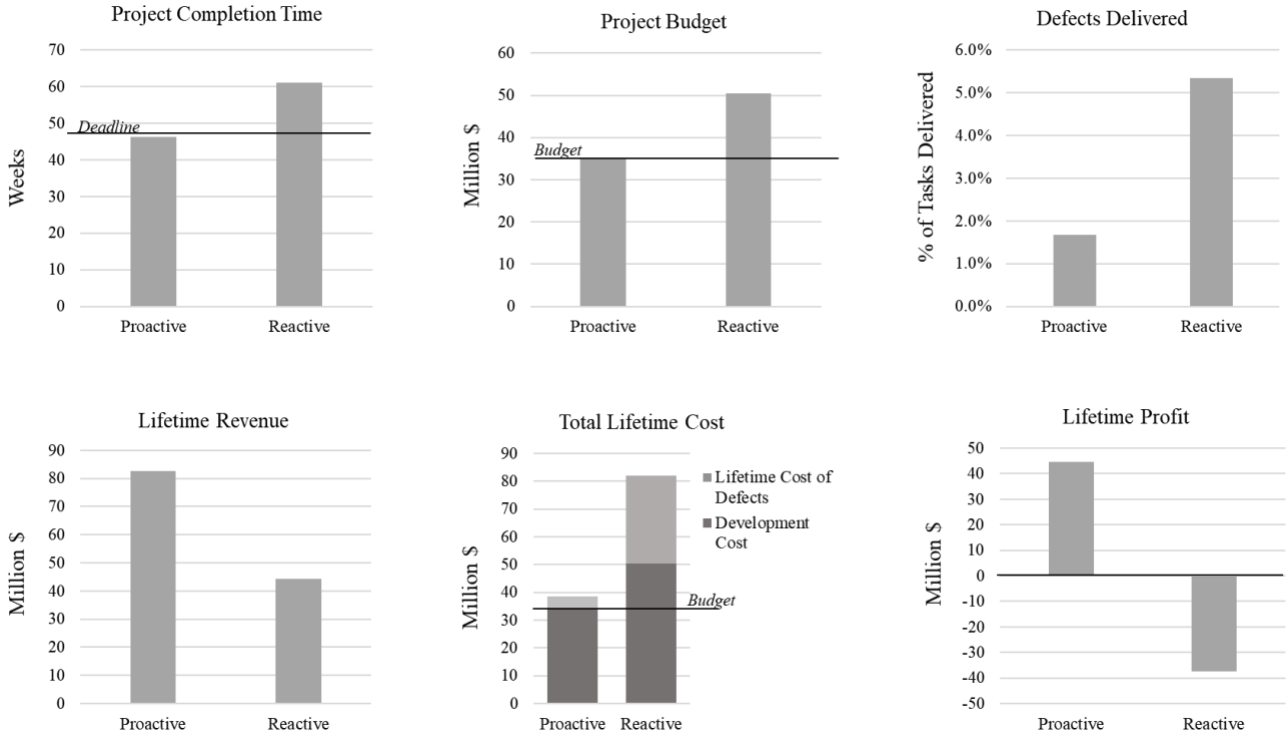

Figure 3. Reactive vs. Proactive project outcomes.

Experimental Results with Experienced Managers

We use the MIT Sloan Project Management Simulation to help people, from undergrads and MBA students to experienced managers, learn about the dynamics of complex projects. Here we report the performance of seasoned managers and engineers, many with extensive experience with complex projects in successful organizations. The participants were enrolled in an online executive education program on systems thinking offered by MIT. After exposure to various approaches to systems thinking and systems engineering, including the management of complex projects, they ran the project simulation using the product development scenario. Participants could play as many times as they liked, then summarized the lessons they learned in a memo to their superior describing how their real organization should scope and manage projects in the future.

Figure 4 reports the results from a session with 269 participants held in July 2020. We focus on results from the participants' first simulation, as it best reveals their initial mental models about how to manage projects effectively. Overall, participants did poorly. For time to market, 85.5% performed worse than the Proactive benchmark, and 2.6% were worse than the Reactive benchmark. Approximately 73% had costs higher than the Proactive benchmark, and 4.8% had higher costs than the Reactive benchmark. Only 15.6% released the product to the public with fewer defects than the Proactive benchmark, and 58.7% released the product with more defects than the Reactive benchmark.

