Lean Software Management: BBC Worldwide Case Study

Journal:	*Transactions on Engineering Management*
Manuscript ID:	Draft
Manuscript Type:	Research Article
Specialty/Area of Expertise:	02 R&D and Engineering Projects Department
Keywords:	Agile, Lean, Software, CMMI, lead time

Lean Software Management:
BBC Worldwide Case Study

***Abstract*: This case study examines how the lean ideas behind the Toyota Production System can be applied to software project management. It is a detailed investigation of the performance of a 9 person software development team employed by BBC Worldwide based in London over a 12 month period.**

The data was collected by one researcher in 2009. It involved direct observation of the development team, the kanban boards, the daily stand up meetings, semi structured interviews with a wide variety of staff, and statistical analysis.

The evidence shows that over the 12 month period, lead time to deliver software improved by 37%, and consistency of delivery rose by 47%. Output increased, while defects reported by customers fell 24%. The significance of this work is that it shows how to considerably improve the performance and consistency of Agile software development techniques.

The conclusion is that the adoption of lean and kanban ideas have enabled the maturity of an Agile software process to move rapidly towards CMMI Level 4. The faster, more responsive delivery has reduced both technical and market risks. The drawbacks are that it may not fit well with existing corporate standards and managers may find their new role challenging.

***Index Terms*:- Lean, software, agile, CMMI, lead time**

I. INTRODUCTION

Lean thinking is important because it can reduce error rates to 1 per million units. It has been shown without question to have the potential to at least double the productivity of both manufacturing and service organisations. It also significantly reduces the time taken to deliver new products while substantially reducing cost. The evidence from Toyota (Japan), Porsche (Germany) and Pratt & Whitney (USA) confirms this. (Womack et al 1990, 1997)

Applying the ideas from the Toyota Production System (TPS) (Shingo, 1981) or lean thinking to the management of software projects therefore promises great improvements. This case study records the practical experience gained between April 2008 and October 2009 by the London based BBC Worldwide when it applied lean thinking to managing software development.

BBC Worldwide is the main commercial arm and a wholly owned subsidiary of the British Broadcasting Corporation (BBC). Its mission is to create, acquire, develop and exploit media content and brands around the world in order to maximise the value of the BBC's assets for the benefit of the UK licence payer. In 2008/09 BBC Worldwide generated profits of £103 million (before exceptionals) on revenues of £1.004 bn. (BBC, 2010)

The basis of lean is the absolute elimination of waste (Ohno 1988). This requires a focus on the flow of work through the system to ensure that material is produced only when it is needed and in the exact quantities required. This enables near zero inventory levels to be

approached which makes production more flexible and also allows sources of defects to be quickly identified.

Waste is defined as anything that does not produce value for the customer. The objective is to achieve the just-in-time delivery of materials. The lean techniques are only of use if there is a commitment to eliminate waste and make fundamental, continuous improvements to the production system.

'Perhaps the most striking difference between mass production and lean production lies in their ultimate objectives. Mass producers set a limited goal for themselves – "good enough", which translates into an acceptable level of defects, a maximum level of inventories, a narrow range of standardised products. To do better, they argue, would cost too much or exceed inherent human capabilities. Lean producers on the other hand, set their sites explicitly on perfection: continually declining costs, zero defects, zero inventories, and endless product variety.' (Womack et al, 1997)

II. LITERATURE REVIEW

The first recorded experiments with lean software development were by Middleton (1993). Microsoft reported how the lean mistake proofing of a software process eradicated whole classes of errors. Tierney (1993) Hou (1995) for the US Department of Defense concluded lean techniques were the only way forward. Morgan (1998) from Cummins Engine Company '…provides some compelling evidence that the ideas of lean manufacturing are indeed applicable, in principle, to software development.' Hamilton (1999) with the Department of Defense concluded that: '…shifting to lean principles improves cycle time reduction and overall quality in the software development process.'

Lean software development is an evolutionary, incremental approach as advocated by Gilb (1988). Although it has different intellectual roots it has much in common with Agile software development (Cockburn, 2002). But the reliance on data within lean means that it has the quantitative rigor required by CMMI Level 4 (Chrissis et al, 2004). The mathematical basis of lean is well described by Hopp & Spearman (2001).

Timberline Software in Oregon in 2002 with 450 staff was the first recorded full industrial implementation of lean software development. They reported considerable improvement but their focus on setting a common tempo or 'Takt' time for software development based on similar sized work units has now been superseded. (Middleton et al 2005)

The lean software ideas were developed by Poppendieck (2003) and Middleton & Sutton (2005). Anderson (2003) and Ladas (2008) provide valuable insights and perspective. Moving upstream and applying lean thinking to influence project selection and definition creates great benefits (Seddon, 2005). The proceedings of the first Lean & Kanban Software conference (Willeke et al 2009) and the work of Shalloway et al (2010) show adoption is spreading. But there is a clear need for more rigorous case studies of implementations.

III. RESEARCH METHODOLOGY

The research question was to establish the benefits and costs of using lean and kanban to manage software projects. The research method was for an experienced researcher to observe and write up the operation of the BBC Worldwide Webmedia department's software

processes. The 7 visits to London of 2 – 3 days each took place between June and October 2009. These were supplement by numerous phone calls and e-mails.

The advice of Eisenhardt & Graebner (2007) that with case study research '…close adherence to the data keeps researchers "honest".' was followed. The pure positivist approach is that truth can always be discerned from untruth; and that the truth can be discerned either by deduction or by empirical support and by no other means (Jankowicz, 1991). The interpretivist model is that that an in-depth understanding of a phenomenon may only be gained by studying it in context from the participants' perspectives (Walsham, 1995). For this study the position of Wynekoop and Russo (1997) that both approaches are useful was adopted.

Seaman (1999) emphasises the importance of 'triangulation' in gathering data from as many different sources as possible to assist accuracy. Data was collected from:

- The most mature software development team, called Digi-Hub.
- Semi structured interviews with developers, project managers, business analysts and managers.
- Walking through the operation of the kanban boards that visually displayed the flow of work so enabling it to be controlled.
- Recording the precise operation of the kanban boards.
- Observing the daily 'stand up' meetings at the kanban boards where work allocations were discussed and agreed.
- Review of statistical analysis of the outputs from the system.

