# LEAN SOFTWARE DEVELOPMENT: IS IT FEASIBLE ?

*Sowmyan Raman, The Boeing Co, Seattle, WA 98124.*

## ABSTRACT

The Lean Aerospace Initiative originally began as Lean Aircraft Initiative (LAI) in the summer of 1992 at Massachusetts Institute of Technology (MIT). The refashioning of Lean Aircraft Initiative to Lean Aerospace Initiative (LAI) took place in November 1997 to reflect the addition of Space Sector. A consortium of Air Force and 21 defense firms including the Boeing Co and the erstwhile McDonnell Douglas was originally formed to investigate how the "Lean Principles" that were applied in the automobile industry particularly in Toyota could be adopted in the Aerospace industry including Electronics and Avionics segments [1].

It is no secret that the next millennium will be digital. Software will not only play an important role in the Aerospace systems but almost in every aspect of our life It is the dream of every enterprise to have software developed cheaper, faster, and better. During the last several years many technologies, methodologies and software languages and development tools have emerged as the answers to the above dream. We are yet to accomplish that dream

Modern aircraft is nothing but a bunch of "flying" computers connected together and it is just "Electronics in motion" or flying electronics. It is estimated that 50% of the development costs for any modern airplane is attributed to systems and a substantial portion to software development.

Toyota is accepted not only as a leader in the automobile industry but as a world class manufacturing company which is very successful in adopting lean principles which they call as Toyota Production System (TPS) [2]. This paper will briefly describe what is "Lean Thinking" and explore the feasibility of applying these principles to software development. It will include discussions on the emerging software development methods and Capability Maturity Model (CMM)[9] of the Software Engineering Institute (SEI) which gives us a framework. It will attempt to answer whether these principles of Lean can to applied to Software Development.

## What Is Lean ?

Lean is all about getting the right things to the right place at the right time the first time while minimizing waste and being open to change. The production system that was pioneered after world war II by the Toyota company for the manufacture of automobiles was labeled as "Lean Production" by the authors of the landmark book: The Machine That Changed the World [2]. This book was the direct result of the study by The Massachusetts Institute of Technology 's five million, five year effort, on the future of The Automobile. Lean Aircraft Initiative was formed to extend

0-7803-5086-3 /98/$10.00 ©1998 IEEE

the Lean production paradigm and knowledge-base from this study. A key goal was to articulate, define and apply concepts, principles and practices of Lean production to defense aircraft industry initially and aerospace industry in general latter, through quantitative bench-marking of best industrial practices.

The word *muda* a Japanese word is the center of this whole Lean revolution. *Muda* means "waste" specifically any activity that absorbs resources but outputs no value. Taiichi Ohno (1912-1990) , an executive of Toyota who developed and implemented Toyota Production System (now known as Lean) was horrified at different forms of *muda* which made him develop methods to eliminate them [3]. Some of them he identified are: Mistakes that are subsequently rectified, unwanted production and inventory, processes which are not really needed, unnecessary movement of people/things, waiting, and products and services which do not meet the needs of the customer. Some of the above are obviously more applicable to hardware environment rather than software.

What are the basic concepts of Lean Thinking. These are: Value, the Value Stream, Flow, Pull, Perfection. Let me describe these concepts[4].

Value: Value can only be described by the ultimate customer. It is useful when expressed in terms of specific product which meets customer's needs at a price at a specified time. In the context of aircraft industry Phil Condit, Chairman and CEO of The Boeing Co, has recently said " the first 75 years were defined by constant push for performance. The aerospace industry is going to change to value driven industry" [5]. Airlines want "value" as perceived by their ultimate customers, rather than advanced and exotic technology or super performance.

Value Stream: is a set of actions required to bring a value product or service to the customer. An aerospace example would be a titanium ingot made 10 times the weight of the finished part. By redesigning the ingot to suit several parts , unnecessary forging and machining that added no value were eliminated. By making supply chain transparent to all the players the value stream can be strengthened.