Of course, there are tradeoffs among schedule, cost, and quality. Delivering the project earlier through extensive overtime and corner cutting degrades quality; improving quality may raise costs. Overall performance on schedule, cost, and quality determine the success of the project. Shockingly, none of the participants outperformed the Proactive benchmark on all three metrics. Fewer than 12% did better on two. Worse, the Reactive manager outperformed 64.3% of the participants on at least one metric.

Critically, participants focused on schedule and cost at the expense of quality. On average, participants' projects were 4.5% late and 8.2% over budget, but the average project was launched with an 11.3% defect rate, more than twice the rate achieved by the reactive manager and nearly seven times higher than the proactive manager. Such poor quality dooms the product in the marketplace. Schedule and cost are highly salient and feedback on them is continuously available, whereas quality is harder to assess. Like the reactive manager (Figure 2), participants who focused on controlling cost and hitting the schedule caused initial understaffing, excessive overtime, late hiring, corner cutting, inadequate testing, and other impacts that trigger the vicious cycles that lead to significant delay, disruption, and quality degradation.

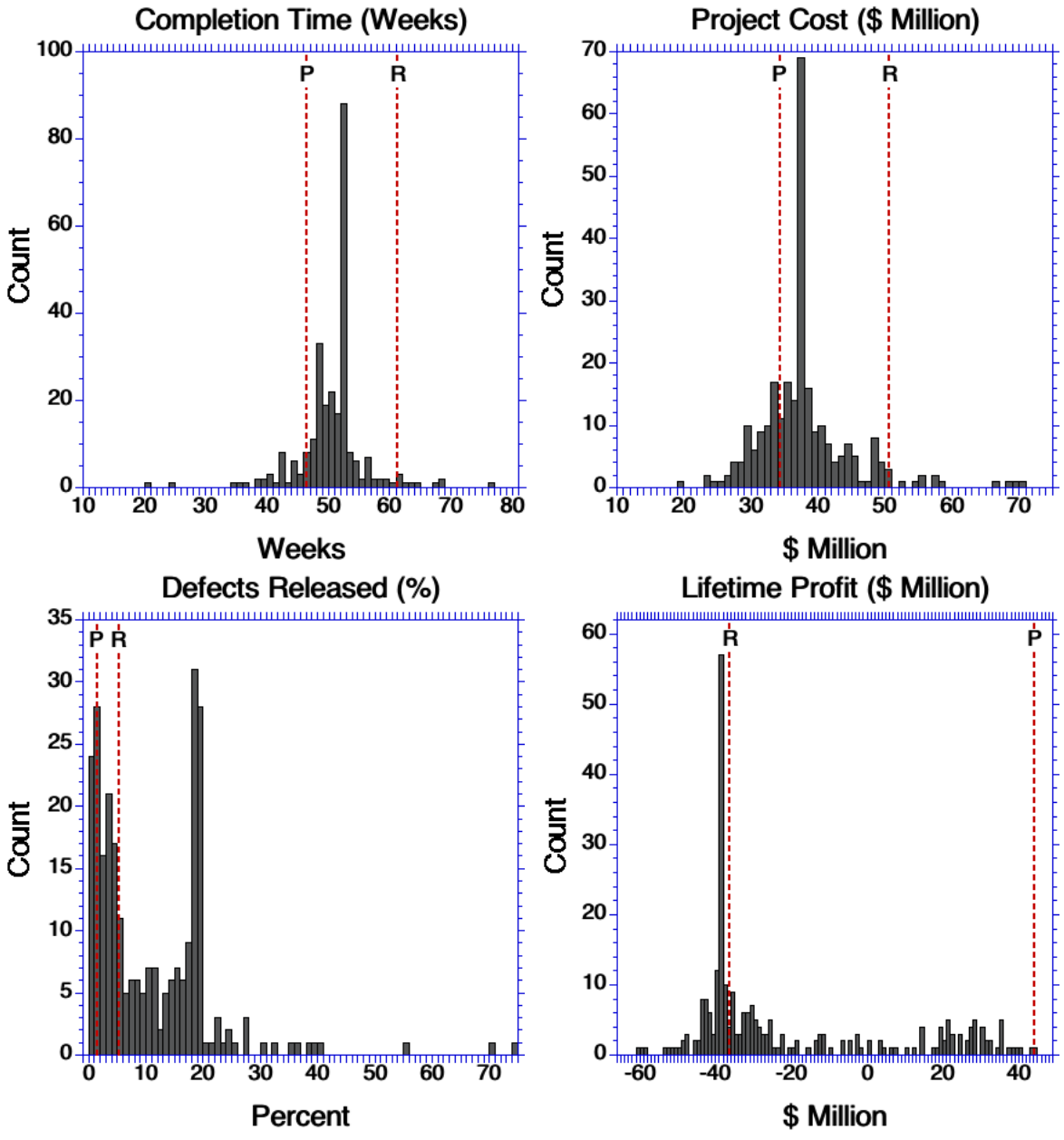

Figure 4. Participant results (N = 269) for schedule, cost, quality, and lifetime profit, compared to the Reactive (R) and Proactive (P) benchmarks.

Implications for Project Management

Project managers commonly overestimate productivity and work quality, often assuming there will be little or no rework, and request budgets and timelines that are inadequate. Relentless cost pressure means project managers are given smaller budgets and less time than they request and told to "do more with less." Consequently, projects often begin with unrealistically aggressive deadlines and inadequate resources and staff. The rush to complete the project as quickly as possible leads the project to start with scope ill defined, increasing the number of late scope changes imposed by customers, marketing, and project team members themselves. Schedule pressure builds, leading to excessive overtime and rapid unplanned hiring, triggering fatigue, corner cutting, inadequate testing, more work done out of sequence, and other impacts that slow progress, further eroding progress and leading to still more overtime, late hiring, and so on. These reinforcing feedbacks, described in the companion paper, become vicious cycles causing extensive delay and disruption. The project is delivered late, over budget, with low quality, and often, serious, even life-threatening defects released to the customer. Examples abound, including GM's defective ignition switches, Merck's Vioxx, Pfizer's Bextra, the Samsung Note 7, and the Boeing 737 Max 8.

Facing pressure to control costs, managers tend to rely first on overtime, hiring additional people only later. Doing so harms the project in multiple ways: extended overtime leads to fatigue, higher error rates, less effective testing, and increased attrition, requiring even more late hiring. Late, unplanned hiring dilutes experience, increases coordination problems, and leads to worksite congestion, degrading project performance before people can gain sufficient experience, build new networks of collaboration and trust, and optimize the worksite.

