2003

Pursuing lean objective-oriented software development

Jack Cook

Pursuing Lean Object-Oriented Software Development

Software quality can be characterized as meeting customer requirements at the agreed cost within the established timeframe. Currently, the sad state of software quality indicates new software development approaches might be warranted. One of the most dramatic examples of software project failures is the London Stock Exchange (called Taurus) developed eleven years late and 13,200% over budget. According to the Standish Group, known for its high quality, independent primary research, analysis, and advice, only 16% of software development projects are successful whereas 31% are cancelled due to defects. Average cost for "failed" projects is 189% of the original estimate, almost double what was budgeted. Average time for software development projects is even more than double the original estimate. Finally, an average project delivers only 61% of the specified functions and requirements (Corr, 2002). If such statistics were applied to manufacturing, process throughput times and the quality of final outputs would be unacceptable. Due to the difficulty of monitoring progress and defining the problem as well as the solutions, the degree of precision required, and the rapid pace of technological advances, many software developers accept that projects will be delivered late and over budget.

The high demand for software, shortage of skilled people, limited financial resources, and the need for quality software requires the software development (SD) process to be re-examined. Businesses cannot afford failure. One alternative is to examine how lean manufacturing techniques can be applied to SD. The transformation of object-oriented software development from traditional to lean using the techniques and examples in lean manufacturing is not only innovative but also challenging. Lean manufacturing and object-oriented software development have not often been thought of as going together.

After researching both areas, lean manufacturing and software development clearly have parallels that facilitate the application of lean techniques to object-oriented software development. In order to produce high quality software, certain requirements need to be met before lean can be applied to SD. After these requirements have been satisfied, certain steps need to be taken to help ensure success. Foremost, lean

manufacturing is not a quick fix to improve software quality. The journey is long and difficult and may not be suited for every company.

This paper explains the major lean manufacturing techniques and their successful application to the software development environment. The introductory sections discuss basic lean techniques, their benefits, and shortcomings. Subsequently, the need for lean manufacturing techniques in the current software development environment is established. Next, the preliminary steps, barriers, and pitfalls of implementing lean procedures in application development are illustrated. Finally, recommendations are provided for implementation of lean techniques in a software development company.

CURRENT SOFTWARE DEVELOPMENT ENVIRONMENT

Why Object-Oriented Software Development

Object-oriented programming (OOP) allows software developers to produce objects that collaborate together to produce software that better models their 'problem domains' than parallel systems created with traditional methods. Think of objects as standardized parts in a manufacturing environment. Once designed, their properties are understood and they may be incorporated into many products. Object-oriented (O-O) systems are "easier to adapt to changing requirements, easier to maintain, more robust, and promote greater design and code reuse" (Bahrami, 1999, p. 5). In an O-O environment, software is a "collection of discrete objects that encapsulate their data and the functionality to model real-world objects" (Bahrami, 1999, p 11). OOP is a development approach many companies use today since it allows programmers to work with higher levels of abstraction, making objects more reusable. An O-O approach integrates and intertwines stages of software development more than traditional software development. For example, analysis does not have to be completed to start the design phase. Also, the design can still be updated and adjusted to user feedback while the programmers begin implementation. Finally, an O-O approach promotes reusability of prewritten code. However, in practice, reusability is more of a dream than a reality. Therefore, this paper will provide some reasons why applying lean principles to object-oriented software development is not only cost-effective but also functional and effective.

Absence of Industry-Wide Coding Regulations and Standards

Currently, the SD industry is very competitive, proprietary, and uncooperative. The absence of industry-wide standards leads to duplication of unnecessary code and objects. A precompiled library of general objects should be generated and sold or rented to SD companies. This would allow businesses to shorten software development time by writing only the innovative, application-specific objects and code.

