Introducing the Time In State InSITe Visualization

Larry Maccherone
Larry@Maccherone.com
Carnegie Mellon University
Rally Software

Abstract

This paper introduces a new visualization type, the Time In State InSITe visualization, and a particular instantiation, the Time in Process (TIP) Chart. The paper demonstrates their value in providing insight into software and systems design work. It is intended to be used in situations where misapplications of control limits often occur and provides guidance about the problems with control limit calculations used in this domain to motivate the reduction in misapplication. Lastly, it introduces a framework (named ODIM) intended to guide metrics choices and avoid misapplications like that which has happened with the use of control limit calculations.

1. Introduction

> *No data hath meaning apart from their context. ~unknown*

Various forms of behavior (control) charts have long been used to provide insight about manufacturing processes. The lean software and systems movement has adopted many of the practices of lean manufacturing, in particular successfully borrowing the concept of kanban and adapting it to fit software and systems design work. Unfortunately, for every set of concepts that can be cleanly adapted to a different environment, there are always some practices that should not be adopted. Worse still, sometimes their misapplication in the new environment can do harm. The commonly-used practices for calculation of control limits on control charts for lead times of software and systems design knowledge work is one such case. It provides limited value at best, and as commonly misapplied or misinterpreted, can lead to poor decisions.

This paper introduces an alternative to control limits that is more suited to software and systems design work. In section 2, I briefly introduce this Time In State InSITe approach to visualizing such data along with the Time in Process (TIP) Chart instantiation of this analysis which is focused on the analysis of lead times. In section 3, I explain the various means of calculating control limits and why each is not ideal for this domain. In section 4, I come back to the TIP chart with a deeper explanation of the insight it provides and instantiations of the Time In State InSITe visualization other than the TIP Chart. Section 5 introduces a framework for making future metrics and visualization choices that should help avoid false starts like that experienced by the lean software and systems design

domain in the case of control limit calculations. Future work and conclusion are sections 6 and 7 respectively.

2. Introducing Time In State InSITe analysis and the TIP Chart

Data visualization is like photography. Impact is a function of perspective, illumination, and focus. ~Larry Maccherone[1]

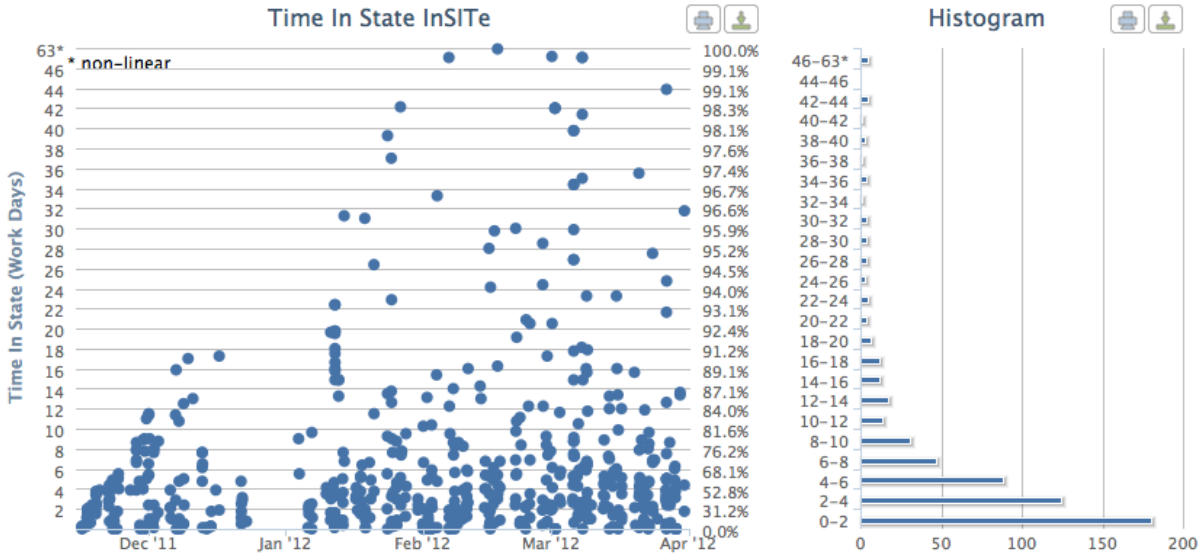


Figure 1 *- TIP Chart for lead time (In-Progress to Accepted) (580 data points)*

Figure 1 is an example of the proposed visualization. The general approach is called Time In State InSITe analysis. Each Time In State InSITe visualization is calculated with the "State" being defined by a particular predicate. This predicate can be used to measure any logical state: blocked time, queue time, in-progress/process time, etc.

