Lean Six Sigma Gold

# Urgent Support:
# A Reduction in Urgent Customer Requests Resolution Cycle Time

*Charles Hollingsworth*
*November 2006*

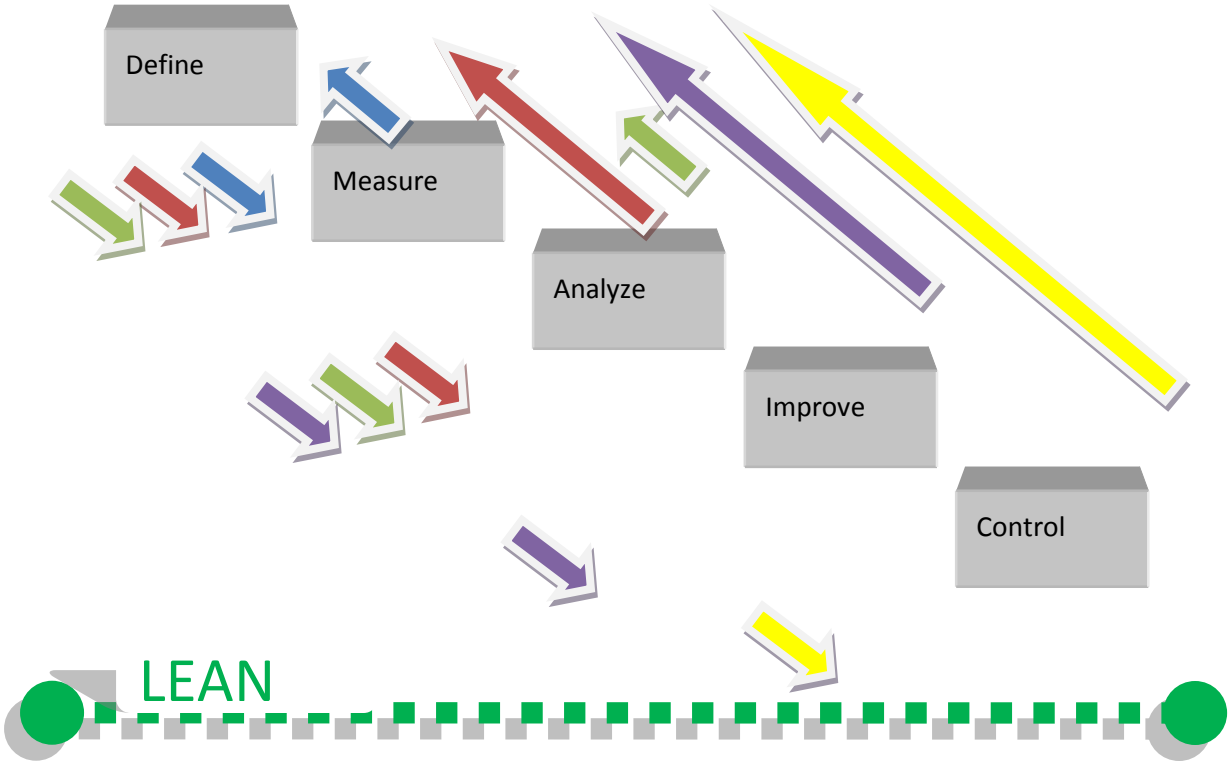
**Abstract**
This is a documentation of the application of Lean and Six Sigma methodologies to the specific process of receiving customer requests and resolving the request. Specifically, the customer requests are software support tickets for online ecommerce applications. The organization is a software development company which exclusively develops the software that is being supported by their *Customer Services* department. This project to reduce cycle time for requests follows the DMAIC process while iteratively applying Lean methodologies within the DMAIC steps to identify constraints and define the appropriate problems to solve. Finally, suggestions are made to control the process in the future and establish requirements for a decision support system to monitor the variation.

I have structured this document according to the DMAIC process to narrate project evolution.

## Contents

## PROJECT FLOW

As with most Lean Six Sigma projects, this specific project was an iterative process. I will explain the final DMAIC sections as they pertain to the eventual value of the project. Firstly, I want to establish a ground work for appreciating the amount of iterative questioning of myself and all those involved in the project, not only for assembling information for the various tools used in the process of accomplishing the objective, but for defining the objective itself.




