

# Business Goals Count, Not Organization Size

Larry Lumsden, *Cúram Software*

*Software development is part of a business and serves business goals—no matter the organization's size.*

**Q**uestion: Given the different characteristics of small and large organizations, can they apply the same software engineering techniques?

My answer is yes. Sales, profits, and success are the goals of any business, large or small. Many factors can influence success, but when software development is an important business constituent, the relationship between the software product's characteristics and business goals is strong. We don't always recognize this, of course, because most of us became engineers for nobler reasons than serving a business—the intellectual challenge, the free T-shirts, the 2 a.m. beer and pizza to celebrate unit tests that are finally passing. But ultimately we must allow the choices we make in software development to be influenced by business goals, or else risk business failure.

Few software engineers would deny the critical influence of development processes on products and time to market. Yes, there are good and bad software processes, though the truly bad don't usually survive long (once the initial consultancy wave subsides). The key issue is that many processes aim to solve particular problems, so they prioritize some product development characteristics over others. (I'm focusing here on business-related characteristics, although there are many other types as well.) To make the right

choice, we must understand our business's most important characteristics and know which processes best support them.

## Clarifying choices

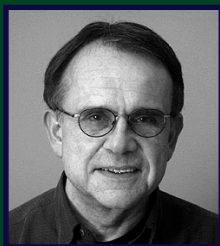
Organizations of greatly differing sizes might be developing software products to serve similar business goals, and vice versa. These organizations, large or small, might successfully use the same software processes when they require products with similar characteristics in order to succeed. Let me illustrate this choice with an example.

Consider two very different business areas:

- producing a solution for a single customer according to a fixed-price contract, and
- producing a software tool for a worldwide market.

If we're developing software according to a fixed-price contract, then the predictability of schedule, effort, and functional content will be critical to success. However, if we're producing a new shrink-wrapped product for a large market, then absorbing feedback from early releases is critical—far more important than predictability. The development process we select must reflect these priorities, no matter the organization's size.

*Continued on page 56*



## In Software Processes, Organization Size Matters

Wolfgang Strigel, QA Labs

**T**o the question of whether small and large organizations can apply the same software engineering techniques, my answer is “I don’t think so.”

The gist of the question lies in the definition of “software engineering techniques.” I agree that small and large organizations have the same overall objective when developing software: both want to produce quality software that satisfies customers’ needs. Why then would they not both apply industry best practices to achieve that goal? Well, philosophically speaking, they should. But practically, the path to optimal development results can vary significantly between small and large teams.

To illustrate the point, I’d like to consider two organizations at the extreme opposites of the size spectrum. One is a large aerospace company, which I’ll call Mega. Its core engineering team is located in the corporate headquarters at the Mega industrial park, but other developers are located in offshore teams in India. Mega typically develops systems with teams of 100 members or more. The other company is called Tiny and consists of one lonely developer located in the famous Tiny basement.

I can’t see any reason why both organizations wouldn’t use the same development methodology or the same techniques for testing. Of course, I could discuss differences

such as the cost of development tools, test automation tools, and so on, but I think the different options here are obvious and don’t address a more fundamental issue.

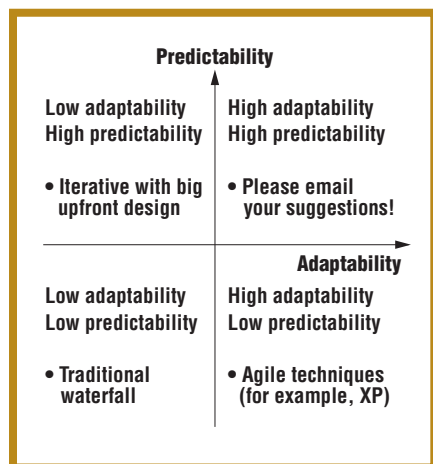
### Communication challenges

The fundamental difference in my mind is communication. Coordinating and synchronizing people is one of the biggest challenges in our profession (as it is for most multicomponent systems). Without the need for controlled communication to meet this challenge, the project management profession wouldn’t exist. We can compare a project manager to an orchestra conductor. Solo musicians don’t need a conductor; they can focus on the score and their personal flavor for delivering the performance without concern for other musicians. Members of a full orchestra don’t have this flexibility.

Software development is a team sport (except in the case of Tiny), and coordinating communication among team members is critical to success. For example, controlled communication is essential in requirements management. I submit that even Tiny should use documentation in this area. Documented requirements, whether in traditional or agile environments, represent a major difference between chaotic, ad hoc development and disciplined approaches. However, Mega

*Apart from mission criticality, organization size is the most important factor for determining software engineering techniques.*

*Continued on page 56*



**Figure 1. Process choices for two product development characteristics: schedule predictability and requirements adaptability.**

The “magic quadrant” is a simple way of representing these process choices. For example, figure 1 categorizes two general business-related characteristics: schedule predictability and requirements adaptability.

It’s possible to plot all software engineering techniques in this quadrant, reflecting how well they support these two characteristics. If schedule predictability is critical, then processes toward the top of the graph are the ones to use, whereas the processes on the right favor adaptability. (Processes on the top right might exist, but they might also involve a

trade-off with another characteristic that the business must consider.)

It’s common to view agile techniques as suitable for small companies and “Iterative with big upfront design” processes as the right choice for large mature organizations. But the business characteristics might well determine that exactly the opposite is the case.

