

## The Weak Link in the Supply Chain

**John D. McGregor**, Clemson University and Luminary Software, U.S.A.

### Abstract

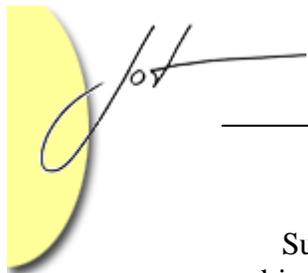
Component-based software techniques have promoted a market of components that can be purchased and integrated to produce a product. Companies that produce software-intensive products using purchased software are experiencing the uncertainty and frustration that comes from including software development companies in their supply chain. In this issue of Strategic Software Engineering I will discuss how companies are addressing these issues and what more can be done.

## 1 INTRODUCTION

As the newly appointed vice president for manufacturing for a major automotive manufacturer, you are surprised to learn that over one-third of the cost of your luxury models comes from software and electronics and that the next generation of luxury cars for which you are planning now will likely host 1 gigabyte of binary code [Salzmann 04]. You are dismayed to learn that half of all software development projects are delivered late, over budget, or with reduced functionality [Standish 94]. After all, automotive suppliers must achieve a 98 – 99% on-time, to-standard delivery record [Gould 04] to be considered good. You now see your goal of reducing order-to-delivery time of a luxury car as nearly impossible. A mechanical engineer by education and experience you feel unprepared to meet these challenges.

The reliability of a company's supply chain is a fundamental ingredient in accomplishing the company's strategic objectives. As the software content of most hard goods products increases, the potential for impact on the company's supply chain by software development also increases. An increasing number of companies count on software development efforts to be on-time, within budget, and to standard as an integral part of a larger product development effort.

Lets be clear about what I mean by "supply chain." When you produce any product, the resources needed to create that product are obtained in a variety of ways. Automotive manufacturers rely on producers of tires, seats, axles, and much more to supply some of the basic parts of each car and some they build themselves. The auto manufacturer's ability to create a car depends on all of those suppliers doing their jobs. In turn, the suppliers' ability also depends upon their suppliers.



Supply chain management (SCM) extends from a company's suppliers, both direct and indirect, to its customers. Supply chain management seeks to reduce time to market, reduce costs, and maximize sales by managing the flow of "stuff" and information about the stuff. This includes alerting customers to changing delivery dates. Managing a customer's expectations is much easier a few months before a scheduled delivery that won't happen than it is a week before. For an industry that invented the term vaporware this is tough to do but essential to maintaining business-to-business relationships.

The supply chain extends back to the company's suppliers, and to their suppliers. A failure anywhere along this chain threatens timely deliveries forward in the chain. Supply chain management seeks to reduce the probability that problems will occur and the cost that will be incurred if they do occur. SCM requires collaboration among the sets of suppliers.

The software development community has made progress toward on time, on standard deliveries, but we still have much to do. That's what I would like to discuss in this issue, but I will stay away from the purely business issues and focus on technical activities related to strong supply chains. First I will describe our manufacturing process and how it interacts in a supply chain. Then I will discuss what we and our customers are doing from a supply chain perspective.

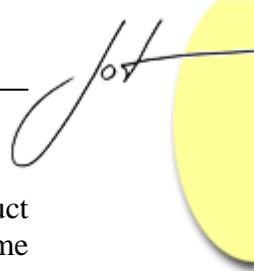
## 2 OUR MANUFACTURING PROCESS

Software is different. That is what we like to say and hope that it explains why we can't produce quality products in a timely manner. It doesn't. The cell phone manufacturer who misses their market because the software browser was delivered late by a vendor isn't interested in excuses, just damages.

Software IS different, but that just means we have different problems and need solutions that are different from those in hard goods manufacturing. The manufacture of the physical embodiment of our products, our deployment media, is seldom a problem. Our problems come in what would be the equivalent of the design process of hard goods. Our manufacturing process is essentially the software development process.

One problem is that software is too soft. There are no immutable laws of physics that constrain our design. Designers are faced with a mind boggling array of choices. Projects often are late because of all the design possibilities, the difficulty in selecting among them, and the irresistible urge to continue to tinker.

On the other hand, as a software product grows in size, the software becomes hard quickly. Technologies such as component-based development are intended to prevent pre-mature hardening of the software product. Components provide pieces of software that are, hopefully, just the right firmness. They are the right size to have a complete specification and have the right structure to be able to adapt, via different implementations, to changing non-functional requirements. Their size is a tradeoff between firmness and economic feasibility.



