

Adaptive Agility

Todd Little, Forrest Greene, Tessy Phillips, Rex Pilger and Robert Poldervaart
Landmark Graphics
tlittle@lgc.com

Abstract

To maximize the velocity of business value delivery Alistair Cockburn talks of having a process that is “barely sufficient.” At Landmark Graphics we developed some guidelines as to what “barely sufficient” means for our various software projects. We examined over 60 projects and observed two primary attributes that influenced the type of process used: complexity and uncertainty. We provide a scoring model for plotting projects onto a four quadrant graph, which we use to categorize projects into dogs--simple projects with low uncertainty, colts--simple projects with high uncertainty, cows--complex projects with low uncertainty, or bulls--complex projects with high uncertainty. We adapt our agile process from a core set of barely sufficient practices that all projects use and add processes and practices according to a project's profile. One key benefit of this approach has been identifying project drivers and providing early guidance to project teams so that they can start with a process that is close to appropriate.

1. Introduction

Agile Software Development has become increasingly popular among development teams looking to shed unnecessary process overhead in order to maximize the velocity of business value delivery. Alistair Cockburn¹ talks of having a process that is “barely sufficient,” and Jim Highsmith² suggests something “a little bit less than just enough.” However, it is a challenge to understand just what is “sufficient” for any given project.

At Landmark Graphics we have experienced working with a number of different software practices and processes and have over the last several years begun to get a better handle on some guidelines as to what “barely sufficient” might mean for software projects. One thing that has become quite apparent is that what is “barely sufficient” for one project may be insufficient for another, or overhead for yet another. When looking at our project history, we observed two primary attributes that influenced the type of process used: complexity and uncertainty. Complexity includes project composition such as team size and criticality, while uncertainty includes both market and technical uncertainty. To better quantify the complexity and uncertainty, we came up with a scoring model and plot the results for each project on a

four quadrant graph. As is tradition, we used animal names to represent the four quadrants:

- Dogs – Simple projects with low uncertainty
- Colts – Simple projects with high uncertainty
- Cows – Complex projects with low uncertainty
- Bulls – Complex projects with high uncertainty

We developed a “barely sufficient” process for any given project based on a core set of common practices. Depending on complexity and uncertainty, additional practices are recommended. This did not involve significant change for us as most projects had already naturally taken on this emergent behavior, although in several cases they did not start out that way. One benefit of this approach has been identifying these project drivers and providing earlier guidance to project teams so that they can start with a process that is close to appropriate.

2. Company Background

Landmark Graphics is the leading supplier of software and services to the upstream oil and gas industry. Our software portfolio, which ranges from exploration and drilling to data management and decision analysis, includes more than 60 products consisting of over 50 million lines of source code. We develop and maintain this product suite with a little over 200 software developers and a total R&D staff of about 400, including domain specialists, testers, writers and program managers. A key business value of our application suite comes from the integration of these products through a common data model with over 800 tables, 12 major entities, and 90 data types. We release our products on a regular basis, with release cycles varying between 3 months and 18 months.

Our products are graphical, highly interactive and some are computationally intensive. Despite the fact that some products are 20+ years old, it has been critical for us to ensure that they contain the latest geoscience and engineering technology. This has required that we have many domain specialists (geophysicists, geologists, petroleum engineers, mathematicians) involved in developing these products. In many cases the developers are domain specialists themselves, while in other cases we are able to team up domain specialists with computer scientists.

Many of our products came through acquisitions over the past 20 years and have been subsequently retrofitted into

our integrated product suite. Newer products have been developed with integration in mind. One of the complications from the acquisitions is that we have geographically dispersed development teams with 5 primary development centers in Houston, Austin, Denver, Calgary and Stavanger. We have also been utilizing an offshore development center in Islamabad, Pakistan.

