12

Motivation for Becoming Predictable

How often have you heard the expression, "We never have the time to do it right but always have the time to do it over"? This theme is unfortunately common for many companies developing software. While this mode of operation may have been acceptable in the past, the global economy of the new millenium is forcing companies to become more efficient. Those organizations unable or unwilling to learn how to do it right the first time are going to be left in the dust.

Software development in the twenty-first century is changing dramatically. New paradigms such as "Extreme Programming" [1] and developing software on "Internet time" [2] are putting increasing pressure on organizations to develop better software more quickly with fewer resources.

Do more with less. Software development organizations are developing more products that are more complex and more difficult to create. To staff up for new projects, organizations are competing, fiercely at times, for a limited pool of experienced people. The tight labor market for software developers and QA professionals is expected to continue well into the first decade of the twenty-first century, since the demand for qualified people is expected to far exceed available resources [3, 4]. Doing more with less means that there are fewer people available for testing. So the testing that is done must be that much more effective.

Do it faster. As a result of the global economy, many organizations are now facing competition from new players in other parts of the world. These new players are developing products faster and at lower cost. Further,

the ability to develop products on Internet time is affecting all businesses, not just those that are developing software for the Web. Product development, testing, and release cycles that once took years are being compressed to months and sometimes weeks. So, not only do organizations have to do more with less, they also have to do it faster. Doing it faster means that the verification and validation activities have to be completed in less time.

Do it with higher quality. Customers are becoming more demanding and more selective in deciding which software products to buy. Unreliable products with exotic features are less attractive than products that contain the features customers actually need and are very reliable. Compressed schedules and understaffed projects, coupled with the need to release very reliable products, result in extraordinary pressures on people within development organizations.

Organizations need a way to cope with these conflicting demands. Doing more with less, doing it faster, and doing it with higher quality all mean that there is little room for error. More than ever before, organizations must learn how to get it right the first time. A predictable development process is crucial to getting it right the first time.

The goal of predictable software development is simply to delight your customers by consistently delivering what you promised when you promised it.

Now let us look at why this is so important.

12.1 Introduction to Predictable Software Development

For every software project, management wants to know three things: (1) when will coding be done; (2) when will testing be done; and (3) when will the product be released?

These are legitimate questions for management to ask. Unfortunately, when an organization is unpredictable, it is very difficult to provide accurate answers to these questions. Why? Well, knowing with certainty when a product will be released requires, among other things:

- A clear definition of what "done" means with respect to coding and testing;
- A process for writing and reviewing requirements (so that developers know what features to code and testers know what features to test);

- A process for designing software;
- A process for planning, estimating, and scheduling software development activities;
- A process for planning, estimating, and scheduling software verification and validation activities;
- A process for controlling changes to requirements, designs, code, and tests;
- A staff that has been trained in critical skills such as accurate estimating and scheduling, requirements writing, project management, inspections, verification and validation, and testing;
- A commitment from management to follow the process agreed to for the project.

New projects often begin with great optimism—a naive expectation that somehow this project is going to be different, that it will be successful. For some reason, management expects that the outcome for new projects will be different, even though the organization continues to use the same unpredictable processes that resulted in failure on earlier projects.

After the initial euphoria ends, the hard reality sets in. The project team quickly recognizes that the same unpredictable processes used on the last unsuccessful project are being applied yet again to the new project. Lessons learned from the previous failure have not resulted in any changes to the process, because there was no management commitment to improve. Managers and executives need to understand that having a predictable software development process is vitally important to the long-term success of their business.

Many software development organizations lack discipline. These organizations either do not have or have but do not follow a written software development process. As a result, they are not able to accurately predict when key events (such as code complete, test complete, and product release) will occur. What many companies fail to recognize is that various parts of the organization need to know when things will happen so that:

- Marketing can plan product-rollout events;
- Customer service can alert customers to new software updates;
- Training can prepare updated course materials and schedule new training sessions;

- Technical writers can prepare updates to online help and have printed manuals ready for product launch;
- Managers can plan resource assignments for the next project.

As a result of the inability to predict when things will happen, many software development organizations suffer from a lack of credibility. No one believes dates from the development group, because it has never met a date. As a result, management frequently sets (or allows to be set) unrealistic release dates for products and makes (or allows to be made) unreasonable commitments to customers.