Our manufacturing process has a supply chain. In fact some software product companies do nothing but buy component parts and integrate them. Others buy some portion of their product and build the rest. We all buy tools. Poor quality or late delivery of some of these products is easily handled but others can be devastating. The failure of a vendor to release the completed new version of a library is manageable unless the new version fixes a serious bug or provides a new feature that we relied on in designing our next product. Failure in a supply chain is magnified as the delivery date slips in one company after another up the chain.

Our supply chain is not limited to other software companies. Admittedly, the failure of a hardware vendor to deliver a standard workstation can be quickly remedied by going to a different vendor but the failure of a vendor to delivery a new high performance server to the testing team can mean unavoidable testing and deployment delays. Services such as consulting and training must also be delivered on time to have the appropriate impact on development effort.

The current movement toward outsourcing can have significant impact on our supply chain. Automotive manufacturers usually require suppliers to have facilities close at hand so that deliveries are quick and reliable. It would seem to be different in our information world where software can arrive instantly from anywhere in the world; however, can we rely on a supply chain that goes around the world and relies on bi-directional flows of understanding among humans? It depends on our tools and processes.

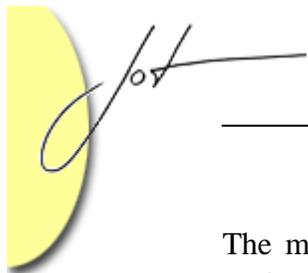
Software is in the supply chain of a large percentage of products in the marketplace, including our own. Let's consider some of the same techniques that hard good manufacturers use to ensure supply chain integrity, and then we will see how these apply to us.

### 3 WHAT ARE THE AUTOMOTIVE SUPPLIERS DOING?

As I have already said, automotive suppliers achieve high levels of delivery quality so let's consider four of their practices [Vasilash 04].

- *Automate SCM* – Manufacturers automate as much of the supply chain management process as possible.
- *Use best practices* – Manufacturers focus on improving their manufacturing processes by incorporating best practices.
- *Integrate suppliers and customers* – Manufacturers strive for collaboration up and down the supply chain to improve the acceptability of products being pushed up the supply chain.
- *Establish visibility to mitigate risk* – Manufacturers seek to mitigate risk by gaining more visibility into suppliers' ever changing capabilities and customers' ever changing needs.

They are not perfect. They still use a push model, where products are pushed through the supply chain to customers, particularly between auto manufacturers and auto purchasers.



The manufacturers are looking toward being able to implement a pull model where a customer can custom order and receive their product in a reasonable amount of time, which is seldom the case now.

Our supply chain is similar to those in any industry. Our problems are similar and our actions are similar so I will not address the activity of automating SCM in this column. I will describe how we are, or should be, doing the other three practices.

## 4 WHAT ARE WE DOING?

Software development has changed, for most of us, since vaporware stole our credibility. A number of factors are coming together to make our manufacturing processes more predictable and our role in a supply change more reliable.

### Process Definition and Improvement

Can you imagine each chemist at Michelin making tires differently? Or worse, every chemist making every batch of tires differently every time? Much attention has been focused on repeatable and manageable software development processes and much effort has been expended to mature our profession in this area. The Capability Maturity Model (CMM<sup>®</sup>) and Capability Maturity Model-Integrated (CMMI<sup>®</sup>) products of the Software Engineering Institute (SEI) are giving companies a step-by-step guide to improving their process definitions [SEI 04a]. The Software Engineering Body of Knowledge (SWEBOK) gives companies guidance on the content of those areas described in CMM documents[SWEBOK 04].

Organizations that use techniques that improve quality, such as inspections, actually are the most productive as well. This is because they waste less time searching for defects than those companies that don't address quality until late in the process. Efforts such as the Personal Software Process (PSP<sup>®</sup>), and Team Software Process (TSP<sup>®</sup>) initiatives at the SEI are providing avenues for software manufacturers to improve their quality and productivity[SEI 04b]. Microsoft and the SEI recently announced the results of a study in which developers reduced their defect rate from 25 defects/KLOC to 7defects/KLOC [Kimberland 04]. They experienced a 35% cost savings on the project due to a reduced need to fix defects.

These results and others show the falacy of the old approach: "There is never enough time to do it right but always enough time to do it over."