Our first foray into synchronized integration began in 1995 and concluded with the industry's first such product offering in 1997. This was enormously successful, but had its challenges. Coordinating multiple product teams in multiple geographic locations turned out to be quite difficult. Fortunately the integration vision permeated the company and product teams did what they needed in order to ship the product suite. Since 1997 we have continued to ship both integrated synchronous releases and individual product updates. Beyond the coordination challenges, there are other considerations. Individual product teams have different market needs and must remain technically competitive, yet the overall market need for integration is quite compelling.

3. Landmark's Development Process

As Landmark grew through acquisition, it likewise acquired a number of different software development processes. As most of the acquisitions were small, relatively young companies, there was minimal process definition in place. These companies were quite successful in developing their individual products and most of the acquisitions were of industry leading products within their domain. While these products were individually successful, Landmark's value proposition was in creating integrated solutions utilizing these best of class applications. This required modifying each of the applications in order to integrate properly with the full Landmark suite of applications.

To better support developing integrated products, Landmark sought out a standardized development process framework that would provide consistency across product teams. One of the product teams had been exploring the Microsoft Solutions Framework (MSF)^{3,4} milestone-based-iterative development framework as early as 1995 and it looked to fit our needs quite well. About this time, Landmark set out to deliver its first synchronous release with all products on a common datamodel and a common install. By 1997 we began to encourage teams to utilize MSF, however teams were not required to follow any particular process. By 1999 the company believed so strongly in MSF that there was a re-organization in order to align with the MSF functional team roles.

What happened next is a somewhat predictable aspect of human dynamics. Some people looked to MSF and chose to interpret it as a rigorous definition of a waterfall process. It is our opinion that MSF is far from a waterfall solution and is much more aligned with agile

development. What is interesting is that the waterfall interpretation came from two different types of people: those that believed that what Landmark needed was a rigorous waterfall process, and those that would do anything to oppose a rigorous waterfall process. It is possible that the opposition arose due to the vocal strength of the waterfall advocates. Debates aside, teams nominally continued as they had been and shipped products. Overall, the use of MSF was effective in establishing a core vocabulary and a set of core practices that most teams utilized. Some teams followed more of the MSF framework, while others started to experiment with other agile methods such as XP.

By 2002 a new management team was concerned that MSF was too heavyweight and was slowing down the development process. We suspect that this perception was mostly a result of the strong vocal presence of the former waterfall advocates. Nonetheless, it was an opportunity to fine-tune our development process.

Murray Roth, the new Executive Vice President of R&D coined the new acronym for the development process: RAPID: "Robust Adaptable Process for Innovative Development." The authors, along with sponsor Karl Zachry, took on the task to define this new process.

Our team of senior functional managers and project managers spent some time assessing what teams were doing, what was working and why. Meanwhile, we had been following the evolution of the agile community and found ourselves aligned with Highsmith's Adaptive Software Development² and Cockburn's Crystal¹ methods. We particularly liked the meta-methodology of Crystal, as we knew that we had issues of scale that needed to be addressed. While we liked the Crystal framework, we felt that there were more project attributes that influenced the type of agile approach to be used than spelled out in Crystal. At the time that we were doing this assessment, the work of Boehm and Turner⁵ had not yet been published. Subsequently we did modify one of our attributes (Team Capacity) based on their work. Our attributes include something similar to all of their attributes except for organizational tolerance which was irrelevant since we were only looking at one organization. Once we had the list of attributes that we thought were influencing project dynamics, we started to look for commonality. Our objective was to have something that was simple to assess yet provide useful information. In other words we were looking for a "barely sufficient" solution for helping us identify a "barely sufficient" process for our projects. We recognized that fundamentally the attributes grouped into two primary concerns: complexity and uncertainty. Using these attributes we generated a quick survey (see Tables 1 and 2) that projects teams could use to assess their project.

4. Complexity Drivers

The first set of attributes that we identified we grouped under the category of complexity. The complexity of a project is a characteristic of the project structure. We developed a system to score each project's complexity based on the following attributes:

- Team Size
- Mission Critical
- Team Location
- Team Maturity
- Domain knowledge gaps
- Dependencies

4.1. Team Size

In Cockburn's Crystal methods¹, team size is used in the determination of Crystal "color", with darker colors requiring additional process ceremony. In a very similar fashion, we see team size as a major contributor to the project complexity.