In contrast, superior performance requires decisions that are highly counterintuitive. Projects should be staffed initially at much higher levels to provide the resources needed to address rework and late specification changes, and thus limit overtime, corner cutting, inadequate testing and other harms these actions triggers. Detailed design should not start until the scope is firm, and late changes should not be accepted at all, or, if accepted, only during high-level design. Costs are initially higher, but lower overall.

Cutting corners or testing should never be tolerated. But building norms for high quality requires resources and staffing sufficient to prevent pressures causing people to cut corners and compromise testing, and then hide these decisions from others, creating a "Liars' Club" (Ford and Sterman 2003).

Why don't people learn to avoid these pathologies? Why don't more organizations provide their projects with sufficient resources and realistic schedules, avoid late changes, and create incentives, norms, and systems that prevent corner cutting and inadequate testing?

First, project performance depends on the nonlinear interaction of multiple feedbacks, accumulations and time delays. The mental models and software tools people use to manage projects often fail to include these critical feedbacks. Even if they did, the complexity of these interactions is far greater than people's ability to simulate them mentally. Instead, people tend to blame trouble on suppliers, customers, poor quality workers, or other "bad apples", not the complex feedback system in which well-intentioned, capable people are embedded, a system that creates powerful pressures that shape people's behavior.

Second, the outcome feedback people receive systematically perpetuates poor mental models and blaming behavior. Projects in trouble often trigger firefighting, with a few workers and managers working heroically to save the project (Repenning 2000, 2001). Such heroes are likely to be promoted, reinforcing culture and norms that reward firefighting. But where do these heroes get the time and resources needed to fight the fire? The answer is often cutting corners, testing, and quality; poaching people from other projects; and failing to document their work, coordinate with others, coach and mentor trainees, and invest in the capabilities needed for long-run success. The benefits of firefighting are immediate and highly visible, while the costs are diffuse and delayed. Firefighters who succeed will be rewarded and promoted, while the organizational capabilities needed to prevent future fires erode further, creating a

capability trap (Repenning and Sterman 2002). In contrast, building the capabilities for effective project management increases short run costs to individuals, even as doing so prevents costly project failures later. As one executive explained, "Nobody ever gets rewarded for solving problems that never happened" (Repenning and Sterman 2001).