This diagram is intended to track the cycles of the process. Each color represents a cycle. As the project was finally defined, the iteration continued as new information was discovered during subsequent phases. Lean methodologies were applied throughout, but it drawn here to also express the impact on the DMAIC process.

## DEFINE

More appropriately, *pre-define*, began with an elevator speech to the CFO of the organization. She was interested in Six Sigma and asked me to think of some way to apply the techniques in the organization.

Shortly after, I approached her with an idea to evaluate how we were handling customer support requests in the Client Services department. I knew that any self-initiated effort to improve customer support would be an easy sell as the organization was “suffering” from significant growth in customers over the previous eight months and increasing complexities among the software applications being built and deployed to customers. To combat the significant rise in customer support tickets, the support staff tripled. Still, the perceived support was actually reducing as investment in the function was dramatically increasing. Obviously, there was a disconnect somewhere…

After I received support from the CFO, I met with the Project manager and Director of Engineering to let them know upon what I was embarking. Although my efforts were not going to impact my schedule, I knew that there may be some impact to others that I would be engaging in my effort. I received full support from all parties to adjust our schedule as needed. The primary contact, the Client Services manager, was the most important resource to convince. He is an expert with our entire product line and has been working in software support for over three years. Fortunately, he is not territorial and was committed to helping with anything that I needed.

We began with an initial pre-Kaizen event in which we tried to hone in on our goal for the project. We both agreed that "make things better" was a little too broad, but at least that proved that everyone was willing to work. Our initial pre-Kaizen ended with the following guidelines:

### d1. Pre-Kaizen

| **PRE-KAIZEN** | |
|---|---|
| Title | Fast resolution of production software support issues. |
| Purpose | Reduce Cycle Time for Customer Production Software Errors. |
| Business Need Addressed | Customers consistently complain that we are not responding quickly to production defects. Specifically, they complain that these defects directly impact their ability to generate revenue each minute that they encounter the errors. Addressing this constant direct contact that we have with the customers can be an opportunity to create a positive interaction and eventually a competitive advantage. A 10 time reduction in cycle time for customer issues is our goal. |
| Problem | Calculate current times to resolve issues |
| Scope | Improve relationship with customers and proactively improve the software quality perception of all stakeholders by identifying the "bang for the buck" problems that are the most valuable to the customer. |

After we framed the project, we assembled those who would be the best representatives to include in a Kaizen event. This was important for several reasons. I needed to show that we were committed to improving the situation and we needed to express to these key people that we would need their help in selling any changes that we would need to make. Especially any that would directly impact customers. In addition to ourselves, we decided to include other software support reps, account executives, and a couple of software developers. This would give us functional coverage to include

- Front-line support (support reps)
- Customer expectation management (account executives)
- Back end higher tier support (developers)

As we discussed the efforts, we started to extract the issues that everyone perceived as the current problems with supporting our customers. What began to surface was interesting. Although anytime we miss customer expectation is an opportunity for improvement, it became obvious that the biggest decrease in customer satisfaction came from handling urgent software requests. We defined these as requests which involved all or part of the software being "down." Since these are ecommerce sites, the perception is that any time offline directly impacts revenue. Obviously, this significantly impacts customer perception. In addition, these urgent requests are the biggest disruption to support reps, account execs, and developers. Since, in each case the employee has to completely drop what they are doing and respond to the situation. The practical impact is even greater in that all three stay distracted for the duration of the resolution as everyone "jumps in" and stays involved "'til the bitter end." Consequently, we amended the Charter to focus on Urgent requests (shown in red).

