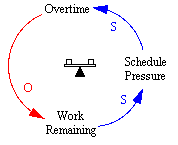
**Project Systems**

**Distinguishing Fact from Fantasy**

A common mental model relative to software reliability is ... "Yes, we could design highly reliable software. The problem is we can't afford it. It will take too long and cost too much. Our most important priority is to get the product to market. If we don't get our product to market quickly, we won't make sales and profit; and, after all, profit is what counts the most."  
  
Based on each of our own personal experiences, this really does make sense. In every personal project we've ever undertaken, we've discovered that making a higher quality product requires more care and therefore more time. The problem is that extending our personal experience to an organizational level is often not valid. In fact, it is quite common that the system-level consequences of our actions are extremely counter intuitive.  
  
What if designing reliable software is actually less expensive at an organizational level than designing lower quality software? What if, as asserted by Putnam & Myers in "Measures for Excellence", "When productivity improves, errors seem to decline, or, as others put it, when more emphasis is put on quality, productivity increases."?  
  
If producing higher quality software does result in higher overall organizational productivity, this would change the entire approach our organizations take in producing software. As a consequence, what would be the effect if we were to consider an alternative mental model?  
  
Such a model might be: "Yes, we design highly reliable software because we can't afford to not do so. It takes too long and it costs too much to correct defects once they're in the product. If our engineers aren't fixing problems in response to customer problems, they can be creating new products and features. After all, our most important priority is to make sales and maintain customer satisfaction to produce the highest possible profits."  
  
The end result desired based on both of these world views is the same. How we endeavor to accomplish this result, and our success in doing so, depends on our mental model of the software development process.

**Systems Thinking and Software Project Management**

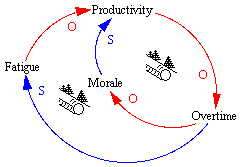
The system of interactions in a software development project is a complex web which we must understand to avoid the unintended and counter intuitive consequences that that cause project overruns and even project failure.  
  
The frightening reality is that many software development projects get caught in what's known as the "Impossible Region." Causal loop diagrams of the complex web of effects of such things as, for example, overtime and rapid staffing effects on project quality, schedule and cost show that projects are not just "problems to be solved," but "messes" in the truest sense of the word. We greatly underestimate the magnitude of the unintended consequences of such interactions.

  
**Figure 1**

This initial representation indicates that at the **Work Remaining** with regard to the current project schedule increases it will tend to produce **Schedule Pressure**.

This **Schedule Pressure** with tend to promote **Overtime** to reduce the **Work Remaining** with regard to the current project schedule.

This structure represents a balancing loop where **Overtime** is used to counteract the **Work Remaining**.  
  
The following structure points out a couple unintended consequences of increasing **Overtime** to combat the **Work Remaining** situation.

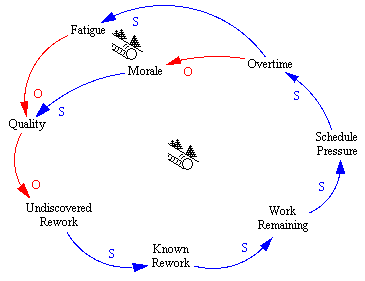
  
**Figure 2**

While the **Overtime** is intended to counteract the **Work Remaining** it has a couple additional influences.

If **Overtime** increases sufficiently it will begin to depress **Morale**, which will subsequently influence **Productivity** to decrease.

The decrease in **Productivity** will then tend to increase the **Overtime** required. This structure represents a viscous reinforcing loop moving opposite to the direction desired.  
  
At the same time an increases in **Overtime** influence **Morale** to decline it is promoting an increase in **Fatigue**.

The increase in **Fatigue** will then tend to reduce **Productivity** further increasing the **Overtime** required. What we have is another viscous reinforcing loop moving opposite to the direction desired.  
  
If these unintended consequences weren't enough annoyance, the following structure points out a couple additional unintended consequences.

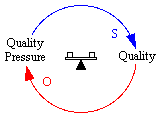
  
**Figure 3**

While **Morale** and **Fatigue** are influencing **Productivity** declines, **Morale** and **Fatigue** also have the nasty habit of influencing **Quality** to decline.