Flow: Once the unnecessary steps are eliminated the remaining value creating steps need to be made to flow. Instead of organizing functionally, development or production activities should be organized as continuous flow with responsibilities for complete processes or sub-processes. This type of arrangement eliminates sub-optimization of any functional group which does not contribute to the ultimate output.

Pull: The production system should be developed such that the customer pulls the product. Every down stream process should pull from its upstream process. This way wait and inventory in between processes can be avoided.

Perfection: This is a principle based on the concept that there is no end to the process of reducing effort, time, space, cost and mistakes. Basically this is the Japanese concept of *Kaizen*[7] or continuous improvement. Always try to be better , cheaper and faster and there is nothing like one has reached the perfection.

## Software Development - Today

Professor Niklaus Wirth laments that almost all software have grown "fat". He contributes this tendency to growing hardware performance and the unnecessary complexity built in the software. Obviously "fat" software needs "fat" software development [6].

The causes for complexity are: (1) software vendors adopt almost any features that users may want and (2) monolithic design with all conceivable features. Complex problems

require complex solutions and resulting complex and "fat" software. Complexity is also viewed as power by many. Vendors use the strategy of customer dependence rather than customer education which is more profitable.

Sophisticated software are complex and good solutions are developed by iterative and time-consuming thought processes. In view of time constraints software inadequacies are corrected by quickly conceived additions increasing the complexity and size of the software. Prof.Wirth recommends good engineering by gradual stepwise refinement of the product which we understand as continuous improvement or *kaizen* [7].

### *Lean Thinking in Software Development*

Let us see whether the basic Lean principles indicated earlier can be applied to software development.

Value: value is what the customer perceives as value or useful to him. In the mass market software industry Microsoft has been very successful. According to Michael Cusumano, Microsoft and Bill Gates value making money and they believe that delivering value to the customers only will make the most money [8]. They do not value technology for the technology's sake. Most of the successful software technologies Microsoft adopted were invented elsewhere including Windows.

There are enough stories about completed software that were not usable by the customers. The value for the customer from his point of view, need to be understood at the stage of Requirements development. SEI Capability Maturity Model [9] includes Requirements Management in level 2 which is one of the early stages of Capability Maturity. Usability by the customer is very important as that contributes to his value.

The question that arises is how can we extract the "value" as perceived by the customer. Rapid Prototyping is one of the processes that may help here. By rapidly prototyping and reviewing with the customer one can develop the customer's requirements and his value. The erstwhile McDonnell Douglas Aerospace has used RAPIDS process very successfully. Microsoft uses flexible incremental approach by incrementally changing evolving features targeting specific customer activities and values. The "spiral" software development model as well as "stacked spiral model" use the concept of prototyping [19]. One way to achieve speedy prototyping is to use "software components". Object-Oriented technology plays an important role here.

Quality Function Development (QFD) is a team-based technique that helps in identifying and translating customer requirements [10]. QFD has been used successfully in hardware environments and its application in software is not yet dominant. Many defense programs like The Joint Advanced Strike Technology is employing QFD. QFD is based on the popular article "House Of Quality" appeared in the Harvard Business Review [11]

First and foremost is to understand clearly what the customer wants and what he considers as "value".

Value Stream: is a series of actions for developing a value product for the customer from order to delivery. By identifying the stream and those activities which actually add value, it would be possible to eliminate activities that do not contribute any value. Gordon Bethune the Chairman and CEO of the Continental Airlines in his recently published book: From Worst to First [12], cites how even after improving the on-time arrivals of the flights, the customer's satisfaction was poor due to the delays in getting their baggage. It is important that the whole "value stream" has to be improved to add value to the product.

In software development it is not just enough to understand customers requirements but is equally important that the design is developed meticulously using good engineering by gradual stepwise refinement of products. Sloppy design to save time will deliver a product with less value during the life of the product. A good design architecture and a Programming language based on object oriented technology will ensure value to the customer.