4.2. Mission/Safety Critical

Also in the Crystal methods, mission criticality or project importance is used to determine the type of development methodology. If there are lives or essential moneys or lives that are at risk with the project, it must be treated differently than if the only cost of failure is the investment in the project. We take a similar view, although for our approach we view the importance of the project and what is at stake as one of the indicators of complexity.

4.3. Team Location

Everyone in the same room enables high bandwidth communication amongst the project team. A vastly distributed team or one in which a significant portion of

the team is in a multi-hour time zone shift can add to the project complexity. This can be a difficult attribute to assess since a team that has one or a few dispersed members may not drastically increase its complexity. We have advised teams to use their judgment on this assessment.

4.4. Team Capacity

An established team of experts that has been working together for a number of years on incremental enhancements to a product line can almost anticipate what other team members are likely to need and do. This is contrasted with a brand new team of relative novices. In many ways this attribute is similar to the Cockburn Shu-Ha-Ri Level utilized by Boehm and Turner⁵.

4.5. Domain knowledge gaps

Landmark's products include leading edge technologies used by specialists in the oil and gas exploration and production domain. It is critical that the product team have full time access to the domain specialists to resolve ambiguity and produce the desired product. We have found that this is greatly simplified when the developers are domain specialists themselves, and much more complex when access to domain knowledge is limited.

4.6. Dependencies

This attribute is a measure of the degree to which the project team is dependent upon 3rd parties or upon other projects within the company. In general, more dependencies will increase project complexity. Established 3rd party dependencies may be given reduced weight if the team has a consistent track record of working with a stable version.

5. Uncertainty Drivers

Table 1. Complexity Attributes, with range from low (left) to highly complex (right)

Attribute	1	3	5	7	10
Team Size	1	5	15	40	100
Mission Critical	Speculative	Small user base	Established market	Mission Critical with large user base	Safety Critical with significant exposure
Team Location	Same Room	Same Building	Within Driving Dist	Same Time Zone +/-2	Multi-site, World Wide
Team Capacity	Established Team of experts	New team of experts	Mixed team of experts and novices	Team with limited experience and a few experts	New team of mostly novices
Domain knowledge gaps	Developers know the domain as well as expert users	Developers know the domain fairly well	Developers require some domain assistance	Developers have exposure to the domain	Developers have no idea about the domain
Dependencies	No dependencies	Limited and/or well insulated	Moderate	Significant dependencies	Tight Integration with several projects

The uncertainty of a project is dependent upon market conditions and upon the choices the development team chooses to make. We consider the following attributes to be the primary indicators of the project uncertainty:

- Market Uncertainty
- Technical Uncertainty
- Project Duration
- Dependents/ Scope Flexibility

5.1. Market Uncertainty

If the market need is well known then the project is unlikely to need significant steering. Conversely, if the market needs are not well understood, then it will be critical to be able to steer the project to the desired goal rather than the initially stated objective. This attribute is similar to the requirements change attribute utilized by Boehm and Turner.

5.2. Technical Uncertainty

Mature products utilizing proven technology do not have much technical uncertainty, although sometimes with our products we have uncertainty surrounding the new domain technology added to an existing product. On the other hand it is common on new products to want to utilize the latest technology and these projects will have a high degree of technical uncertainty.

5.3. Project Duration

The longer the project is scheduled to go prior to its release, the more chance there is for the technical or market uncertainty to have an impact on the project.

5.4. Dependents/ Scope Flexibility

The degree to which other projects are dependent upon this project can limit the amount of steering that can be tolerated by the other projects. It is not acceptable to be continually modifying interfaces when those changes have ripple effects on a number of other projects.