The decline in **Quality** is often not immediately realized but enters into the structure as **Undiscovered Rework**. This **Undiscovered Rework** is work that needs to be done, we just don't know about it yet as it's undiscovered.

As **Undiscovered Rework** increases it will tend to influence us to find more of the rework thus increasing the **Known Rework.**

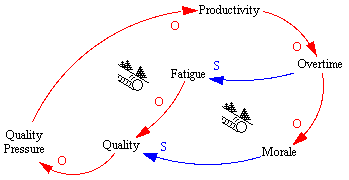
An increase in **Known Rework** just serves to increase the amount of **Work Remaining** which increases **Schedule Pressure**, and subsequently increasing the need for **Overtime**. Here we are again, with two viscous reinforcing loops taking us in exactly the direction we don't want to go.  
  
A short time ago we addressed the manner in which decreases in **Morale** and increases in **Fatigue** tend to cause **Quality** to decline. The following structure alludes to another influence resulting from a decline in **Quality**.

  
**Figure 4**

As **Quality** declines, and once the decline is realized, there will be an increased focus placed on **Quality**.

This is represented in the above diagram by **Quality Pressure**.

As **Quality Pressure** increases it will tend to increase the resultant **Quality**. Thus we finally have a balancing loop which moves something in a desired direction.  
  
Yet, don't get too comfortable for the following structure provides an insight into an unintended consequence of **Quality Pressure** which isn't quite so beneficial.

  
**Figure 5**

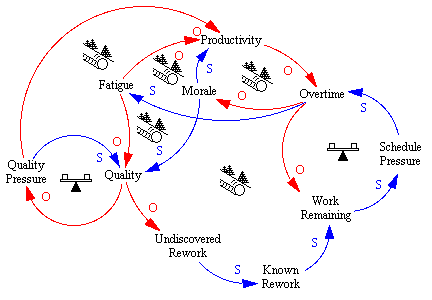
Once a decline in **Quality** is realized there is an increased emphasis on **Quality**, i.e. **Quality Pressure**.

This increase in **Quality Pressure** will serve to improve **Quality** as shown in Figure 4. Yet, an increase in **Quality Pressure** also serves to decrease **Productivity** because there is more of an emphasis on getting it right than getting it done.

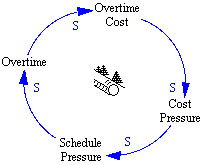
So this decline in **Productivity** serves to promote more **Overtime** which increases **Fatigue** and decreases **Morale**.

The end result being a tendency for **Quality** to decline. Thus we have two more viscous reinforcing loops which simply indicates that the more **Quality Pressure** applied the more will be needed.

Doesn't seem to make much sense does it? Welcome to the dysfunctional reality of organizational life!  
  
When we merge the structures in Figures 2, 3, 4, and 5 with Figure 1 we end up with Figure 6. The current elaboration of the understanding.

  
**Figure 6**

Before you let yourself become overwhelmed by the complexity of this diagram you had best fasten your seat belt as we're only about half way there.  
  
**Overtime** has this real nasty habit of costing more than regular time so there are some implications of increasing **Overtime**.

  
**Figure 7**

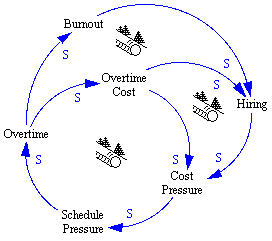
An increase in **Overtime** brings with it an increase in **Overtime Cost**.

As **Overtime Cost** increases there is an increased emphasis on cost which shows up as **Cost Pressure**.

The **Cost Pressure** is interpreted by the management of project in such a way that it shows up as additional **Schedule Pressure**.

This increased **Schedule Pressure** then leads to even more **Overtime**.

Here we have but one more viscous reinforcing loop in which actions influence the overall effect to be just the opposite of what is desired.  
  
Overtime and Overtime Cost have a couple more influences.

