OBSERVATIONS FROM THE ANT HILL: WHAT ANTS AND SOFTWARE ENGINEERS HAVE IN COMMON

Joe Schofield, CQA, CFPS

Ants and people, ant work and software engineering; they are not as unrelated as one might think. Perhaps the ideas in this article provide an *ant*idote for your organization or project; or perhaps the challenges presented in this article that confront software projects are the *ant*ithesis of your own organization, and you do not *ant*icipate any immediate changes. Regardless of the outcome, you now have another context in which to examine the people and productivity in your organization. Do not be too surprised if you pay closer attention the next time you pass an ant hill.

JOE SCHOFIELD, CQA, CFPS, *is a Distinguished Member of the technical staff at a national laboratory in New Mexico. One of his current assignments is guiding an organization of about 150 software engineers toward SW-CMM® maturity which provides personal opportunity for thinking and learning about software process improvement. He is a CFPS, a CQA, a faculty member at the College of Santa Fe, and a licensed coach in the State of New Mexico. Joe enjoys writing and presenting, as evidenced by dozens of publications.*

SOME OF LIFE'S BEST LESSONS CAN BE found in the most unexpected places. You might expect to find an array of diet books at the grocery checkout or a police sobriety checkpoint on a busy holiday. But would you really expect to discover parallels between the behavior of ants and software developers? Probably not until now.

ABRIDGED RELEVANT HISTORY

In 2001, an organization within Sandia National Laboratories investigated the value of incorporating the Personal Software Process (PSP) and the Team Software Process (TSP)[1] in its process improvement efforts. The organization of approximately 150 software engineers was already committed to using the SW-CMM[2] (Capability Maturity Model for Software) as a framework for improvement. The PSP, TSP, and SW-CMM are products of the Software Engineering Institute and are portrayed as working together to accelerate organizational process maturity.

The PSP classes were introduced to enhance personal skills while SW-CMM Level 3 processes were being formalized. The structure of the PSP class requires students to build nine software programs. The programs reinforce the concepts taught in the class, such as estimating, sizing, and statistical analysis of the software. These principles are easily applied to other areas of software engineering, such as defect density and productivity. The principles are fundamental to understanding software process measurement, and then improvement.

The classroom experience also provides a forum for collecting other data of interest; in this case, lines of code measures. The subsequent statistical analysis of this data presents serious challenges to using lines of code as a reliable predictor of size or function. The PSP course provides a fertile, well-controlled laboratory for validating the reliability of, or lack thereof, counting lines of code. In contrast, other software groups may find that the number of screens, buttons (functions) on a screen,

The collection and analysis of the lines of code from the PSP class exercises also inspired the somewhat absurd notion of counting ants to predict software size.

database tables or objects, and number of reports provide more useful estimating primitives. Alan Albrecht realized this over 30 years ago when he introduced *function points*. Studies by the Development Support Center,[3] MIT, and Eckerd Drug found consistency in different parties counting applications of up to 19K function points to be accurate within .7, 10, and .7 percent, respectively.

The collection and analysis of the lines of code from the PSP class exercises also inspired the somewhat absurd notion of counting ants to predict software size. The analysis of the size of the programs written in the PSP class indicates a variation in product size as great as 22:1.[4] To put this number in context, consider the following aspects of the PSP course:

- Nine programs are developed in the PSP course; only instantiations of the same program are used for size variation comparisons in this study; that is, only program assignment number 1s are compared to other program assignment 1s.
- Each program is written according to a predetermined specification.
- Each program is counted using the same counting guidance; that is, blank and comment lines are not included in the counts.
- Each programmer writes only one instantiation of a program, but reuse is encouraged in the course.
- In one set of programs written with the same language, program 2, which counts the lines of code in a program, was used to count the other instantiations of the program written in that language. In no case did any programmer over-count lines of code compared to the average count from the other peer programmers. Thus, there is no inflation of the number of lines of code written as a result of an error in the lines of code counting software.
- The programmers did not know that their programs were going to be included in a study of the utility of counting lines of code.