By identifying the stream and the activities that actually add value it would be possible to eliminate steps which do not contribute to the value for the customer. CMM helps in providing a framework. In SEI CMM level 2 the emphasis is to get the processes under control and establish repeatability. Once the processes are defined and repeatable focus on the process (level 3) and process improvement (level 5) are possible. By defining the process one can eliminate those processes that do not add value[9]

Flow: The third concept in Lean Thinking is Flow and we should make value flow. Flow is to make every activity in the product development take place continuously with no stoppage, no back flows and no scrap. Translating this to software development at Microsoft would be syn-and-stabilize approach according to Cusumano. In this approach developers activities as individuals and as members of teams working on different features are continuously coordinated as project proceeds. Microsoft people refer to this as "nightly build" or "Zero-defect" process[8]. The term "build" refers to integrating partially completed or finished pieces of software during development process to see what functions work. This structure, while allowing people enough flexibility to be creative and evolve product's details in stages, results in coordination of individual engineers and teams. Developers are required to test features with customers and refine their design during development. Scrapping of efforts can be avoided by putting in place processes that ensure continuous product development as described above.

Pull: Pull is an interesting concept in Lean. It means that no one upstream should produce a good or service until the customer downstream asks for it. This has been the basis for "Just In Time" (JIT) inventory concept in a manufacturing environment. In software environment JIT may not be applicable but it is needless to emphasis that only what is required by the customer should be built in the software. The software community has a tendency to include features which they think are "nice" for the customer to have. The result is "fat" and unnecessarily complex software.

Perfection: This last concept in Lean Thinking is to continuously move towards perfection or *Kaizen*. This is quite opposite to the reengineering concept that has emerged past few years. SEI CMM emphasizes continuously improving the process and defect prevention as key process areas at the highest level 5. There are two aspects in perfection. The first is eliminate defects while developing the product. The second, is improve the processes so that all "waste" can be removed. Michael Hammer the author of "reengineering" has recently conceded in his current book "Beyond Reengineering" [13] that reengineering has failed since process focus was not integrated in the reengineering efforts. Process Management defined as one of the Key Process Area under SEI CMM level 5 is very essential for moving towards perfection. By establishing appropriate processes defects in the software development can be avoided. Component-Based software development enables to use already proved components thus ensuring defect free [14].

## Lean Software Development

The literature survey has produced only two articles on Lean Software Development. The work of Alexander C.Hou under the auspices of Lean Aircraft Initiative is an exhaustive paper encompassing Lean Hardware and Software System Development[15]. The author has described and discussed five methodologies : (1) The Rapid Development Process, (2) The GriTech rapid development process, (3) Ptolemy-supported hardware and software co-design (4) The RASP (Rapid Prototyping of Application Specific Signal Processors) design methodology and (5) Clean room software engineering. He has recommended study of ARPA/Tri-service RASPP program, ARPA STARS (Software Technology for Adaptable, Reliable Systems), and F22 Avionics development case study. Prof Wirth has argued for Lean Software and has not discussed Lean Software Development . He has however recommends simple object-oriented programming language Oberon instead of C++ which is becoming a very popular language.

### *Japanese Experiences*

Michael Cusumano has described in detail the Japanese experiences of improving Software development through what he calls as "Software Factories" [16]. Software Factory is a concept or a philosophy that some software can be produced in a manner more akin to engineering and manufacturing than craft practices. Hitachi, Toshiba, NEC, and Fujitsu have successfully adopted this concept. So if one accepts the Software Factory Philosophy then application of Lean principles can also be accepted easily.

Even though the Software Factory was originally proposed in US it did not live very long in US. However US Department Of Defense had subsequently developed the concept of software development " component-by-component" rather than "instruction-by-instruction". This was called "mega-Programming" [17]. Emergence of Object-Oriented programming languages rekindled interest in "re-use" of software components and there is considerable interest now in the industry for Component Based Software Development. Software leaders like Microsoft have developed Class libraries for object oriented languages which promote re-use.

The "re-use" help in achieving in improved quality and reduced flow time and cost since already developed, tested and proven components are used. Reuse is a very dominant principle under Lean.