To most closely match the visualizations that I'm proposing to replace, namely control charts of lead times, the predicate will correspond to some definition of lead time which means that it is measuring Time In Progress or Time In Process, so I am giving the nickname of TIP Chart to this particular form of Time In State InSITe visualization. Figure 1 was calculated using the "in state" predicate of ("In-progress" <= KanbanColumn < "Accepted"). It is the lead time for Rally Software's own development teams of the 580 most recently completed work items.

Visualization features include:

[1] Inspired by "Visualizing data is like photography. Instead of starting with a blank canvas, you manipulate the lens used to present the data from a certain angle." - Paul Butler (Facebook)

- X-axis of the scatter (left) part of the visualization is linear unbroken calendar time. A point appears above the position on the x-axis for the moment that it last satisfies the "in state" predicate. For Figure 1, the date of the last time a work item transitioned out of state (which is usually when it transitioned past "In-progress") is where it will appear along the x-axis. Items that are currently still in process will not appear on the chart.
- The far left y-axis indicates the number of cumulative work days that the work item spent in that state. A work item will appear on the y-axis according to how many work days it was "in state". Work days is calculated by counting the work hours that a work item spent "in state" and dividing by the number of work hours in this particular definition of a work day. So a work item that was started at 4pm and completed at 10am the next day would only have been "in state" for two work hours or 2/8 (0.25) of a "nine to five" work day. Weekends and other non-workdays are excluded. A work item can transition into and out of the "state" multiple times during its life. The cumulative time spent "in state" is what matters.
- The y-axis to the right of the scatter area indicates the percentile coverage. So, according to Figure 1, 31.2% of all work items completed in two days or less. The 70% coverage line for this set of work items is somewhere between six and eight days. 99.1% of all work items completed in 44 days.
- So that outliers do not compress the distribution, the top band of the visualization is optionally non-linear. A heuristic is employed to detect significant increases in the gaps between points in the distribution as you move to greater time-in-state values. When detected, the remaining points are collapsed in the last band. We intend to shade this band to better indicate its non-linearity.
- The right side of the visualization is a histogram showing the distribution of the time in state. The horizontal axis on the histogram indicates the count of work items; 180 work items were completed in 0-2 days; 125 in 2-4 days; etc.
- The vertical axis labeling each bar in the histogram exactly aligns with the data and scaling on the scatter chart, so if you counted the points below the first horizontal line on the scatter chart, it would add up to the count in the lowest bar on the histogram.

3. Control limit miscalculation and limited usefulness

> *Control is an illusion, you infantile egomaniac. Nobody knows what's gonna happen next: not on a freeway, not in an airplane, not inside our own bodies and certainly not on a racetrack with 40 other infantile egomaniacs. ~Days of Thunder*

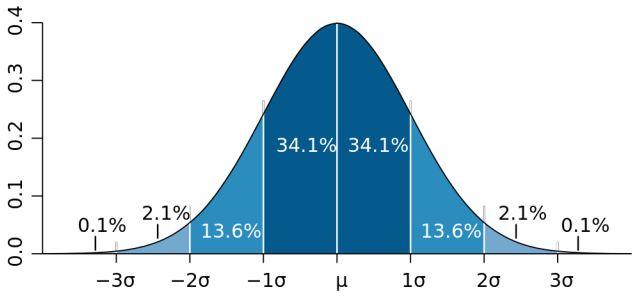
It is common practice in the lean software and systems world to analyze lead time[2] data by

[2] "Lead time" is used to characterize the time that an individual work item spends in process. Unqualified, it generally refers to the full lifecycle of the work item but it is often qualified to refer to a portion of the overall lifecycle. This paper analyzes a lead time data set calculated from the moment work was started by developers

displaying them on a scatter chart and adding control limits. The displaying of the data in a scatter chart is noncontroversial. However, there are several ways to calculate control limits, none of which have been demonstrated to be efficaciously applicable to software and systems design work.