## d2. Charter

| CHARTER | |
|---|---|
| Title | Fast resolution of Urgent production software support issues. |
| Purpose | Reduce Cycle Time and eliminate frequency of Urgent Customer Production Software Errors. |
| Business Need Addressed | Customers consistently complain that we are not responding quickly to production defects. Specifically, they complain that these defects directly impact their ability to generate revenue each minute that they encounter the errors. Addressing this constant direct contact that we have with the customers can be an opportunity to create a positive interaction and eventually a competitive advantage. A 10 time reduction in cycle time for customer issues is our goal. Also, we need to eliminate 50% of the urgent production support issues. |
| Problem | Calculate current times to resolve issues |
| Scope | Improve relationship with customers and proactively improve the software quality perception of all stakeholders by identifying the "bang for the buck" problems that are the most valuable to the customer. |
| Roles and Responsibilities | Stakeholders: Software Support, QA, Development, Account Management, IT<br>Sponsors: George Chapman, Director of Engineering<br>Team Members: Brian Glover, Manager of Software Support<br>Black Belt Candidate: Charles Hollingsworth |
| Resources (non-human) | Task Support Database |
| MILESTONES/MEASURES | Project start date: 9/1/06<br>Planned project completion date: 11/30/06 |
| How will we know if we are successful? What are the measurable benefits the project is targeted to deliver? | ▪ Decreased cycle times from ticket open to resolution.<br>▪ Increased customer satisfaction during status meetings with Account Executives<br>▪ Decreased interruption in development and QA which negatively impacts scheduled software releases<br>▪ Important:<br>▪ Set expectations with customer<br>▪ Meet those every time for critical issues<br>▪ On call Developer?<br>▪ Stakeholder Analysis |

## d3. Scope

| SCOPE | Measure | |
|---|---|---|
| **Problem**<br>▪ During the third quarter of 2006, we lost customers to a competing software platform and another high profile client has demanded that we devote full-time resources to their support issues and scalability testing. Since the competing platform is essentially equal regarding product line and costs, a significant component in their decision to move is software | Customer Requirements | Measure |
| | Fast Resolution to Urgent Ticket | Throughput Time |

support and scalability. Considering only the contracts that have already been lost, the monthly revenue loss is approximately $80,000 dollars. The loss of just one "Tier 1" customer would cost approximately $5,000,000 per year in revenue.

**Goal**

- Reduce cycle time on software support issues to under an hour.
- Eliminate defects that need to have to be addressed in the first place!

**COPQ**

- Perceived software quality. Many of the support issues are not bugs in our software, but rather hardware/platform (Windows 2000/2003, IIS, SQL Server) malfunctions; however, given our relatively unsophisticated customer base…*everything is OUR fault!*
- Direct costs are lost customers and any downtime in customer's software resulting in a reduction in our maintenance fees and rebates to the customer (add metric here).

**VOC**

- We will be using software support issues from our call support database which contains detailed complaints with time stamped ticket initiation and documentation explaining each step in the resolution process. We will also gather direct candid feedback from account managers which are effectively brokers which represent all of our customers. "They want to know progress," but "IT ALL BOILS DOWN TO WAIT TIME."

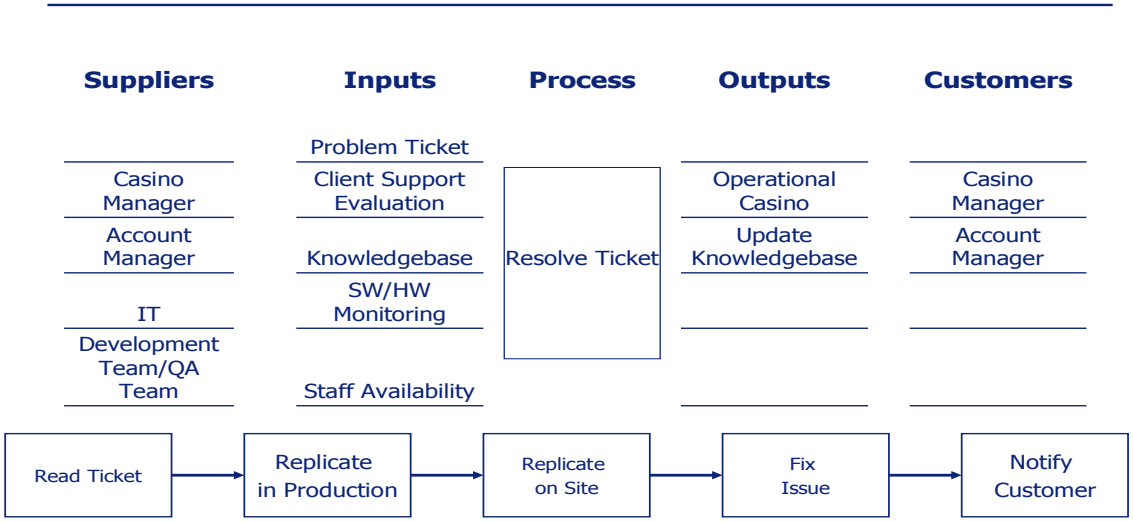
| Want to know progress at every stage | Ticket Update Frequency |
|---|---|

