

Context-Adaptive Agility: Managing Complexity and Uncertainty

Todd Little, *Landmark Graphics*

Once you categorize your projects as dogs, colts, cows, or bulls according to their complexity and uncertainty, you can adapt your process by adding practices according to each project's profile.

Agile software development has become increasingly popular among development teams looking to shed unnecessary process overhead to maximize the velocity of business value delivery. Alistair Cockburn talks about a “barely sufficient” process,¹ and Jim Highsmith suggests something “a little bit less than just enough.”² However, understanding just what’s sufficient for any given project is a challenge.

At Landmark Graphics, we’ve worked with various software practices and processes and over the last several years have begun to better comprehend some of the guidelines that define “barely sufficient” for our software projects. One thing has become quite apparent: what’s barely sufficient for one project can be insufficient for another yet overly bureaucratic for another. As we looked at our project history, we found that two primary attributes influenced the type of process we used: complexity and uncertainty. To better quantify these attributes, we devised a scoring model and plotted each project’s results on a four-quadrant graph. Similar to the Boston Consulting Group’s Boston Matrix (see the related sidebar on page 32), we used names to represent the four quadrants:

- *Dogs* are simple projects with low uncertainty.

- *Colts* are simple projects with high uncertainty.
- *Cows* are complex projects with low uncertainty.
- *Bulls* are complex projects with high uncertainty.

Using this matrix, we’ve developed an approach to help determine what process practices are “barely sufficient” for any given project. We start with a core set of common practices. Then, depending on complexity and uncertainty, we can recommend additional practices. We found that most of our projects had already taken on this emergent behavior naturally—although in several cases they didn’t start out that way. By identifying these project drivers, we can provide earlier guidance to project teams so that they can start with a process that’s close to appropriate.

Table 1**Complexity attributes and their scores***

Attribute	Complexity score				
	1	3	5	7	10
Team size	1	5	15	40	100
Mission criticality	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 distance	Same time zone +/- 2 hrs.	Multisite, worldwide
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	None	Limited, well insulated	Moderate	Significant	Tight integration with several projects

* Minimally complex = 1, highly complex = 10.

Company background

Landmark Graphics (www.lgc.com) is the leading supplier of software and services for oil and gas exploration and production. 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 release products regularly, with release cycles varying between three and 18 months. Landmark has about 200 software developers and an R&D staff of about 400 geographically dispersed through primary development centers in Houston, Austin, and Denver in the US; Calgary, Canada; and Stavanger, Norway. More details on Landmark and our development process history are available in an earlier report on this work.³

Development process overview

We started by investigating Alistair Cockburn's Crystal methods¹ and Barry Boehm's risk-based approach⁴ to blending agile and plan-driven software development. Although we liked the Crystal framework of categorizing projects by size and criticality, we felt there were more project attributes that influence how to best manage a project. We enumerated all the critical attributes we'd experienced that had influenced how our successful projects had been run. Once we had this list of attributes, we looked for commonality and found, as I said earlier, two primary concerns: complexity and uncertainty. Using these attributes, we generated a quick survey that project teams could use to assess their projects.

At the time we were doing this assessment, Boehm and Richard Turner hadn't yet published their work on balancing agility and discipline.⁵ In that work, they categorized projects on the basis of five attributes: size, criticality, dynamism, personnel capacity, and organizational culture. Our approach both extends and simplifies their ideas. We use more attributes in our evaluation but then simplify by grouping them into two categories—complexity and uncertainty.

Complexity drivers

A project's structure determines its complexity. We developed a system (see Table 1) to score project complexity on the basis of

- team size,
- mission criticality,
- team location,
- team maturity,
- domain knowledge gaps, and
- dependencies.

Team size

Cockburn's Crystal methods use team size to determine Crystal "color," with darker colors requiring additional *practices and development rigor*. Similarly, we see team size as a major contributor to project complexity.

