CyonResearch™

The Road to the Future: How Product Development Can Make the Journey More Profitable

A Cyon Research White Paper
September 15, 2003

Cyon Research Corporation
8220 Stone Trail Drive
Bethesda, MD 20817-4556 USA

phone: 301-365-9085
fax: 301-365-4586
Web: www.cyonresearch.com

The Road to the Future: How Product Development Can Make the Journey More Profitable

A Cyon Research White Paper
September 15, 2003

Executive Summary

The most important asset a manufacturer can possess is the ability to consistently create winning products. From a customer's viewpoint, a winning product is one they want to buy. From a manufacturer's viewpoint, a winning product is one that meets the spec they started out with, including hitting the targets for time-to-market, cost, manufacturability, and quality.

The capability of a company to create winning products is tied to its product-development processes. While product-development processes vary from company to company, all have five building-blocks in common:

- Information,
- Knowledge,
- Communication,
- Decision, and
- Execution.

For a manufacturer to optimize their product-development process, they must use technology in support of optimizing each of the five building-blocks. The purpose of this software technology is to enable those involved in the product-development process to do their jobs more effectively. Incremental improvements brought about by software that addresses all of the building-blocks can drive step-improvements in a company's product-development process.

Companies which are evaluating product-development tools should pay careful attention to usability, and favor products which are "discoverable." (That is, they can be learned independently of a formal training course by users who are experienced in their domain.) Further, they should specifically look for products which support open interfaces and open data standards, as these are necessary not just for simple interoperability with other systems, but for the effective use of information and knowledge generated by other systems, both inside and outside of the product development process.

The Road to the Future: How Product Development Can Make the Journey More Profitable

Caminante, no hay camino, se hace camino al andar.
Traveler, there is no road; you make the road as you go.
- Antonio Machado

All businesses are looking for a profitable road to the future. The dilemma is that it doesn't really exist. There are guides, roadmaps, and plans—but the road, as the poet Machado said, has to be made as you go. And for every company, that road is unique. No one will lead—and none can follow.

If you are a manufacturer planning your direction for the future, you'll have no lack of business "gurus," authors, and consultants who are willing to provide you with advice—sometimes for a healthy fee. Yet there is a basic concept that is possibly so obvious that it's easy to forget:

The most important asset you can possess is the ability to consistently create winning products.

This is not saying your most important asset is a winning product. Your ability to *create* winning products—not just occasionally, but consistently—is what really matters. There are two perspectives as to what constitutes a winning product. For your customers, it's easy: A winning product is one that they want to buy. It is a product that offers a combination of value, quality, functionality, esthetics, and innovation that is, to them, superior to any other alternatives. For you, a winning product is one that meets the specification you started out with. That means it hits all the targets that you laid out for it, including—especially—time-to-market, cost, manufacturability, and quality.

Over time, various pundits have made pronouncements related to sustainable advantages, important assets, and the like. Arie de Geus wrote in Harvard Business Review that "the only competitive advantage the company of the future will have is its managers' ability to learn faster than their competitors." In his research paper, "Working with Tacit Knowledge," Joseph A. Horvath, Ph.D, of the IBM Institute for Knowledge Management, said that that knowledge is "the only sustainable competitive advantage." While these and other similar statements are brilliant distillations, they are also very abstract. What isn't abstract is the ability to use all that learning and knowledge to consistently deliver products that meet specs.

Consider some of the benefits that winning products bring to your company:

- They are the seeds from which customer loyalty grows;
- Customers will pay more for them;
- They are easier to sell and market;.
- They are invariably easier and less expensive to manufacture;
- They make better use of your fixed-cost assets;
- They have lower service and warranty costs.

While there are many functions that are important to a manufacturing enterprise, none can have as positive an effect on your bottom line as the ability to produce winning products.

How can you consistently create winning products?

There are questions that no one but you can answer, and this is one of them. The capability of your company to create winning products is tied to your particular product-development processes. Though these processes may resemble those of other companies, they are at their essence unique.

Product development is among the most complex of human endeavors, touching a variety of disciplines, spanning organizational boundaries, and involving the interplay of myriad business, technical, and social issues.

