

Phillip G. Armour

## The Case for a New Business Model

*Is software a product or a medium?*

**T**here is no doubt the triumph of modern production has been the development of the manufacturing process. From Eli Whitney's cotton gin, through Simeon North's use of standardized rifle parts, to Frederick W. Taylor's scientific management concepts, the concept of assembly and manufacturing achieved its ultimate expression in Henry Ford's Dearborn assembly plant. Underlying the techniques these pioneers devised was one principle: Product is king. This principle is so omnipresent in business, it is largely unquestioned. The application of similar structures to the business of the production of software is also somewhat unquestioned. The push is for standardized parts, for process, for automation, and for repetitive operations. But there is an underlying misassumption we make in this drive for engineering credibility and consistency. It is this: *Software is not a product.*

TERRY MIURA

"What do you mean our software is not a product?" the general

manager recoiled as if bitten. "We've been pounding it into people's heads that it is. We've been saying it's the most important product we produce! This *non-product* is currently replacing key

hardware elements of our old product line at a rate of 20%–50% per year!"

A nerve had obviously been touched.

"Let's see..." the GM continued, ticking off points on his fingers. "...We design it, build it, test

it, package it, and ship it. We maintain it, replace it, patent it, own it, license it, and most of all get paid for it. How much more of a product could it be?" He sat back with the triumphant air of a poker player laying down a straight flush.

But software is not a product. It is a medium. Specifically, it is the most recent in a series of media that exist to store knowledge.

### The Five Media of Knowledge

- **DNA.** Most evidence points to the development of DNA around eight billion years ago. DNA exists to store knowledge. It is a set of Turing machine instructions on how to create life. Knowledge is painstakingly stored in this medium by the universality of life. A passing grade is survival. Species that don't learn don't graduate to

the next level, they get expelled. DNA was the first knowledge storage medium.

- **Brains.** Maybe 2.5 million years ago, nature chose the human race to try an experiment. Our ancestors grew a brain that could store information, erase it,

## Software is not treated as a medium, it is treated as a product, and this is the problem. The product is not the software, the product is the knowledge that goes into the software.

restore it, change it, and add to it. They could learn. Humans have little instinctual (DNA-stored) knowledge. More than any other animal, we use our brains as a place to store and restore knowledge. The brain was the second knowledge storage medium.

- **Hardware.** Humans have been called “tool makers.” Tools and people get along very well, and for good reason. However, the value of a tool such as a hand ax is not in the rock itself, but in how it was selected and modified. The knowledge of the toolmaker is what makes the difference. A rock is just a rock. A good hand ax is designed from the right material and selected for the job, flaked and chiseled to the right angle for the type of cutting it will perform. It is just heavy enough, sharp enough, strong enough, and big enough. Otherwise it is just a rock. It contains what management guru Peter Drucker calls “solid knowledge.” Hardware was the third knowledge storage medium.

- **Books** Although books were probably invented in China about 7000 B.C., they didn’t make much of an impact until they became cheap and available around 600 years ago. Books gave access to knowledge that had hitherto been locked up in

brains. They made knowledge portable through space and time. Books were the fourth knowledge storage medium.

- **Software** About 50 years ago, the human race found another place to store knowledge. After a slow start, this medium is now growing at an astonishing rate. Vast numbers of people are employed in gathering knowledge from myriad sources in other media, understanding and classifying it, translating it into this medium, and then attempting to validate this knowledge. There is a reason for all this activity. This medium has valuable characteristics, characteristics the other media do not have. Software is the fifth knowledge storage medium.

The value of a hand ax is in the skill and knowledge that went into its design. The value of a book is not in paper and ink, but in the ideas contained within. Equally, the value of software is not the code, but what the code does. Or rather, the knowledge that the code contains. It is easy for me to write code. It is not easy to write code that “works,” because then I have to know what “works” encompasses—what does the user want to do? In what environment will the code run? Under what circumstances will the customer employ the functions in the code?

To do what? When? What other users or other functions will be employed at the same time? Etc., etc., etc. Each of these et ceteras is a component of knowledge I must first understand, and then insert into the software.

### Characteristics of Knowledge Media

If we look at the characteristics of each of these media, we see why we are so busy transcribing knowledge into software. Knowledge in each medium has different levels of persistence (how long knowledge stored will remain), update speed (how quickly the knowledge can be changed), intentionality (how much the knowledge storage and change can be done deliberately), and “activeness” or ability to affect the outside world (if and how knowledge can be applied to action). The following are characteristics of each of the five knowledge media:

- Knowledge in DNA is persistent, but updates rather slowly. We don’t have much ability to intentionally change it. It can grow a physical artifact to affect the outside world.
- Knowledge in the brain is very volatile, though this allows us to change it quickly. It is generally intentional and through our bodies, we can apply the knowledge to the outside world.
- Knowledge in hardware is quite persistent, but it is not usually easy to update. It is intentional and of course it exists to affect the outside world.
- Knowledge in books is quite persistent, though slow to update. While books are intentional, they have absolutely no capacity to change the world.

# The Business of Software

- Knowledge in software has all the characteristics we value: it is persistent, quick to update, intentional, but most of all it is *active*.

Drucker, in *Post Capitalist Society* pointed out that pharmaceutical companies no longer spend money making drugs. They spend money learning how to make drugs. And where do they then put this knowledge? Into software, of course. Why? Because there it is persistent, updatable, intentional—but mostly active.

## The Medium of Choice

These valuable characteristics mean that software is the medium of choice. Excluding for a moment DNA, there are only four places we can “put” knowledge once we get it: we can leave it in brains, we can build a hardware device, we can write a book, we can develop software. If we need variability in the knowledge, or we have already put the knowledge of how to manufacture things into software (which we increasingly have accomplished), we are presented with only three places we can place knowledge: brains, books, or software.