Furthermore, this seemed to make sense considering Six Sigma and variation. The specific "variation" caused by unexpected urgent situations essentially impacted several other resources and schedules from four departments. This does not even consider the expense associated with rework and retesting.

## MEASURE

### m1. SIPOC

The first step in mapping our process required a SIPOC analysis. An interesting characteristic of the SIPOC is that the primary suppliers (*Casino Manager, Account Manager*) are also the customers of this process. This is understandable given that these groups create the support tickets and are consequently delivered a solution represented by a resolved support ticket.



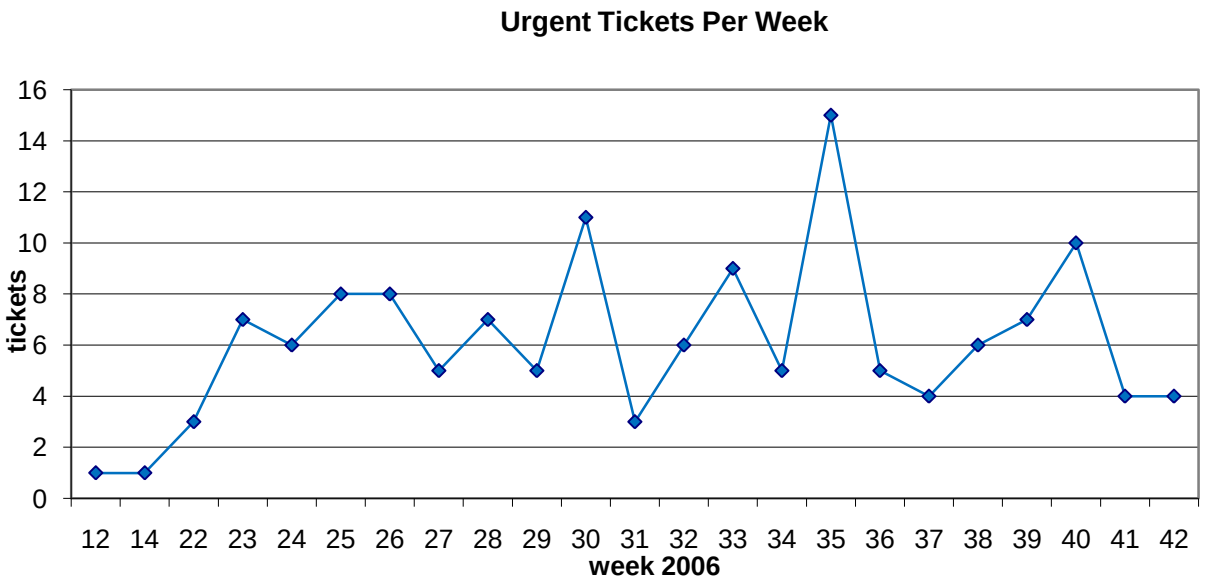

To perform the measure phase, I had to get my hands on data. Fortunately, I had access to our task support database. Unfortunately, the items in the database were not explicitly categorized. For instance, the tickets were not tagged as “Urgent” and were not classified by type or application. So, I tested many techniques (time open, string matching for urgent keywords, etc.) for writing SQL queries to filter and scrub the support database to focus on tickets involving urgent requests.

Painstakingly, I reviewed a thousand tickets and manually assigned categories to the tickets.