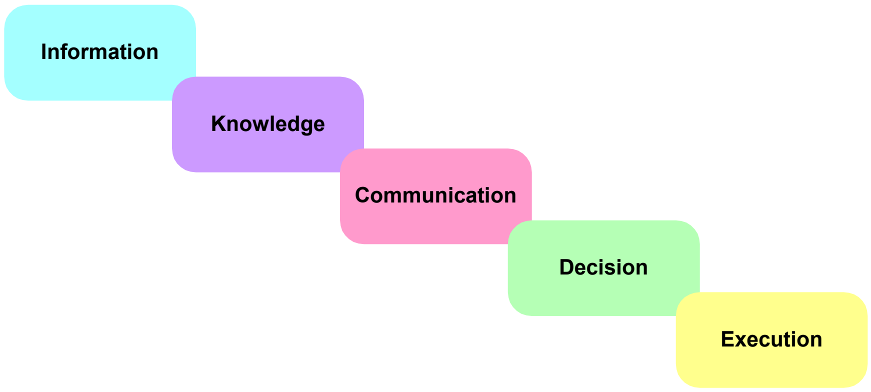
There are numerous methodologies that claim to optimize product development. Concurrent Engineering, ISO-9000, QFD, VA/VE, Six-Sigma—the list goes on and on. And beyond the sheer variety, each methodology seems to come from an entirely different mindset.

The various product-development methodologies are analogous to martial arts, each with its own black-belt practitioners. But, just as in martial arts, if you put them all in a cage and make them fight it out, none of them will come out on top. They'll all get beaten by a street fighter who borrows the best elements from each, adapts them to his needs and style, and practices them until he achieves mastery.

So, while we can't tell you exactly how to build a product-development process that will consistently create winning products, we can describe the building-blocks you'll need to design and build that process yourself.

The Five Building-Blocks of Product Development

Einstein said, "Make everything as simple as possible, but not simpler." The simplest breakdown of the product-development process is into five basic building-blocks:



These building-blocks may be put together in different ways by different organizations, and may be emphasized differently by various product-development methodologies, but we've not seen any examples where even one of the building-blocks could be weak or missing, and not effectively compromise the result.

Information

Information, as a practical matter, has no meaning of its own. It is just data placed into a context. But, when coupled with human interpretation, it becomes the basis for knowledge. And it provides the underpinnings for making decisions.

Information falls into two broad categories: *Formal* information is explicitly labeled, and can be found in specifications, catalogs, Web pages, databases, and books such as the Machinery Handbook. *Informal* information is either not permanently or explicitly captured, or is captured with no intention of generating formal information. Informal information might include that which comes from conversations, general knowledge, or even that which can be found in the "favorites" list of your Web browser.

Both formal and informal information can suffer from problems, particularly lack of context and lack of explicitly stated assumptions. Good information is characterized by four qualities[1]:

1. **Transparency** requires knowledge of the source, how it was obtained, and what was omitted.
2. **Reliability** may be explicit, implicit, or unstated. A press release might include a statement "this information is deemed to be reliable, but is not guaranteed." A dimension on a drawing might include a tolerance of +/- 0.010". Most often,

[1] Sam Vaknin, Ph.D., *Knowledge and Power*, http://samvak.tripod.com/nm061.html

information comes with no explicit statement of its reliability. It is up to the user to evaluate the information, and come to a conclusion as to its reliability. This evaluation will often factor in transparency, familiarity with the source, apparent balance, context, and how "true" the information rings.

3. **Comprehensiveness** implies that nothing relevant has been omitted. One particularly relevant part of information is context. Without it, the information may actually be meaningless.
4. **Organization** is a quality that makes information both comprehensible and useful.

Consider this example: At the beginning of a project, a development team might be given a set of requirements, including the following criteria[2]:

- Functional performance
- Human factors
- Reliability
- Lifecycle concerns
- Resource concerns
- Manufacturing requirements
- Vendor concerns

While some of the requirements might be based on "good" information, often enough they are based on information with no pedigree. One reason might be that the people involved in establishing the initial requirements for the project are basing their work on "gut feel," or on incomplete, uncertain, or evolving information. A more common reason is that there may be no process that accounts for the source of the information.

The quality of the information at the beginning of a project is important because it is the raw material with which decisions about the project are made. Though it may be a practical impossibility to have all good information in advance, it is a factor that can make a significant difference in the project cost and schedule.