Japanese have been using Software Factory concept and continuously improving the software processes. These have helped many Japanese companies to reach SEI CMM level 5. Fujitsu for example has software groups functioning at CMM level 5 and they do certify some of their software "Zero defect". This is based on "perfection" principle of Lean. Many Japanese firms attribute their higher quality and productivity to software reuse but they also define reuse as systematic way of integrating components that have been certified as "zero defect" [18].

### *Flexible Software manufacturing*

Software Factory concept introduced the factory model to software development. Object-Oriented technologies promoted reusability to a great extent.

Paul Bassett comments that the most difficult problem in software reuse is not technical rather it is establishing a manufacturing culture[19]. The craftsman's

mentality of distrusting what others do has been the main source of obstacle in implementing aggressive reuse in software development. He argues that to manufacture software (not develop) managers must support development processes, infrastructure and most importantly culture. Development processes should include processes for developing reusable components and for developing systems. The infrastructure must align with a reuse strategy. Both execution and construction standards may be the answers to this problem. Culture is a very dominant in changing any processes and software processes are no exception. Most of the problems in the direction of software manufacturing are people related.

It is claimed that by using Object-oriented reusability technologies based on Framework have achieved 70% reduction in schedule and 84% in costs [19]. Both Flow time and cost reductions are the basic goals of Lean thinking.

## Lean Enterprise Model

Before a Lean Software Development model is suggested, a brief review of the Lean Enterprise Model as developed by Lean Aircraft Initiative is worthwhile.

Lean Enterprise model of LAI is based on the meta-principles: Responsiveness to change and Waste Minimization. The enterprise principles consist of Right thing at Right Place, Right Time, and in Right quantity/quality, effective relationships within the Value Stream, Continuous Improvement, and optimal First Delivered Unit Quality. This model suggests enterprise level metrics like flow time, stakeholders satisfaction, resource utilization and quality yield. This model also recommends the following 12 overarching practices:

(1) Identification and optimization of enterprise flow

(2) Assuring Seamless information flow

(3) Optimization of capability and utilization of people

(4) Decisions at lowest possible level,

(5) Integrated product and process development

(6) Mutual trust and commitment

(7) Continuous focus on the customer

(8) Lean leadership at all levels

(9) Challenging of existing processes

(10) Nurture a learning environment

(11) Ensure process capability and maturation

(12) Maximize stability in changing environment.

The above apply to any enterprise including those which are involved in the development of software.

## Lean Software Development Model

While it is true that software development is very different from normal hardware product development and production the Lean principles can still be applied. There is always an argument that Software development is a creative work as such not amiable to hardware development practices which I believe, is not true. Lean thinking is applicable to any activity or processes including software development processes.

My first suggestion is that Software Factories model based on reusability of software components should be adopted. Industrializing software development is the first step.

There are several technologies emerging. IEEE has a Reuse Steering Committee (RSC) with a scope related to the analysis, design, implementation, verification,

validation, documentation and maintenance of reusable software assets, as well as their supporting infrastructure in the creation and maintenance of new software systems. RSC will ensure coordination of standards related to software reuse and their general harmonization with the more general body of software engineering standards. Emerging programming languages particularly those based on Object-Orientation like C++, Java, Visual Basic, and Ada to mention a few facilitate reusability.

Concurrent Engineering has been effectively used to reduce the flow time of product development in the hardware world. Software product development can benefit by using this concept. Software development concurrency often involves the use of system prototypes that allow the software architect to build a skeletal system and try it out in the real world environment. The "spiral" software development model as well as "stacked spiral " model use this concept of prototyping [17]. Prototyping helps not only to resolve design issues but also the most important software requirements issues.

SEI's CMM[9] should be adopted to move the organization to higher levels with emphasis on the processes.

Continuous Improvement and "perfection" concept of Lean can be followed by using what Wirth calls as "stepwise refinement" or "synch-and-stabilize" approach used at Microsoft. Microsoft also assigns the responsibility of ensuring that the "build" is not broken by the individual's software component or code fragments to the individual developer. Thus the waste introduced in the build by defective code is eliminated.

