Teaching Cognitive IV&V

Tim Menzies
CSEE, WVU
tim@menzies.us*†

March 17, 2006

The goal of this workshop is to outline teaching methods for IV&V engineers. Initially, this paper will be very negative about that goal, arguing that there is are major differences between NASA software development practices and software development practices elsewhere. In that (very pessimistic) view, if NASA wants to train IV&V analysts on the specifics of NASA IV&V then they require a NASA-run university focusing on NASA applications to teach NASA SE principles to NASA IV&V analysts.

That pessimism will be reversed by the second half of this paper. Using some cognitive theory, a new definition of IV&V will be proposed which could be used as the basis of a general IV&V education program.

1 Uniqueness of NASA

In the past, NASA has been a rich source of SE research. One of the outstanding examples of that work was the NASA Software Engineering Laboratory (SEL) experiments [1]. In the period 1976 to 1994, Basili and his colleagues reported impressive improvements in a particular product line of satellites at NASA Goddard. Many of the quality improvement method developed there are often cited and widely applied including:

- the Goal-Question-Metric paradigm [2]
- clean room testing [14]
- perspective-based inspection [15]

While the contribution of these methods is indisputable, it would be misleading to represent the SEL experience as follows: "it worked at NASA, so it should work across the industry". What is not generally appreciated is that the NASA SEL experiments were conducted in an environment somewhat distant to a standard software development shop. In the SEL environment:

- For several decades, stable and co-operative relationships existed between contractors and clients;
- The work focused on a single product line;
- The gap between development and test can be very large

Many of the features of the SEL environment still hold at NASA. However, few of these features exist in general software engineering industry where:

- Projects last (much) less time than one decade.
- Contractor/client relationships can be quite volatile. Over the space of several decades, a single client can switch contractors multiple times.
- Contractors and clients rarely focus on a single goal such as a single product line. The modern software environment is quite varied and prone to rapid change. During a successful contractor/client relationship, market forces can make clients direct their contractors towards multiple goals.
- Modern interactive development tools allow developers to dramatically reduce the time between development and testing and deployment.

The last point is the most critical since it shows the great difference between software development practices within and without of NASA. When developing according the spiral [3] or agile [4] development model, the time between finishing and using software in its target environment may be quite short. Hence, even within the time allocated to building the first version of a project, feedback from the field may result in changes to project goals. For example, the ECLIPSE open-source integrated development platform is undergoing continual revision and changes due to feedback and new plug-ins developed by the ECLIPSE community. This is a very different situation to NASA where it may take years before a finished piece of software is used in its

*Submitted to CSEET 2006 Workshop on 21^{st} Century IV&V, April 20, 2006 Turtle Bay, Oahu, Hawaii http://db-itm.cba.hawaii.edu/cseet2006/cseetworkshop.php.

†Earlier drafts of this paper are available from http://menzies.us/pdf/06teachivv.pdf.

target environment (e.g. Pluto).

Hence, much of what has been learned about the general SE industry may not apply to NASA IV&V. For example,

- The top level of the SEI capability maturity model includes a feedback phase where lessons learned can be feed back into process change. Such continual process improvement is common method for storing and improving corporate memory.
- Nevertheless, an IV&V team cannot propose process improvement for NASA projects Or, to be more precise, IV&V *can* propose process improvements to projects but projects will rarely act on such recommendations. IV&V works best when the developers feel that the IV&V team is there to augment, not replace, their current process. Hence, it can be counter-productive to take a dictatorial stance while performing IV&V. Instead, it is best to adapt the IV&V task to the development processes, and not the other way around.

For another example, consider the differences between V&V and IV&V. In V&V, the development and test teams work together, often at the same physical location. Hence, there is a *fat pipe* of information flowing between developers and test engineers. On the other hand, IV&V is linked to projects via *very thin pipes*. In the case of NASA, several layers of management can exist between:

- The IV&V analysts working for an IV&V contractor company ...
- ... who reports to a NASA civil servant at NASA's IV&V facility ...
- ... who co-ordinates with a NASA civil servant at the development site ...
- ... who supervises a contractor group ...
- ... that hires a developer ...
- ... that actually does the development.

Each link in this chain takes time and effort to traverse. Since this chain crosses organization boundaries, the information flow is also constrained and restricted by contractual issues. For example, the developer may decline to maintain a log of defects seen during development since that log was not included in the deliverables negotiated as part of the original contract.

In summary, there are significant differences between NASA and the general software engineering industry and techniques that work in general may not work in the particular case of NASA IV&V.