- progressives
- cashier
- casino client/art change
- credit system
- processor setup
- CD create
- processing transactions
- admin access
- coupon
- admin education

Once I identified the categories for the tickets, I then applied these categories to about 140 tickets that I was able to classify as *urgent*. I ran a time series analysis on the tickets to track workload and concurrent outstanding tickets over the previous months. Shown below are samplings of the measurements.

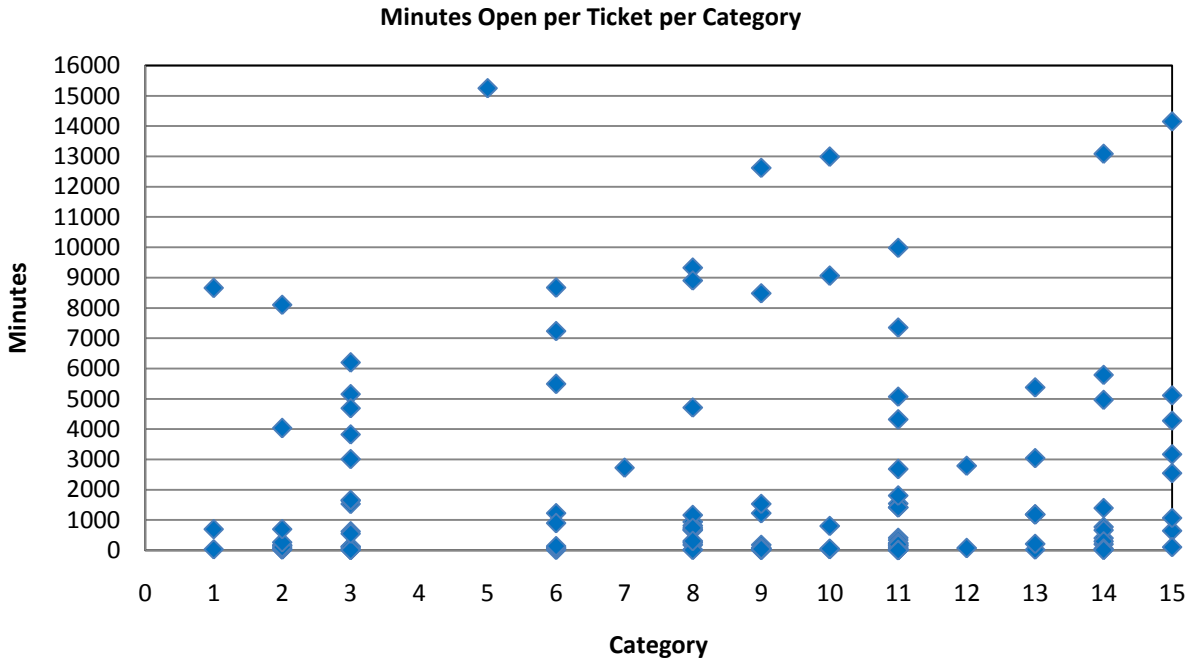
### m2. Urgent Tickets per Week




This is a time series plot for the tickets opened per week. There are some significant spikes in the opened ticket amount over this six month period.

