Software Development Productivity

Productivity is the prime determinant of our standard of living. If the revenue generated per work hour goes up, income levels rise. If productivity lags, wages also decline. When all else is equal in a market, the more productive company will enjoy greater profits. Thus the key to sustaining and increasing wages in the software development industry is year-to-year improvements in software development productivity.

Over the decade of the 1990's, productivity increases in the technology sector resulted largely from increased capability of the underlying technology and intense customer demand for new technology. In recent years, however, software development productivity has stagnated as the demand for newer, faster technology has flattened. Thus it should come as no surprise that wages in the software development profession have flattened or declined. In order for the software development industry to see rising incomes, year-to-year increases in software development productivity must be generated.

As the technology sector matures, it can no longer depend on increasing growth in the underlying technology to fuel productivity increases. The time has come for serious efforts to increase productivity through more efficient use of labor and more effective value propositions for customers. This is how more mature economic sectors have been increasing productivity for decades.

But first we need to define what we mean by productivity improvement in the software development industry. Traditionally, we have measured productivity as thousand lines of code (kloc) per labor hour. However, the key process of a development activity is the transformation ideas of into products. To measure the real productivity of software development, we need look at how efficiently and effectively we turn ideas into software. So perhaps we should start with a new definition of software development productivity:

1. For companies that develop and sell software as a product, productivity may be defined as the revenue generated per employee.

2. For internal IT organizations, productivity may be defined as increased revenue realized by the supported business per dollar spent by the IT organization.[i]

There are three basic approaches to productivity improvement:

1. Reduce product costs by eliminating investments in product features that customers do not find valuable.

2. Reduce indirect costs by streamlining processes and eliminating inefficiencies in development, delivery and support.

3. Increase revenue by adding more value to a product so customers will pay more for it.

Let's explore each of these approaches to improving software development productivity.

1. Reduce Software Development Effort

   About two-thirds of the features of a typical software system are seldom or never used. Only twenty percent of the features are used frequently.[ii] Eliminating extra features that no one really wants represents the single largest opportunity for increasing software development productivity in most organizations.

   Extra features are generated by a software development process that attempts to nail down the features in a system at the beginning of the development process. Typically, customers are asked to decide at the start of a project what features they want. Often they have little incentive to keep the features list short, but they are penalized if the forget to include something. Can there be any wonder that a feature list generated with such incentives contains far more features than are really necessary?

   The biggest opportunity for reducing software development effort is to limit overproduction of features by developing features on an as-needed basis. For many companies, this may require a paradigm shift in architecture and design. However, it is becoming increasingly apparent that there are many disciplined approaches to software development that provide for an emergent architecture. Key approaches include:

   a. Automated test harnesses developed at the same time as the underlying code

   b. Refactoring the code on an on-going basis to keep its design simple and clean

2. Streamline the Development Processes

   The measure of maturity of an organization is the speed at which it can reliably and repeatedly execute it's key processes. A mature software development organization is one that can rapidly, reliably and reliably translate customer needs into deployed code. Too often, rapid software development has been equated with sloppy work, so attempts to streamline the development process are often looked upon with suspicion. However, in industry after industry, when sequential development processes are replaced with concurrent development processes, costs are slashed, quality is improved, and development time is dramatically reduced.

   Organizations which record year-on-year productivity improvements spend a lot of time focusing on streamlining key processes while increasing their reliability. They do this by focusing on the flow of value through the process. There are three main techniques used to do this:

   a. Value stream mapping is a tried and true approach to streamlining value-creating processes. A value stream map of the current state is invaluable

for spotting waste; a value stream map of the desired future state is a roadmap for process improvement. Poppendieck.LLC helps organizations use value stream mapping to improve software development productivity.

b. Kaizen events are a typical implementation vehicle for streamlining operational processes. In software development, a modification of Kaizen events is usually required, because software improvement efforts usually take more time than is generally allocated for Kaizen events. We facilitate effective Kaizen events for software development.

c. An Integrated Product Team (IPT) is frequently used to facilitate information flow across an entire development team. Software development IPT's involve not only architects, designers and developers, but also those responsible for deployment, operations, customer support, and maintenance.

3. Increase Customer Value

Understanding customer value well enough to obtain additional revenue is the third key to increasing software development productivity. In general, customers cannot be relied upon to tell a software development organization how to increase the value of its offerings. In fact, customers generally think of software as a tool that should adapt to their needs over time. This makes understanding customer value elusive not only during a software development project, but even after deployment. And yet, the price that can be charged for software is directly related to understating how to increase its value proposition.

We are not likely to increase software's value proposition unless we increase our understanding how customers might use our software to create value for themselves. There are three steps to increasing customer value:

a. Iterative development with frequent releases creates a short feedback loop between customers and developers. The key to successful iterative development is to prioritize the feature list, implement features in order of business value, and deploy them as soon as possible. The short feedback loop created by iterative development and early deployment not only limits the development of unnecessary code, it brings to light innovative new uses of technology that can significantly improve business results.

b. The next step to understanding customer value is to understand how our customers create value for their customers. Increasing the support of key value creating processes for customers is the most likely source of increased revenue for software development. One method of discovering how our customers create value is to focus on their key processes and map their value stream.

c. The final step to providing customer value is to link organizations through partnerships that focus on the overall productivity of the combined

enterprise. Peter Drucker noted in *Management Challenge for the 21st Century*[iii] that "In every single case…the integration into one management system of enterprises that are linked economically rather than controlled legally, has given a cost advantage of at least 25 percent and more often 30 percent." The bottom line is that when organizations work together for their mutual benefit rather than optimizing the results of the individual organizations, a large increase in overall productivity can be realized.

Software development organizations have always emphasized process; however, the focus has been on predictable delivery of pre-defined scope rather than increased productivity. Not surprisingly, processes which did not value productivity did not deliver productivity increases; quite often they decreased productivity instead. We need to move from processes which define scope early to processes which allow only the most valuable scope to be addressed. We need to move from slow processes with many wasteful steps to streamlined processes which eliminate non-value-adding activities. We need to focus on increasing revenue potential over cutting costs by discovering how we can help our customers increase their productivity.

---

[i] Productivity for internal IT organizations may be also defined as increased revenue realized by the supported business per employee in the IT organization, but such a definition would ignore make-vs-buy decisions, not to mention outsourcing.

[ii] Johnson, Jim, Chairman of The Standish Group, 'ROI, It's Your Job,' Published Keynote Third International Conference on Extreme Programming, Alghero, Italy, May, 26-29, 2002

[iii] *Management Challenge for the 21st Century*, Peter Drucker, Harper Business, 2001, p 33.