In Easterbrook's (2008) terms this is an exploratory case study. The research is asking knowledge questions focused on recording and understanding how the system was operating.

The key features of a lean system are:

- Levels of work in progress (WIP) e.g.: requirements, designs, and code, are deliberately kept as low as possible.
- Work is 'pulled' into the software development system only when there is capacity to work on it, rather than 'pushed' in regardless of capacity available.
- 'Autonomation' where technology is used to speed the flow of work through the process.
- Process improvement and waste elimination are routine.
- Visual indicators used extensively so even a causal observer can see the status of the work in progress.

IV. RELIABILITY OF THE DATA COLLECTED

With a case study there is a danger of bias in the data collected which would undermine or destroy the validity of the results reported. The following areas were reviewed to identify any possible distortions in the data.

A. Time line

The implementation of lean and kanban was started in April 2008. Due to the necessity to stabilise the processes and adapt to the changes, data collection did not start until 3 months

later in August 2008. The data used in this paper refers to the 12 months from October 2008 to October 2009.

B. Size of projects

The data shows that the size of the projects being handled over the 12 months of the study did fluctuate but there is no clear trend towards smaller projects. This means the results are not biased by project size.

Month	Small	Medium	Large
01/11/2008	11	5	0
01/12/2008	5	13	2
01/01/2009	7	5	5
01/02/2009	3	0	0
01/03/2009	18	9	1
01/04/2009	13	2	5
01/05/2009	7	4	0
01/06/2009	5	3	6
01/07/2009	15	1	0
01/08/2009	13	4	1
01/09/2009	8	4	0
01/10/2009	7	2	0
01/11/2009	5	3	1

While the units of work going through the system were deliberately made as small as possible, the size and nature of the projects themselves were unchanged.

C. Complexity of work

If the projects became less complex over time then this could be a source of bias in the results. The list of all the work undertaken was reviewed. It was clear the work was very varied and came from different sources. There was no evidence to suggest the complexity had changed. A reduction in the complexity of the work was therefore not distorting the results.

D. Governance arrangements

The structure was:

Business Board (Strategy & Budget)
Project Board (Detail & authorise specific work)
Product Owner (reconcile Business & Customer wants)
Users requesting work (10 Sign off work completed)
End users (200 - 300 people)

Note: End users are those internal to BBC Worldwide. The digital assets created were ultimately used by millions of people.

This governance structure had been unchanged since before April 2008. But over the period of the study it was reported that stricter identification of the business benefits was required

before projects were authorised. This may mean projects are better thought through as regards return on investment but at the technical level the work was unchanged.

E. Composition of team

The team personnel were the same since October 2008 with the same Project Manager. The data reflects the work of all the team, not just selected high performers. Also all work carried out, including low priority and legacy improvements tasks is reported. The skills of the team improved over the 12 months.

F. Engineering Practices

Work to improve engineering practices started in April 2008 which involved the following:

Test Driven Development (unit tests)
Automated Acceptance Testing (main suite completed April 09)
Source Control Software
Bug tracking software
Decoupling – improving legacy software (April – July 08)
Minimal Marketable Feature (MMF) introduced April 09

This resulted in higher test coverage and releases increasing from one every two weeks to almost daily. The same engineering practices were in place but their use was consolidated and improved over the period studied.

G. Overview of data collected

The size, complexity, type and volume of the work handled did not change materially during the 12 months of the study. The team and the governance structure remained comparable. The engineering practices were improved but most were in place by October 2008. Certainly a stable team with better tools would be expected to show improvement over a 12 month period. The only other difference was the introduction of lean thinking which will now be explored in more detail.

V. DIGITAL HUB (Digi-Hub) TEAM

In 2009 the team had operating costs of £1.5m and a development budget of £675K. It was made up of 9 staff: Project Manager, Business Analyst, Software Architect, Tester, Lead Developer, 3 Developers and a Support Developer. It was working on a mix of developing new software and software maintenance. The technology used was C#, .NET, MS SQL Server, and legacy Connected Service Framework (CSF) code.

In April 2008 the kanban boards were installed and all the value adding stages of the development life cycle (value stream) were drawn onto them. Restrictions on the amount of work in progress (WIP) allowed at each stage were put in place. All work at each stage was then listed on the boards. The Project Board agreed priorities to be released in the next 3 months.

To break the work down into smaller units that could be delivered more quickly, the concept of Minimum Marketable Features (MMF) was adopted. This is a chunk of functionality that

delivers a subset of the customer's requirements that is capable of returning value to the customer when released as an independent entity. These are then broken down into Stories (New Features) and then further into Tasks which are just 'To Do' items.

A. Office layout and work flow

Office layout is a key component of success. In the Toyota production lines there are lights displaying the status of production at any time; the same principle can be applied to software development. Information radiators and kanban boards were placed all around the work space to ensure that progress on a project was completely transparent and available for all to see. (Fig. 1) This enabled team members to be self managing.

The strategic direction and prioritisation of work was still set by the Project Board and the Steering Group, but the delivery team now had a much clearer idea of their capacity and current work in progress. In the Digi-Hub project, 2 kanban boards (A,B) and 4 information radiators (C,D.E,F) were used and positioned as shown in the diagram below. The layout of the boards evolved as the projects and staff understanding progressed.