Mission and safety criticality

Also as in the Crystal methods, our approach treats mission criticality or project importance as a major influence on development

Table 2**Uncertainty attributes and their scores***

Attribute	Uncertainty score				
	1	3	5	7	10
Market uncertainty	Known deliverable, possibly defined contractual obligation	Minor changes in market target expected	Initial guess of market target likely to require steering	Significant market uncertainty	New, unknown, and untested market
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 incremental research involved	New technology, new architecture; might be some exploratory research
Project duration	1–4 weeks	6 months	12 months	18 months	24 months
Dependencies, scope flexibility	Well-defined contractual obligations or infrastructure with published interfaces	Several interfaces Scope isn't very flexible	Scope has some flexibility	Some published interfaces Scope is highly flexible	No published interfaces

* Minimally complex = 1, highly complex = 10.

methodology. If the project puts lives or business-critical functions at risk, we must treat it differently than if the only cost of failure is the project investment.

Team location

Having everyone in the same room enables high-bandwidth communication among the project team members. A widely distributed team or one in which a significant portion of the team is located several time zones apart can increase project complexity. This can be a difficult attribute to assess, because a team that has one or a few dispersed members might not drastically increase its complexity. We've advised teams to use their judgment on this assessment.

Team capacity

An established team of experts that have been working together for years on product line enhancements can almost anticipate what other team members are likely to need and do. This contrasts with a brand-new team of relative novices. In many ways, this attribute is similar to the Cockburn Shu-Ha-Ri personnel capacity that Boehm and Turner use.⁵

Domain knowledge gaps

Landmark's products include leading-edge technologies used by specialists in oil and gas exploration and production. It's critical that the product team have full-time access to the domain specialists to resolve ambiguities and produce the desired product. We've found

that this is greatly simplified when the developers are domain specialists themselves and much more complex when access to domain knowledge is limited.

Dependencies

This attribute measures the degree to which the project team depends on third parties or on other projects within the company. In general, the more dependencies there are, the more complex the project will be. You can give reduced weight to established third-party dependencies if the team has a consistent track record of working with a stable version.

Uncertainty drivers

A project's uncertainty depends on market conditions and project constraints. The primary indicators of project uncertainty (see Table 2) are

- market uncertainty,
- technical uncertainty,
- project duration, and
- other projects' dependencies on that project, and scope flexibility.

Market uncertainty

If the market needs are well known, the project probably won't need much steering. Conversely, if they aren't well understood, the ability to steer the project to the desired goal rather than to the initially stated objective will be critical. This attribute resembles the requirements change attribute that Boehm and Turner use.

We've found that one primary amplifier of market uncertainty relates to the number of customers. A single customer or single customer voice typically has less uncertainty than when there are multiple customers with multiple voices.

Technical uncertainty

Mature products using proven technology don't encounter much technical uncertainty, although sometimes we've experienced uncertainty with new domain technologies added to an existing product. On the other hand, project teams building new products often want to use the latest technology, so these projects will have a high degree of technical uncertainty.

Project duration

The longer the project takes from start to product release, the more chance there is for technical or market uncertainty to affect it.

Dependencies and scope flexibility

The degree to which other projects depend on this project can limit the amount of steering that the other projects can tolerate. Continually modifying interfaces isn't acceptable when those changes affect other projects.

Quadrant assessment

The project's overall complexity and uncertainty score is calculated on the basis of the aggregation of the individual complexity and uncertainty attribute scores:

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

$$\text{Uncertainty} = 2^{\sum \log_{10} y_j}$$

where i and j are the complexity and uncertainty attributes, respectively, for a given project, and x_i and y_j are the individual complexity and uncertainty attribute scores from Tables 1 and 2. In effect, the $\log x$ terms are scaled information measures.⁶

The 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 we cross-plotted the complexity and uncertainty values for our portfolio (see Figure 1). We found that the projects in a given quadrant were quite similar, as were the successful approaches used for managing those projects. Figure 2 summarizes each quadrant's proper-