Armed with the line counts of several students, in several classes, and classified by language, and in some cases, education level of the software engineers, these numbers were loaded into a spreadsheet where some comparison, averages, and variances were calculated. With variances as high as 2200 percent, a statistician was asked to verify the "normality" of the data, the degrees of confidence, and coefficient of variation. The statistical analysis of the normalized data confirmed the lack of reliability of using lines of code as a predictor of size or effort. Then came the ants!

I began to pay attention to a colony of ants outside my office. I was curious to determine if there was less variation between the ants I could observe than with lines of code. I sampled the ants over a period of two months. Extrapolating the ants counted, I postulated, perhaps the ants would be a better predictor of size than KLOCs (thousand lines of code). As I finish this article, the variation of ants statistically (i.e., the coefficient of variation) is greater than that of the lines of code; that is, the number of ants outside my office are not a more reliable predictor of software size.

But there was much more to be learned through these samples. While less quantitative, the lessons learned were of immediate interest and value. Thus, the context for the "observations from the ant hill" that follow.

OBSERVATIONS FROM THE ANT HILL

Ants and Progress

Sometimes you do not see any activity and you believe nothing is getting done — and you are right. The irony of the saying "work smarter not harder" applies here. Ants have about 250K brain cells; humans about 10 billion. While it takes about 40K ants to equal the brain power of one human, this is not always obvious; that is, sometimes just a few ants look to be superior. Use measurable progress results and mature project management practices such as *earned value* to track performance. Use reliable sizing measures such as *function points* to assess progress towards customer needs. Avoid the practice of counting lines of code as a measure of size or progress. Writing lines of code might be best equated to riding a stationary bike — lots of activity but not necessarily going anywhere.

With 10K species of ants, and some with 700K members in a colony, one would think we could learn something about managing project tasks from ants. Ants live a few weeks to a couple of years, usually. This workforce churn is typically higher than occurs for most software projects. Software projects last from a few weeks to a couple of years. Because most software engineers live long enough to contribute to multiple projects, they should be able to plan and execute better than ants.

Some ants get "out of the box" when they see something interesting. Some software engineers may dabble outside the scope of the project if it interests them. "Out-of-the-box"

Usually, higher people churn rates diminish productivity for software teams.

software engineers are sometimes the source of creeping requirements and technology churn. I once observed a manager accuse a team of expanding the scope of the project without the customer's input. The team, he suggested, was improvising on requirements by adding business rules to the process model. Indeed, the team had been reporting progress, just not the progress expected under the project plan. Brooks calls this growth "fills" and "embellishments" in his classic book *The Mythical Man-Month*.

You may not see ants for days. You may not be able to see progress with software developers for days either. Sometimes, this is called slack, or training, or status reporting, or something. But you better know what to call it! According to a study conducted by the Standish Group in 1995, one third of projects were canceled and more than one half "challenged" — many due to lack of progress.

Some ant holes are bigger than others; some software projects are bigger than others. Bigger holes accommodate more ants, and bigger projects seem to accumulate more staff, needed or not. Sometimes, smaller holes are a better fit for team members. Be wary of projects whose staff numbers grow faster than requirements dictate. Large projects often assume inordinate overhead. Michael Hammer refers to this as "non-value-adding work, and sometimes as waste!"

The number of workers and amount of activity do not necessarily expedite the completion of any given task. Heuristics reveal that the ants collide and step all over each other. Teams without configuration management practices will often do the same thing with their software changes.

Some ants spread out in random fashion. Random fashion, random work, random results — this sequence requires abundant good fortune for sustained success. A documented process is more likely to bring consistent results; and when combined with continuous process improvement, desired results.

When you watch ants, someone will likely wonder, "what are you doing." This is also a question that gets asked of software project leaders; but the answer should be different. Project teams must be able to demonstrate and report actual progress — not overstating of progress, profits, or performance. When my friend Dave and I were "shooting" the digital images that would inspire these lessons, two people who once worked with us spotted us hunched over the ant bed. "What are you doing?" they smirked. "What's it look like?" I retorted, "We're taking pictures of ants for an article on software productivity."