## Business priorities

Undoubtedly, large organizations do confront some particularly difficult problems. Communication is an obvious challenge. However, many large companies retain a small company’s attributes by keeping their development teams small, colocating the business and technical people, flattening the organization hierarchy, and so on. These are ways of creating the dynamics of a small organization in a large one. It means that teams within a large organization can implement low-overhead, agile processes on projects that need the adaptability they facilitate.


Two further examples from my own experience illustrate the priority of business characteristics in choosing processes:

- Company A was an early and very successful adopter of agile techniques, but as the company grew (by acquiring other companies), its organization was distributed over three

continents. It then became important to be able to transfer projects between development sites without also transferring all the engineers. This led A to adopt a process with more residue (in the form of documentation) than pure agile processes.

- Company Y produced software for the telecommunications industry, so it had to select a process that could show compliance with international standards, immediately limiting the choice.

For both A and Y, an aspect of the business itself—not merely the organization’s size—was key in selecting the development process.

The right process is largely a function of the business characteristics that drive a software product’s development. When large and small organizations have the same business goals for a development project, they will both benefit from the processes that support those goals. 

**Larry Lumsden** is vice president of engineering at Curam Software in Dublin, Ireland, responsible for application development, tools and architecture, and quality assurance. Contact him at [llumsden@curamsoftware.com](mailto:llumsden@curamsoftware.com).

must manage the interaction of many team members who might contribute to the requirements specification and are often physically separated. Tiny, on the other hand, communicates requirements only with the customer.

We also need to consider team motivation. Even after Tiny grows to 10 or 20 team members, it still will be easy to

sit around a table, explain the importance of process, and build enthusiasm. Everybody on the team can be addressed directly to make sure the whole team is on the same page regarding the need for good software engineering practices and their commitment to apply them. Techniques can be adjusted as required and this can be a consensus de-

cision. At Mega, the mandate to apply defined processes may come from the Chairman of the corporation, but each project team has its own dynamics and own challenges. Communicating enthusiasm and ensuring commitment is a lot more difficult. Deviations from the defined process may need formal approval. The commitment to apply the

defined techniques must be reflected in everybody's personal objectives.


## Change control

Similarly, change control and code management (or configuration management) are significantly more complex at Mega in comparison with Tiny. Increased complexity calls for different techniques to establish control methods and mechanisms. The management controls provided by project managers and team leaders increase in direct relation to the organization's size. External control mechanisms such as quality assurance also become more meaningful in large team situations.

The process-improvement community is well aware of the different tech-

niques required for large as opposed to small organizations. The community has developed many CMM assessment-method derivatives to address the different needs of medium-sized and small companies. These custom-tailored assessment methods not only reduce the scope of the assessed software engineering techniques but also drop certain disciplines altogether (for example, sub-contract management, organizational process, and intergroup coordination).

**M**ega and Tiny give an extreme example of the differences in software engineering techniques required for large and small organizations. As Tiny grows, it must gradually introduce more sophisticated and robust software engi-

neering techniques. Small companies get easily overwhelmed and sidetracked by too much process. Techniques as well as processes must be tailored to an organization's needs. Apart from an application's mission criticality, which determines the degree of required risk management, organization size is the most important factor for determining the most suitable software engineering techniques. 

**Wolfgang Strigel** is general manager of QA Labs, a US Technology Inc. company. His special area of interest is software engineering process and, more recently, software testing in the context of pragmatic business needs. Contact him at [strigel@qalabs.com](mailto:strigel@qalabs.com).

## Larry Responds

The fulcrum of this debate is whether we're open to the idea that organizations have much to gain from a careful consideration of all characteristics of the software engineering techniques that are available to them. It's a commonly held view that there are large company processes and small company processes. But many developers in large companies also feel bogged down by administration and bureaucracy, whereas developers in smaller organizations are confounded by chaos and lack of structure. Small and large organizations can learn a lot about processes and techniques from each other.

Small companies and teams were the first champions of agile techniques. Their subsequent widespread adoption has been fuelled, in part, by developers' enthusiasm and willingness to use them, but also by the absolute focus on user needs that they foster. Large companies have now adopted agile techniques and gained many of these benefits, a clear example of techniques that both small and large organizations can use.

If you're in the innovation business, your processes had better help you be innovative, regardless of your size. This means facilitating the emergence of new requirements and feedback. If your products must perform exactly as specified, you'd better have process support for that—even if you're a single developer working in a basement. There's much more to choosing a process than how many developers you've got. Consider your business goals, and choose wisely.

## Wolfgang Responds

Larry is right that software development is a business, no matter the business's size. But let's face it: Wal-Mart and a corner store are both in the consumer-goods business and both have the business goals of making a profit and avoiding undue risk. However, because of Wal-Mart's size, the logistics of just-in-time delivery and related shelf life are critical to its profitability, while the corner store only needs to worry about the shelf life of vegetables. Wal-Mart needs to maximize shelf-space allocation and exposure. The corner store has a limited number of shelves and doesn't need a process to reallocate space. Surely, they require different processes.

I strongly disagree with the proposition that large companies tend to retain small-company attributes. This is the stuff of textbooks. It would be nice if the real world took heed, but anybody who's consulted with large IT shops knows that reality is quite different. Granted, product companies tend to be more aware of the benefits of a flat organization, but even they rarely implement the small-team atmosphere successfully.

Of course, the business objective matters for both small and large companies. But it's also self selective. Tiny wouldn't develop mission-critical software for aircraft control, and Mega wouldn't bother developing a stand-alone application for a one-time use. So, the companies can and should use different processes, even if they're both in the business of software development. Hence, size does matter.