6. Quadrant Assessment

Based on the values for the project, we calculate the overall Project Complexity and the Uncertainty as follows:

$$Complexity = 2^{\sum \log_{10} x_i}$$

$$Uncertainty = 2^{\sum \log_{10} y_i}$$

where x_i and y_i are the individual complexity and uncertainty attribute scores. In effect, the log x terms are scaled information measures.⁶

The choice of equation is equivalent to rescaling each attribute between 1 and 2 and then computing the product. We chose this approach because it made sense and seemed to give good results when values of Complexity and Uncertainty are cross plotted. The results for our portfolio are plotted in Figure 1. We found that the projects in a given quadrant were quite similar and that the successful approaches used for managing the projects were also similar. The properties of these quadrants are described below, and summarized in Figure 2.

Table 2: Uncertainty Attributes, with range from low (left) to high uncertainty (right)

Attribute	1	3	5	7	10
Market Uncertainty	Known deliverable, possibly defined contractual obligation	Minor changes in market target expected	Initial guess of market target is likely to require steering	Significant market uncertainty	New market that is unknown and untested
Technical Uncertainty	Enhancements to existing architecture	We think we know how to build it	We're not quite sure if we know how to build it	Some "r"	New technology, new architecture. May be some "R"
Project Duration	1-4 week	6 months	12 months	18 months	24 months
Dependents/ Scope Flexibility	Well defined contractual obligations or Infrastructure	Scope is not very flexible.	Scope has some flexibility	Scope is highly flexible	Independent

RAPID Quadrant Assessment

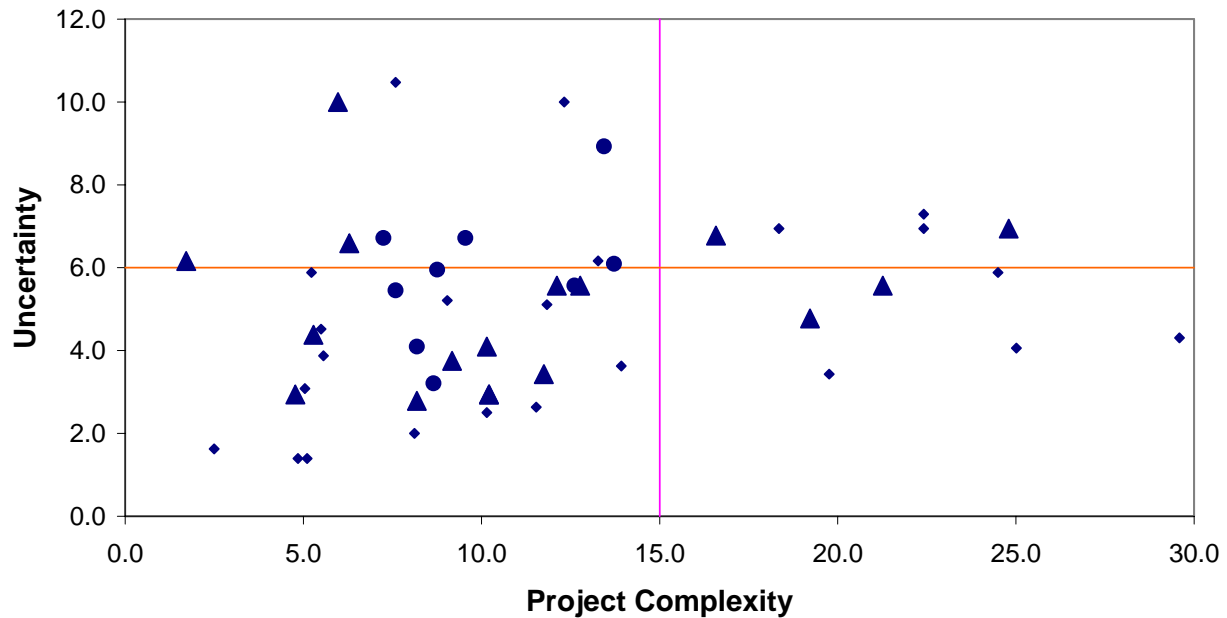


Figure 1. Project complexity versus project uncertainty for projects from 3 divisions.