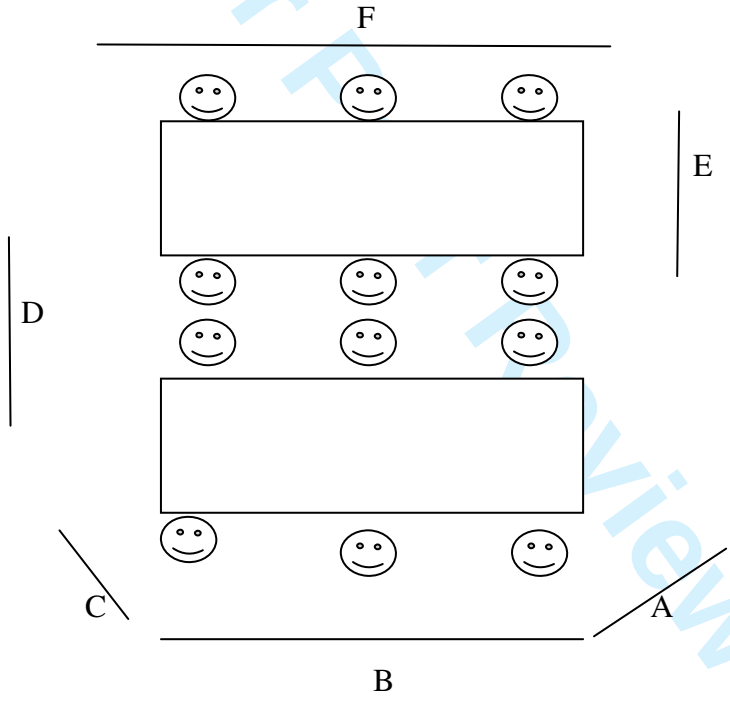

Fig. 1. Layout of Kanban Boards and Information Radiators

It is important that work flows are kept as stable as possible. This is because sudden peaks and troughs of work are disruptive and will damage productivity. It is therefore necessary to try and influence the upstream work flows as much as possible. This information is captured on kanban board A: the ideation pipeline (Fig. 2).

Proposed Ideas	Ready for decomposition	Decomposed Engineering Ready	Released

Waste

On hold

Fig. 2. Kanban Board A: Ideation Pipeline

Any ideas or potential work from customers were recorded on a card and retained in Proposed Ideas in case they trigger suggestions from the team. Any work which is abandoned or has its priority changed is also recorded. Once the ideas have been clarified and broken down into small deliverable units then they are ready to be 'pulled' to the next board when the team has capacity to work on them. This kanban board B (Fig. 3) tracks the progress of Minimum Marketable Features (MMF), Stories and Tasks:

Goal	Objective	MMF	Engineering Ready	Dev. Ready	Dev.	Dev. Complete	QA Ready	Engineering Complete	UAT	Release Ready

Work stream 1

Work stream 2

Work stream 3

Work stream 4

Expedite

Fig. 3. Kanban Board B: (Development Phase)

If there are problems in Development (Dev) these will quickly become apparent as they will reach their Work In Progress (WIP) limit of 4 Minimum Marketable Features (MMF) and become a visible bottleneck.

B. Daily Stand Up

The daily stand up lasts for about 15 minutes, and normally starts about 10.15 a.m. each morning. It is carried out with all team members standing in front of kanban board B which tracks the Development Phase, see above diagram (Fig. 3). This is because this is where the bulk of the work is carried out. The daily stand up is vital for the operation of the lean system. It is essential to facilitate the identification and removal of blockages and bottlenecks, and update the status and prioritisation of work items.

The structure of the daily stand up rhythm is given on information radiator C (Fig. 4). Firstly, everyone checks to ensure their work status is correctly displayed. Secondly, anyone who is 'blocked', unable to progress due to something outside their control reports this, and appropriate action is decided to remove the obstruction. Thirdly, any clusters of cards indicating a bottleneck are noted and the people reorganise to alleviate this. Lastly the work is reviewed to see if priorities have changed or it the work flow can be improved. There is an

Expedite work flow which can be used in exceptional circumstances to accommodate urgent items or if there is a high priority late change.

There is no need or time in a large team for an individual report from each person. It is more effective to just flag problems to be resolved. The kanban boards make it clear to all the team the exact status of progress, blockages, bottlenecks and it also signals possible future issues to prepare for. This shared information enables the team to self organize to ensure the work flows smoothly. The different coloured cards used are listed below (Fig. 4) and enable the exact status of all the team's work to be seen at any time.

Key		Stand up Rhythm (Daily)
Data	Card	1. Is the board correct?
Blocker	= Purple	2. Blockers
Bug (in QA / UAT)	= Pink	3. WIP / Bottlenecks
Fixed delivery	= Small red (attached to card)	4. Work
Expedite	= Small yellow (attached to card)	
New Feature	= Yellow	
Live Defect	= Red (bug found by customer, high priority)	
Technical Story	= Green	
Something missed	= White	
Kaizen	= Blue	

Fig. 4 . Information Radiator C: Kanban Board and Team Performance Indicators

Information Radiator D: Release Notification and daily support process tasks board is used to ensure that any scheduled releases or other operations that have to be carried out during the week are visible. It acts as a reminder and check.

Information Radiator E: Architecture, estimating and breaking down projects
The board was used to record decisions on the architecture for the software, initial estimates and how the work had been broken down into MMFs.

Information Radiator F: Kaizen Board and Technical Debt
The team vote on which items they wish to work on to reduce technical debt or improve the lean system. Reducing technical debt involves work such as improving poor legacy code or making a modification that could increase future productivity. Legacy software can be a severe constraint on current productivity. It is therefore necessary to explicitly reduce any technical debt by allowing time for improvements to be made, even though these are invisible to, and not requested by the customers.

VI. PERFORMANCE DATA ON THE LEAN SOFTWARE SYSTEMS

Lean systems are typically rich in data to enable a team to be self organising and initiate continual improvement. To evaluate the effectiveness of lean software management over 12 months of operation some of the data used by the team is presented here. The team monitored the time taken for work to flow through various parts of the value stream. They also tracked quality and throughput measures.

A. Lead Time

Lead time is the total elapsed time from when a customer requests software to when the finished software is released to the customer. It is measured because it tracks how quickly and reliably software is delivered to customers. Lead Time is defined as the number of working days the work takes measured from kanban board A: Decomposed Engineering Ready to kanban board B: Release Ready.

The main work requests are New Features which are a subset of a Minimum Marketable Feature (MMF). Other work would include Technical Features and Live Defects. Decomposed Engineering Ready means the customer has agreed to proceed and then the Lead Time clock starts. The items are then created for the Engineering Ready input queue. Items are 'pulled' into Engineering Ready only when capacity becomes available. The Lead Time clock stops when User Acceptance Testing is complete and the items have reached Release Ready.

The lead times were analyzed using time series statistical process control charts. These are a powerful tool as they can show trends over time and the amount of variance in a process. A reliable process will have low variance, so a key objective is to continually reduce variance. Fig. 5 illustrates how the top lines, Upper Control Limits, continually decline meaning that lead times are becoming more predictable. To show trends the periods on the charts have been split from November 2008 - March 2009, April – June 2009, and July to October 2009.