The absence of company-wide practices and standards for the design and coding of "quality" reusable objects raises numerous issues. The lack of standards stems from two main sources. One source is the absence of an industry standard specifying how programmers should code or the techniques on how to produce "good" code. Thus, taking an object created in one company and trying to implement it in another can be very difficult and costly especially in regards to the compatibility and the integration processes.

The second source is the lack of communication between the systems analyst and the programmer on what the customer requirements are and how to translate them into code. "One problem area is an uncomfortable 'join' between the work of business analysts, responsible for understanding and specifying the business function required, and programmers, responsible for designing and building the implementation on a particular hardware or software platform" (Bevington 2000, p. 315). Unclear requirements may lead to functionality and implementation errors. Some of the causes of such errors may be the misinterpretations between the analysts and the programmers as well as the ineffective or insufficient exchange of information between the two.

Unnecessary, Single-Use or Unused Code

Currently, there is a large abundance of inefficient and redundant code. Programmers, pressured by deadlines and individual goals (competition, creativity, and showing off personal skills and knowledge), tend not to produce reusable code. The absence of suitable incentives to recycle existing objects and use them in other applications is one of the most common roadblocks to promoting reusability. Since reusability is one of the major ways to cut costs and shorten product development time, it is crucial that management establishes such incentives throughout the entire organization.

The quickened pace of software development also discourages reusability. Recovering and reusing code takes time and effort. If suitable code cannot be found, valuable time has been wasted.

Finally, the programmer may lack the resources from which to extract existing code and objects. The absence of standardized libraries and databases creates roadblocks in software development. Therefore, industry-wide standard library of objects should be compiled and thoroughly tested to increase the efficiency of current SD processes.

Information Encapsulation

OOP incorporates the concept of information encapsulation. Information encapsulation is the idea that the "nuts and bolts" of an object are hidden from the user. The internal functionality of an object is represented through a graphical user interface. It may be a button, a checkbox, or a menu. All the user can see is that visible piece, but behind that is all the information pertaining to the implementation of a particular object. The information ("nuts and bolts") is the definition of the object and its functionality. The user utilizes the objects as a mean to an end without the necessity to understand its implementations

Information encapsulation is an important step from structured programming. It provides somewhat of a tamperproof system that a user can feel comfortable using. Tamperproof means that the system is "designed to prevent improper or harmful alteration or to provide evidence of such alteration" (Stackpole 2002, p.11). The coded functioning objects are not available to the user to edit, therefore, making less room for alterations.

LEAN MANUFACTURING TECHNIQUES

This section describes the basic principles of lean manufacturing (refer to Figure 1) while summarizing the benefits and shortcomings of some of the most essential techniques. "Lean manufacturing is a systems approach. Each phase builds on the previous one, anchoring the system as a whole. All processes that deliver value and all processes that support the delivery of value must be integrated. Skip any phase or element, and the system will be ineffective. Leadership, technical components, and value-adding activity must be balanced, blended, and synchronized. Companies that plunge into lean manufacturing can find themselves

overwhelmed by obstacles. "Lean" is not a panacea for everything that's wrong with your manufacturing operation; it's a long-term process" (Allen 2000, p. 54).

Insert Figure 1 Here

Since lean is a philosophy that all organizational levels must strongly support, adopt and believe in and not an instant problem-solving program, it requires a change not just in the company's vision and scope, but also in the management style. Since management is often the most resistant to change, a certain approach to overcoming this resistance is necessary. This approach might involve a demonstration of statistical benefits, potential savings and quality improvements after the institution of the new techniques in the organization. The employees, on the other hand, are frequently more concerned with the ease of implementation and operation of the new processes and equipment, than with the overall company savings. Personal benefits and the ability to preserve individual value to the company even with the new procedures and methods are also of great significance. Their adaptation process can be improved if the employees are included in the transformation and decision making processes, resulting in "ownership". This will increase their sense of vitality, necessity and importance in the organization.