Of these, keeping knowledge in my brain helps only me, and I am liable to forget it. Storing knowledge in a book helps retain it, but there it doesn't actually do anything. Storing my knowledge in software allows the knowledge to be executed.

As an example: if I happen to know how the U.S. tax codes are interpreted, there are three things I can do with that knowledge: I can leave it in my brain and become your accountant; I can write a book on how to interpret the U.S.

tax codes; I can put this knowledge into software that you can execute.

The advantages of the last option are overwhelming and explain why, across the planet, accountants are putting accounting knowledge in software. Also, chemical engineers are putting chemical engineering knowledge in software. Scientists and business people are writing code. Doctors and nuclear engineers are programming computers. Why? Because this makes their knowledge executable, it makes their knowledge usable.

It is important to note that, just because an accountant puts accounting knowledge into software, he or she does not become a software engineer. It is not knowledge of software that is being captured, it is knowledge of accounting. Software is just a place to put it. Ditto electrical engineers and scientists. Software is simply the storage medium.

But software is not treated as a medium, it is treated as a product, and this is the problem. The product is not the software, the product is the knowledge that goes into the software. But this is not the way we manage it. The product focus is so strongly entrenched that it causes us to release software that does not contain the correct knowledge in the mistaken belief that we are shipping product. It causes us to design and build systems that, if they do contain the “correct” knowledge, store it in such a contaminated way, that it is impossible to reconstitute it from the code. It causes us to spend copious amounts of time documenting the code (the book form), or alternatively leaving the

only version of the precious and costly knowledge in the collective heads of the developers (the brain form), where it will evaporate within the year.

Even more critical, if software is not a product, then software development is not a product-producing activity—it is a knowledge-acquiring activity. And a knowledge acquisition business is very different from a product production business.

It is easy to write code. In fact, we really don't spend time writing code or even building systems, we spend time trying to figure out what the heck the systems should accomplish. And too often the activity of building a system is actually just the mechanism we use to acquire the knowledge.

## Seven Pointers to Managing the Real Product

So why do we manage software as if it were a manufactured product? This is partly an accident of history—the traditional knowledge storage media of books and hardware have a physical presence that must be managed. But software does not. There is a difference between the design of a car and an actual car. There is not the same difference between the design of software and the actual software.

We know the consequences of this product orientation, but how would a knowledge orientation be different? Here are some pointers.

First, while software (and books and hardware) can store knowledge, only a brain can create it. Representational models, modularity, structure, class, and type associations and all the other paraphernalia of software development are not required

## The Business of Software

**COMMUNICATIONS  
of the ACM  
can direct your  
advertising message  
to over 80,000  
highly educated  
subscribers who belong  
to the world's oldest  
most respected  
Computer Association**

**Developers,  
programmers,  
software engineers,  
and technical managers.  
80%  
of whom design, write  
or customize software.  
67% to 90%  
of whom don't regularly  
read other computer  
publications.**

**If your advertising has  
been missing from**

**COMMUNICATIONS  
of the ACM,  
you've been missing  
the chance to reach  
72% of our subscribers  
involved in purchasing  
Applications Software,  
and 66%  
involved in purchasing  
Software Development  
Tools!**

**For information about  
advertising  
call Lynne Lancaster,  
Advertising Director  
212-626-0685 ,  
e-mail:  
lancaster@acm.org.**

by software, the machines, or even (arguably) the knowledge—they are required by the people. A computer doesn't care whether code is structured or unstructured. We care. To understand the needs of systems, we have to understand the needs of humans.

Second, use the language, representational models, and methodologies to manage the knowledge content of the domain. But do it in a *usable* form.

Third, structure the workplace to learn, rather than build, establishing an expectation that the job is to acquire knowledge.

Fourth, set up systems that store the knowledge as it is acquired in a usable form, that make the knowledge available in a usable form, to others who are trying to acquire further knowledge.

Fifth, train developers in a different way: first in the domain in which the knowledge operates, second in the ability to acquire knowledge, third in the structuring mechanisms (the methodology) of the representational form. Arguably, most developers would hardly need to be trained in software knowledge.

Sixth, treat all activities and events as knowledge acquisition. Some of the most catastrophic product “failures” are actually the source of the most pertinent and useful knowledge—if we don't throw it away.

Seventh, somehow establish knowledge-based (not product-based) asset accounting methods that assign value to the knowledge, and monitor its development and use.

Note, none of these steps reference translating the knowledge into a “product”—the traditional

“coding” step. Why not? Well, because that's rather easy to accomplish once we have the knowledge. Though we could simply assign “software engineering” as a domain in its own right, and treat it accordingly. Also, the implication is that these needs of the human place more of a stamp on the methodology than do the needs of the domain. And so they do. In reality the needs intersect, and the method used must support both the domains and the people trying to understand the domains. But the human need is paramount. It is only humans that can understand, and only people that can acquire knowledge. And speaking of people, it is commonly said that “people are our most valuable resource.” In software development this is not true. In software development, people are the only resource.

More than anything, the change in the business model requires a change in perspective and attitude, away from the “have we shipped yet?” approach. Away from “more overtime, because more hours equals more product.” From “we failed” to “we learned,” and from “what did we produce?” to “what did we find out?” from approaches and processes whose job it is to put some form of knowledge into an executable form in the shortest possible time, to processes that acquire, catalog, store and retrieve domain knowledge.

Oh, and we will produce a product in the end. But it's not *the* product. ■

---

**PHIL G. ARMOUR** (armour@corvusintl.com) is a vice president and senior consultant at Corvus International Inc, Deer Park, IL.

---

© 2000 ACM 0002-0782/00/0800 \$5.00