As you might expect, the unrealistic dates set by management are rarely met. This creates a lose-lose situation—your customers lose, since their plans may be predicated on your unrealistic schedules and unreasonable commitments, and your employees lose, since no one wants to be associated with a project that fails.

On projects with unrealistic schedules, the schedule can only be reduced so much (the time required to get the minimum feature set coded and tested). Developers, being eternal optimists, fail to anticipate problems, and coding frequently takes longer than expected. QA engineers fail to accurately estimate how many tests are needed and how long it will take to write and execute them. This problem occurs because most people have never been trained in how to develop accurate estimates and build realistic schedules. When development time expands, time for software V&V activities is reduced (usually owing to a commitment made to a key customer to ship on a certain date). When V&V tasks are cut from projects, the organization will find fewer bugs and customers will find more bugs. This makes the QA staff frustrated and customers unhappy. It also means that the organization will need to do unplanned bug-fix releases.

Simply stated, the more predictable an organization is, the more likely the software V&V activities discussed in Parts I–III of this book will be performed. When those activities are performed, they will significantly increase the ability of the organization to find and fix bugs before the software is shipped to customers, thereby decreasing the need for unplanned, expensive bug-fix releases.

Lack of predictability impacts your bottom line. Unplanned bug-fix releases represent a significant cost to the organization, as shown in Figure 12.1. Management determines how to use scarce and expensive resources. You can decide to use these resources to deliver bug-fix releases,

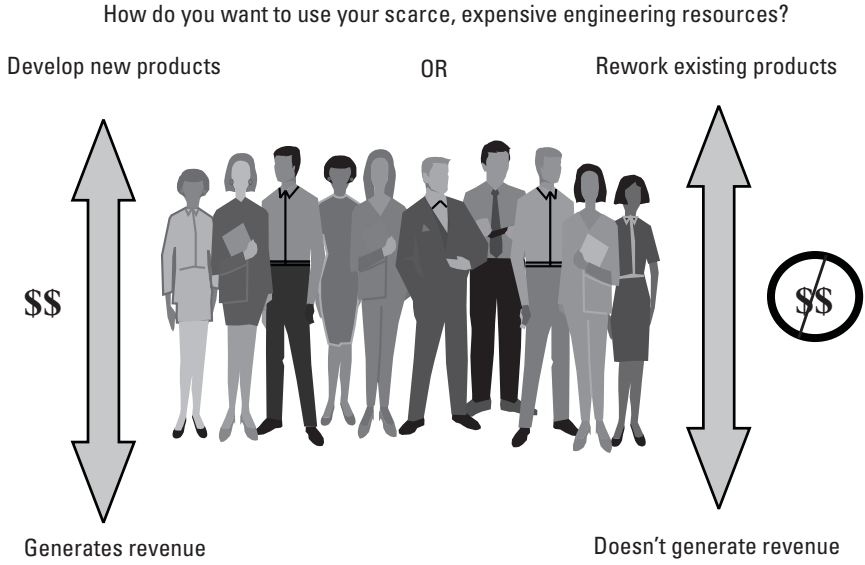

Figure 12.1 The real cost of unplanned bug-fix releases.

which typically don't generate any revenue, or to work on new features and new products, which do generate revenue. The choice is yours to make.

Lack of predictability negatively impacts customers. In unpredictable organizations, customers are unsure of when new products and updates to existing products will be released. This makes it hard for customers to plan to migrate to new software releases. Further, since unpredictable organizations are unable to develop accurate schedules, they tend to release software with far too many bugs, adding to customer dissatisfaction.

Lack of predictability negatively impacts employees. No one wants to be associated with projects that fail. In unpredictable organizations, failed projects are the norm. And experience has shown that there is a very strong correlation between customer satisfaction and employee satisfaction.

Predictable software development can be achieved when management takes the lead to change the behavior of the organization. Management needs to be focused on the elements identified in Figure 12.2.