  
**Figure 8**

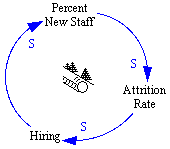
Prolonged **Overtime** has a tendency to lead to **Burnout** which means **Hiring** must occur to replace or augment resources.

Yet **Hiring** only serves to increase **Cost Pressure** also, creating another viscous reinforcing loop.  
  
Also, in an attempt to minimize **Overtime Costs** additional resources are hired. And, because of the time delays involved, **Hiring** only serves to increase **Cost Pressure**.

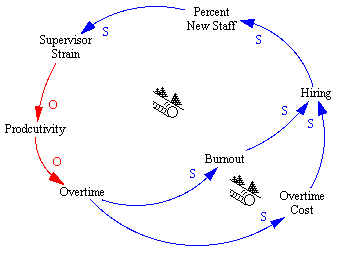
We there for have another viscous reinforcing loop driving **Cost Pressure** to increase **Schedule Pressure** leading to more **Overtime**.

Does it sound like things are going down hill fast?  
  
Now as Fredrick Brooks stated in "The Mythical Man Month" more than 20 years ago, "Adding additional resources to a late software project only makes it later," has a very solid foundation.

What follows are some of the unintended consequences of **Hiring**.

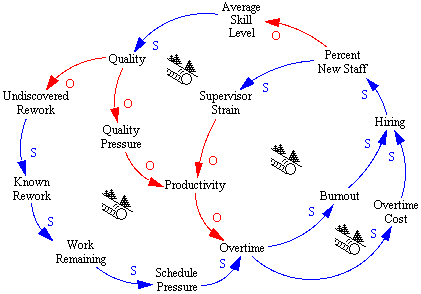
  
**Figure 9**

**Hiring** serves to increase the **Percent New Staff** which tends to increase **Attrition Rate** which simply servers to require more **Hiring**. You guessed it, another viscous reinforcing loop.

  
**Figure 10**

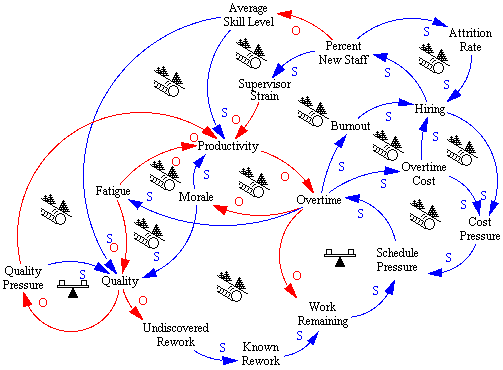
As the **Percent New Staff** increases it tends to produce **Supervisor Strain**.

As **Supervisor Strain** increases it influences a decline in **Productivity** and an increase in **Overtime** and we're back to the same part of the model presented in Figure 8. Yes, but another influence which is part of two viscous reinforcing loops. Are you beginning to feel there is no hope in sight?  
  
Percent of New Staff has another influence just as miserable as described in the next figure.

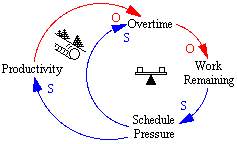
  
**Figure 11**

As **Percent New Staff** increases it decreases the **Average Skill Level** of the resource pool.

This has a tendency to decrease **Quality** which feeds right into the viscous reinforcing loops described in Figure 3 and Figure 5.  
  
Now when we combine the implications developed in Figures 7 thru 11 with Figure 6 we have a nightmare even I'm not happy looking at.

  
**Figure 12**

Schedule Pressure has a couple additional influences that should be mentioned.

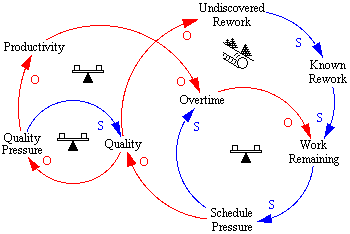
  
**Figure 13**

**Schedule Pressure** serves to increase **Overtime** thus reducing the **Work Remaining** and finally decreasing the **Schedule Pressure**.

This balancing loop is supported by a virtuous reinforcing loop as **Schedule Pressure** tends to increase **Productivity**.