The data shows that adopting lean has allowed the mean for lead time to be reduced by 37% or 8.4 days, from 22.8 to 14.4 working days from November 2008 to October 2009.

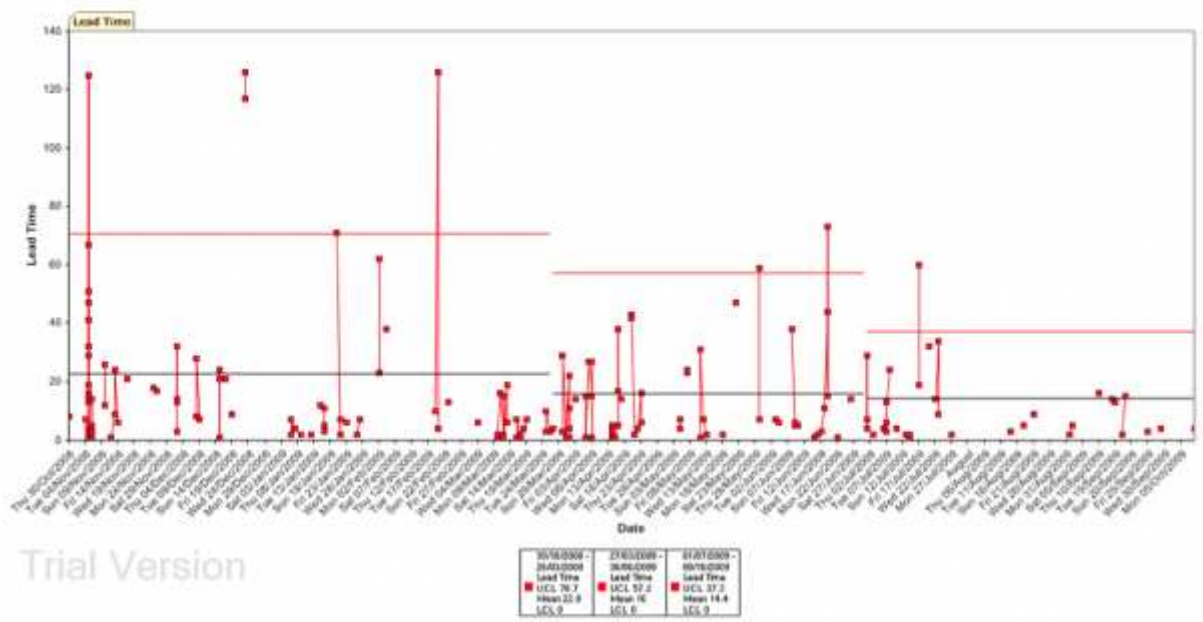

Fig. 5. Variance of Mean Lead Time

The 37% drop in lead time does demonstrate improved ability to respond to the needs of the business. The spread of variation has also dropped significantly shown by the upper control limit in Fig. 5 reducing 47% from 70.7 to 37.3 working days. This means that now the team is delivering new functionality faster and with considerably more predictability.

Projects are broken down into Minimum Marketable Features (MMF), which are then broken down into New Features, so this is tracking when customers first start to receive New Features and MMFs, rather than when the entire project is complete and delivered. Lead time was reduced by 37% and variation in delivery time by 47% in 12 months. This enabled the business to receive new software faster and with more predictability.

B. Development Time

The Development Time measure gives insight into the efficiency of development. This portion of the value stream was directly under the team's control and not subject to delays from upstream, downstream or 3rd parties. It does not include Engineering Ready, QA or related queuing times. Development time is recorded in working days, from kanban board B stages: Dev. Ready to Dev. Complete. The work units are either Stories or Tasks, which can be either standalone or part of an MMF.

The periods on the statistical process control chart (Fig. 6) has been split from February – March 2009, April - June 2009, and July - October 2009 to show trends. The results show the mean development time for all the work completed has reduced 73% from a mean of 9.2 to 2.5 working days over the 9 months.