To become predictable, organizations need to learn how to balance quality, features, and schedule. While tradeoffs are made all the time, organizations need to understand and assess the implications of these tradeoffs. Further, organizations need to learn how to balance the needs of people,

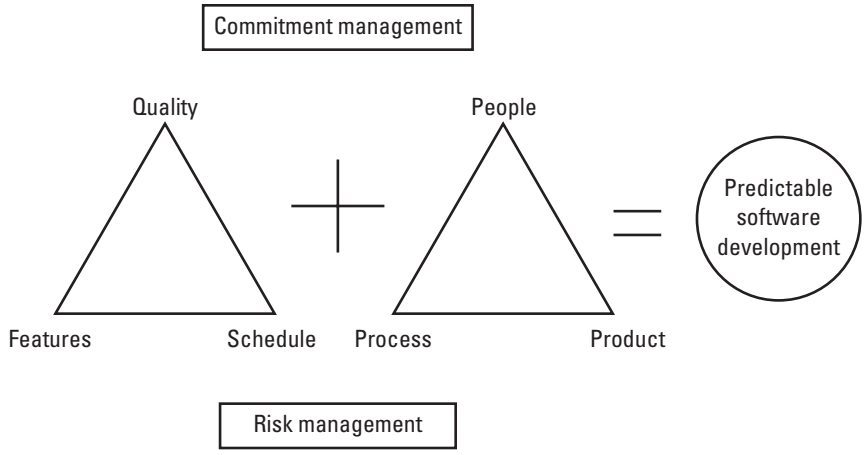

Figure 12.2 Elements of predictable software development.

process, and product. Tradeoffs here affect productivity as well as customer and employee satisfaction.

Also required is the ability to manage commitments and to manage risks. Managing commitments (internal and especially external) is essential so that the organization can consistently exceed commitments. Many complex software projects are fraught with risks. Risks on many software projects may be tacitly understood, but all too often they are not actively managed. Effective risk-management skills can make the difference between success and failure.

12.2 Characteristics of Unpredictable Organizations

How can you tell if your organization is unpredictable? Listed below are some characteristics of unpredictable organizations.

- The organization frequently over-commits and under-delivers.
- Project schedules are consistently not met.
- Customer perception of product quality is low.
- It is difficult to plan for new product releases and product rollout.
- It is difficult to plan the resources required for future products.

- Customer satisfaction is low and probably not being measured regularly.
- Employee satisfaction and employee morale are low.
- Many unplanned bug-fix releases are needed.
- Revenue projections are frequently not met.

From working with dozens of companies, I have identified several root causes of unpredictable behavior. These include:

- *Unrealistic schedules.* Unrealistic schedules can result from several causes, including lack of training in estimating and scheduling, allowing other organizations to set development and validation schedules, and not keeping or using information from past projects to help develop more accurate estimates.
- *Poor project management.* Software project management is a difficult and unrewarding job. Project managers frequently become scapegoats for failed projects. Good project managers are rare and worth their weight in gold. Without a doubt, one of the most frequent reasons that projects fail is due to poor project management. In many cases, the root cause of this problem is not the project manager, but rather, how management measures the project manager's performance. Usually, project managers are measured on their ability to get products released. Therefore, they will do whatever it takes to get the product released, including cutting features and reducing quality.
- *Crisis mentality.* For many organizations, working in "firefighting" mode is the norm. These organizations move from one crisis to the next. It is a mystery how anything gets done. What we should have learned by now is that working from crisis to crisis is not the most effective way to use scarce, expensive resources. This mode of working frequently leads to burnout and frustration.
- *Rewarding of wrong behaviors.* In many organizations, management's goals and objectives are not aligned with the individual performance goals and objectives of the staff. For example, in organizations where management complains about poor product quality, you would likely not find any mention of the word "quality" in the performance plans for the staff. Rather than encouraging the behavior that is desired, management knowingly or unknowingly does the opposite. For example, management inadvertently encourages firefighting

behavior by rewarding "heroes" who resolve the crisis-du-jour. In fact, in organizations that work in crisis mode most of the time, in many instances it is the so-called "heroes" who frequently cause the crises in the first place.

- *Lack of measurement.* Many organizations are unable to measure the amount of effort required to develop, document, and test a software release. Some organizations are unable to answer basic questions, such as how big is the product (using any particular size metric); how long did it take to develop and release; and how many bugs were found?

12.3 Characteristics of Predictable Organizations

Predictable organizations exhibit the following characteristics:

- Under-commit and over-deliver;
- Master skills for estimating tasks and building realistic schedules;
- Use project postmortem information to refine estimating and scheduling skills;
- Make effective use of scarce, expensive resources;
- Rarely operate in firefighting mode;
- Require few unplanned bug-fix releases;
- Follow a documented software development process;
- Actively manage risks and commitments;
- Measure quality and customer satisfaction regularly;
- Measure employee satisfaction regularly;
- Recognize the connection between customer satisfaction and employee satisfaction.