### Reuse

Many reuse schemes have been tried and most have failed to deliver strategic value. Why do we keep trying? Software has more predictable quality and can be manufactured in a

---

<sup>®</sup> CMM, CMMI, PSP, TSP are registered marks of the SEI.



more predictable time box by using pre-existing parts if those parts are appropriately designed and implemented.

What has been wrong with many of the reuse approaches? Vague requirements. Assignments such as “Build this component to be as flexible as possible” are not sufficiently constrained to allow for intelligent engineering. The engineer decides on one dimension of flexibility when the next project may really need a different type of flexibility.

Design for reuse *in context* is necessary. This can be accomplished in several ways. One way is standardization. Standard architectures, industry consortium standards, and government standards provide the opportunity for multiple vendors to form a competitive market place. An example are the web browsers that run on wireless platforms using the wireless access protocol (WAP), which can now be purchased from multiple vendors.

My favorite technique for achieving reuse in context, as should be obvious from previous columns, is a software product line where elements are used with a minimum of modification [Clements 01] on several products. The assets in a product line are only intended to be used within the constraints of the products defined in the product line. The financial viability of building the components to be reusable across the fixed number of products in the product line is evaluated as part of developing the business case. The viability is established before resources are expended on developing the components.

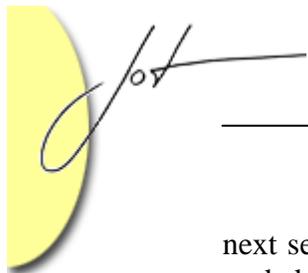
A third way to achieve high quality reuse in context is through a comprehensive library or framework built around an architecture. The Eclipse project is not a product line but it is intended to be the basis for a large number of products of a specific kind. It is a set of projects developing comprehensive libraries of classes that work around a specific architecture for a reasonably specific purpose. In this case, elements are more often extended before being used than used as is, but those extensions are anticipated and compositional.

In each case, there is a learning curve. Using Eclipse as the basis for only one product is not a cost effective approach. The amount of effort to understand the nuances of the dependencies between classes in the library is considerable and must be amortized over several products.

## **Risk Management**

Risk assessment and mitigation are project management best practices. Many management processes now include risk-based techniques for using risk to adjust for the sensitivity of their estimates to uncertainty. The two attributes of any risk are the probability that the event will occur and the cost if it does occur. Risk assessment depends on collecting data from which the probabilities can be computed. Mitigation works to either reduce the probability of occurrence or the cost of occurrence.

The probability of an anticipated risk event occurring can be reduced by the use of software engineering “best practices.” For example, an unambiguous, complete requirements model reduces the likelihood of missing functionality in the product. In the



next section I will discuss “due diligence” in which the customer seeks to determine the probabilities of risk events, in which they have an interest, occurring in our work.

The cost to the customer of a risk event occurring can be reduced by close collaboration between supplier and customer. Many successful supplier relationships operate on two levels to maintain visibility into each other’s processes. The business relationship is maintained by management to ensure compliance with the contract and any applicable laws and regulations. A technical relationship is also established in which a technical contact makes periodic reviews of progress at the supplying company. The technical connection will identify risks to timely, quality deliveries while the financial connection may identify budget busting situations if the contract allows for increased costs to be passed on to the customer.

### Quality Assurance

A number of activities are grouped under the heading of quality assurance, sometimes but not always including testing. Over time the focus of QA efforts has moved from system testing after the development is completed to points much earlier in the development process. Activities such as training in new techniques, active inspections, and the adoption of tools that provide checking of their output are all quality assurance activities that reduce faults and increase our on-time delivery.

Quality assurance has two impacts on the supply chain. First, the presence of a well-defined QA program can give customers visibility into the company’s efforts to produce to-standard products. Reviews of the results of inspections and audits of training records can substantiate claims of quality development. Second, the QA program can improve the on-time delivery for the company by reducing the faults that cause product defects. Time is saved since there are fewer defects to chase down. With a comprehensive QA program, problems are known earlier and are more easily handled at the root cause.

## 5 WHAT ARE OUR CUSTOMERS DOING?

Our customers evaluate our role in their supply chain based on three attributes [SCC 03]:

- *Reliability* – How often do we achieve full delivery on-time, to-standard?
- *Responsiveness* – How quickly can we deliver a product once it is commissioned?
- *Flexibility* – How well do we respond to changes in the marketplace or domain?

They perform due diligence in selecting their vendors and practice risk management concerning the vendor’s role in their supply chain.