The information used in the initial phases of a project comes from a variety of outside sources, including competitors, distributors, customers, suppliers, as well as from inside sources, including manufacturing, maintenance, and previous projects. The challenges in obtaining this information are substantial—first in identifying its source, then in capturing it, and finally in actually building it into the beginning of a project.

Common sources of formal information are ERP, SCM, and CRM systems—all of which are becoming more common, even in mid-sized manufacturing enterprises. A challenge with these systems is that the information they keep may not be tuned for the needs of the product-development process—and in some cases, getting the information to the right people at the right time may take some work.

[2] David G Ullman, Ph.D., *12 Steps to Robust Decisions,* Trafford Publishing, 2001

The one repository of information that is most often tuned to these needs is PDM (Product Data Management), which is now offered as a part of a larger Product Lifecycle Management solution by a number of software vendors. Because these solutions are mainstream applications for engineers (often used as a project-control panel), they are a natural choice for populating with project information—even that informal information that doesn't strictly meet the criteria for "good quality," but that is valuable nonetheless.

A second practical repository for project information is in the models created with engineering software. This can be in the form of component parts, assemblies, or even system models. Incorporating project requirements into a system model can be particularly valuable on complex projects, where the important driving parameters can be carried throughout all the part, assembly, and analysis models.

Knowledge

Knowledge can be defined in a variety of ways. One definition is that knowledge is the result of a person interpreting some information. The fact that it might be 105 degrees in the shade outside is just information. It becomes knowledge when a person says it's too hot to be walking to the corner store.

Possibly a better definition, at least in the context of product development, is that knowledge is "know-how," usually related to a process—whether it be something as simple as how to bake a cake, or as complicated as how to design a plastic-injection mold.

Knowledge "gurus" talk about explicit knowledge and tacit knowledge. Explicit knowledge is that which you know you know, and that is codified in some formal fashion. Tacit knowledge is the stuff that's inside someone's head that they don't particularly think about knowing.

The problem with tacit knowledge is that it's not something that can be easily replicated. When an employee leaves, they take their tacit knowledge with them. Further, companies may have a goldmine of knowledge and experience, yet not be able to tap it, because there is no map that shows where it is.

For the participants in the product-development process, the most important sources of knowledge are their peers, both inside and outside the organization. These are the people who most often know how to do something worth learning, and who can relate it in an understandable way. There is no substitute for being able to sit down with someone and have a discussion. This is the oldest and most effective form of knowledge-sharing.

While simply sharing knowledge may be helpful in providing support for decision-making, its effect on execution—actually doing the work—is not as dramatic as when knowledge can be captured and incorporated into knowledge-based software tools.

Engineers have come to expect a certain level of knowledge support in their software applications. In CAD, for example, parametric models and assemblies can include a significant amount of embedded knowledge. When driven by a spreadsheet or other data source, they are capable of modeling complete families of products.

CAD, CAM, and CAE applications often include built-in knowledge-based "wizards" which use process-centric workflows to automate complex tasks, such as modeling plastic injection molds, creating high-speed machining toolpaths, or representing real-world loads for analysis. More advanced CAD applications can work iteratively with CAE and other simulation applications to develop solutions to problems based on stress, thermal, fluid, or dynamic analysis.

Beyond traditional product-development applications, there are Knowledge-Based Engineering (KBE) tools. KBE tools, some of which have been commercially available for over 15 years, have been well-proven to save time and cost in the areas in which they have been applied. Published case-studies report verifiable savings that range from at least 50% to (literally) 97.5%. Yet, despite this, KBE has not caught-on in any mainstream sense.

The limited acceptance of KBE seems to be the result of some pretty fundamental issues, coupled with some complex human psychology. First, KBE tools have often been quite expensive, limiting their acceptance among smaller manufacturers. Second, the effort to develop an application using traditional KBE tools has been substantial—requiring trained specialists to write tens to hundreds of thousands of lines of code in a specialized language. Plus, KBE applications have to be kept up-to-date, or they will fall into disuse. Third, people tend to shy-away from systems they don't understand. They learn from talking to other engineers—not from using KBE applications.[3]

A practical solution to the past problems with KBE is to integrate knowledge capture and reuse capabilities into traditional product-development applications. A critical factor in the success of this strategy lies in making these capabilities easy enough to use so that users who are not "knowledge specialists" can build their own tools, and at the same time, providing enough capabilities so that knowledge specialists can build more comprehensive integrated systems.

Communication

Can't we just talk? Or would email be better?