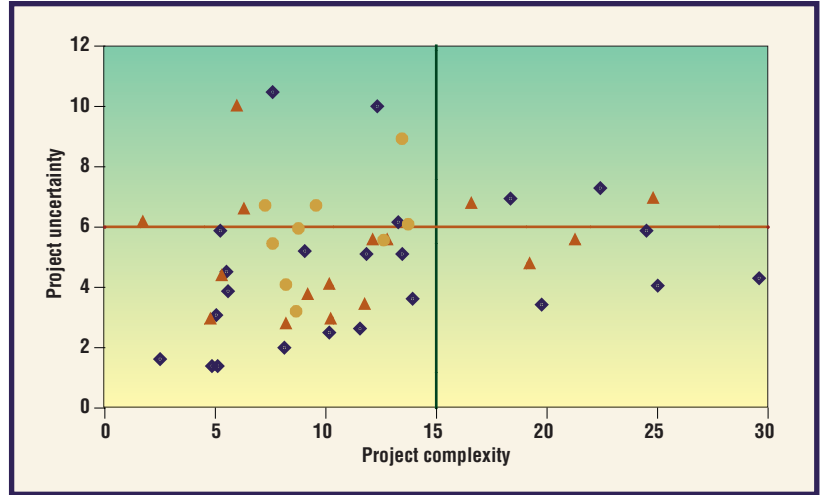


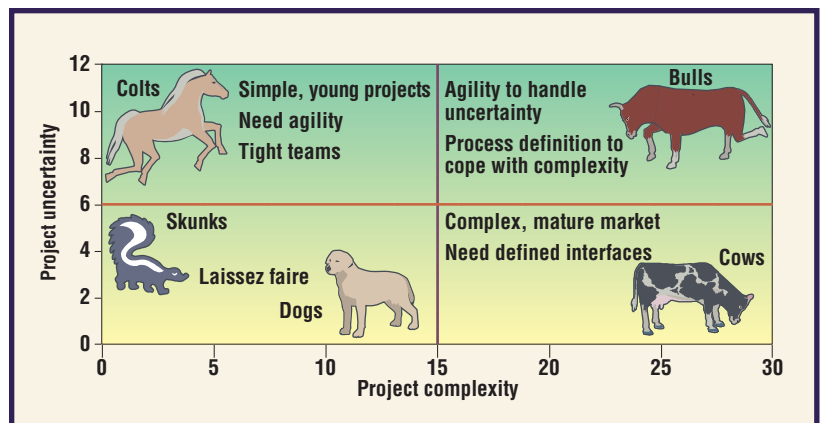
Figure 1. Project complexity versus project uncertainty for projects from three divisions. The symbols represent different divisions in the company.

ties, and the following sections describe them. Putting these ideas together, we borrowed from the concepts of the Boston Consulting Group (see "The Boston Matrix" sidebar) to create our own Houston Matrix.

Dogs: Simple projects with low uncertainty

Dogs are typically mature products being developed by small teams. With this type of project, which isn't particularly complex or uncertain, the best thing to do is to let the development team do its job to ship the product. Some projects in this quadrant have some uncertainty, but we can keep their duration short to limit the uncertainty's impact. Prototype or skunk works (R&D) projects, or *skunks*, often fit in this category. For both dogs and skunks, additional process ceremony and documentation are unnecessary and inefficient, so we run them using only the minimal core set of practices that we use for all projects in all quadrants. This approach is similar to Cockburn's

Figure 2. Houston Matrix quadrant assessment.



The Boston Matrix

The Boston Consulting Group's Boston Matrix (see Figure A) is a common tool for product portfolio management. Products are plotted on the chart according to market growth and market share.

Question marks, also known as *problem children*, show potential to gain market but have not yet established a strong market position. The objective is to convert them into *Stars*.

Stars have established a strong market share and continue to show market growth. Because they're in growth mode, they still require significant investment to generate market growth.

Cash cows are prior stars that have started to lose market share. They don't require much investment and as a result are highly profitable.

Dogs, or *pets*, are established products that either haven't

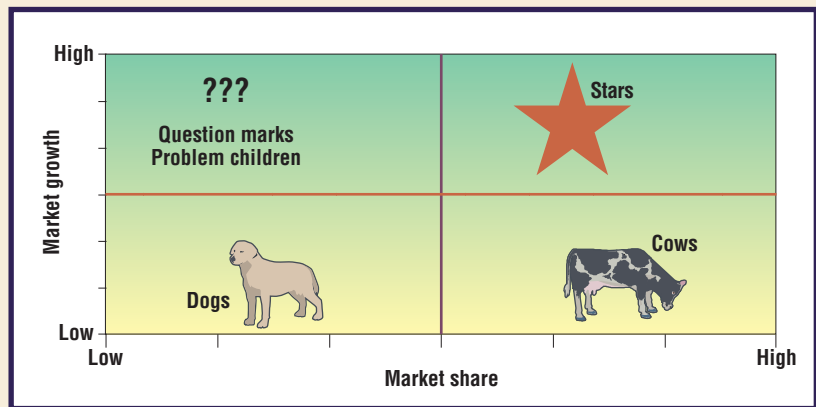


Figure A. The Boston Consulting Group's Boston Matrix. (figure courtesy of the Boston Consulting Group)

penetrated the market or are former cash cows that have lost market share. These products should be killed, divested, or made profitable by reducing expenses.