These pathologies cannot be solved solely by more sophisticated tools that improve the productivity and management of complex projects, including automation, digitization, and artificial intelligence. Without a paradigm change in the mental models and behaviors of managers and workers, productivity and cycle time gains from technological innovation will simply lead to even more complex projects, more aggressive schedules, and tighter budgets. No matter how good our technology, pressure to "do more with less, faster" will keep causing projects to be launched with inadequate resources, triggering schedule pressure, unplanned overtime and hiring, fatigue and skill dilution, corner cutting and insufficient testing.

Indeed, despite unprecedented technological progress since the age of the pyramids, projects today continue to be late, expensive, and wrong, often with disastrous consequences. The designers of the Vasa, which in 1628 sank minutes into its maiden voyage, drowning approximately 150 people, would hardly recognize a modern vessel, and would be amazed by our aircraft. But they would understand well the intense pressures to meet aggressive targets that led to the crash of two 737 Max 8 aircraft minutes after takeoff, killing 346.

What is required to catalyze that paradigm change? Important questions for research and education remain. How much management simulation experience is needed to catalyze meaningful learning? How can people learn generalizable lessons that transfer from one project to settings with different scope, timeline, budgets, and market pressures? What kinds of feedback and coaching will help participants improve their understanding of complex project dynamics and use those insights in real projects? Simulation has long been essential to design and test complex systems, and to train people to operate them safely. It is even more important to use simulation to design better project management processes and help people learn how to manage complex projects safely and effectively. System dynamics provides the framework, simulation methods and models, and successful examples to help organizations transform the management of complex projects.

References

Abdel-Hamid, T. and S. Madnick (1991) Software Project Dynamics: An Integrated Approach. Englewood Cliffs, NJ: Prentice-Hall.

Boehm, B., Abts, C., Brown, A., et al. (2000). Software Cost Estimation with COCOMO II. Englewood Cliffs, NJ: Prentice-Hall.

Christensen C. M. (1997). We've Got Rhythm! Medtronic Corporation's Cardiac Pacemaker Business. Boston, MA: *Harvard Business School Press*.

Cooper, K. (1980) Naval ship production: A claim settled and a framework built, Interfaces 10(6), 20-36.

Cooper K. and K. Reichelt (2004). Project changes: sources, impacts, mitigation, pricing, litigation, and excellence. In The Wiley Guide to Managing Projects, Morris PWG, Pinto JK (eds). Wiley: Hoboken, NJ; pp. 743–772.

Cooper, K. and G. Lee (2009) Managing the Dynamics of Projects and Changes at Fluor. Ken Cooper Associates. http://www.kcooperassociates.com/files/SD_Paper_for_Reprint_V3.pdf.

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, The Journal of the Operational Research Society. 51(3), 291-300.

Ford, D. and J. Sterman (1998a) Expert knowledge elicitation for improving mental and formal models, *System Dynamics Review* 14(4), 309-340.

Ford, D. and J. Sterman (1998b) Dynamic modeling of product development processes, *System Dynamics Review* 14(1), 31-68.

Ford, D. and J. Sterman (2003a). Overcoming the 90% Syndrome: Iteration Management in Concurrent Development Projects. *Concurrent Engineering: Research and Applications* **11**(3): 177-186.

Ford, D. and J. D. Sterman (2003b). The Liar's Club: Concealing Rework in Concurrent Development. *Concurrent Engineering: Research and Applications* **11**(3): 211-220.

Ford D., Lyneis J. (2020) System Dynamics Applied to Project Management: A Survey, Assessment, and Directions for Future Research. In: Dangerfield B. (eds) System Dynamics. Encyclopedia of Complexity and Systems Science Series. Springer, New York, NY. https://doi.org/10.1007/978-1-4939-8790-0_658

Gartner Group (2020) "Gartner Survey Finds That 45% of Product Launches Are Delayed by at Least One Month." Accessed December 12, 2020. https://www.gartner.com/en/newsroom/press-releases/2019-09-09-gartner-survey-finds-that-45-percent-of-product-launches-are-delayed-by-at-least-one-month.

Godlewski, E., G. Lee, K. Cooper (2012) System Dynamics Transforms Fluor Project and Change Management. Interfaces, 42(1), 17-32.