### Due Diligence

Companies are becoming aware of the need to investigate suppliers before including them in the supply chain. Few of us would hire a contractor to remodel our kitchen without asking for references and checking on the contractor’s prior job performance. But



I have consulted on at least one legal action where the CEOs of two companies were golfing buddies and a supplier relationship was entered into with no due diligence.

Even though the quantitative management of projects is not required until Level 4 in the CMM, companies that expect to supply software-intensive products to other companies should not wait until they are moving to level 4 to systematically collect data. Industry data indicates that projects are slowly becoming more reliable; however, as stated above, half of all software development projects are delivered late, over budget, or with reduced functionality. If you are significantly better than that average you need to be able to prove it.

From a supply change management perspective, due diligence seeks to determine the quality of a company's on-time, to-standard record. But we would like to go beyond these basics. What is the average time to respond to a requirements change? What is the turn over rate in domain experienced personnel? Do they have a plan for maintaining currency of knowledge in their personnel?

The information collected during due diligence is input to the contract negotiation process. We may not reject a vendor because of what is found during due diligence, but we may adjust what we are willing to pay. Assume more risk, pay less.

### **Risk Management**

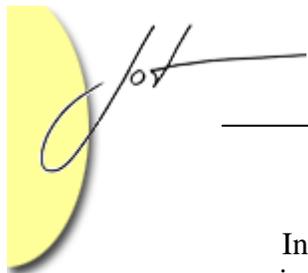
Our customers are as concerned about minimizing risk as we are, maybe more. They evaluate our reliability for future use. Our failures to deliver will come back to haunt us in future marketing when customers press for reduced prices to compensate for increased risk.

Our customers, the savvy ones anyway, are requiring an open process where they can monitor and evaluate the quality and timeliness of our work. This often means that valuable engineer time will be required to participate in joint requirements, architecture, and design reviews. The good news is that if we are proactive we can use these reviews to our advantage. Select the areas in which your staff has questions and suggest to the customer that those areas be reviewed. Use the review time to get your questions answered. Drive discussions deep into areas where you feel you have good ideas.

## **6 WHAT ELSE SHOULD WE BE DOING?**

### **Benchmarking**

How are we doing compared to other vendors in our market in terms of on time, on standard delivery? I had an operation a few years ago that was reasonably serious. In talking with a surgeon that I might have perform the surgery, the surgeon listed the possible complications, the average number of times each complications occurred for the general population of surgeons, and then his average for each complication. This was very useful information to have in choosing my "vendor."



In our business the data can be difficult to collect cleanly. We allow changes in the requirements for a product later into the development process than almost any other industry. How should these changes reasonably be treated in terms of what delay does the change impose? Many companies provide “pre-releases” to customers. When do we really deliver and when is it really complete?

The expanding market for components provides a simplifying assumption. When we make a purchase we get almost immediate delivery. This makes “when” easier to identify.

### **Data-based decision making**

Quantifying attributes of product development within a market, collecting data upon which to base decisions, is the single most important missing ingredient in our discipline. But equally important is the cultural view that such numbers must be suspect. Actual data shows that implementing techniques such as inspections during development has the highest return on investment of any process improvement investment [Jones 94] yet not every organization does inspections. Too often managers make decisions based on their personal biases and their belief that the latest silver bullet will save them. Use pilot projects, which may be real projects that are trying one specific new technique, to collect data and establish the viability, or lack thereof, for the technique.

Process improvement, actually process modification, should be based on experience data rather than random decisions. I once wrote a column for the Journal of Object-Oriented Programming titled “Let’s Don’t and Say We Did. [McGregor 98]” The premise was that a manager sees a new buzz word, say pair programming, and wants to be seen as a cutting edge individual. So the manager says, “yes we are doing pair programming but it is too wasteful to have two developers at one computer. We only need one!” When the benefits of pair programming are not realized, the manager says “well these new ideas never seem to work.” Using small pilot projects to explore a new technique reduces the risk of the new technique but may still provide sufficient data for a manager to see the validity of the new technique.

### **Comprehensive production planning**

Few companies create a comprehensive production plan for their software products. Planning one project at a time does not achieve any economies of scale or scope. By default, personnel develop expertise in the tools of their trade. Choosing a new language for every product or changing tools often imposes a learning overhead that is hard to compensate for.

Software product line development includes a production planning practice that ties the goals of the organization to the techniques used to build products [Chastek 02]. This goes to my basic view of engineering, “Building to a purpose.” Production planning relates each low level construction step back to the more general goals of the organization.