These characteristics will be discussed further in subsequent chapters.

12.4 Management Can Change the Organization

Being a parent who has helped raise two children, I learned the value of positive reinforcement—rewarding and recognizing good behavior and

providing negative incentives for bad behavior. I adapted these principles to managing technical people and found that they worked just as well with adults as they did with children.

Management has the ability to change the organization. Management controls the organization's human resources, determines how resources are allocated to projects, and most importantly, determines how people are evaluated and measured. What management often fails to recognize, however, is that to change the culture, you need to change the way people behave. In most organizations, the way people behave is directly related to how they are measured.

This is particularly true for software engineering organizations, where technical challenge and peer recognition are very important. The notion that management can change the culture of an organization by changing the way people are measured is not new. It has been acknowledged for many years [5, 6]. Unfortunately, in many organizations, management hasn't recognized it.

The following are some specific actions management can take to help create an organization that behaves in a more predictable manner.

- *Measure individual performance based on objectives that are directly related to overall corporate goals.* Most organizations have general corporate goals such as increasing market share and improving quality and customer satisfaction. Frequently, management fails to define specific goals and objectives for individuals that are based on the corporate goals. Setting individual goals and objectives is vitally important, since we know from experience that people will be conscientious about those things that they are being measured on.

  A client having quality problems asked me to come up with recommendations for improvements. I asked to see the performance plans for the software engineering staff. Not one person had "improve code quality" or " reduce defects" as an objective. My recommendation was to incorporate such objectives into the performance plans. The client did this and over time the quality problem disappeared.

- *Learn to develop accurate, realistic schedules, and then meet them.* The literature is full of horror stories about project failures, cost overruns, and missed schedules [7]. The ability of an organization to define an accurate and realistic schedule is critical. Yet few organizations know how to do this. Why? Well, there are probably many reasons, but what I have observed is that most people have never been trained,

while in school or on the job, in how to accurately estimate a task, build an accurate schedule, and then meet that schedule. Even in those organizations that conduct project postmortems, there is little or no improvement in the organization's ability to develop accurate schedules that can be met. Skills required to develop accurate estimates and build realistic schedules are discussed in Chapter 14.

- *Follow a documented software development process.* One of the reasons that organizations are unpredictable is that either they don't have a documented software development process or they don't follow the process they do have. We know from experience that to successfully develop complex products you need to rely on a proven process. This is as true for software as it is for any complex product. For software, the process must address both development activities and verification and validation activities, as described in Chapter 3 and Appendixes G and H.

  Without a documented process, people will spend a significant amount of time arguing about what to do. It will be harder to develop accurate estimates and schedules. Groups like software QA and documentation that need to have specific information will not know if or when that information will be available. Further compounding the problem is the claim from some software engineers that having to follow a process will diminish their ability to be creative. Nothing could be further from the truth. Ask hardware engineers if following a process diminishes their ability to innovate. It doesn't. I've found that software engineers who resist following a process do so out of fear that they will actually be held accountable for some task (such as writing a design specification) when they would rather spend all of their time writing code. This topic will be further discussed in Chapter 15.

- *Hold people accountable.* Accountability is a concept that is totally foreign to many in the software industry. However, it is essential for those organizations that want to become predictable. Everyone in the organization, from the CEO on down, needs to be held accountable for doing his or her job. On a project team, people need to be held accountable for meeting their schedule, assuming of course that the people doing the work set their schedules. Development should be held accountable for delivering what was promised, not just to the external customers but to internal customers (such as software QA and technical documentation) as well. Software QA should be

held accountable for writing and executing tests based on the SRS and for performing the testing as defined in the schedule.

Management needs to give people the responsibility for defining what they will do and when they will do it by. Once this happens, people need to expect to be held accountable for delivering what they promised. When people miss their commitments, managers need to understand the reason and determine how to get things back on track by working collaboratively and cooperatively with everyone involved.

Note that accountability extends to managers as well. Oftentimes management is responsible for providing resources, buying equipment and tools, and taking care of other things that project teams need to meet their commitments. This topic will be further discussed in Chapter 16.

