corporate culture (and process) led the team leader, Don Estridge, to relocate the team as far away from Armonk as he could [2]. Even in the most prosaic software development, there is always some element of creativity that requires this freedom. The most creative of knowledge acquisition always addresses the 2nd Order Ignorance (2OI), as we find out things we didn't know that we didn't know.

The learning team is a research group. The output is some intellectual model of whatever has been, is being, or must be, learned. The key need for this is the construction of shared mental models and cognitive languages between team members. Constructing these models has strong elements of creativity, but does not necessarily produce a product; often the output is a better idea of what *might* work. For true research groups, most of the effort is spent in finding approaches or processes that will unveil what is not known. Therefore, this kind of team is focusing on the 2OI–3OI levels.

The challenge for most software teams is that all types of processing are necessary. Usually a software team is not just one of these. It may be a learning team in gathering requirements, experimenting with different ways of interacting with the customer or environment. Then it may be a creative team in designing the system, tactical in implementing it, and problem-solving in debugging it. This is why defining software processes is so difficult, and why rigid processes do not work well.

The approach to defining process must take these issues into account. The peril of not doing so is that the process will force the wrong answer.

Putting This All Together

We can summarize these points as follows:

- We can only have a well-defined process for those (0OI and 1OI) things that are well defined and we have done before.
- We cannot have a well-defined process for things (2OI and 3OI) we do not know how to do.
- Defined process only really supports tactical and problem-solving activities, and may be harmful in creative and learning activities.
- Almost all software projects have elements of tactical implementation that can benefit from well-defined development procedures, creativity, and learning hampered by restrictive process.
- Non-restrictive processes (guidelines) are necessarily vague and therefore are not much use.
- The ideal process applies at the wholly predictable and mechanical level.
- The purpose of process is to liberate us from the mundane.

Therefore, we can assert that the purpose of process is to take those things we already know how to do successfully, and to package them so they are entirely predictable and invariably successful. The process serves to free up our brain from reprocessing those things that really do not need to be reprocessed.

What Does This Leave Us Doing?

The freeing of brain cycles from the repetition of the more predictable and mechanistic activities of our projects gives us the bandwidth to work on the unknown aspects of building systems. These are the unstructured, unpredictable, undefined, variable, new, and, yes, creative activities for which we cannot have a well-defined process. In OI terms, the job of dumb process is to take care of 0OI and some of 1OI. Our job as clever and inventive developers is to take care of 2OI and 3OI. The fact that software developers have a high growth need strength or need for personal development (read: learn new things) [1] is evidence of this.

To be free to create, of course, is exactly what engineers and developers want. The fact that they often view process as preventing them from doing this is probably the most legitimate criticism of process as it is often done. C

References

1. Couger, D.J and Zawacki, R.A. *Motivating and Managing Computer Personnel.* John Wiley and Sons, 1980.
2. Larson, C.E. and LaFasto, F.M. *Teamwork: What Must Go Right/What Can go Wrong.* Sage Publications, 1989.

Phillip G. Armour (armour@corvusintl.com) is a vice president and senior consultant at Corvus International Inc., Deer Park, IL.