This increase in **Productivity** then tends to decrease **Overtime** increasing the **Work Remaining**.

This increase in **Work Remaining** then supports the continued **Schedule Pressure**.  
  
**Schedule Pressure** also has an effect on **Quality**.

  
**Figure 14**

**Schedule Pressure** serves to influence **Quality** to decline.

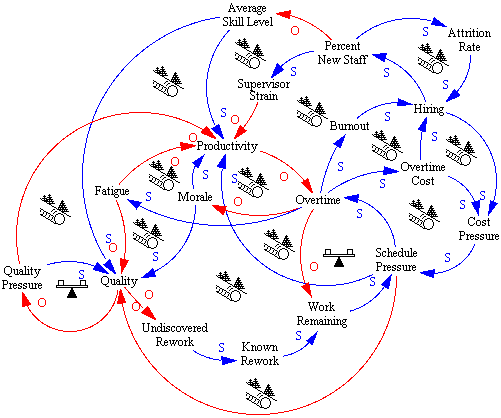
This decline in **Quality** results in an increase in **Quality Pressure** which serves to decrease **Productivity** resulting in an increase in **Overtime**.

The increase in **Overtime** then serves to reduce the **Work Remaining**. This is a balancing loop such that an increase in **Schedule Pressure** tends to reduce **Schedule Pressure**.

The decrease in **Quality** due to the increase in **Schedule Pressure** serves to increase the **Undiscovered Rework** thus increasing **Known Rework** and the **Work Remaining**.

The increase in **Work Remaining** influences an increase in **Schedule Pressure**.

This is a viscous reinforcing loop where an increase in **Schedule Pressure** tends to influence additional **Schedule Pressure**.  
  
Now, combining the structures in Figure 13 and 14 with Figure 12 we have:

  
**Figure 15**

If this is reality is it any wonder we have such difficulty getting projects done on time and within budget?

**Reality Revisited**

Our standard approaches for managing and controlling projects (including reviews, work breakdown structures with earned value-based tracking, and PERT/CPM and Gantt scheduling) are not adequate to understand, and guide us to prevent, problems caused by these dynamics. Using them is like driving by looking in the rearview mirror.  
  
For example, projects often get a bad start due to underestimating the effort and the time required. Project underestimates often end up causing seemingly never-ending difficulty and would cause Mr. Rogers to ask, "Can you say Death Spiral?" Underestimates can put projects in what might be called the "Dead Meat" region where they are subjected to large and simultaneous quality, schedule, and cost pressures.

This region is larger than one might think because the effort required on a project goes as the cube of the code size and the inverse fourth power of the development time (see "Measures for Excellence" by Putnam and Myers, 1992). Seemingly minor underestimates in code size and/or duration-required can cause a major underestimate in the effort required.  
  
While managers have little control over projects, they do have great influence in avoiding the unintended and counter intuitive consequences that that cause projects to falter. Systems thinking can help managers, engineers and programmers understand the dynamics of project system, their part in the system, and the varieties of policy feedback that cause project performance problems.  
  
Such a systems perspective sheds light on what doesn't work, and on what does work, in managing software projects. For example, demanding excessive overtime and hiring personnel too rapidly definitely don't work because they have an adverse impact on quality and productivity -- and ultimately on project schedule and cost.  
  
Among the things that work are to

* do excellent planning including product specs, project plans and test plans before starting development,
* guard band schedule beyond the minimum development time because, for example, a 15% schedule guard band saves ~50% in required manpower,
* identify independent, parallel development opportunities because two decoupled sub-projects take about one quarter the manpower of one large project of the same size,
* test as soon as possible to avoid the effect of defects on downstream code, and
* before the project starts, identify optional functions that can be worked on later, or dropped, if the project gets in trouble.

Special thanks to Bob Powell, Ph.D., MBA, for contributing the conceptual foundation for this article. Bob is a consultant in continuous improvement and learning organizations based in Colorado Springs. Contact him at (719) 599-0977 or at [scuba@usa.net](mailto:scuba@usa.net)