Testing is a very important and also very expensive phase in the software development. In Japan quality control or quality assurance is defined as the development, design, manufacture and service of products that will satisfy the customer's needs and at the lowest possible cost. The customer's satisfaction with product quality is an end in itself. Inspection by specialized inspectors have been minimized in Lean factories. Inspectors whose activities are outside the manufacturing process perform operations with no value added and thus impact productivity and cost. The responsibility for the quality of the product lies with the producer. This would be a radical thinking in software development. In the current scenario extensive testing is done at various levels, unit test, Integration test, functional test and final tests by the end user (Alpha and Beta tests). There is scope to adopt Lean thinking here by delegating the responsibility to the developer or the integrator for quality. This is an area which requires some investigation. It is also evident that if software components already produced are used testing will be minimum. We understand that as the use and age of the software component increases its quality also does.

## Conclusion

I have discussed above what is meant by Lean and the various software methodologies/technologies that contribute to Lean thinking in software development. The next steps in the software community is to move from a craft to an engineering discipline, industrialize the software development utilizing factory model and Lean Thinking. Reusability, rapid prototyping, spiral model, object-oriented technologies, component-based software development , concurrent engineering, quality function deployment are some of the concepts that will help in the direction of Lean

The question whether Lean Software Development is Feasible can easily be answered with "yes".

## References


(1) Lean Air II , Winter 1998, A Publication of the Lean Aerospace Initiative, Massachusetts Institute of Technology, Vol 5, No 3.

(2) J.P.Womack, D.T.Jones, and D.Roos, The Machine That Changed The World, Simon & Schuster, 1990.

(3) Taichii Ohno, Toyota Production System, Productivity Press, 1988.

(4) J.P.Womack, and D.T.Jones, Lean Thinking, Simon & Schuster, 1996.

(5) "Condit, Stonecipher Speak Out", Boeing News, an internal Boeing company magazine, June 12, 1998.

(6) N.Wirth, "A Plea for Lean Software", IEEE Computer, Vol 28, No 2; February 1995.

(7) M.Imai, Kaizen, The Key to Japan's Competitive Success, McGraw-Hill Publishing Co, New York, 1986.

(8) M.A.Cusumano, and R.W.Selby, Microsoft Secrets, The Free Press, 1995

(9) M.C.Paulk, W.Curtis, M.B.Chrissis, and C.V.Weber, Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-2, Software Engineering Insitute, Carnegie Mellon University, Pittsburgh, PA 15213, 1993.

(10) Y.Akao, Quality Function Deployment, Integrating Customer Requirements into Product Design, Productivity Press, 1988.

(11) J.R.Hauser and D.P. Clausing, "House of Quality", Harvard Busines Review, May/June 1988.

(12) B.Gordon, From Worst to First: Behind the Scenes of Continental's Remarkable Comeback-A Flighgt Plan for Success, John Wiley & Sons, 1998.

(13) M.Hammer, Beyond Re-Engineering: how the process-centered organization is changing our work and our lives, Harper Business, 1996.

(14) A.W.Brown, Component-Based Software Engineering, Selected papers from the Software Engineering, IEEE Computer Society Press, 1996.

(15) A.C.Hou, "Toward Lean Hardware/Software System Development: Evaluation of Selected Complex Electronic System Development Methodologies, Report-Lean 95-01, Lean Aircraft Initiative, Massachusetts Institute of Technology, Cambridge, 1995.

(16) M.A.Cusumano, Japanese Software Factories, Oxford University Press, 1991.

(17) Draft Department of Defense Software Technology Strategy, Office of The Director of Defense Research and Engineering, Washington DC, 1991

(18) J.Vu, "Boeing Scientist Tours Japan", unpublished Boeing internal document.

(19) P.Bassett, "Managing for Flexible Software manufacturing", IEEE Computer, July 1998