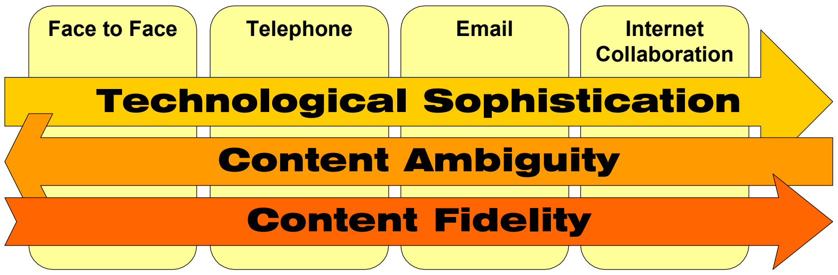
It depends on what you're trying to accomplish. If you're trying to transfer knowledge, talking is better. If you're trying to transfer information, email is better. As a rule, the technological content or sophistication of a communication method is directly

[3] Whitney, Dong, Judson, and Mascoli, "Introducing Knowledge-Based Engineering Into an Interconnected Product Development Process", M.I.T. Center for Innovation in Product Development, Cambridge, MA, White Paper Jan. 27, 1999.

proportional to its ability to support content fidelity, and inversely proportional to its ability to support content ambiguity.

In a face-to-face conversation, it's easy to convey ambiguous concepts or tacit knowledge. Consider something as simple as explaining how to tie a shoe: Easy in person, harder on the phone, and just not worth doing on email.

At the same time, more sophisticated communication methods are better for conveying information that requires fidelity. Want to know the value of pi to 1,000 places? Don't say it, email it.



Within a product-development process, all forms of communication have their place and must be balanced and integrated. On the fuzzy front-end of a project, face-to-face is still indispensable. On the back-end, for example, when building tooling or communicating with suppliers, Web collaboration can be a lifesaver.

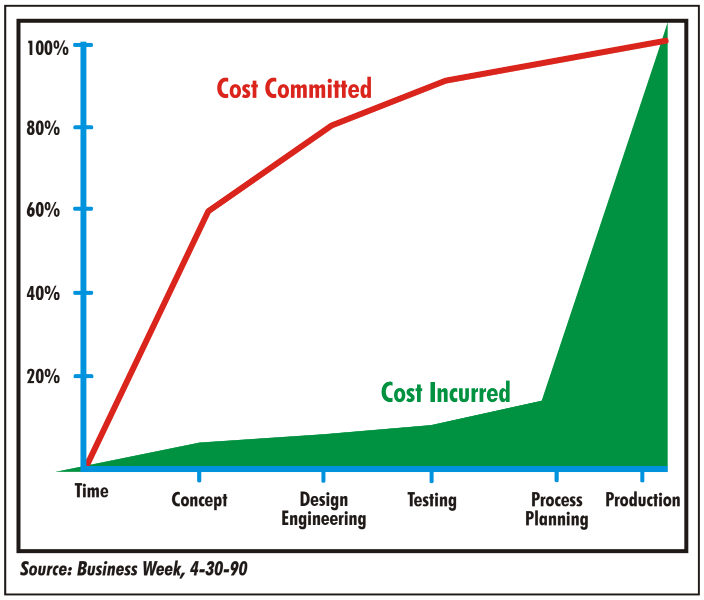
Despite technological advances, project communication still requires thoughtful management. Problems in communications are more likely to be related to human factors than to technological factors.

Decision

It may be impossible to estimate the number of decisions that are made throughout the course of a product-development process. Big decisions—such as what overall concept to use—are easy to identify. But for every big decision, there are innumerable small decisions—from what kind of bolts to use, to what kind of wrench to use to tighten them.

The problem with having so many decisions to make is determining the right levels at which to make them. Small decisions are almost always made by individuals who, given the necessary information, have the knowledge to make the decision. (Or, lacking sufficient information, have the experience of making a similar decisions in the past.) Big decisions more often are made by consensus by groups of people. Yet it's often hard to know what decisions are big, and what are small. Lockheed discovered this, when they found that using cadmium-plated wrenches to tighten the titanium bolts on the SR-71 Blackbird stealth fighter airplane would cause the bolts to fail at high temperatures.

