Laws of Software Motion

The theme of this issue deals with consulting. As consultants, the authors know that consulting is all about leverage and motion. Of course, being card-carrying geeks (punch cards, that is), the authors immediately made the connection between motion and Newton. Newton, after all, formulated the "Laws of Motion."

Newton's laws govern the movement of the universe. Unfortunately, we've noted that software doesn't seem to follow Newton's Laws of Motion. In fact, software often seems not to follow any laws of motion at all. Thus, the authors have developed what we modestly call the "Cook-Leishman Laws of Software Motion."

These so-called laws can provide understanding and guidance to program managers, developers, and users. These laws help indicate the relationships between time, cost, effort, and program progress.

Law No. 1

Newton's First Law – An object in uniform non-accelerated motion (or at rest), will remain in the same state of motion unless an outside force acts upon it.

Cook-Leishman's First Law – Any software intensive program not given adequate force (motivation) will degrade and cease to progress.

Any program slowly making progress at a steady rate will eventually grind to a halt unless continual outside force is applied to it. Programs will not coast. It takes active program/project management and focused risk management to keep a program continually and successfully proceeding [1]. If you do not have an active risk management plan, you run the risk of having your program stall.

If by some rare chance a program quits making progress, it takes a lot of effort to get it moving again. In fact, lack of progress usually means that programs tend to degrade. Personnel are reassigned, budgets shrink, and out-of-sight, out-of-mind thinking takes over.

Law No. 2

Newton's Second Law – There is a relationship between force, mass, and acceleration that is specified as F = ma.

Cook-Leishman's Second Law – There is a relationship between the forces required to accomplish a software intensive program/project. This relationship correlates requirements (R), changes (C), external influences (X), and timing (T) (schedule), thus F = f(R,C,X,T) where f stands as function that relates R, C, X and T.

Often, in the real world, there is actually an inverse relationship between the software force F and R, C, X, and T. In large programs that are well staffed and well funded, relatively small efforts or influences can make a change; this is due to the fact that the small changes require relatively minor effort with respect to the entire program. On the other hand, in small programs, small efforts or influences might reflect major changes to the entire effort. In addition, in some programs, major efforts have to be made to implement relatively small changes.

Changes can occur based upon outside forces (changes) that seem to have no influence upon your program. With interservice programs, multiple users and multiple sources of requirements can have ripple effects in which insignificant events can have major impacts upon your program.

In the first law, risk management (and risk mitigation) is required. In the second law, configuration management is the major player [2]. Without a configuration management plan that puts you in proactive instead of reactive mode, changes will eventually stop forward progress. At this point, Law No. 1 will apply.

Law No. 3

Newton's Third Law – For every action, there is an equal and opposite reaction.

Cook-Leishman's Third Law – For every action, there are varying reactions. Some are small, and some might have exponentially greater force. You really don't know what the consequence will be. The same action at a different time might have totally different consequences.

Small changes early in the program have minor consequences. The same changes during design might cause relatively major rewriting of both code and design. During integration and systems testing, the changes might cause a catastrophic delay in final delivery. Timing is everything. Attempts to improve quality might have positive effects one time, but the same attempt might not help at a different time (or for a different organization). You have to keep the end product in focus, and remember that the process has to be tailored for each product, not vice versa. [3] To put Law No. 3 in a nutshell, "Timing is everything."

Are these laws really laws? Of course not! In reality they are, at best, general guidelines or hints. They are based on an examination of many software projects under varying constraints. Certainly, software development is far too complex to be summed up in a few so-called laws. After all, even Newton wasn't totally correct[1].

However, these so-called laws do help provide general guidelines to help you keep your program moving forward. Have a strong risk management and risk mitigation plan. Implement configuration management, and use risk management and configuration together to proactively predict upcoming changes, thus mitigating major cost and schedule impacts. Focus on the product and quality, and modify the process to accommodate the needs of your program.

To sum it up, you can help your program greatly by applying Cook-Leishman's Fourth Law.

Law No. 4

Cook-Leishman's Fourth Law – Read (and heed!) CROSSTALK regularly.

-Dr. David A. Cook
AEgis Technologies Group
and
-Theron Leishman
STSC/Northrop Grumman

References
1. Leishman, Theron R., and Dr. David A. Cook. "Requirements Risk Can Drown Software Projects." CROSSTALK Apr. 2002: 4.
2. Leishman, Theron R., and Dr. David A. Cook. "But I Only Changed One Line of Code!" CROSSTALK Jan. 2003: 20.
3. Cook, Dr. David A. "Confusing Process and Product: Why the Quality Is Not There Yet!" CROSSTALK July 1999: 27.

Note
1. The authors can see the e-mail now. Yes, Newton was correct given the understanding of physics at the time. Quantum theory and Einstein have changed a few things. Newton's laws do not apply either at relativistic speeds (near the speed of light), or at the very small level, where quantum mechanics must be used. Newton's laws, of course, do apply to all generally observed behaviors.