3.1 Using mean and standard deviation (Gaussian) to calculate control limits

A normal distribution is not required for you to place your data on a behavior chart[3]. However, it is misleading to annotate that chart with limits that are calculated using the assumption of a normal (Gaussian) distribution.



The naive approach for calculating control limits assumes a Gaussian distribution and uses +/- 2 or +/- 3 standard deviations from the mean. The assumption is that everything to the left of the +2 sigma line would be 97.7% of all possible outcomes. 99.9% of all outcomes are to the left of the +3 sigma line. Incorrectly assuming that the distribution is Gaussian and calculating standard deviation and mean, would indicate to the casual user that 97.7% should occur in 25.4 days. However, since we have sufficient data points, we can identify the actual value necessary to achieve 97.7 percentile coverage -- 38.5 days. At 25.4 days, only 94.1% (almost 6% below 100%) of all occurrences are covered, so if you calculated the risk of paying reimbursement fee to customers whenever you went over your service level agreement (SLA) of 25 days, assuming you'd only miss it in 2% of the cases, you'd have miscalculated your risk of exposure by almost a factor of 3.

Further, the data set analyzed above comes from a mature team that has been using kanban to provide feedback on the process for almost two years. While others may have a different experience, this data set is one of the more stable sets that I have analyzed. I have seen lead time data sets where the 2 sigma SLA recommendation would be off by a factor of 3 and the risk exposure be off by a factor of 20.

3.2 Shewhart's limit calculation

From the 1890s, control charts relied upon assumptions about the dispersion. In 1942, W. J. Jennett working with Walter A. Shewhart[4] proposed Individual X and XmR chart methods

---

until the moment that it was accepted for release. "Cycle time" is often used by the lean software and systems world to mean what I intend here with the use of "lead time". However, "cycle time" has a different definition in the manufacturing world where it measures the time between individual work items leaving the system. "Lead time" is favored to avoid confusion with the manufacturing definition.

[3] Donald J. Wheeler, 2008-08-05, Quality Digest, http://www.qualitydigest.com/inside/quality-insider-column/do-you-have-leptokurtophobia.html

[4] Donald J. Wheeler, 2010-02-10, Quality Digest, http://www.qualitydigest.com/inside/quality-insider-column/individual-charts-done-right-and-wrong.html

for calculating control limits that simply relied upon the average moving range. In the lean software and systems world, this approach has become widely accepted and is often referenced as "Shewhart's method" although the full extent of "The Shewhart Procedure" encompasses more ground than just the Individual X and XmR charts. The primary advantage claimed for these charts is that they make no assumption about the shape of the distribution. However, this claim only confuses matters for software and systems design applications because the charts themselves are only really applicable when monitoring a process for special cause variation. "When this happens [discovery of a point outside of the limits], the important questions are no longer questions about the process location or the process dispersion[3], but rather questions about what is causing this process to change. Discovery of the assignable cause is paramount."

So, the intended purpose of this chart is the rapid identification of dramatic variation in process output that requires attention -- at the very least, to identify the variation as an anomaly, but perhaps to adjust the system to avoid similar future variation. However, the tight monitoring for process changes is generally not a critical factor for success of software and systems design projects like it is in manufacturing where it is best to prevent the system from drifting far enough to result in defective product. Yet, when a user who has no background to evaluate the applicability to his situation sees data points outside of a "control limit", he is going to assume that is not a desirable situation. The chart's mere existence in an ALM tool may alter behavior whether intended or not.

Moreover, this method's biggest claim to fame might actually be its biggest weakness. In fact, when applied to dispersions commonly found in software and systems design work, it can give even more misleading insight than the naive approach identified in section 2.1. For the data set in Figure 1, Shewhart's method recommends an upper limit of only a little over 7.0 which actually corresponds to a percentile coverage of only 72% leaving 28% of the data points in our data set uncovered. When used inappropriately to inform SLA decisions, it could lead to a risk exposure that is 14 times greater than assumed by the user.

I must note that the 2-point moving average used to calculate the Shewhart upper limit of 7.0 is not the only valid alternative in Shewhart and Wheeler's toolbox (although it is the default for many implementations). Using a day's worth of points as the sub-group and/or the median plus median moving range alternatives are likely to improve the prediction.