6.1. Dogs – Simple projects with low uncertainty

Dogs are typically mature products being developed by small teams. With these types of projects where they are not particularly complex and do not have much uncertainty, the best thing to do is to let the development teams do their job to ship the products. There are also projects in this quadrant that have some uncertainty, but the duration is kept very short to limit the impact of the uncertainty. Prototype or skunk-works projects often fit into this category. For both dogs and skunks we find that additional process ceremony and documentation is unnecessary and inefficient, thus we run these projects using only the minimal core set of practices that we use for all projects in all quadrants. These projects are run similar to Cockburn's Crystal Clear. For our portfolio this quadrant contains approximately 60% of the projects.

6.2. Colts – Simple projects with high uncertainty

New products will usually have both market and technical uncertainty. If teams are kept small then they can react quickly to adapt to the uncertainty. The metaphor of the young colt aptly describes these projects. They are just getting started and have a lot of energy and freedom. Most of our project teams that have had success with Extreme Programming (XP)⁷ fit into this quadrant. We

have also found that daily standup Scrums⁸ are effective in this quadrant. Approximately 20% of our projects are colts.

6.3. Cows – Complex projects with low uncertainty

The mature products and product suites that continue to have large project teams are usually the cash cows of the organization. In addition to the obvious similarity to cash cow, the cow is a good metaphor for these projects as they are quite large but do not move particularly fast. These projects have less need for agile steering, and often may actually have need for disciplined change control in order to reduce the impact when there are many dependent projects or customers. Projects in this quadrant may still be agile, but require defined and published interfaces for the dependent projects. They also require more direct project and program management, looking at issues such as critical path and cross team communication. Many of our cows are integration projects involving a number of projects, typically dogs. We have utilized a team of team leaders, something quite similar to a "Scrum of Scrums,"⁸ to manage many of these projects. Cows comprise about 10% of our projects.