Decisions are also time-dependent. A rule of thumb is that up to 80% of a project's cost is committed during the ten percent of the project, even though that cost may not be incurred until the late phases.[4] Though the number is only a rule of thumb, the effect of making well-considered decisions early in the process is clear.



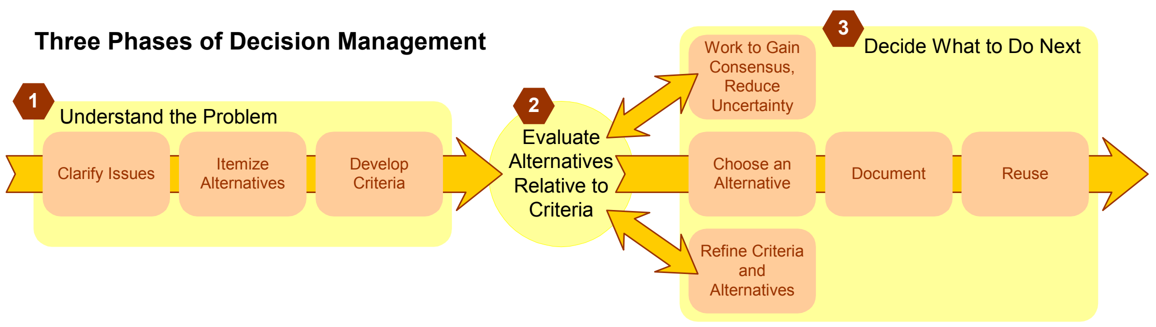
The problem with moving decisions to the beginning of a process is that doing so often means making them without information that might be available later. The old adage applies: "If I knew then what I know now, I'd have done it differently."

Signs of decision-related problems in a product-development process include:

1. Projects are late or over-budget
2. "Final" decisions are later revisited
3. Poor stakeholder buy-in
4. Decisions made by edict or the most forceful person
5. Expertise is underutilized
6. Low confidence in decisions
7. Decisions are not justified, recorded, reused

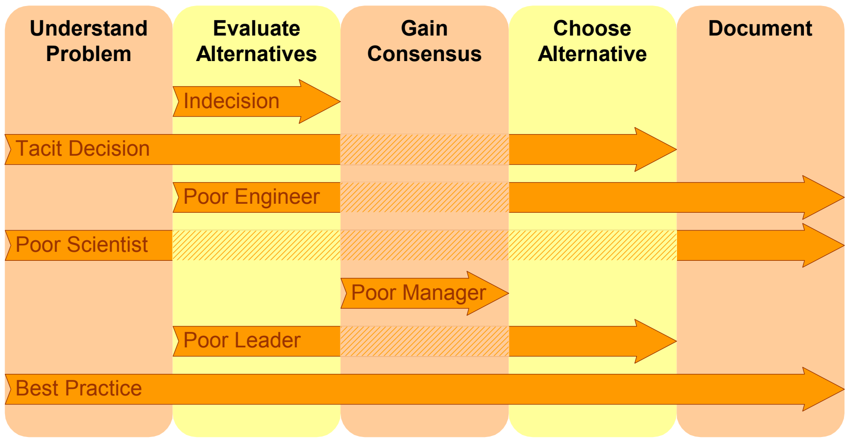
These signs all derive from an inadequate or non-existent decision-management process. Consider the following process, which consists of three major phases:[5]

[4] "A Smarter Way to Manufacture," Business Week, 4/30/90
[5] This process, and the list of signs of decision management failures, are based on the work of David G Ullman, Ph.D., author of *12 Steps to Robust Decisions,* Trafford Publishing, 2001


Three Phases of Decision Management
1
Understand the Problem
Clarify Issues
Itemize Alternatives
Develop Criteria
2
Evaluate Alternatives Relative to Criteria
3
Decide What to Do Next
Work to Gain Consensus, Reduce Uncertainty
Choose an Alternative
Document
Reuse
Refine Criteria and Alternatives

”Understanding the problem” requires the support of information, knowledge, and communication. The second and third phases of the decision-management process, depending on the decision complexity or importance, may involve anything from one experienced person making a choice, to a committee (or even a board of directors) fighting it out. This is the area where decisions often break down, typically for lack of the process being explicit, for failure to use best-in-class evaluation tools (simulation software being notable), or for failure to formalize the process of gaining consensus.