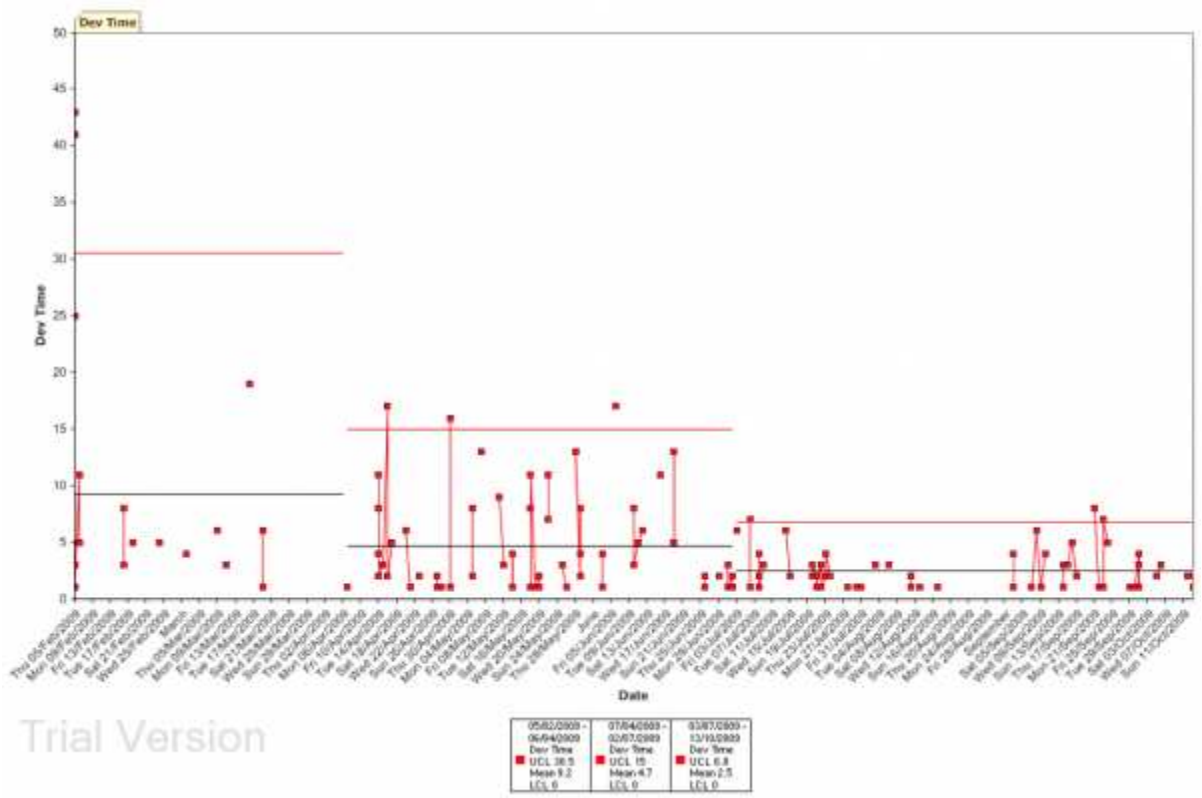

Fig. 6. Variance of Mean Development Time

The spread of variation fell by 78% from 30.5 days to 6.8 days in the 9 months that data was recorded indicating improved predictability of delivery.

C. Throughput

Throughput is defined as the number of items released to customers. An upward trend would be expected as the code base was decoupled, projects were broken into the smaller units of

MMFs, Stories and Tasks, and cycle time was reduced. The chart below (Fig. 7) shows the number of releases per month from November 2007 – October 2009. There was a release freeze in February hence the drop for that month.

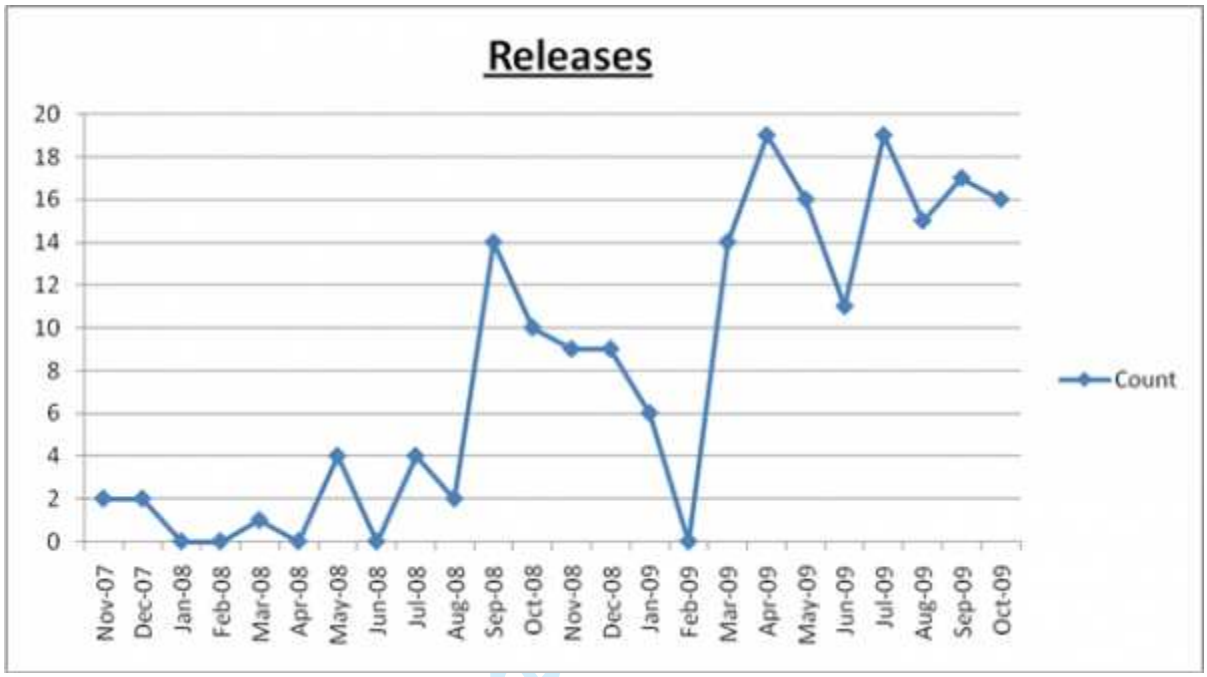

Fig. 7. Release Frequency

The Releases chart does not show how much value is being delivered but it does show an 8 fold increase in releases from 2 in November 2007 to 16 in October 2009. This indicates an improvement in configuration management discipline and capability. The more frequent releases reduce both technical and market risk by allowing customers to evaluate tangible product rather than just progress reports.

D. Live Defects per Week

Live Defects are bugs reported by customers. It is vital that the reduction in lead and development times and the increase in throughput are not at the expense of quality. Live Defects are recorded on Red kanban cards and added to the Dev. (Development) stage of kanban board B. The chart below (Fig. 8) is split between October 2008 – June 2009 and July – September 2009. It shows that the mean number of bugs reported by customers each week fell by 24% from 2.9 to 2.2.