- *Proactively manage risk.* Most software development projects are fraught with risk, yet few project managers proactively manage risk. On projects where risk management is ignored, I frequently see several "replanning" activities. Replanning is often just a euphemism for dealing with something that wasn't expected but could possibly have been avoided. Many times, replanning activities can be prevented if proactive risk management is used from the outset. Proactively managing risk not only helps ensure that development and software QA will meet their schedule, it increases the likelihood that the project will be successful. Risk management is further discussed in Chapter 16.
- *Manage internal and external commitments.* In many organizations, salespeople frequently make unrealistic commitments to customers in order to sell product. This often results in undue pressure on project teams to deliver. When they can't deliver (because the commitments were unrealistic), customers become unhappy. Meanwhile, the salesperson has received their commission check and is off with other customers making more unrealistic commitments.

  The underlying problem here is an organizational one. There is a disconnect between sales and development. And it all goes back to how salespeople are measured. In many organizations, salespeople are measured on the dollar amount of sales they book, not on meeting commitments to customers. In fact, some customers have finally smartened up and have signed agreements that have penalty clauses in them based on agreed-to delivery dates and feature sets.

Surprisingly, even with these financial incentives in place, management still has failed to hold salespeople accountable.

Management needs to manage commitments made to external and internal customers. One way to do this is to set expectations lower. By doing this, the organization has a much better chance of meeting or exceeding expectations. All too often, organizations set expectations unrealistically high and frequently fail to meet them. As a customer, we are generally very satisfied when we receive exactly what we expect. If we receive more, we are very happy. If we receive less, we are usually not happy. Therefore, management should encourage the organization to undercommit and overdeliver.

In addition, people who make commitments to customers need to be held accountable for meeting those commitments. The same is true for commitments made to internal customers. Managing commitments is discussed in more detail in Chapter 16.

- *Measure what happens.* It goes without saying that in order to become more predictable, organizations need to measure what happens. A few simple measures that are tied to the overall corporate goals should be sufficient. For example, What is the average amount of schedule slippage on projects? How many known defects are products being shipped with? How many unplanned bug-fix releases were there last year? Measurement is discussed in Chapters 7, 10, 13, and 14.

Management must recognize that they have the ability to change the behavior of their organizations. By learning how to do this, management can significantly improve the organization's predictability.

12.5 Summary

As observed by Jones [8], "One of the most important topics in the entire software quality domain is the relationship between software quality and software project management. Indeed, Deming, Juran, and Crosby have asserted that the main source of quality problems can be assigned to management rather than to technical workers."

Management must provide leadership to help organizations behave in a more predictable manner. Management must understand that they can change how people behave by changing how people are measured. If you

want to improve schedule accuracy, measure people on their ability to develop accurate schedules and then meet them. If you want to improve product quality, measure people on product quality. If you want to improve customer satisfaction, reward people based on achieving improvements in customer satisfaction.

Don't believe that this works? Jay Bertelli, CEO of Mercury Computer in Chelmsford, Massachusetts, challenged his management team in January 1999 to double the company's stock price by the end of the year. As an incentive to meet this goal, each executive was promised a new Porsche. By the end of 1999, the company's stock price had tripled. Bertelli had 20 Porsches delivered to the company's headquarters—one for each executive and two loaners for top performers among the staff [9].

References

[1] Beck, K., "Embracing Change with Extreme Programming," *IEEE Computer*, October 1999, pp. 70–78.

[2] Cusumano, M. A., "Software Development on Internet Time," *IEEE Computer*, October 1999, pp. 60–70.

[3] Schafer, M., "Hiring for Keeps (Or at Least for a While)," *Software Magazine*, April/May 2000.

[4] Wilkinson, S., "Hired Guns—Beating the IT Staffing Shortage with Contract Workers," *Datamation*, May 1999.

[5] Weinberg, G. M., *The Psychology of Computer Programming,* silver anniversary ed., New York: Dorset House, 1998.

[6] Lister, T., and T. DeMarco, *Peopleware: Productive Projects and Teams*, 2nd ed., New York: Dorset House, 1999.

[7] Yourdon, E., *Death March: The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects*, Upper Saddle River, NJ: Prentice-Hall, 1999.

[8] Jones, C., *Software Quality: Analysis and Guidelines for Success*, Boston, MA: International Thomson Computer Press, 1997.

[9] *Boston Globe,* Business Section, April 27, 2000, page C4.