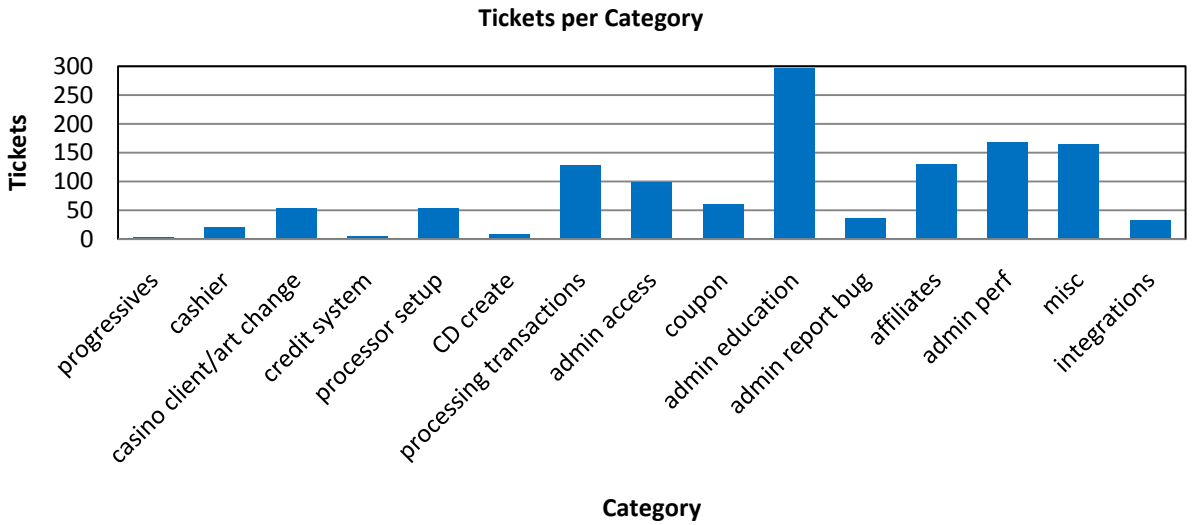
Subsequently, I graphed the number of tickets per category to get an idea of how the load was distributed and how long each ticket remained open. Also, I wanted to validate the categories that I had created. As shown in table m2, some categories that were represented in the original total ticket dataset did not have *urgent* tickets attributed to them for more than a six month period.

### m3. Minutes per Ticket per Category




In order to get a visual idea of how many tickets were submitted in their respective categories, I charted the tickets for the nominal amount of tickets that existed in the system. Obviously not all tickets are "created equally" considering complexity, reproducibility, etc, but with each ticket there are similar overhead costs associated with the customer entering a ticket and a representative opening and reading and categorizing and reproducing each ticket no matter the ultimate complexity of the issue. The m3 chart considers all tickets for this observation, since all tickets submitted impact the quality of service and resolution time for urgent requests.

### m4. Tickets per Category



After studying our initial data and honing the goal and definition of the project we created SQL reports which allowed us to instantly produce and instantly reproduce these measurements over any time range. For this project we continued with our analysis using the six month prior time frame. From a business perspective, we knew that this particular time frame encompassed many support and product scenarios that would be representative of worst case scenarios in the future.

## ANALYZE

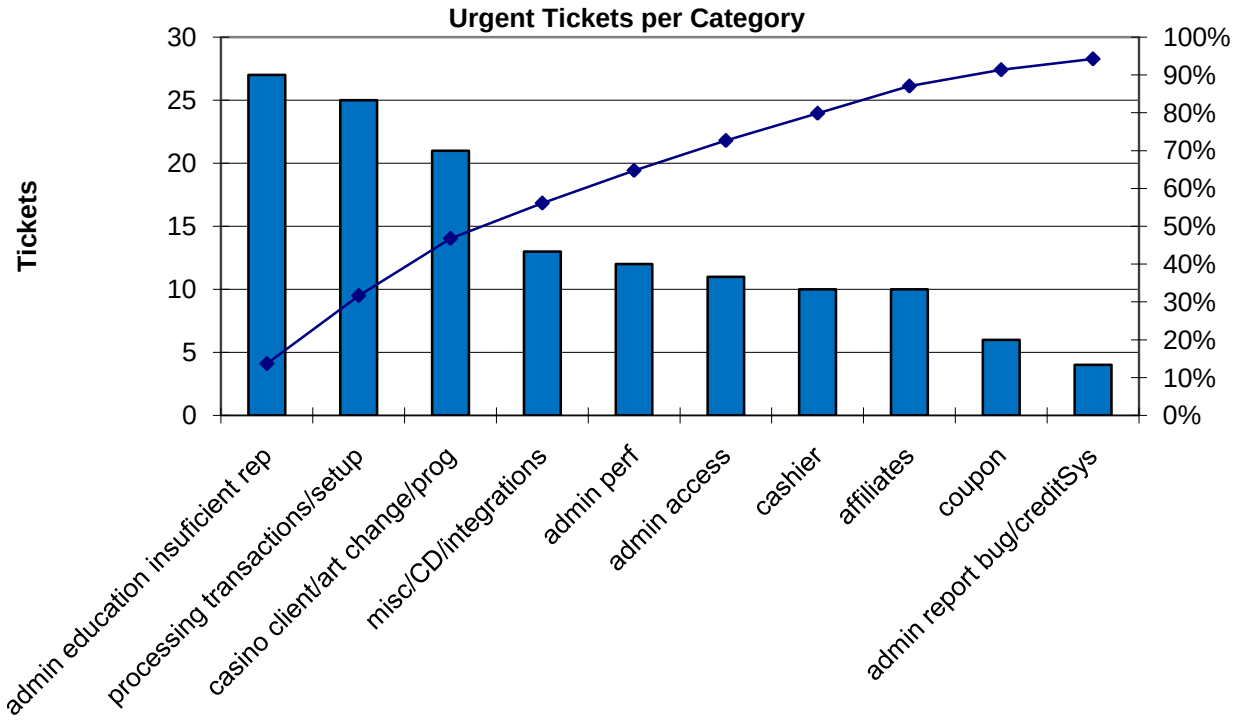
As we set up the *Measure* reports, instantly, there were obvious improvements to be made. However, I stressed to those involved that although the measurements pointed out some obvious "low hanging fruit," we needed to complete our analysis to prioritize our efforts. Based on other projects I had been involved with in the past, I was sure that attacking the biggest *constraints* would dynamically cascade throughout our prioritizations as we systematically improved our processes.