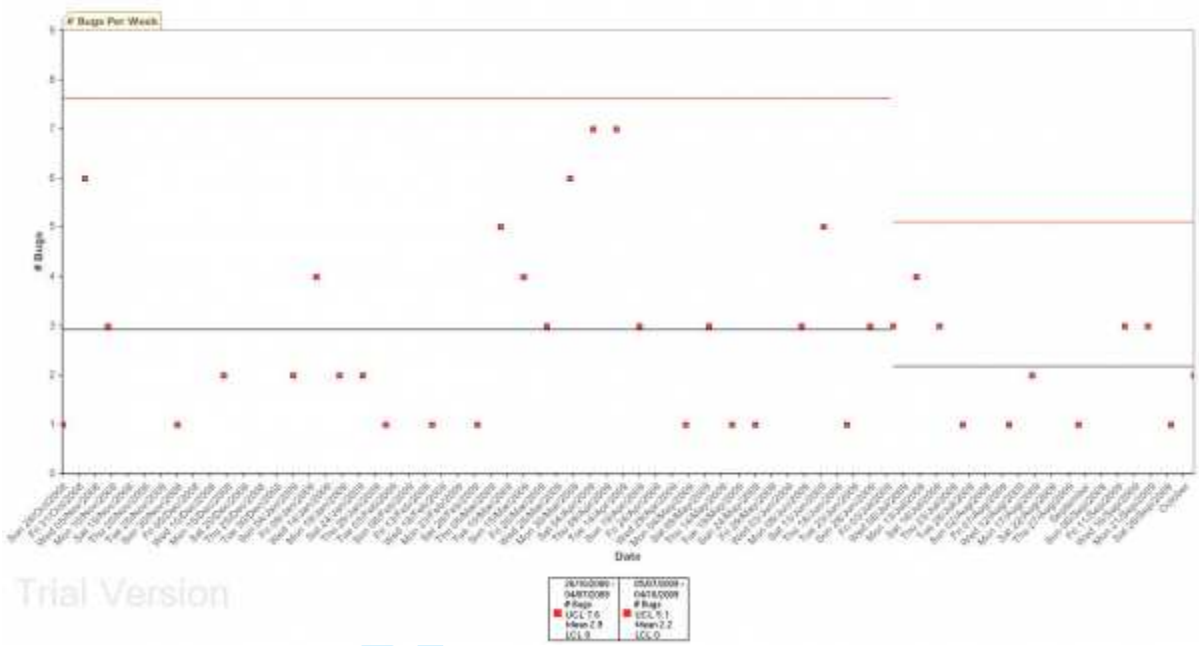

Fig. 8. Variance in Live Defects Reported per Week

As the number of bugs reported by customers was low, more data collected over a longer period of time would be preferred. With this caveat, this data does show the rate of defects is predictable and under statistical process control.

The number of bugs open at any one time has fallen by 33% from an Upper Control Limit of 7.6 to 5.1. This indicates they were being fixed more quickly, possibly due to the improving structure of the code base. The necessity of allowing software developers time to improve the quality of their code was often mentioned by the team, as a big factor in the improved bug rates. As legacy issues (Technical Stories) were resolved the bug rate had fallen, so allowing more Customer Stories to be completed. So while teams are customer focused and responsive to customer needs, they need to pay down any technical debt to increase their productivity levels.

E. Continuous Improvement

The daily stand up is concerned to identify and remove anything that is preventing progress. To do this 'blockers' are actively identified, assigned, tracked, escalated and removed. This is a mechanism for making continuous improvement routine. Evidence of the effectiveness of this is shown below in a statistical process control chart (Fig. 9). The periods on the chart have been split from September 2008 - March 2009; April – June 2009, and from July - October 2009.

Over the 12 months the number of working days items were blocked was reduced by 81% from a mean of 25.8 days to 4.9 days. The outlier in 2008 was a result of waiting for a 3rd party to complete their work (a special cause). This is powerful data to use when discussing performance with 3rd parties.

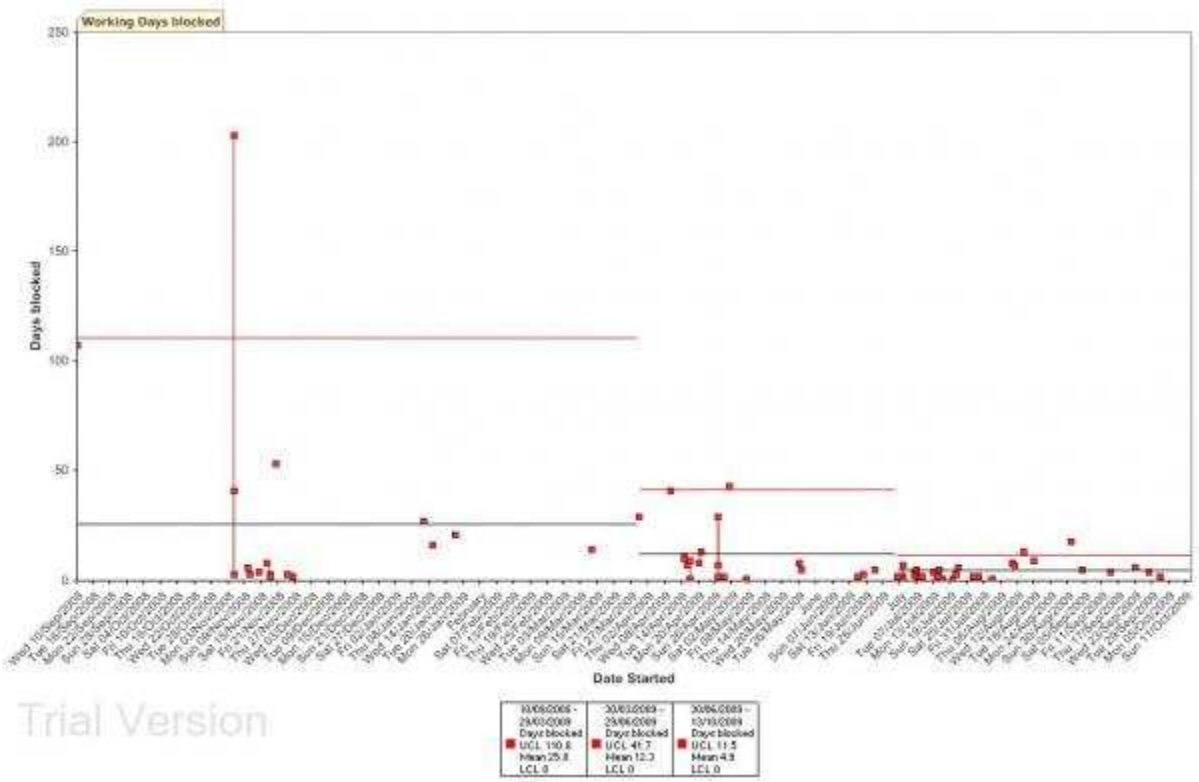

Fig. 9. Continuous Improvement: issues identified and time to resolve

Actively looking for and recording blockers increased the number of blockers raised which is beneficial. These were then being removed at a faster rate by the team as more experience was gained. This data was also used in Retrospectives and quarterly reviews. Re-occurring blockers were investigated and root-cause analysis performed. The 81% reduction in the number of working days that work was 'blocked' is evidence of effective continual improvement.

VII. AGILE v. LEAN

This case study may appear to record an Agile / Scrum like approach where care has been taken to collect data. But the lean approach described here does have significant differences from Agile.