2 Cognitive IV&V

A general theory of IV&V has at least four requirements:

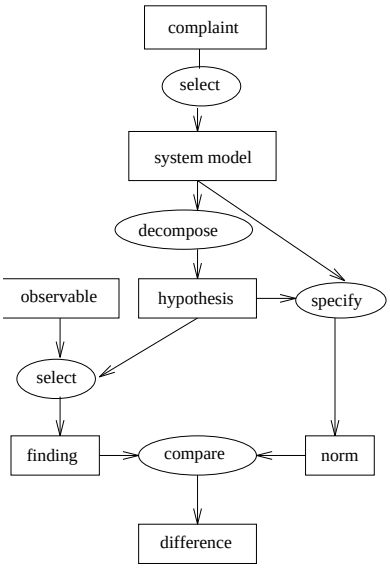

Figure 1: Diagnosis

R1: It must take into account the restrictive nature of the information flow between the IV&V team and the projects.

R2: It should summarize past IV&V experience in a format suitable for the education of new IV&V engineers.

R3: It should offer a direction into the future; i.e. it should offer new and useful insights into IV&V.

R4: In order for NASA to lever the experience of other organizations, it should be based on more than just the NASA IV&V experience.

To meet the requirements {R1,R2,R3,R4}, we turn to Gardner et.al.'s *cognitive patterns* [8]. Gardner's central thesis is that understanding human-based processes requires understanding how humans think. Cognitive patterns research aims at modeling the patterns repeatedly employed by practitioners. Once isolated, such patterns may be used in many ways:

- As rules of thumb for tool designers and maintainers
- As terminology to document practices, problems, and solutions
- They can explicate structures not recorded any other way.

Cognitive patters can be recorded using the notation of Figure 1 in which rectangles are data structures and ovals are functions. Given a `complaint`, a cognitive pattern for *diagnosis* can be represented as follows. A `system model` is `decomposed` into `hypothetical` candidate faulty components. A `norm` value is collected from the `system`

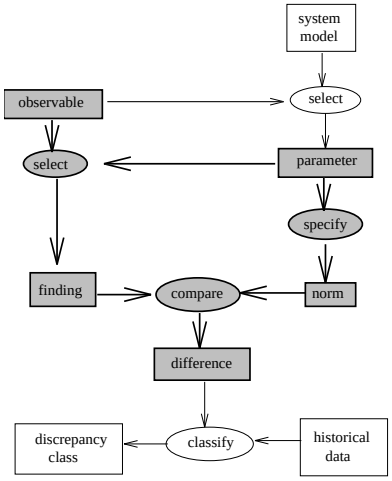

Figure 2: Monitoring. The shaded nodes show the intersection of diagnosis to monitoring.

`model`. An observation for that candidate is requested from the `observable`s (stored internally as a `finding`). The candidate hypothesis is declared to be the diagnosis based on the `difference` between the `norm` value and the `finding`.

As another example, Figure 2 shows a cognitive patterns for *monitoring*[1]. A `parameter` is selected from a `system model`. The model's expected `norm`al value is generated from the model and collected from the `observable-s` (stored as a `finding`). The current state of the monitoring system us reported as a `discrepancy class` after comparing the `finding` with the expected `normal` value.

Note that Figures 1 & 2 do not imply a particular execution order of their functions. Conceptually each function can be driven forwards or backwards to connect inputs to outputs or visa versa. For example:

- The heuristic classification pattern of Figure 3 could be driven from data to solutions to perform diagnosis; i.e. given the `data`, execute forwards `data-abstraction` then `heuristic match`, then `refinement`.
- Alternatively, it could be driven from solutions to data to perform intelligent data collection; i.e. given `solutions`, execute backwards `refinement`, then `heuristic match`, then `data abstraction`. In this backwards reasoning, the generated `data` items become requests back to the environment in order to rule out certain possibilities.

[1]Note that I say "a" cognitive pattern and not "the" cognitive pattern. Elsewhere, I argue that cognitive patterns can vary from domain to domain [11].

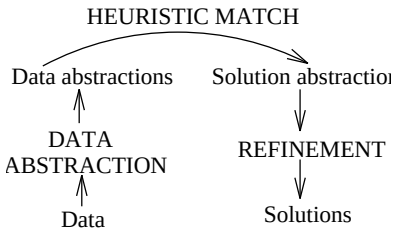

Figure 3: Heuristic classification

3 Discussion

This section how well cognitive patterns satisfy {R1,R2,R3,R4}.

3.1 R1: restrictive information flow

IV&V was characterized above as a process of watching over projects across a thin pipe of information. Note that this process is very similar to a medical practitioner treating a human patient:

- The human body is a very complex device comprising multiple systems interacting over complex feedback loops.
- Doctors diagnosis human illness despite a lack of data; e.g. just a garbled description of symptoms from a sick and confused patient; or a confusing set of measurements from blood tests; or from cloudy shadows on an X-ray.
- Further, they successful modify the behavior of the complex bodies under their control using treatments selected, for the most part, via budgetary concerns.

That is, the diagnosis and monitoring patterns described above show how professional M.D.s *or* IV&V engineers understand and control complex systems about which they have minimal information.