3.3 Assuming an asymmetric distribution

It is possible to assume an asymmetric distribution and calculate asymmetric control limits with the same calculated probability coverage as the +/- 2 or +/- 3 standard deviations of a Gaussian distribution. The distributions are generally Poisson whose probability distribution function is known. It can also be modeled fairly well with a log-normal distribution. In fact, for the lead time data set visualized in Figure 1, a log-normal assumption does a fairly good

job of predicting the actual percentile coverage. The +1 sigma (84.1%) is predicted to be 11.3 compared to the actual of 12.9 and the +2 sigma line (97.7%) is predicted at 28.7 which is closer to the actual 38.5 than the naive Gaussian assumption of 25.4.

However, there are problems even with this approach.

1. It's still not perfect. The exposure risk of an SLA at 28.7 is still 125% greater than predicted by the model.
2. It is both harder to comprehend as well as model asymmetric distributions. As of this writing, I do not believe that any tool vendor targeting the software and systems design space has put such a control limit calculation into their product.
3. The distribution is only *generally* Poisson. Actual distributions for data that I have observed often exhibit fatter tails than the models would predict and often multiple humps indicating a bi-modal or multi-modal situation. The sample data set in Figure 1 exhibits both. Careful cleaning of a data set from a mature team with significant stability can result in a decent fit. However, to responsibly embed such a chart into an ALM tool would require that significant logic be put into place to confirm the applicability.

With my background, it would have been an interesting problem to design such logic and include it in the Rally Software ALM tool as I could have done from my position as the Analytics Product Owner. However, that would have been wasteful because such a model does not provide any additional value and would add significant cost of implementation and comprehension complexity.

Rather, I am now proposing the Time In State InSITe visualization and the TIP Chart instantiation for the lean software and systems design domain that is both easier to comprehend as well as capable of providing more useful insight.

4. Usage and insight

> *A point of view can be a dangerous luxury when substituted for insight and understanding. ~Marshall McLuhan*

4.1 General insight from our example

The most obvious pattern in the example visualization in Figure 1 is the lull over the holidays. The duration of the lull was increased by the timing of a company-wide hack-a-thon just prior to Christmas.

More interesting is the lower level of dispersion prior to the holiday. During that time, almost all of the development effort at Rally was focused on a due-date critical delivery of our new Rally Portfolio Manager (RPM) product. All long-term work was put aside temporarily to focus on this goal. A development organization benefits from opportunities for

experimentation and research, but when the cost of delay is high, it is also useful to focus. This chart helps demonstrate the dramatic effect that due-date driven focus can have. The success of lowering lead-time in such a situation should factor into future decision making when other tight market opportunity windows become apparent.

The temporary putting aside of longer term work also shows up in the flurry of work completed a week or so after returning from break. There was work that was close to completion before pivoting to RPM. Not only were carrying costs incurred, it is clear that there were context switching costs because there is a week or two delay before work started rolling out again.

The weekend lulls are also obvious (moreso in the context of a larger visualization than is possible in this paper) and a careful examination shows that more work is completed on Mondays. Is this work that was completed on the weekend and only accepted on Monday or is it a function of paperwork tracking work completion is more often updated at Monday morning meetings?

4.2 The value in outlier analysis

Another unfortunate outcome of the display of control limits is that it tends to send the message (whether intentional or not) that work items that fall outside of the limits are best prevented. When David J. Anderson has mentioned the reduction in variability as a potential element in the recipe for Kanban success[5], it has always been as the last item and, he "believe[s] that you fix other things first and let variability reduce over time."

I would state it even more strongly, I have yet to see a kanban usage situation in software and systems design work, where tight control of lead times is the highest priority for the team. It is often couched in terms of improved predictability when advocated by management or agile coaching. This shifts the focus away from the diagnostic benefit of understanding what is really going on, to one of simply improving the metric. One trick to improve this metric is to artificially break the work down. In this circumstance, the metric itself becomes meaningless because the artificial breakdown means that individual work items no longer deliver independent value. The predictable delivery of increments of value is what matters, not the delivery of artificially broken down work. Ironically, emphasis on this metric might have the opposite effect on actual predictability because the true lead time is not available for feedback, having been hidden by the artificial breakdown of work.

Enabling a discussion around the details of the work items represented by the points at the top of the scatter chart can lead to insight about how to improve flow and increase value delivery. All the more so when the visualization is interactive and allows immediate drill down into the history and details of the work item. The focus is then on the improvement of the team's ability to deliver value rather than on improving the metric.