A. Push v Pull

Scrum has time boxed iterations with a fixed release cadence. It is therefore in essence still a push, batch model. Lean uses WIP limits to ensure a team is not overloaded. Work is 'pulled' in when the team has capacity. Teams don't sign up for arbitrary deadlines, as support issues occur that blow things off course. Arbitrary deadlines tend to lead to game playing and poor quality as attempts are made to shoehorn work into the reduced time.

B. Reliance on data

Scrum has 'Inspect and Adapt' in their Retrospectives which is a trailing indicator. The boards are not so important because Scrum they focuses more on the people rather than the work. The Scrum 'stand-up' directs attention to the people and what they did yesterday and what they are doing today.

In contrast, lean enumerates the work not the people. In lean thinking data is seen as a source of empowerment. The team is expected to collect and analyse their own data so they can control and improve their own work. The lean approach uses the kanban boards to expose the problems and expects the team to take action. This is a leading indicator. The lean 'stand-up' focuses on the work and what the team is going to release. Lean uses the data to help the team look up and down stream to enable innovation.

C. Continual Improvement

Scrum uses 'Retrospectives' but benefits from these are largely anecdotal and not quantifiable. The concept of 'velocity' measured as number of feature/story points delivered per iteration is often used. But there is a risk that velocity estimates and number of feature/story points delivered are too subjective and at risk of manipulation. These teams don't put the number of feature/story points delivered under Statistical Process Control (Deming, 2000) so the natural variation in results is not known.

The dearth of data in the Agile community generally is remarkable. A major study seeking evidence to support Agile (Dyba & Dingsoyr, 2008) reported 'the strength of evidence is very low, which makes it difficult to offer specific advice to industry.' (p. 853)

In contrast lean uses 'lead time' which is much harder to game as it records total time from when a customer requested the work to when the finished work was received by the customer. Lean looks at 'blockers' or impediments as 1st Class items to be addressed. Lean seeks data to be used by the team for self management and routine process improvement.

D. Multi skilling / collaboration

With Agile, the Scrum Master has the 'impediment list' but it is not the team's responsibility. With lean because of the WIP limits and the visibility of the kanban board, the staff can't 'cherry pick' what they would like to work on if they are blocked. The must help with the bottlenecks and items blocking work. The focus of the daily 'stand ups' is on the flow of work not on the individual reports and performances. Staff, regardless of skill, must help with the bottlenecks. The objective is to deliver value as quickly as possible to the customer.

VIII. DISCUSSION

The data shows that over 12 months this 9 person software team has achieved better quality, predictability and throughput. The use of Statistical Process Control indicates a team that is managing process improvement in a quantitative way in line with CMMI level 4.

The main use of the kanban boards was to give the team a clear, current picture of where exactly they were. Customers rarely visited the kanban boards and did not attend the daily stand up meetings as they both contained more detail than customers needed and would be time consuming. Also a team may be working for several customers so much of the meeting would not be of interest. Customers were encouraged to attend but overall it was not feasible for them to be closely involved with the kanban boards.

Managers can see the status of projects from the kanban boards, but this was not the boards' primary function. Managers may ask questions when the kanban board shows several red notes, denoting live defects, or if excessive 'waste' was being recorded, but otherwise they

would not be closely involved with the kanban board. The regular meetings for the Steering Group or Project Committee would not be held at the kanban boards.

The boards evolved with changes to layout being made roughly on a weekly basis. They were a living tool reflecting both the evolving project and the changing understanding of the team of their work. The social value of the daily stand up cannot be underestimated. The need to daily share your progress and problems with your peers is a powerful motivator and source of discipline.

The lean approach can handle big, complex projects. The constraint is the ability of the human mind to handle complexity. So any large project can be broken up into smaller projects. A master kanban Board can then be used to record and summarise the progress of all the smaller projects. There is no need or benefit from one gigantic kanban board recording everything. Scale or complexity of projects was not observed to be a problem.

The lean and kanban approach worked better when it was supported by:

1. A stable, experienced team with low staff turnover.
2. A project manager who knew the skills and abilities of their people.
3. Good source control software to manage different versions of the software product.
4. Bug tracking software.
5. Well automated release and deployment processes for software.
6. Automated unit and coverage testing that reduces live bugs.

Lean and kanban is therefore not a substitute for good software engineering tools and practice, but supplies information that enables a team to self-manage. The transparency kanban boards provide does assist software development teams to rapidly raise their maturity levels.

Lead time could potentially be further reduced as customers were batching work for User Acceptance Testing (UAT), which was slowing the process before Release Ready. An investigation of how to make UAT easier for customers could be beneficial. The other area that could possibly be improved is in the 'fuzzy front end' (Smith & Reinertsen, 1998). The time from when an idea is logged on kanban board A: Proposed Ideas to Decomposed Engineering Ready is currently not recorded and this could be significant.

The concept of a fixed 'heart beat' of production or Takt time used in lean manufacturing to ensure that the rate of production meets the rate of demand was not used. Takt time does not work with software or services because each unit of work is different as are the abilities of each software developer. It would therefore be impossible to schedule production in this way. The solution for software is to have incremental development with frequent releases. This provides rhythm to production and ensures a direct connection with the customer. When this is combined with the daily stand ups, the work effort can be directed and controlled as the customer requires.

IX. CONCLUSION

The data presented here is felt to objectively record the behaviour of this software team. No significant bias was detected from variations in: project size, project complexity, governance, team composition, and engineering practices.

The lean ideas of: low work in progress, statistical process control, pulling work when capacity is available, self managing teams empowered by data, daily routine of continual improvement, and multi skilling did help to achieve the results recorded.

Lean also provided a framework that encouraged other beneficial improvements, such as rewriting parts of the legacy code, developing team skills and reducing staff turnover. The quantitative and qualitative evidence indicates that the lean ideas do provide a remarkably effective way to manage software projects.

A. The data recorded the following benefits

1. The mean lead time to deliver software to customers was improved by 37% (8.4 working days) with a 47% gain in predictability.
2. Development time was reduced by 73% from a mean of 9.2 to 2.5 working days. Predictability of delivery improved by 78% from 30.5 to 6.8 days.
3. Product releases increased 8 fold which reduced both technical and market risk.
4. Live defects per week reported by customers fell by 24% from 2.9 to 2.2. The total number of defects open at any one time fell by 33% showing problems were being fixed more quickly.
5. Continual improvement was effective with an 81% drop in the number of working days items were blocked.
6. High staff morale, motivation and pleasure in their work were observed.