Basic Principles

Since flexible and multifunctional employees can help the company adjust and prosper in the always-changing environment, a certain amount of standardized training and lean education is necessary for both workers and management. This includes cross-functional teamwork, lectures and workshops on lean principles, and education on all aspects of manufacturing. While training employees, management needs to include them in the decision making process to achieve greater success. Employee involvement is not just allowing them to make one minor decision; it is the process of empowerment, enablement, and encouragement for continuous feedback, cooperation, and suggestions.

The major principles of lean manufacturing also include the elimination of waste and production of zero defects. If an activity in the process is not adding value to the product, it is a waste. Muda (waste) can be found everywhere. It is in the unnecessary complexity of a product, the labor used in production, the space

taken up by the materials and the output, the defects, the raw materials, the inventory on hand caused by overproduction, the travel distance of a product from materials to the final output, the time, and the skills. Standardization – the identical sequence, material, tools, training, orientation, and steps to complete the process – is the goal of lean manufacturers because it will reduce process variance and lead at first to fewer, then hopefully zero, defects. "Materials should flow through a Lean plant without interruption attaining impressive throughput rates at exceptional quality levels" (Piszczalski 2000, p. 26).

Another crucial factor is supply chain networks that encourage cooperation, benchmarking, and continuous improvement. Such networks encourage *just in time* delivery and manufacturing, and smaller lot sizes. However, "lean manufacturing is not just about supply chain or logistics. It also involves a paradigm shift in corporate level decision making that touches on customer relationship management, accounting, information sharing, and profit and loss accountability" (Sharood, 2001, pg. 160).

Finally, in order to effectively determine the efficiency of the lean techniques in the manufacturing process, an organization must implement statistical quality and process control. A substantial number of variation measurements should be recorded at different points of production and any irregularities should be dealt with immediately. Moreover, the employees should be educated about both their and the company's progress in meeting the set requirements and the problems arising along the way.

Benefits

When comparing the benefits and the shortcomings of lean manufacturing techniques in the production process, it is critical that the benefits outweigh the shortcomings, making it profitable to implement lean in an organization. However, a company must understand both the pros and cons for a successful process implementation. Lean manufacturing practices encourage the company to become more agile and responsive to the environment, leading to profitability in a world where customer requirements are ever changing. They ultimately help reduce production cost by encouraging waste decrease and defect elimination, standardization and reduction of process variations, pull production control with a lot size of one that lowers overproduction, and lastly, continuous improvement to exceed customer expectations.

The implementation of lean techniques in the production process contributes to the "process time cut up to 80 percent, staff empowerment and motivation, and vastly improved quality of output data" (Wight 2000, p. 39). In summary, lean practices are a set of valuable strategies and tools to advance current manufacturing processes and achieve maximum productivity with increased customer satisfaction.

Shortcomings

Implementing lean techniques requires a clear understanding of the involved factors and potential drawbacks. The transformation process requires a certain level of commitment and training. Employees need to be educated about the major lean theories and principles as well as trained to perform ad hoc problem solving, identify process waste and defects, and collaborate in the multifunctional teams.

Most importantly, if the implementation was not instituted efficiently and properly, a great sum of money would have already been spent, making the recovery very expensive and complicated. Jim Lewis points out in his article "Get Ready…Get Set…Get Lean!", don't leap into a lean manufacturing approach with both feet. The transition requires a methodological, step-by-step approach to ensure success" (p. 43). Lean implementation requires a certain amount of time and money to be spent upfront on education, process transformation, new factory layouts and the purchase of new equipment, if necessary. Therefore, management needs to be aware that the transformation is not going to be immediate and cost beneficial right away.

DRAWING THE PARALLELS

Manufacturing parallels software development in many ways that have not been capitalized on so far. This section attempts to uncover the most evident resemblances between the two and show the appropriateness of lean manufacturing techniques to software development.