[5] David. J. Anderson, 2008, http://agilemanagement.net/index.php/Blog/an_alternative_recipe_for_success/

There is value in your outliers. It's a fine line, but the trick is to focus on understand what caused them without creating a culture afraid to ever show them.

4.3 InSITe visualizations other than the TIP Chart

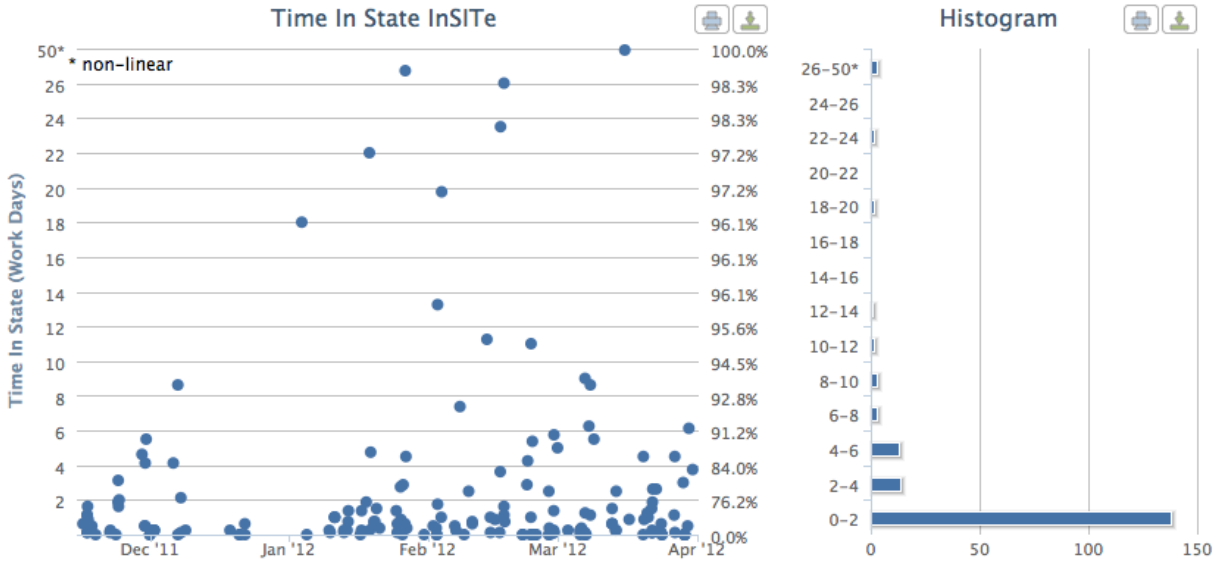


Figure 2 *- Time In State InSITe Chart for blocked time (181 data points)*

Figure 2 covers a subset of the data points in Figure 1, however, these are the 181 out of 580 work items that were marked as blocked at some point in their lifecycle and the time measured is only the time spent in blocked.

Interestingly, the top outlier in this set is at 50 days which tells us that a significant portion of the max time of 63 days for the highest time in state value in Figure 1 is blocked time. That gives us quantitative feedback that when work is blocked, it has a much higher likelihood of resulting in longer lead times. Because it matches our intuition, that is not terribly interesting. However, this chart allows us to drill down to the details about the blocked items and examine the reasons. Doing so in this case reveals that cross-team and vendor dependency issues account for almost all work items that were blocked at least five days.

It is up to the teams to decide what to do with this information. It may be that the cost to mitigate cross-team dependencies exceeds the value that would provide, but this visualization at least gives you the quantitative insight you would need to start to make an informed decision in this regard.

I suspect that the minor hump centered around 13 days in figure 1 could be controlled for by factoring issues that were blocked into another population but the analysis to determine this conclusively is beyond the scope of this paper. I mention it to illustrate how the use of

this visualization and altering the definition of "in state" can lead to actionable insight.

5. The ODIM framework for making metric and visualization choices

better ***Measurement*** **=>** *better* ***Insight*** **=>** *better* ***Decisions*** **=>** *better* ***Outcomes***

I offer this framework for making measurement and visualization choices that I believe will help the lean software and systems domain avoid costly false starts like the one experienced with the misapplication of control limit calculations:

- Effective **measurement** and visualization provide better **insight**.
- Better insight leads to better **decisions**.
- Better decisions lead to better **outcomes**/results.