RAPID Quadrant Assessment

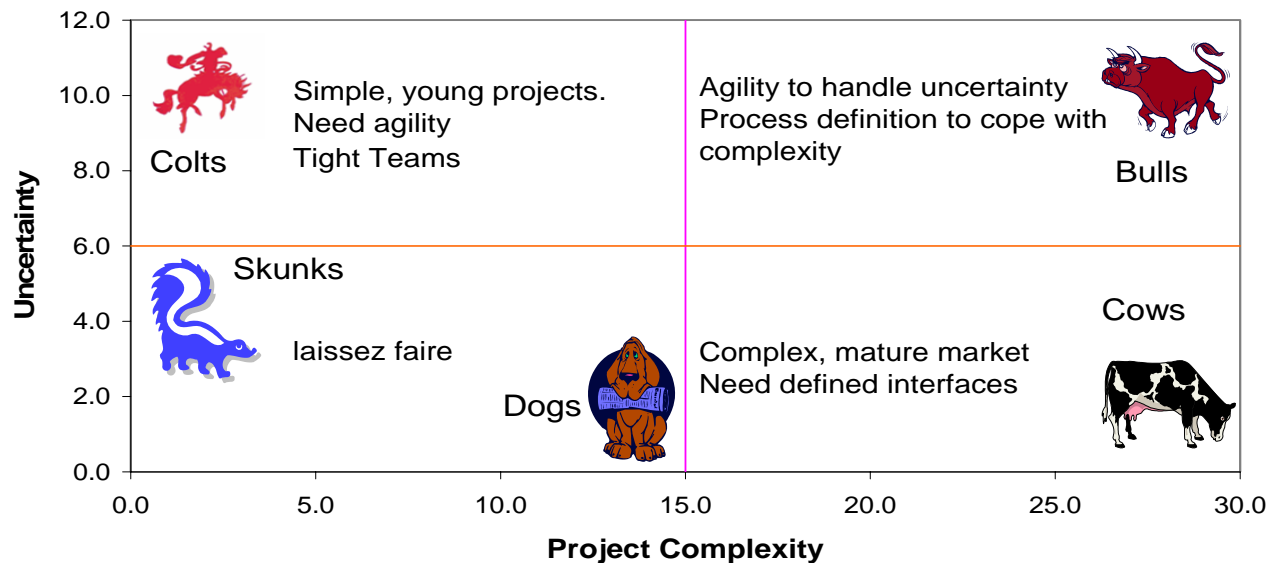


Figure 2: RAPID Quadrant Assessment

6.4. Bulls—Complex projects with high uncertainty

Projects that are highly complex and have high uncertainty create problems on all fronts. They need to be quite agile in order to steer through the uncertainty, yet they require some process ceremony in order to manage the project complexity.

The metaphor of the bull is quite appropriate. These projects are large and can get out of control quickly if not careful. They have high visibility throughout the organization, as they are often new products that have strong investment. In our case many have been next generation products intending to supplant existing cash cows. Expectations are high, yet uncertainty and complexity are equally high. These projects require much of the same process ceremony as the cows, yet must be structured in a manner that enables agile steering. Iterations must be more frequent and communication channels must be very efficient. We have found that these projects require the best program managers that can work with agility and cut through complexity. In our portfolio approximately 10% of our projects are bulls.

7. Core RAPID Process

The quadrant assessment is a key early work product of the RAPID process. We have constructed the overall RAPID framework to incorporate Core processes that all projects follow and then utilize the project assessment to

determine which other process activities should be utilized. The core processes include:

- Aggregate Product Plan
- A/B/C List
- Quality Agreement
- Continuous Integration
- Expert User Involvement
- Project Dashboard

7.1. Aggregate Product Plan

This is a very concise statement of the project objectives produced by the Product Manager. It contains the following information for this release:

- Target date
- One sentence product vision
- High level list of “A” priority features committed (see below)
- Short description of the strategic fit
- List of the target markets that will be pursued
- Supported platforms

7.2. A/B/C List

Desired features are categorized into 3 priorities, which we conveniently name A, B, and C. The team estimates effort requirements and works with the Product Manager’s estimate of value to maximize return on investment. Only “A” features may be communicated to customers. The following is our definitions for our A/B/C items:

- A. MUST be completed in order to ship the product.
- B. SHOULD be completed in order to ship the product.
- C. MAY be completed prior to shipping the product if time allows.

Since we are contracting to deliver all the “A” items, we allow for uncertainty by limiting the schedule to no more than 50% “A” features. Over the course of the project as “A” items are completed we utilize any remaining schedule to complete the “B” or “C” items. It is common to reprioritize during the project, particularly at iterations, although “A” items are usually not dropped unless proper customer expectations have been set. This approach and how it is used to maximize value delivery is described further in a recent article by Little⁹.

7.3. Quality Agreement

The team works with the Product Manager to reach agreement on quality targets for the release. We have modified Rob Thomsett’s¹⁰ quality agreement approach to utilize the A/B/C prioritization. We feel that this gives us consistency in our discussions and also provides a bit more granularity than Thomsett’s On/Off approach.

7.4. Continuous Integration

For all our projects we utilize configuration management and build at least nightly. A number of projects have started utilizing a continuous build process. Most of the projects using continuous builds started as Colts, but some of these are now Bulls or Cows and still find the use of continuous builds to be beneficial.

7.5. Expert User Involvement

We have always found it critical to have expert users involved in the development. Most of our complex projects have a dedicated expert user, usually someone that has been a former customer. Nearly all of our testers are also expert users, and many of the developers are expert users themselves.

7.6. Project Dashboard

We developed a web based interface for reporting common information about the project status. This information is recorded at least weekly by the project managers and provides an excellent portfolio dashboard to view project health. Information that is available includes the aggregate product plan, quality metrics, top active risks, any revisions to the release estimate, etc.

8. Adaptive Processes

Additional processes and practices are added to the Core set based on the project attributes. The project quadrant provides guidance for the types of processes and practices to be added.