Crystal Clear.⁷ Approximately 60 percent of our projects fall into the dog quadrant.

Colts: Simple projects with high uncertainty

New products will usually have both market and technical uncertainty. If we keep our teams small, they can react quickly to adapt to those uncertainties. The metaphor of the young colt aptly describes these projects. Colt projects are just getting started and have a lot of energy and freedom. Most of our project teams that have had success with Extreme Programming⁸ fit in this quadrant. We've also found that short daily stand-up meetings such as those advocated by Scrum⁹ are effective in this quadrant. Approximately 20 percent of our projects are colts.

Cows: Complex projects with low uncertainty

The mature products and product suites that continue to have large project teams are usually the organization's cash cows. In addition to the obvious connection, the cow is a good metaphor for these projects: cows are quite large but don't move particularly fast. Cow projects have less need for agile steering; in fact, they might need disciplined change control to reduce their impact when many projects or customers depend on them. Projects in this quadrant might still be agile, but

they need defined and published interfaces to the projects that depend on them. They also require more direct project and program management, including looking at issues such as cross-team communication and critical path. Many of our cows are integration projects involving a number of projects, typically dogs. We've used a team of team leaders, something quite similar to a "Scrum of Scrums,"⁹ to manage many of these projects. Cows constitute about 10 percent of our projects.

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 to steer through the uncertainty, yet they require some process ceremony to manage complexity.

The bull metaphor is quite appropriate. Bull projects are large and can get out of control quickly if the team isn't careful. They have high visibility throughout the organization, as they're often about emerging products that have strong investment. In our case, many have been next-generation products that management intended to supplant existing cash cows. Expectations are high, yet uncertainty and complexity are equally high. Approximately 10 percent of our projects are bulls.

Core process practices

The quadrant assessment is a key early work product of our development process. First, we incorporate into our overall process framework the core practices that all projects must follow. Then we determine, on the basis of the project assessment, which other process activities we should use. The core practices include

- an aggregate product plan,
- an A/B/C list,
- a quality agreement,
- continuous integration,
- expert-user involvement, and
- a project dashboard.

Aggregate product plan

The product manager produces this concise statement of the project objectives. For each release, it contains

- the target date,
- a one-sentence product vision,
- a high-level list of priority features that are committed (I'll discuss this in a moment),
- a short description of the strategic fit,
- a list of the target markets that will be pursued, and
- the supported platforms.

A/B/C list

We categorize all the desired features into three priority levels:

- Priority A features *must* be completed to ship the product.
- Priority B features *should* be completed to ship the product.
- Priority C features *might* be completed before shipping the product if time allows.

The team estimates effort requirements and works with the product manager's estimate of value to maximize return on investment. We communicate only the Priority A features to customers.

Because we contract to deliver all the A items, we allow for uncertainty by planning the schedule so that A features consume no more than 50 percent of the overall planned effort. As we complete Priority A items over the course of the project, we use any remaining schedule time to complete the Priority B or C items. We often reprioritize during the proj-

ect, particularly at iterations, although we usually don't drop A items unless we've set proper customer expectations. (Another article describes this approach and how we use it to maximize value delivery.)¹⁰

Quality agreement

The team works with the product manager to reach agreement on quality targets for the release. We've modified Rob Thomsett's quality agreement approach¹¹ to use A/B/C prioritization. We feel that this gives us consistency in our discussions and provides a bit more granularity than Thomsett's on-off approach.

Continuous integration

For all our projects, we use configuration management and build at least nightly. A number of projects have started using a continuous-build process. Most projects using continuous builds started as colts, but some of these are now bulls or cows and still find continuous builds beneficial.