Building on our charts from the *Measure* phase and the business expertise on the team, the initial categories began to congeal into the ultimate set of Consolidated Categories. These are the ticket categories (with their categorical code) that will be referred to from this point in the document.

- cashier (2)
- casino client/art change/progressives (3)
- processing transactions/setup (6)
- admin access (9)
- coupon (10)
- admin education insufficient report (11)
- admin report bug/creditSystem (12)
- affiliates (13)
- admin performance (14)
- misc/CD/integrations (15)

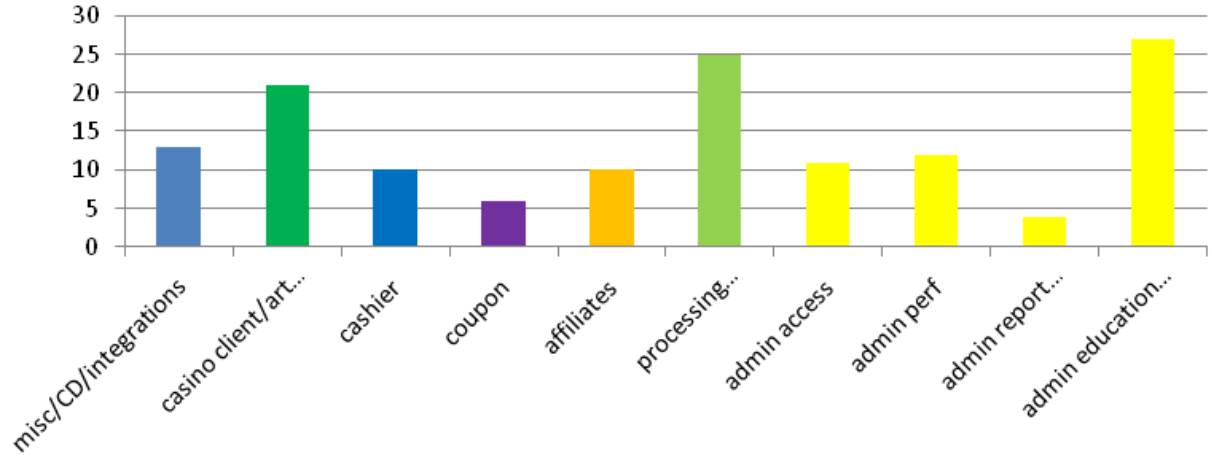
The Pareto analysis offered here gives a scope of the situation.