Process Inputs, Throughputs, and Outputs

Manufacturing, on the most basic level, is a combination of inputs, processing (throughputs), and outputs. Raw materials, engineering specifications and designs, and, finally, customer requirements are just some of the inputs coming into the process. Next, the raw materials are transformed based on the specifications and

designs into real products so that customer requirements are achieved. As a result, the outputs of the system are the final products that meet or exceed customer expectations and satisfy engineering specifications.

In object oriented software development, the process elements (inputs, throughputs, and outputs) are still the same even though they each undertake a different description. For example, the inputs become the system analysts' specifications, customer requirements, and programmers' coded objects. Customers are also often present during the processing phase, where the throughputs become customer reactions to the application prototypes. The objects and their interactions are the final software package. During the entire development stage, system analysts' specifications are updated to correspond to changing client's requirements. Finally, the outputs of the software development process are similar to those of manufacturing. They include the final application matching customer specifications as well as the library of objects and documentation that can be later used for training and process standardization.

Therefore, it is obvious that although the two processes might seem disconnected and unrelated, so far the major players in manufacturing and software development are very similar (refer to Figure 2 for summarized correlation of process inputs, throughputs, and outputs).

Insert Figure 2 Here

Waste Correlations

The elimination of waste was discussed earlier in relation to basic lean manufacturing principles. Software development has very similar areas of waste. The following table summarizes the correlations of muda in both processes.

Insert Table 1 Here

Therefore, if the lean techniques help resolve most of these problems in manufacturing, they have the potential to, if not eliminate, then at least substantially decrease the same problems in software development.

IMPLEMENTATION OF LEAN MANUFACTURING TECHNIQUES IN OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Numerous parallels exist between manufacturing and software development. The following section applies the major lean techniques that are so effective in increasing productivity and efficiency in manufacturing to object-oriented software development. Lean software may be the solution to minimizing defects, reducing numerous occurrences of waste and maximizing efficiency and overall improvement software quality.

Preliminary Steps

Determine Reasons To Go Lean – The first question a company must ask itself before truly dedicating itself to lean is "Why pursue lean?" This is the single most important question that must be addressed before implementation occurs. If convincing rationale and supporting examples can be found, then lean should be attempted. A company must first examine its entire system. The entire system should be documented in full and investigated. Every process should be looked at carefully to determine if it adds value to the system processes. Problem areas, especially the wasteful processes, are identified and removed.

Organizations must weigh both positives and negatives of applying lean techniques to their system. Usually there are four scenarios companies are confronted with when applying lean techniques:

- Many benefits while having little or no negative repercussions to the system
- Some benefits but also some minor negative consequences to the system
- Few benefits and major system setbacks
- No benefits and major setbacks to the system

If an organization fits into the first two scenarios then applying lean techniques is a viable alternative. Utilizing lean techniques creates a system that has *just in time* inventory (or objects as in the case of software development), produces high quality, low cost products, and overall produces less waste. If, after examining the system processes and applying lean techniques, the desired changes do not materialize, then a company has wasted valuable resources and more than likely created new problems.

Customize the Transformation Process – Corporations wishing to implement lean cannot use another company's transformation process without adaptation. Companies have different infrastructure systems, as well as different cultural, physical, and environmental aspects of a business that all must be transformed. Customization involves adjusting lean principles and implementation strategies to fit a particular organization's goal, mission statement, or processes.

Train and Empower Employees – Minimizing resistance to implementing lean techniques can be accomplished by including all employees from the beginning in the transformation process. Employee training and empowerment is essential. Workers will feel that the new system was not forced upon them but rather they were involved in the decision-making process. Information about upcoming changes should be circulated regularly throughout the entire organization. Company's statistics, policies and agendas should be clearly stated and posted for reference. Finally, easy and effective suggestion systems should be instituted for gathering employee feedback and focusing on quality improvement.

Following the informative stage, employees should be properly trained in lean principles. People should think lean, produce lean, and recognize waste. Continuously training in new technologies, processes, and principles is another requirement. Cross-training in a variety of different business areas ensures that the majority of workers have the necessary knowledge and hands on experience. When workers understand the diverse business aspects, it becomes easier for them to interact and communicate with each other, thus increasing the productivity of software development.