Expert user involvement

We've always found it critical to have expert users involved in development. Most of our complex projects have a dedicated expert user, usually a former customer. Nearly all our testers are also expert users, as are many of the developers.

Project dashboard

We developed a Web-based interface for reporting typical information about project status. This portfolio dashboard is an excellent way to view project health. Available information includes the aggregate product plan, quality metrics, top active risks, and any revisions to the release estimate, among other things. Project managers record this information at least weekly.

Context-adaptive process practices

We can add additional processes and practices to the core set on the basis of the project attributes. The project quadrant provides guidance for the types of processes and practices to be added.

Dogs and skunks

As I previously mentioned, these projects are generally sufficient with just the core practices.

It's unwise for an organization to have more bull projects than bull project managers.

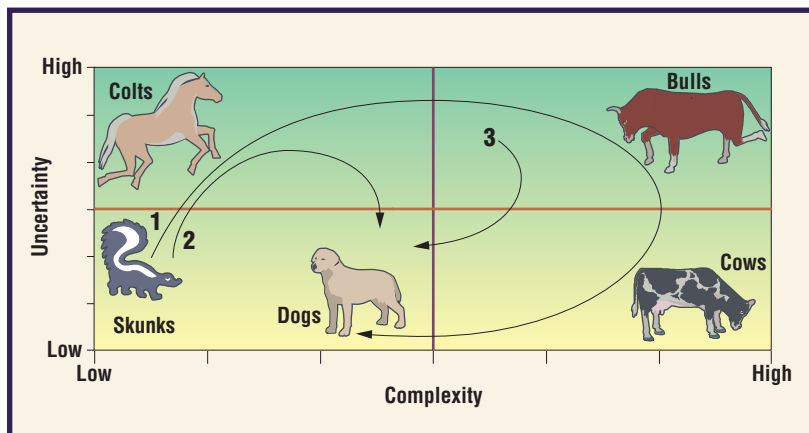


Figure 3. The life cycle of most products in our portfolio moves through many of the quadrants in our Houston Matrix.

Colts

These high-uncertainty projects benefit from three additional practices that help them cope with uncertainty: short iterations, daily stand-up meetings, and automated unit tests.

Cows

Although these projects don't have much uncertainty, they require additional processes to deal with complexity. Such activities might include more rigorous requirements management (we use a requirements tool), functional specifications for interface definitions, relatively detailed project plans with critical-path identification, and projects broken down into subprojects and coordinated by a team of leaders or a Scrum of Scrums.

Bulls

These projects are quite difficult to control: they can be large and complex and require steering to cope with uncertainty. To run successfully, they require much of the same process ceremony we use with cows and much of the agile steering we do with colts. Iterations must be more frequent than with cows but longer than for colts, and communication channels for these projects must be efficient. Most important, they require the best and most seasoned project managers, who can understand how to work with agility and cut through complexity, balancing the dichotomy. We expect that most organizations have only a few project managers with the requisite capacity to manage these projects. As such, it's unwise for an organization to have more bull projects than bull project managers. (See Marjorie Farmer's experience report for more on the trials and tribulations of managing a bull project.)¹²

Adjusting project constraints

Your teams might discover during the quadrant assessment that their project is either more complex or uncertain than they'd thought. Sometimes, you can adjust a project to reduce either complexity or uncertainty. In particular, we've often found that decomposing larger projects into subprojects has helped reduce complexity.

Product life cycle

Products in our portfolio tend to have a life cycle that moves through the various quadrants in Figure 3, again similar to the Boston Matrix. Our most successful products follow path 1 and start with low complexity and moderate uncertainty as skunks, move to greater uncertainty and complexity as colts, and then become successful and turn into highly uncertain and highly complex bulls. Over time the complexity dies down and the product becomes a cow, eventually becoming a dog.

Many products follow path 2 and never need to get particularly complex. There's nothing wrong with this, as a high percentage of these products end up being profitable.


While we've had numerous attempts to start products on path 3 directly in the bull quadrant, we haven't yet seen success with this approach. Our only successful bull projects are those that have first begun as colts or dogs.

At Landmark we strive to deliver software that maximizes business value. We believe that agile development approaches are aligned with this philosophy and have helped us improve our value delivery.