3.2 R2: Summarize past experience

The high-level notation of cognitive patterns has been used many times to offer a high-level documentation of complex systems; e.g. [5, 6, 8, 10, 16, 18] Hence, these patterns can be the basis for educating IV&V practitioners. Students can introduce themselves to IV&V by browsing the libraries of patterns. Existing

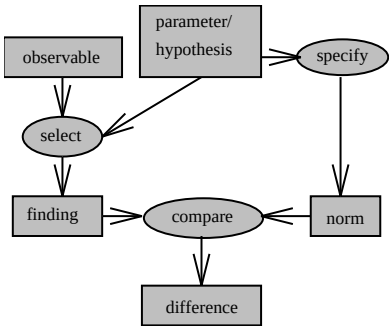

Figure 4: Overlap between diagnosis and monitoring

patterns could be explored (looking for commonalities or improvements). In the literature, there are many examples of such an analysis:

- Scheduling, planning and configuration are actually the same problem, divided on two dimensions ("goal states known or not" and "temporal factors considered or not" (Figure 12.3 of [17]).
- There exists a common sub-graph between Figures 1 & 2 (see Figure 4).

3.3 R3: Directs the Future

Much work is required to fully characterizing IV&V in terms of the diagnosis and monitoring cognitive patterns. For example:

- All the terms of Figures 1 & 2 would need to be mapped into observable and controllable at the IV&V center.
- Historical logs would have to be explored in order to determine what observable best distinguish different findings.

3.4 R4: Based on more than just NASA

Cognitive patterns are a well-studies domain and much of that literature could be used to inform the NASA IV&V process. Diagnosis is a well-studied problem [9] with many general algorithms [12, 13]. For example, *probing* is the task of finding the cheapest observable that could most reduce the hypothesis set. Much has been written about how to design such probes including the use of entropy measures [7].

4 Conclusion

This paper has argued for a radical restructuring of IV&V education. Traditional SE methods may not apply to the particulars of the IV&V experience. However, there are interesting similarities between IV&V and tasks like diagnosis and monitoring. Education designed at this more abstract level can take advantage of a wealth of research of cognitive patterns.

References

[1] V. Basili, F. McGarry, R. Pajerski, and M. Zelkowitz. Lessons Learned from 25 Years of Process Improvement: The Rise and Fall of the NASA Software Engineering Laboratory. In *Proceedings of the 24th International Conference on Software Engineering (ICSE) 2002, Orlando, Florida*, 2002. Available from `http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/83.88.pdf`.

[2] Victor R. Basili. Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Technical Report CS-TR-2956, Department of Computer Science, University of Maryland, College Park, MD 20742, September 1992.

[3] B. Boehm. A Spiral Model of Software Development and Enhancement. *Software Engineering Notes*, 11(4):22, 1986.

[4] B. Boehm. Get Ready for Agile Methods. *IEEE Computer*, pages 2–7, 2002.

[5] B. Chandrasekaran. Design Problem Solving: A Task Analysis. *AI Magazine*, pages 59–71, Winter 1990.

[6] W.J. Clancey. Model Construction Operators. *Artificial Intelligence*, 53:1–115, 1992.

[7] J. DeKleer and B.C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1 1987.

[8] Karen M. Gardner, Alexander R. Rush, Michael Crist, Rober Konitzer, James J. Odell, Bobbin Teegarden, and Robert Konitzer. *Cognitive Patterns: Problem-Solving Frameworks for Object Technology*. Cambridge University Press, June 1998.

[9] W. Hamscher, L. Console, and J. DeKleer. *Readings in Model-Based Diagnosis*. Morgan Kaufmann, 1992.

[10] D. Marques, G. Dallemagne, G. Kliner, J. McDermott, and D. Tung. Easy Programming: Empowering People to Build Their Own Applications. *IEEE Expert*, pages 16–29, June 1992.

[11] T.J. Menzies. OO Patterns: Lessons from Expert Systems. *Software Practice & Experience*, 27(12):1457–1478, December 1997. Available from `http://menzies.us/pdf/97patern.pdf`.

[12] P. Pandurang Nayak and Brian C. Williams. Fast Context Switching in Real-time Propositional Reasoning. In *Proceedings of AAAI-97*, 1997. Available from `http://ack.arc.nasa.gov:80/ic/projects/mba/papers/aaai97.ps`.

[13] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–96, 1 1987.

[14] Richard W. Selby, Victor R. Basili, and F. Terry Baker. Cleanroom Software Development: An Empirical Evaluation. *IEEE Transactions on Software Engineering*, SE-13(9):1027–1037, September 1987.

[15] F. Shull, I. Rus, and V.R. Basili. How Perspective-Based Reading Can Improve Requirements Inspections. *IEEE Computer*, 33(7):73–79, 2000. Available from `http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.77.pdf`.

[16] L. Steels. Components of Expertise. *AI Magazine*, 11:29–49, 2 1990.

[17] D.S.W. Tansley and C.C. Hayball. *Knowledge-Based Systems Analysis and Design*. Prentice-Hall, 1993.

[18] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. KADS: a Modeling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4:1–162, 1 1992.