Individuals within an organization have their own decision styles, emphasizing different steps of the process, and sometimes ignoring steps. Whether a result of DNA, training, or experience, there seem to be some commonalities among groups of people. Though every group has many individuals who recognize (and try to use)” best practices,” there are always those who seem determined to provide fodder for the Dilbert comic strip.



Because of the tendency of people to skip steps, critical decisions should always be supported by an explicit process.

The failure to document and reuse decisions is as problematic as any other breakdown in a decision-management process. It forces decisions to be remade (instead of simply revisited) on subsequent projects—often when those with the relevant knowledge and experience are no longer available.

One way to promote the reuse of decisions is to codify the decision process in software, so that it can be used broadly throughout an enterprise. Since a documented decision process will include references to all the relevant information and knowledge supporting the decision, all the elements are in place to support this.

Execution

Execution is easy. Just do it.

Possibly it's not that easy. Executing any task, no matter how trivial, requires up-front information, knowledge, communication, and decisions. Even going to lunch with a friend involves these building-blocks.

Yet, execution is an individual building-block. A machine tool requires nothing more than a chunk of metal, a cutting tool, coolant, and electricity to execute its program and create a part. If a production line is carefully designed, it can run with no need for new information, knowledge, communications, or decisions. If you doubt it, ask Fanuc, Ltd., which has a robot-assembly plant in Japan that can run up to 30 days with the lights out, and the air-conditioning and heating turned off.

While it is possible (though exceedingly difficult) in manufacturing to completely separate execution from all other considerations, in product development it is completely impossible. Product development is an inherently iterative process, requiring a dynamic stacking of all the building-blocks of product development to actually create something useful. Execution is just one building-block out of five.

A problem that has faced manufacturers since the dawn of the industrial revolution is that execution, particularly in a product-development process, has always been a bigger building-block—or, rather, stumbling block—than it should have been.

Why? For the human race, execution is inseparably tied to the use of technology. Over two millennia ago, Archimedes, arguably the "father of engineering," relied upon the technologies of mathematics, language, writing, and drawing to execute the designs for his inventions. From Archimedes' time, until the invention of the digital computer, technology provided little to make those fundamental tools easier or faster to use.

Despite tremendous progress during the last 50 years in creating computer-based tools to support the needs of engineers and others involved in product design, it has not been until very recently—as improvements in processors, memory, storage and networking have driven computing power up and costs down—that they have come into widespread use. The maturation of these tools, pushed by competitive pressures, is even more recent, having truly started in the mid-1990s and being still in progress today.

Technologies that support execution in manufacturing, such as the printing press, the Jacquard loom, and the milling machine, have been mature for some time. They passed the threshold of becoming "good enough" to transform their industries long ago. And while innovation still continues, it is in a context where improvements are not likely to precipitate fundamental changes in how products are manufactured.

By contrast, the technologies that support product development, particularly the main tools of CAD and CAM, have only recently crossed the threshold to become good enough so that they are not a constraint to execution. Some products in this space have reached levels of functionality, reliability, and usability where they have changed in nature, becoming appropriate tools not just for extensively trained users, but for casual users as well. Other products in the space are improving rapidly with the lessons gained from those ahead of them.

The nature of tools (which are implementations of technologies) is that their acceptance is intimately tied to their usability. If people had to use large format cameras, requiring them to set shutter speeds and apertures using a zone system with a hand-held meter, and forcing them to use fixed focal length lenses, ground glass focusing screens, and single sheet film holders, they'd not take too many photos. Fortunately, modern cameras (whether film or digital) have progressed in usability to the point where they present few problems—even for those who refuse to read the manual, and would rather discover on their own how to use it.

It's even more important for product-development tools to be usable and discoverable than it is for other tools. There is no correlation between the competence of an engineer in their domain, and their ability to figure out overly complex software. And, while the cost of software training may be calculable, the cost of lost productivity from fighting inadequate software is incalculable.

Within the product-development software industry, we've seen strong progress in the last two or three years towards improving product usability, but our sense is that there is still much more work to go.

