TITLE: "PRIDE VERSUS AGILE/EXTREME PROGRAMMING"

by Tim Bryce
Managing Director
M. Bryce & Associates (MBA)
P.O. Box 1637
Palm Harbor, FL 34682-1637
United States
Tel: 727/786-4567
E-Mail: timb001@attglobal.net
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"If we built bridges the same way we build systems in this country, this would be a nation run by ferryboats."
- Bryce's Law

INTRODUCTION

Every so often we are asked how "PRIDE" compares to the latest IT industry fad. Recently, we have been asked how "PRIDE" compares to "Agile" or "Extreme Programming" methods. After studying the latest trade propaganda on the subject, I was going to originally entitle this article *"Comparing Apples with Oranges"* as there are substantial differences, but let me be more descriptive.

In a nutshell, Agile/Extreme Programming (A/EP) is a departure from traditional methodologies such as "PRIDE" and is intended to accelerate the development of software through interviewing techniques to determine what the end-user wants and using power programming tools to build them. Further, the emphasis is to get something up and running as fast as possible and continually modify it until completion. To their credit, the advocates of A/EP openly admit the scope of their effort is on software only and not on total systems development.

The differences between software and systems are substantial. As we discussed in "PRIDE" Special Subject Bulletin Number 10 (*"Managing Design Complexity"* - Feb 07, 2005), the average information system is approximately 50 times larger in terms of complexity as opposed to developing a single program. Building a major system requires vision, planning, organization, and patience, something that is in short supply in today's corporate world. As a result, IT departments turn to A/EP to conquer small programs in the hope they will somehow mesh with the overall corporate system architecture and data base later on. Inevitably, this rarely happens which leads to data redundancy issues and more re-writes.

REQUIREMENTS

An important part of A/EP is how to specify the software to be built. Special techniques are recommended on how to interview end-users and ascertain how the computer program should work. Here, questions are asked regarding the functionality of the software and the appearance of screens and reports in terms of what is mandatory/strategic/optional. From this, a list of features is compiled and prioritized for the Software Engineer to tackle.

Understand this, the questions asked regarding specifications have nothing to do with information requirements a la "PRIDE" but, instead, are geared towards the physical design of the software. (For the difference between Information Requirements and Programming Specs, see:

- "PRIDE" SSB Number 4 (*"Defining Information Requirements"* - Dec 27, 2004)
- "PRIDE" SSB Number 14 (*"What is a good program spec?"* - Mar 07, 2005)

To describe the difference between the two, I am reminded of the story of the IT Director at a shoe manufacturing company who received a call from the corporate Sales Manager asking for some help on a pressing problem. The IT Director sent over one of his analysts to meet with the Sales Manager and discuss the problem. Basically, the manager wanted a printout of all shoe sales sorted by model, volume, type, color, etc. The analyst immediately knew how to access the necessary data and sorted it accordingly thereby producing a voluminous printout (three feet high) which he dutifully delivered to the user.

The IT Director stopped by the Sales Manager's office a few days later to inquire if the analyst had adequately serviced the user. The sales manager afforded the analyst accolades on his performance and proudly pointed at the impressively thick printout sitting on his desk. The IT Director then asked how the manager used the printout. He explained he took it home over the weekend, slowly sifted through the data, and built a report from it showing sales trends.

"Did you explain to the analyst you were going to do this?" asked the IT Director.

"No," replied the Sales Manager.

(continued on page 2)

(continued from page 1)

"Aren't you aware we could have produced your report for you and saved you a lot of time and effort?"

"No."

This is a classic example of the blind leading the blind. The user did not know how to adequately describe the business problem, and the analyst asked the wrong questions. Remarkably, both the Sales Manager and analyst were delighted with the results. The IT Director simply shook his head in disbelief.

Quite simply, there are substantial differences between specifying information requirements and specifying software. Both have their place, but both serve different purposes. A true business analyst (or "Systems Engineer" as refer to it in "PRIDE") investigates the underlying business rationale of the information. The Software Engineer lives in the physical world and is only concerned with how the software will work.

METHODOLOGY

In "PRIDE" SSB Number 26 (*"Methodologies versus Techniques and Tools"* - May 30, 2005) we described the characteristics of methodologies and techniques. Based on our criteria, A/EP is more aptly entitled a "technique" as opposed to a true methodology. The steps involved with A/EP are basically:

1. Listen
2. Design
3. Code
4. Test

...then iterate through this process until completion. Frankly, aside from the programming tools used, I fail to see how this is any different than what is taught in today's college classrooms.

From a programmer's point of view, this is understandable but from an IT manager's point of view, it is hardly a viable means for building enterprise-wide integrated systems.

In the "PRIDE"-Information Systems Engineering Methodology (ISEM), A/EP can be applied in Phase 4-II (Software Engineering), Phase 5 (Software Manufacturing), and Phase 6 (Software Testing), but it serves no other purpose in the other phases of the methodology. This means "PRIDE" and A/EP are compatible if the manager is so inclined to have them work together. In fact, "PRIDE"-ISEM Phases 4-II, 5 and 6, provide the necessary framework for managing such projects.

CONCLUSION

A/EP can trace their roots back to Rapid Application Development (RAD) and Joint Application Development (JAD) techniques introduced in the 1980's. The emphasis of A/EP is on user participation in specifying software and using "power tools" to build programs in small increments that are refined over time. Under this approach, there is no effort to build software in accordance with an overall system architecture. In other words, A/EP addresses a small part of the overall systems puzzle and, as such, casts suspicion over the necessity of the software being produced.

Basically, A/EP is an admission that major systems cannot be built anymore and that IT organizations are content doing small things. This disturbs me greatly as I remember an America that used to accept big challenges, organized and prepared accordingly, and conquered major design objectives, be it a skyscraper, a spaceship, or the major transportation systems of our country. But as I mentioned in our "Bryce's Law" at the beginning, *"If we built bridges the same way we build systems in this country, this would be a nation run by ferryboats."*

Need to do build something small? Use A/EP.

Need to do build something big? Use "PRIDE".

EPILOGUE

I would like to close with a note I recently received from an old "PRIDE" user which I would like to share with you.

"32 years ago or so, I learned PRIDE while at the Newport News Shipbuilding and Dry Dock Company. I have used it's concepts ever since, especially analyzing the current system up front. That gives the customer the most confidence in the analyst and trust begins to develop. I actually get down to the lowest level and do the work. It pays dividends. At another site, I was reviewing the process with upper management, they found duplicate efforts which resulted in reducing manual processing time. I am part of a design improvement team and quite naturally have been suggesting a 'phased' approach which I took from memory. One of your phrases I forgot was 'illustrative output' which was produced during analysis.

(continued on page 3)

(continued from page 2)

When I present 'illustrative outputs' to users, their eyes light-up because they can see I really understand their business (learned from analyzing the current system).

What I do now is make recommendations and watch them get diluted by analysts and developers trying to make their testing or programs easier. This is called being a 'team' player. After a year or so, who gets called to put out the fire? Me.

After all these years, everything is still the same and PRIDE is still the best system development methodology.

Thanks for making the product public."

- W.C. Feurtado

END