One particular metric that has shown improvement is our Estimation Quality Factor.¹³ EQF measures project estimation accuracy, essentially the reciprocal of the estimation error. An EQF of 5.0 represents an aggregate estimation error of 20 percent. As Figure 4 shows, our median EQF has improved steadily from 4.8 to 8.4 since we introduced this approach. It was already respectable compared to the industry median EQF of 3.8 reported by Tom DeMarco.¹³

Any organization will have a distribution of project types, people, and opinions about the right way to do things. No single software development process is the best approach for every project.

The scoring model in our assessment tool isn't intended to be rigorous. However, it's proven useful to the project teams and to senior management. We don't recommend using it blindly; identifying your project's complexity and uncertainty attributes is intended to stimulate thought, not eliminate it.

The assessment has also provided insight into our project portfolio management. Our overall portfolio of projects is distributed across the four quadrants. Bull projects are difficult to run, and an organization with a high percentage of bulls is taking on significant risk. Most of our projects turn out to be in the dog quadrant. Dogs can be loyal and rewarding. Provide them reasonable care and feeding, and they'll provide good results in return. 

Acknowledgments

The experience of the many Landmark development teams provided fodder for this analysis. The coauthors of the original experience report³ (Forrest Greene, Tessy Phillips, Rex Pilger, and Robert Poldervaart) along with Karl Zachry helped to formulate the development process and have provided valuable contributions to this work.

References

1. A. Cockburn, *Agile Software Development*, Addison-Wesley, 2001.
2. J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House, 2000.
3. T. Little et al., "Adaptive Agility—Managing Complexity and Uncertainty," *Proc. 2004 Agile Development Conf.*, IEEE Press, 2004.
4. B. Boehm, "Get Ready for Agile Methods with Care," *Computer*, Jan. 2002, pp. 64–69.
5. B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, 2003.
6. C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Tech. J.*, vol. 27, July and Oct. 1948, pp. 379–423, 623–656.
7. A. Cockburn, *Crystal Clear*, Addison-Wesley, 2004.
8. K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.
9. K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*, Prentice Hall, 2001.
10. T. Little, "Value Creation and Capture: A Model of the Software Development Process," *IEEE Software*, vol. 21, no. 3, 2004, pp. 48–53.
11. R. Thomsett, *Radical Project Management*, Prentice Hall, 2002.
12. M. Farmer, "DecisionSpace Infrastructure: Agile Development in a Large, Distributed Team," *Proc. 2004 Agile Development Conf.*, IEEE Press, 2004, pp. 95–99.
13. T. DeMarco, *Controlling Software Projects*, Prentice Hall, 1982.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.

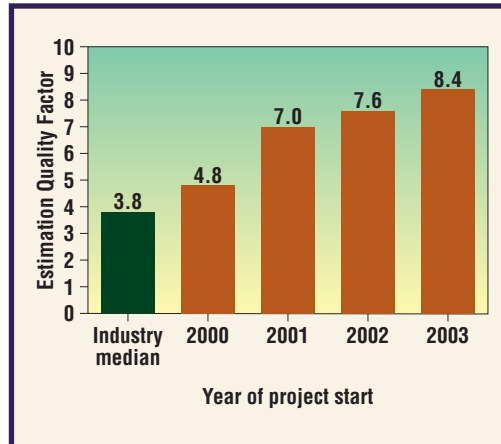
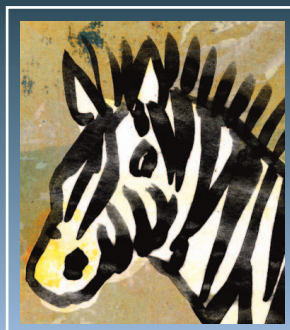


Figure 4. Improvement in our company's median Estimation Quality Factor.

About the Author



Todd Little is a senior development manager for Landmark Graphics. His current interests include generating business value through agile software development. He's a member of the AgileAlliance's board of directors, the program director for the Agile 2005 Conference, a member of the IEEE Computer Society and the Society of Petroleum Engineers, and a registered Professional Engineer in Texas. He received his MS in petroleum engineering from the University of Houston. Contact him at Landmark Graphics, PO Box 42806, Houston, TX 77242; tlittle@lgc.com.



We'd like to hear from you

SEND US EMAIL AT

Software@computer.org