The LSSC community seems to have started with a metric (control limits) borrowed from the manufacturing domain without first considering its fitness for purpose. Rather, when thinking about metrics, it's best to start with the outcomes and work your way back to the best measurement and visualization to achieve those outcomes. Like when reviewing a Kanban board, it's best to start at the right and work your way left, so I give this framework the acronym **ODIM** rather than MIDO. The reversal of the letters in this acronym inspired the GNU-like acronym for the InSITe visualization (InSITe rather than TIS-I).

5.1 Example walkthrough

Outcomes
Setting of SLAs is anticipated to have several beneficial outcomes. Customers can better anticipate delivery which should lead to happier customers and more sales. Contract negotiation informed by accurate risk assessment is much less likely to result in costly outcomes for the organization.

Decision
The decision, in this particular case, is, "What number of days would I feel 'safe' telling my customers we will complete this work in?"

Insight
The next step is to then try to figure out what insight I need to make the best possible decisions around SLAs. To me, it seems like the most useful thing when making a decision like that is to know the distribution of how long other similar work took. That will tell you what percentage fall outside of a proposed SLA settings. I can then think about how risky that is.

Measurement/Visualization
A percentile coverage scatter chart and histogram like I've proposed seems to ideally provide this insight. Control limit calculations, even ignoring their applicability issues in our

domain, do not seem to help with this particular situation. Further, even if the difficulties of correct calculation are overcome, the ODIM framework analysis of our domain would generally not lead one from desired outcomes back to this particular measurement.

5.2 Discussion

I have previously discussed this chain of dependencies between measurement, insight, decisions, and outcomes but this paper represents its introduction as a framework for making metric system choices. It roughly aligns with the thinking in Vic Basili Goal Question Metric (GQM)[6] and the Practical Software Measurement (PSM)[7] framework which has been standardized in ISO/IEC 15939[8].

However, I believe this particular instantiation of the general idea to be a better fit for the lean and agile communities. Business strategies are implemented one small decision at a time by the folks closest to the work. When a strategy fails to achieve its desired outcome, it is often hard to attribute it to one thing. It is often a series of small decisions. In an agile environment, more and more of these decisions are pushed to those closer to the work... to those with more qualitative insight about the context... to the team, team leadership, and individual team members. While my work at Rally includes providing feedback mechanisms targeted at high-level strategic steering, I believe it is critical to provide context-flexible insight to those making the day-to-day decisions that implement those strategies. Decisions about what work to focus on next; on how much automated testing to put into the codebase; on how to spread the work out over multiple team members and small teams; about when to refactor; about when to cut scope and where.

6. Future work

The TIP Chart is in the process of being embedded as a standard chart in Rally Software's product. The more generalized InSITe visualization is being used to provide insight for the development of future benchmarking and performance index products.

Exploration of more InSITe instantiations other than the TIP Chart and blocked time could prove useful.

Controlling for factors found (cross-team dependency) in the initial analysis described above could prove fruitful especially if it can be shown to have a strong correlation with particular outcomes and costs of mitigating can be discerned.

The InSITe visualization was made possible via the creation of a new analytics engine at Rally Software built upon a high-speed in-memory no-SQL database with a data model

[6] http://en.wikipedia.org/wiki/GQM
[7] http://www.psmsc.com/AboutPSM.asp
[8] http://www.psmsc.com/ISO.asp

highly-optimized for time-series analysis... colloquially referred to as Analytics 2.0 or A2.0 at Rally. Further documentation about this platform is forthcoming. However, the exploration of the possibilities that it opens up is just starting and should provide a wealth of possibilities. Most exciting, we are not holding back the availability of this engine for internal use. Its full power is being made available via a new analytics API so anyone who desires can participate in this exploration.

7. Conclusion

Donald G. Reinertsen has arguably done more than anyone to debunk the myth that the lean software and systems design domain can merely borrow from the successful practices of lean manufacturing[9]. He and others like David Anderson and Alan Shalloway have tried to walk a careful line of balancing the value of credibility that comes from a long history of application in manufacturing and ensuring the applicability of practices. The visualization proposed here attempts to follow their lead in this regard.

[9] Donald G. Reinertsen, 2009, The Principles of Product Development Flow: Second Generation Lean Product Development