B. The costs or negatives of the lean approach are the following

1. While capital costs were trivial there is a need for considerable space to display the kanban and information boards. Organizations with offices designed around a corporate 'look' may not welcome walls of Post-it notes.
2. If the organization has a heavy plan driven process with standardised corporate reporting on projects, then this emergent approach will not fit easily. Lean handles risk by being highly transparent, reducing WIP, breaking projects into small parts and frequent deliveries. Lean does not work well with targets, milestones, Gantt Charts and Traffic Light reporting methods.
3. To truly deliver value to customers will require the development team to proactively move upstream to work with customers to define and analyse their problems and then work downstream after release to see if business value was actually created. Organizations may feel the IT teams are going beyond their remit.
4. A self managing team can be challenging as managers need to move towards a facilitating role which they may feel uncomfortable with. Staff may not be used to being encouraged to indentify problems or having to multi skill.

C. Future developments

It is clear that there is much potential in the lean and kanban approach to managing software projects. It shows how the Agile approach can be improved and the quantitative behaviours of CMMI level 4 implemented. The two areas which appear to offer the most scope for further improvement are:

1. Move to also measuring the 'Touch Time' when value is added to the work, to supplement elapsed time in days. This may well suggest ways to further improve the process.

2. To discover how to move further upstream in the organization to ensure a total systems view is taken so that the correct IT projects are identified in the first place.

REFERENCES

Anderson, David (2003) Agile Management for Software Engineering –Applying the Theory of Constraints for Business Results, Prentice Hall

BBC (2010), BBC Worldwide website www.bbcworldwide.com/about-us.aspx

Chrissis, M.B,. Konrad, M. & Shrum, S. *CMMI Guidelines for Process Integration and Product Improvement,* Boston, Addison-Wesley, 2004

Cockburn, A, *Agile Software Development,* Boston, Addison-Wesley, 2002

Deming, W.E. (2000) *Out of the Crisis*, The MIT Press, Cambridge, Mass. (first published 1982)

Dyba, T. & Dingsoyr, T. (2008) Empirical studies of agile software development: a systematic review. Information and Software Technology, 50, 833-859

Easterbrook, S., Singer, J., Storey, M., & Damian, D. (2008) Selecting Empirical Methods for Software Engineering Research, Capt. 11 in Guide to Advanced Empirical Software Engineering, Shull, F., Singer, & J. Day, (Eds.), Springer, London.

Eisenhardt, K. M. and Graebner, M. E., "Theory Building from Cases: Opportunities and Challenges", *Academy of Management Journal,* Vol. 50, No. 1, pp. 25-32, 2007

Gilb, T. (1988) *Principles of Software Engineering Management*, Addison-Wesley, Wokingham, U.K.

Hamilton, T. "A Lean Software Engineering System for the Department of Defense", *Massachusetts Institute of Technology*, MSc thesis, 1999

Hopp, W.J. & Spearman, M.L., (2001) Factory Physics, McGraw-Hill, New York

Hou, A.C. "Toward Lean / Software System Development: Evaluation of Selected Complex Electronic System Development Methodologies", *Report – Lean 95-01, Lean Aircraft Initiative, Center for Technology, Policy and Industrial Development, Massachusetts Institute of Technology, 1995*

Jankowicz, A.D. *Business Research Projects*, Chapman & Hall, London, 1991

Ladas, C. (2008) Scrumban: Essays on Kanban Systems for Lean Software Development, Modus Cooperandi Press, Seattle, WA.

Middleton, P. (1993) *Just-In-Time Software Development*, Proc. 2nd Intl. Conf. on Achieving Software Quality in Software, Consorzio Quality I.E.I.-CNR, Venice, Italy, 18-20 October, pp. 49-56

Middleton, P., Flaxel, A. & Cookson, A., (2005) *Lean Software Management Case study: Timberline Inc.,* Lecture Notes in Computer Science, Springer-Verlag

Middleton, P. & Sutton, J. (2005) Lean Software Strategies, Productivity Press, New York

Morgan, T. "Lean Manufacturing Techniques Applied to Software Development", *Massachusetts Institute of Technology*, MSc thesis, 1998

Ohno, T. (1988) Toyota Production System: beyond large-scale production, Productivity Press, Portland, Oregon

Poppendieck, M & Poppendieck, T. (2003) Lean Software Development: an Agile Toolkit, Addison Wesley, Boston

Seaman, C. B. (1999) Qualitative Methods in Empirical Studies of Software Engineering, IEEE Trans. Software Engineering, Vol. 25, No. 4, pp. 557-572

Seddon, J. *Freedom from Command & Control*, Buckingham, U.K. Vanguard Education, 2005

Shalloway, A., Beaver, G. &Trott, J (2010) Lean-Agile Software Development: Achieving Enterprise Agility, Pearson Education, Boston, MA.

Shingo, S. (1981) A Study of the Toyota Production System, Productivity Press, Portland, Oregon

Smith, P.G. & Reinertsen, D.G. (1998) *Developing Products in Half the Time*, Wiley, New York

Tierney, J. (1993) Eradicating mistakes from your software process through Poka Yoke, Proc. 6th Intl. Software Quality Week, Software Research Institute, San Francisco, C.A. pp.300-307

Walsham, G. (1995) "The Emergence of Interpretivism in IS Research", Information Systems Research, Vol. 6, No. 4, pp. 376-394

Willeke, E., Anderson, D. & Landes, E. (2009) Proceedings of Lean & Kanban 2009 Miami, Wordclay, Bloomington, Indiana

Womack, J.P. and Jones, D.T. (1997) *Lean Thinking*, Touchstone Books, London

Womack, J.P., Jones, D.T. and Ross, D. (1990) *The Machine that Changed the World,* Rawson Associates, New York

Wynekoop, J.L. and Russo, N.L. "Studying System Development Methodologies: an examination of research results", *Information Systems Journal*, Vol. 7, No. 1, pp. 47-65, 1997