Ants and People

Ants are classified by biologists as a type of wasp (*hymenoptera formicidae*). Humans are classified as people (*homo sapiens*). The Slave-Maker ant (*polyergus rufescens*) steals pupae from other ant nests to get workers. Software engineers steal workers from other projects; this practice is labeled matrixing or leveraging. When we steal preexisting software, we call this reuse. In either case, avoid overusing key people resources — they may begin to feel like overcommitted slaves.

Ants have meetings. This fact is the most discouraging news I uncovered in my ant research. When one ant wants to communicate with another, it merely taps the other ant on its head with its antennae. Avoid both these practices: meetings, and tapping other team members "upside" their heads.

The number of worker ants may change quickly; this does not mean that more work is getting completed. Usually, higher people churn rates diminish productivity for software teams. Churn rate is a useful metric for gaining insight into process stability and predictable performance. Brooks' *The Mythical Man-Month* supports the delays triggered by people churn.

Some ants are married. (This "stretch" is not really true.) Their spouses are called uncles. This has nothing to do with software, except when software engineers get married; their spouses are called "lonely." Teams stuck on overdrive are overdue for a crash. Healthy lives outside of work are important too. In their classic book *Peopleware*, DeMarco and Lister describe an employee who "calls in well." After sustained on-the-job "illness," one day the employee realized that his work was the cause of his affliction. Finally one day, the employee was "well" and called-in stating he would not be in "anymore."

Ants and the PSP

Ants reportedly carry up to 20, 50, or 100 times their weight, depending on the source. If you find a worker like any of these, treat that worker really well. Despite all the intrinsic and alternative compensation approaches, money still speaks loudly. The PSP requires team members to document their work time and interrupts,

Do not be too surprised if you pay closer attention the next time you pass an anthill.

which helps team members to focus on commitments and carry their weight.

The observation of a couple of ants side-by-side seemed the perfect reminder for mentoring. Some team members need someone by their side to mentor them. Do not neglect training needs and organizational opportunities to mentor. Mentoring is an attitude, not just a formal program.

Ants often discard items they collect. Sometimes, the seeds they obtain one day, they dispose of the next. Project teams often discard technologies and tools faster than ants discard seeds. Select tools carefully and avoid technology churn. Discourage religious wars over tools, methodologies, and life cycles. Because the PSP tracks interrupts, team members are likely to be more task focused.

An ant may go in one hole and out another; this apparently is its job. Compare the hole to the door in your facility. Some employees are present during work hours — but they are not contributing. They come in one door and leave through another. Movement and progress are not synonymous. PSP weekly reporting at the individual level provides a mechanism for tracking contribution.

MISSED THE POINT?

Ants and people, ant work and software engineering; they are not as unrelated as you might have thought. Perhaps the ideas in this article provide an *ant*idote for your organization or project. Or perhaps the challenges presented in this article that confront software projects are the *ant*ithesis of your own organization, and you do not *ant*icipate any immediate changes. Regardless of the outcome, you now have another context in which to examine the people and productivity in your organization. Do not be too surprised if you pay closer attention the next time you pass an ant hill.

I refined many of the thoughts in this article while on vacation; more specifically, poolside at a Las Vegas resort. This environment triggered the possibility that software measures might also have a relationship to certain human physical dimensions. However, I have not yet been able to devise a non-invasive technique for obtaining the measurement data need for comparison. Stay tuned! ▲

Notes

1. PSP and TSP are service marks of the Software Engineering Institute
2. The CMM is a registered trademark of the Software Engineering Institute
3. The author gratefully acknowledges the assistance of Bill Huffschmidt of the Development Support Center for the information on the Function Point studies.
4. The author gratefully acknowledges the assistance of Iraj Hirmanpour, as associate with the Software Engineering Institute for the data from the PSP courses. Without this data, analysis would not have been possible.