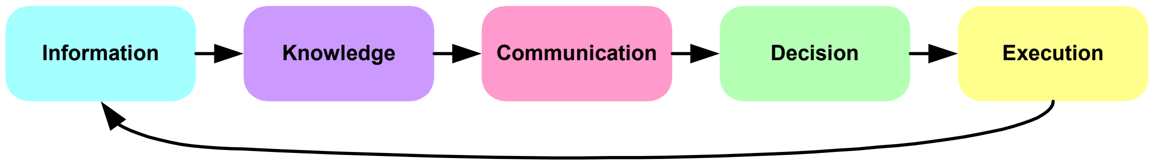
Another area where progress has been made, with more work to go, is in interoperability. Successful execution in product-development requires information and knowledge to be accessible by a wide variety of tools. It's unreasonable to expect that any software company will be able to supply all of the tools necessary to support a complete product-development process. Because of this, it's critical that programs support open data standards and open interfaces, so that information and knowledge can be captured and used wherever in the process it is appropriate.

Even as product-development execution technologies improve with time, there is one threshold they are still far from crossing: replacing engineers and designers with unskilled operators. This is because ultimately, only a person with relevant experience can evaluate or validate the output of a complex process.

Product Development, People, and Technology

Although execution in product development requires the use of technology, the other four building-blocks have no such requirement. Information, knowledge, communication and decision are very human concepts. For a problem of limited scope, it would be possible for a person, or a group of people, to come up with a workable solution without even using a paper and pencil. And while engineers like doing this every once in a while just to prove they can, it's just not practical to do projects of a realistic scope entirely in one's head. It's better to have some technology to make things easier.

And though the particular organization or structure of a product-development process may vary, it is always true that all of the five building-blocks of product development are dependent on the others. Rarely will you find a project that is not fundamentally iterative, with multiple dependencies among different blocks. And in almost every case, the process of execution will disclose something that doesn't work, providing more information, which must be fed back through the process to find something that does work.



Because of this iterative characteristic, any weak block will have a negative effect on the process—no matter whether that weakness is a result of people factors or technology factors. Yet, if technology can compensate for or strengthen an area that has been weak in the past, the positive effect can be dramatic.

With this, we come to our central premise: *a product development process that is capable of consistently creating winning products must be supported by technology which addresses and optimizes* ***all*** *of the building blocks of that process.*

In the last few years, we've seen a number of examples of companies that have implemented technology in a way that goes beyond simple execution. By using technology to improve their management of the information, knowledge, communication, and decisions associated with their product-development processes—even incrementally or partially—companies have seen paybacks that dwarf their investment in software and training.

Evaluating Product-Development Technology

Technology, particularly software technology supporting product-development, is exceedingly complex.   The five building-blocks provide a context for evaluating software in terms of how it supports your product-development process. The following criteria must also be met if the promise of the software is to be realized in practice:

- **Usability/Discoverability.** Software must be usable by those whose job it is to use it. It is a tool—not a career. Even if a program were massively more capable than its competitors, those capabilities would be chronically underutilized if the developer had not paid careful attention to usability. Discoverability is important;

as users' needs change over time, and the cost and time of formal training to address those needs is often not justifiable.

- **Capability/Function.** Any software you use should be capable of doing the job you will require of it. If you're designing airframes or auto bodies, your requirements will be different than if you're developing consumer products. In any of these cases, you will benefit from software that supports process-centric workflows for complex domain-specific tasks.

- **Interoperability/Openness.** If product development is a fluid process, then the disparate tools that support that process must have the ability to work together. This is manifested in support for open data and open interfaces.

- **Incremental improvement.** Software that forces users to make large changes in the way they work will often be unsuccessful. The software vendor's upgrade strategy should manifest as incremental improvements.

- **Continual innovation.** When you buy software, you're generally going to use it for a long time. Seek out suppliers who have a history of delivering quality and innovation.

- **Value.** Value has two components. The first is what is in the package. The second is what you pay for that package. Expensive software isn't expensive if it contributes to your ability to consistently create winning products.

# # #

About Cyon Research...

Cyon Research Corporation was formed by CAD industry consultants Brad Holtz, Joel Orr, and Evan Yares to foster clarity and provide vision to users and vendors of CAD and PLM tools. Current products include: CADwire.net, a leading provider of online news and analysis; COFES: The Congress on the Future of Engineering Software; Engineering Automation Report, A-E-C Automation Newsletter, Extranet News, and The CAD Rating Guide™. More information can be found at: www.cyonresearch.com, 301-365-9085.

Cyon Research™

Cyon Research Corporation
8220 Stone Trail Drive
Bethesda, MD 20817-4556 USA

phone: 301-365-9085
fax: 301-365-4586
Web: www.cyonresearch.com