Finally management should empower their employees. Not only will this increase job satisfaction, improve the working environment and interdepartmental cooperation, but it will also encourage teamwork, independent decision-making and efficient responsibility delegation with shared accountability. Workers will be trained not to continuously depend on management to make all the decisions. They will be provided with enough authority to utilize their individual or team problem-solving skills to establish and execute effective solutions.

Allocating support without removing an individual's responsibility is not only achievable but would also generate even greater feelings of individual accomplishment and contribution. However, choosing the

proper person for a specific task is imperative. While the task is delegated to a particular individual, it is management's job to provide continuous performance feedback and evaluation so that the person can keep track of their progress.

Standardize Software Development Regulations –A company's software development standards should be consistent with the industry's standards to successfully implement lean and be able to reuse objects across many applications. Lean techniques cannot be effectively implemented if the company's software processes are continuously changing. Development process stability is important. Newly instituted standards must be used on a regular basis for lean execution to be productive. Regulations and requirements guide and effectively constrain each programmer's object design and implementation, as well as the overall software development process. Objects, code blocks, methods, and functions will become reusable in various private software packages, or other company's applications. Industry-wide cooperation will increase total quality and improve customer satisfaction.

Commit Fully to the Transition Process – Successfully implementing lean techniques requires the dedication of the entire organization. Otherwise, lean will not be used to its full potential. All operations, analysis, design, and programming aspects of software development must be realized and transformed. Areas that originally might seem inconsequential can be easily overlooked and later crucially impacts testing. Such inconsistencies will create more waste, rework, and recovery. Furthermore, it might discourage continued implementation of lean and prolong the payback period by requiring higher financial resource allocation.

Integrate Various Roles and Encourage Cooperation Among System Analysts and Programmers – In order to accomplish continuous flow in software development (SD), programmers and system analysts must cooperate and collaborate throughout the entire process. Effective communication and feedback about customer requirements, application prototypes, design specification, and implementation details is crucial. The cooperation between analysts and programmers does not stop at effective communication. It should include collaborative idea generation and decision-making relevant to SD as well as continuous resource support from both departments. Finally, adequate time should be allocated to discuss SD processes and potential improvement strategies.

Revenues of the Transformation

"In the last years, the demand for new languages and tools for software development raised dramatically among Programmable Logic Controller (PLC) users, that is why the International Electrotechnical Committee published a normative for PLC programming languages standardization called IEC 61131-3" (Bonfe; Fantuzzi, 2001, pg. 787). The adoption of lean practices into the object-oriented realm ultimately comes about with the standardization of programming methods. The four principle concepts of lean are: eliminate waste, standardize work, produce zero defects, and institute one-piece flow (Piszczalski, 2000). With lean manufacturing, less is best but with IS departments, more is better (Piszczalski, 2000). With standardization, however, IS departments will adopt a lean philosophy.

With the introduction of lean manufacturing techniques into object-oriented software development, programmers will have to produce less code as well as become much more proficient in the field of reusability. "Object-orientation teaches the principle of 'design for change': if we structure the code into modules we can minimize the impact of future changes, i.e. side effects on the structure as a whole" (Valerio; Cardino; Di Leo, 2001, pg. 100). Lean techniques will also increase the ease of program maintenance because the code will be simpler and thus more understandable and readable.

Program components are analogous to the interchangeable parts in manufacturing. These components have already been tested, and need only to be successfully implemented into the new application. Ultimately, the error checking becomes easier because only the new code needs to be thoroughly debugged and tested.

"By following a few straightforward rules and conventions, you can achieve a logically cohesive, loosely coupled, object-based design that facilitates reuse, future feature expansion and substantial error localization and isolation" (Binder, 2000, pg. 138). With the implementation of lean techniques in software development (SD), the improved processes can be expected to decrease SD time. Lower development costs should be accomplished through shortened development times, enhanced reusability, easier maintenance, and improved documentation.