8.1. Dogs and Skunks

These projects do not require any additional process guidelines beyond the core processes. Teams are free to do what they need to in order to ship product. As Alistair Cockburn has referred to Crystal Clear¹¹, these projects take a laissez-faire approach to software development.

8.2. Colts

These high uncertainty projects benefit from a number of additional project practices that help cope with the uncertainty:

- Short iterations
- Daily standup meetings
- Automated unit tests

8.3. Cows

Although they do not have much uncertainty, these projects require additional processes to deal with the complexity. Such activities include:

- More rigorous requirements management; we use a requirements tool.
- Functional specifications for interface definitions
- Relatively detailed project plans with critical path identification
- Projects broken up into sub projects and coordinated by a team of leaders or a Scrum of Scrums.

8.4. Bulls

These projects are quite difficult to control as they require steering to cope with the uncertainty, yet are large and/or complex. We find that to be run successfully they require most of the process ceremony of the Cows, yet much of the steering of the Colts. Most importantly they require the most seasoned project managers that are able to understand how to balance this dichotomy. We expect that most organizations have only a few project managers with the requisite capacity to manage these projects. As such it is unwise for an organization to have more Bulls than Bull project managers.

9. Adjusting Project Constraints

We have had teams discover during the quadrant assessment that their project was either more complex or uncertain than they had thought. There are adjustments

that sometimes can be made to the project in order to reduce either complexity or uncertainty. In particular, we have often found it useful to decompose larger projects into subprojects in order to reduce complexity.

10. Conclusion

At Landmark we strive to deliver our software in a manner that will maximize the delivery of business value. We believe that agile development approaches are aligned with this philosophy.

Within any organization there will be a distribution of project types, a distribution of people and a distribution of opinions about the right way to do things. Our experience has been that there is no single software development process that is the best approach for every project. It is important to look at the project and team conditions to determine how best to run the project. We believe that two of the most critical attributes that impact how to run a project are complexity and uncertainty.

We developed an assessment tool to provide guidance to project teams on how to adapt their project process to manage complexity and to cope with uncertainty. The scoring model is not intended to be rigorous; however it has proven to be useful to the project teams and to senior management. We do not recommend blindly using the assessment tool; the identification of the complexity and uncertainty attributes is intended to simulate thought, not to eliminate it.

The assessment has also provided an insight into our project portfolio management. Our overall portfolio of projects is distributed across the four quadrants. Fewer than 10% of our projects are classified as bulls, These projects are difficult to run and an organization with a high percentage of bulls is taking on significant risk. Likewise, only about 10% of projects are cows and about 20% are colts. Most of our projects are in the dog quadrant. Dogs can be loyal and rewarding. Provide them reasonable care and feeding and they will provide good results in return.

11. Acknowledgements:

The experience of the many Landmark development teams provided fodder for this analysis. Thanks also to Rebecca Wirfs-Brock of Wirfs-Brock Associates for providing valuable guidance in shepherding this paper.

12. References

1. Cockburn, Alistair, *Agile Software Development*, Addison Wesley, 2001.
2. Highsmith, Jim, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House, 2000.
3. Microsoft Solutions Framework, <http://www.microsoft.com/msf/>
4. Cusumano, Michael A. and Selby, Richard W., *Microsoft Secrets: How The World's Most Powerful Software Company Creates Technology, Shapes Markets, And Manages People*, Simon & Schuster, 1998.
5. Boehm, Barry and Turner, Richard, *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, 2003.
6. Shannon, C. E., "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948.
7. Beck, Kent, *Extreme Programming Explained: Embrace Change*, Addison Wesley, 1999.
8. Schwaber, Ken & Beedle, Mike, *Agile Software Development with SCRUM*, Prentice Hall, 2001.
9. Little, Todd, "Value Creation and Capture: A Model of the Software Development Process," *IEEE Software*, Vol. 21, No. 2, 2004.
10. Thomsett, Rob, *Radical Project Management*, Prentice Hall, 2002.
11. Cockburn, Alistair, *Crystal Clear*, Addison Wesley, 2004