House Committee on Transportation and Infrastructure, 2020. Boeing 737 MAX Investigation. https://transportation.house.gov/committee-activity/boeing-737-max-investigation.

Larson, R. & Larson, E. (2009). Top five causes of scope creep ... and what to do about them. Project Management Institute. https://www.pmi.org/learning/library/top-five-causes-scope-creep-6675.

Lee, Se Young. "Note 7 Fiasco Could Burn a $17 Billion Hole in Samsung Accounts." *Reuters*, October 12, 2016. https://www.reuters.com/article/us-samsung-elec-smartphones-costs-idUSKCN12B0FX.

Lee Z, D. Ford. N. Joglekar (2007) Resource allocation policy design for reduced project duration: a systems modeling approach. *Systems Research and Behavioral Science* **24**: 1–15.

Lyneis, J. and D. Ford (2007) System dynamics and project management: a survey, assessment and directions for future work, *System Dynamics Review* 23(2/3), 157-189.

Minerals Management Service 2010. Accident Investigation Report, Thunder Horse Platform. US Department of the Interior, https://www.bsee.gov/sites/bsee.gov/files/reports/safety/050708-pdf.pdf.

Parvan K, Rahmandad H, Haghani A (2015) Inter-phase feedbacks in construction projects. Journal of Operations Management 39:48–62.

Rahmandad, H., and K. Hu (2010) Modeling the rework cycle: capturing multiple defects per task. System Dynamics Review. 26(4), 291–315.

Rahmandad, H. and D. Weiss (2009) Dynamics of Concurrent Software Development. *System Dynamics Review.* 25(3), 224-249.

Repenning, N. (2000) A Dynamic Model of Resource Allocation in Multi-Project Research and Development Systems, *System Dynamics Review* 16(3), 173-212.

Repenning, N. (2001). Understanding fire fighting in new product development. *Journal of Product Innovation Management*, *18*(5), 285–300.

Repenning N., J. Sterman (2001) Nobody ever gets credit for fixing problems that never happened: creating and sustaining process improvement. *California Management Review* **43**(4): 64–88.

Repenning, N, and J. Sterman (2002) Capability traps and self-confirming attribution errors in the dynamics of process improvement. *Administrative Science Quarterly* **47**: 265–295.

SHRM (2018) "SHRM customized talent acquisition benchmarking report 2018." Society for Human Resource Management, 1800 Duke Street, Alexandria, VA 22314, USA. https://www.shrm.org/ResourcesAndTools/business-solutions/Documents/Talent-Acquisition-Report-All-Industries-All-FTEs.pdf.

Stephens C., A. Graham, J. Lyneis (2005) System dynamics modeling in the legal arena: meeting the challenges of expert witness admissibility. *System Dynamics Review* **21**(2): 95–122.

Sterman, J. (2000) *Business Dynamics: Systems Thinking and Modeling for a Complex World.* Boston, New York & Chicago: Irwin/McGraw-Hill.

Terwiesch C, Loch C. (1999) Managing the process of engineering change orders: the case of the climate control system in automobile development. Journal of Product Innovation Management 16(2):160–172.

Ullah, I., Tang, D., Yin, L. (2016) Engineering product and process design changes: A literature overview. Procedia CIRP 56: 25-33. https://doi.org/10.1016/j.procir.2016.10.010.

US BLS (2020) Median years of tenure with current employer for employed wage and salary workers by industry, selected years, 2010-2020. US Bureau of Labor Statistics, Economic News Release, Table 5. https://www.bls.gov/news.release/tenure.t05.htm.

Vanttinen, P. 2020. The never-ending saga of Finland's Olkiluoto nuclear plant. https://www.euractiv.com/section/all/short_news/the-never-ending-saga-of-finlands-olkiluoto-nuclear-plant/.

Williams, T, C. Eden, F. Ackerman, A. Tait (1995) The effects of design changes and delays on project costs, *The Journal of the Operational Research Society*, 46(7), 809-819.

World Nuclear News 2018. Olkiluoto 3 EPR parties agree settlement. https://www.world-nuclear-news.org/C-Olkiluoto-3-EPR-parties-agree-settlement-12031801.html (12 March 2018).

World Nuclear News 2022. Automation adjustments delay OL3 grid connection. https://www.world-nuclear-news.org/Articles/Automation-adjustments-delay-OL3-grid-connection (14 February 2022).