Barriers and Pitfalls

The above section detailed the benefits and efficiency of applying lean manufacturing techniques to software development. This is great for well-established firms, but for new companies, lean may not be the strategy to pursue. Start-ups and some of the smaller companies usually have limited funds to operate their business let alone have the spare capital to create a library of reusable objects or time to research and develop the appropriate implementation plans. On the other hand, large companies may not be willing to invest the necessary capital into a lean transformation as well. Primarily, it may be inefficient to change a current functioning system. Secondary, employees may not be receptive and committed to the transformation process, sentencing it to failure.

"In an attempt to ensure that programs are readable and easily maintainable, many companies have tried to turn programmers into cookie-cutter automata" (Strassberg, 2001, pg. 129). Some believe this is an argument against the introduction of lean manufacturing. Automation and reuse may stifle the creativity of programmers and quite possibly impede their ability to create solutions to new problems. It also constrains a programmer's flexibility. "Programmers are people, and their experience, training, and preferences vary [...] (therefore) forcing different programmers' styles into one mold is a pretty good recipe for errors and wasted time" (Strassberg, 2001, pg. 129). Trying to get every programmer on the same page is very difficult. The training it would take may not be worth the trouble because it still might not produce any results. The reason is the programmers themselves may be happy the way they do things now, and no amount of training can change that.

The last shortcoming that occurs when implementing lean techniques is that the company does all they need to do, but the benefits or return on investments they expected to see do not materialize until years afterwards. For instance, a company may spend millions on training their programmers to reuse their code, instituting a system that rewards code reuse, and the setup of a library to house that reusable code. The benefits in reduced errors, lower costs, and shortened development may not come about until years later. Companies may hesitate before they decide to invest in such an endeavor because they may not have time to wait.

RECOMMENDATIONS

The goal of this paper was to consider the implementation of lean manufacturing techniques in object-oriented software development. As a result, certain practical recommendations will be made to allow the transition and application process to be valuable, efficient, and straightforward. Following are just some of the steps for applying lean principles in object-oriented software development.

Organization Specific Standardization

The transformation process begins with an organization specific standardization. Each organization must find the most appropriate way to implement this principle instead of using the generic idea of instituting manufacturing standards in software development. Standardization in object-oriented software development does not suggest that every analysis, design, development, and implementation process should be the same. What it does suggest is that a company should inaugurate a set of standards, rules, and regulations that employees must follow on a regular basis. These would include the general design principles like object encapsulation, information hiding, hierarchical structure utilization, polymorphism and, finally, inheritance. The design standards can be laid out in a programmer's handbook or a department policy statement so that employees can be easily educated and trained to follow these guidelines. However, those are just some of the object-oriented design principles. Standardization also applies to the lean techniques such as waste reduction during all development stages, statistical quality control procedures where the same measurements will be evaluated for all applications, and the processes themselves. In conclusion, an organization should encourage standardized coding procedures, methods to analyze requirements and implement design principles.

Reduction of Waste in Software Development

Since the lean principle of waste reduction worked so well in manufacturing, it should be applied to software development (SD). The following strategies were influenced by the implementation approaches of lean techniques in manufacturing:

- Eliminate cubicles in the workplace. This will allow easier information flow and communication between employees. All team members, working on the same processes, objects or code sections, should be located in close proximity to each other whether that means relocating them to the same

floor or work area. Finally, cooperating departments should also be adjacent or at least reasonably close to each other. This approach will decrease the waste found in information flow and product or employee travel.

- Customers should be allowed to select features from a predefined list of components and capabilities similar to buying a car. A standard package should be available and all add on attributes should be chosen separately. This will not only clarify user requirements but also eliminate unnecessary and unused features in an application.
- System analysts should work closely with customers to determine a reasonable set of requirements, thus eliminating some of the inputs' waste.
- SD organizations should hire people with appropriate skills for a particular job.
- Finally, continuous testing and debugging will decrease the number of defects in the final product, increasing software quality and customer satisfaction.