## New development technologies

The improved support in programming languages and modeling languages for interfaces has improved the reusability of software. Component-based software design also holds much promise but there is still no widely accepted component-based language. Having a CLASS named COMPONENT is not the same as having COMPONENT as a first-class element in the language. Koala holds some promise but it is not widely used [van Ommering 00].

## Better requirements

We should be giving our suppliers better requirements. Over half of defects are the result of misunderstood, vague, or incorrect requirements. Two simple things to do:

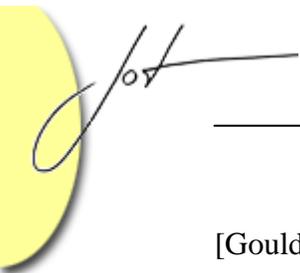
- Test the requirements. We often work with groups of domain experts to develop requirements. Our standard practice is to hold one or two of the experts out of the development group. The reserved experts conduct a “Guided Inspection” of the requirements developed by the larger group [McGregor 99].
- Use a sufficiently rich representation of the requirements. Lists in an Excel spreadsheet are a sign of too little detail and not enough information being communicated. Use cases, with some supporting text, provide the ability to write detailed requirements [Armour 00] and to show dependencies among the requirements. This is a must in an outsourcing situation.

## 7 SUMMARY

Many of us work in situations where our product is sold to be a part of a bigger product and often we are purchasing parts from others for use in our products. I have talked about several practices that are contributing to a more reliable industry and also pointed out some additional practices that need to be used more than they are now. I hope that this discussion will stimulate thinking about your role in the bigger picture of strategic supplier-consumer relationships.

## REFERENCES

- [Armour 00] Frank Armour and Granville Miller. *Advanced Use Case Modeling: Software Systems*, Addison-Wesley, 2000.
- [Chastek 02] Gary Chastek and John D. McGregor. “Guidelines for Developing a Product Line Production Plan”, CMU/SEI-2002-TR-006.
- [Clements01] Paul Clements and Linda Northrop: *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2001.

- 
- [Gould 04] Larry Gould. *Automotive Supply Chain Management: As Good As It Gets?* <http://www.autofieldguide.com/articles/020306.html>.
- [Jones 94] Capers Jones. *Assessment and Control of Software Risks*, Englewood Cliffs, NJ: Youdon Press, 1994.
- [Kimberland 04] Kelly Kimberland. "Microsoft's Pilot of TSP Yields Dramatic Results," *news@sei*, 2004(2).
- [McGregor 99] John D. McGregor. "Validating Domain Models", *Journal of Object-oriented Programming*, July/August 1999.
- [McGregor 98] John D. McGregor. "Let's Don't and Say We Did", *Journal of Object-oriented Programming*, September 1998.
- [Salzmann 04] Christian Salzmann, Thomas Stauner, and Alexander Pretschner. ICSE Workshop: Software Engineering for Automotive Systems.
- [SEI 04a] Software Engineering Institute. The Team Software Process (TSP) and the Personal Software Process (PSP), <http://www.sei.cmu.edu/tsp/>.
- [SEI 04b] Software Engineering Institute. Capability Maturity Model Integration, <http://www.sei.cmu.edu/cmmi/>.
- [Standish 94] The Standish Group. *Charting the Seas of Information Technology*, Dennis, MA: Standish Group, 1994.
- [SCC 03] Supply Chain Council. *Supply Chain Operations Reference Model (SCOR)*, 2003.
- [SWEBOK 04] Software Engineering Coordinating Committee. *Guide to the Software Engineering Body of Knowledge*, <http://www.swebok.org>.
- [van Ommering 00] Rob van Ommering, Frank van der Linden, Jeff Kramer, and Jeff Magee. "The Koala Component Model for Consumer Electronics Software", *Computer*, IEEE Computer Society, March 2000.
- [Vasilash 04] Gary S. Vasilash. *Leveraging the Supply Chain for Competitive Advantage*, <http://www.autofieldguide.com/articles/040206.html>.

## About the author

**Dr. John D. McGregor** is an associate professor of computer science at Clemson University and a partner in Luminary Software, a software engineering consulting firm. His research interests are software product lines and component-base software engineering. His latest book is *A Practical Guide to Testing Object-Oriented Software* (Addison-Wesley 2001). Contact him at [johnmc@lumsoft.com](mailto:johnmc@lumsoft.com).