Cross-Functional Teamwork and Cooperation

Training, education, and empowerment should be the guiding strategies for information systems management. Training in a variety of different areas such as design, programming, and strategic implementation will allow the assembly of multifunctional teams that are capable of performing an assortment of software development tasks. Education in lean principles will encourage employees to recognize problem areas and potential waste independently without management's tedious and constant involvement. If people are aware of what should be done to solve newly occurring problems autonomously, then management can concentrate on investigating new techniques and strategies that will increase productivity and efficiency. This principle also leads into employee empowerment where problem solving and idea contribution is encouraged at all levels. Such involvement contributes to the sense of significance and contribution to the final product. Another lean implementation tactic to be utilized is team collaboration. Members should be aware of the other's progress, project details, and responsibilities. Such cooperation will allow employees to be aware of the objects that are being developed by their teammates that can be reused

later and of the innovative techniques that some of the co-workers are employing for a more productive product development.

Debugging and Testing Throughout the Entire Software Development Life Cycle

Unfortunately, in the current object-oriented software development, even though some of the individual objects are tested upon completion, the programmers wait until the final product is compiled to thoroughly test it. This strategy supposedly saves debugging time during the development stage. However, what the programmers do not realize is that at the end, it will take much longer to debug and test the final application because it will be about 5 times harder to find all the defects and errors, especially if they are caused by incorrect user requirements translations. Therefore, if the software package is tested from the start of its development, beginning with the coding of the first object and method, the resulting product will have fewer errors and deficiencies.

Continuous Improvement

Finally, the most important implementation principle is continuous improvement. Even if transformation from traditional to lean software development was successful, an organization should not stop its search for better quality and process efficiency. Continuous improvement encourages ceaseless system, software, and hardware update to keep up with changing technology. It also promotes exceeding customer expectations and raising the quality bar for its employee to be able to produce a zero defects application.

CONCLUSION

Without a doubt, unless certain transformations are undertaken and new approaches developed, the problems plaguing software development will persist. Reforms and standards must be instituted in order to correct the inconsistencies and setbacks within the process. One developed solution is to incorporate and implement lean manufacturing techniques into object-oriented software development to reduce production costs and development time. Before manufacturing had transformed to lean, similar problems of waste, defects, and rework were present. Lean techniques significantly decreased the number of errors and encouraged a more

effective and productive manufacturing system with just-in-time inventory, less waste, and proficient resource management.

Numerous parallels exist between manufacturing and software development. Therefore, if the lean techniques help resolve abundant problems in manufacturing, they have the potential to, if not eliminate, at least substantially decrease the same problems in software development (SD). Implementing lean techniques in SD would drastically reduce the number of defects and waste, decrease product development time, and institute a set of industry-wide coding standards that promote reusability.

In order to utilize lean principles to their full potential companies must undergo a difficult and time-consuming transformation process. The success of lean implementation in software development depends on a number of factors. A company's internal processes, social structure, physical environment, and mission should all be re-examined with lean principles in mind. Upon successful implementation, object-oriented software development is expected to uncover its true potential for producing reusable objects as well as high quality, lower cost applications that exceed customer expectations.

REFERENCES

Allen, John H. "Make Lean Manufacturing Work for You," *SME*, June, 2000, pp. 54-64.

Bahrami, Ali. *Object-Oriented Systems Development – Using the Unified Modeling Language*, Irwin McGraw-Hill, Boston, 1999.

Bevington, D. "Technical Note: Business Function Specification of Commercial Applications," *IBM Systems Journal*, Vol. 39, No. 2, 2000, pp. 315-335.

Binder, Edward F. "Implementing Object-Oriented Designs in ANSI-Standard C," *EDN*, Vol 45, April 13, 2000, pp. 137-47.

Bonfe, Marcello, Fantuzzi, Cesare. "Object-Oriented Approach to PLC Software Design for a Manufacturing Machinery Using IEC 61131-3 Norm Language," *International Conference on Advanced Intelligent Mechatronics Proceedings*. Como, Italy. July, 2001. pp. 787-792.

Corr, Pat. PhD MIEEE, Queens University, Belfast. "Presentation on Software Quality Assurance," using *Standish Group Statistics* (www.standishgroup.com). 2002. http://www.cs.qub.ac.uk/~P.Corr/ProfPrac/Software_QA.ppt#2 Accessed May 22, 2003.

Lewis, Jim. "Get Ready...Get Set...Get Lean!" *Upholstery Design & Management*, December, 2000, pp. 42-43.

Piszczalski, Martin. "Lean Vs. Information Systems," *Automotive Manufacturing & Production*, August, 2000, pp. 26-28.

Stackpole, William. "Security Realities in Software Development," *Computer Security Journal*, Vol. 18, No. 1, 2002, pp. 9-14.

Sharood, John. "Is Lean Still Meaningful?" *Appliance Manufacturer*, 49. December, 2001. p. 160.

Strassberg, Dan. "In Developing Measurement Applications, Mixing Metaphors Minimizes Muddles," *EDN*, Vol. 46, January 18, 2001, pp. 129-40.

Valerio, Andrea; Cardino, Guido; Di Leo, Vincenzo. "Improving Software Development Practices through Components," *Euromicro Conference*, 2001. Proceedings. 27th. pp. 97-103.

Wight, Oliver. "Lean Manufacturing," *Works Management*, August, 2001, p. 39.

TABLE 1

Waste Correlations in Manufacturing and Software Development

Area of waste	Manufacturing	Software development
Unnecessary complexity	Product features, some never used by the customer, or documentation that makes it harder to understand and use.	Application objects and methods not needed to perform the specified task or the elements and added features that will never be used by the customer.
Labor used in production	At times production is subdivided into overly specific sections that require too many people performing similar or same tasks.	Development teams have an unnecessary large number of members carrying out redundant functions and reworking each other's code and ideas.
Space taken up by materials	Inventory might exceed the needed amount, taking valuable space in a factory. Even though the products or parts are not being sold or used, the company still pays for the space.	The size of an application especially with unnecessary features and inefficient coding is unreasonably large in size. Therefore, a customer might not be able to utilize the application due to the hard drive space restrictions.
Defects	Product defects might exceed the accepted norm, often 0-1%.	Excessive number of program defects due to the lack of testing and debugging, or the lack of acceptable defect rates.
Raw materials	Unused inventory of materials on hand because of incorrect order numbers or the supplier delivery schedule. Just in time delivery not implemented.	Unreasonable customer requirements with unattainable details as well as the large size and complexity of objects.

Inventory on hand as a result of overproduction	Pull production might not be instituted in the company and therefore, too many parts are being produced without being used by the assembly department. At times, one component piles up while the team is waiting for another part.	The programmers are writing code and features that the customer might not be willing to pay for due to poor communication with system analysts. Therefore, now you have too much code or objects that will not be implemented.
Travel distance of a product	Components might be too far away from each other so that the travel time is wasted.	The virtual travel of the code can be excessive. Also, the departments or teams might be located too far from each other whether it is different floors or buildings. This does not allow a clear communication flow that leads to misunderstanding.
Time	In manufacturing, this involves both production and delivery time.	In object-oriented software development, this deals with the time it takes for the development of individual objects and design specifications, as well as the time it takes to deliver the final system.
Skills	Each factory needs unskilled and skilled labor to perform various tasks, however, at times the balance is not achieved properly and therefore, some skills are wasted.	Overqualified employees might be performing non-value adding tasks. Underqualified personnel can be in the inappropriate positions.

FIGURE 1
Lean Manufacturing Principles

FIGURE 2
Process Inputs, Throughputs, and Outputs