### a1. Urgent Tickets per Category



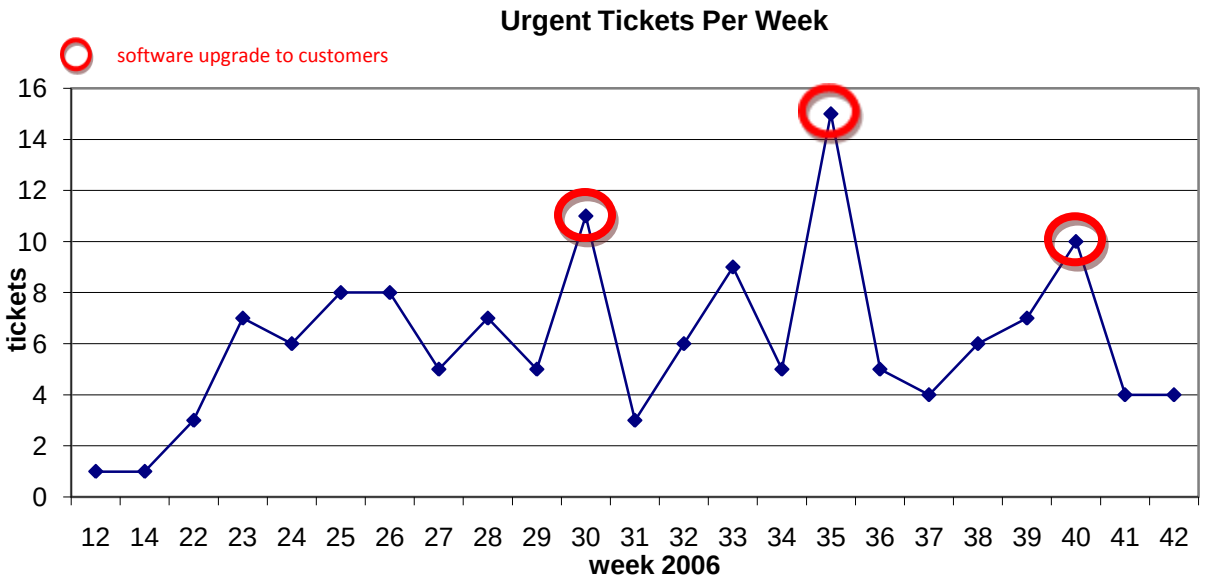

Based on the architecture of our applications and the organization of our product features, we were able to further categorize these consolidated categories so that improvements could be made in an area. This may seem counter to Lean philosophy at first, but, in fact, the nature of enterprise software applications make it much more efficient to make code adjustments and add features to one area at one time. To use a mechanical analogy, it is easier to go ahead and fix several things while you have the engine disassembled. This applies to fixing and adding code as well as testing the code for QA before deployment. Specifically, the four rightmost categories represent issues relating to the same software application. Furthermore, the processing and cashier categories involve another logical function that exist as separate applications but, because of their integration, must be tested by QA and evaluated by customer support simultaneously.

### a2. Urgent Tickets per Category



Reviewing the time series plot from the *Measure* phase exposed an interesting phenomenon. It seems that the spikes mentioned before correspond directly with the rollout of a new software version. Software upgrades are performed by our company for each of our customers. These upgrades are done remotely usually within a two week timeframe.

### a3. Urgent Tickets per Week (spikes)




Given the consolidated categories and time series analysis, I wanted to build a model to calculate the relative impact of ticket submission on ticket resolution times so that wait times could be predicted in the future for customers based on the classification of the request and the current tickets that were simultaneously under investigation in the system. I transformed the categorical codes that I designated for each consolidated ticket category. Using SPSS, I ran a factor analysis on the categories to see if I could further combine any correlated variables. There were no further statistically significant relationships between the ticket categories that I had already functionally defined. Furthermore, other than the .69 correlation score for *ticketUpdates* and *minutes*, there were no statistical relationships between

### a4. Factor Analysis of ticket characteristics

- Day of week opened
- Week opened
- Minutes opened
- Status
- Category
- Ticket Updates

| | week | day of week | status | category | minutes | simlutaneousOpened | ticketUpdates |
|---|---|---|---|---|---|---|---|
| week | 1.00 | -0.17 | -0.10 | -0.17 | -0.35 | -0.81 | -0.13 |
| day of week | -0.17 | 1.00 | 0.01 | -0.14 | 0.05 | 0.07 | 0.01 |
| status | -0.10 | 0.01 | 1.00 | -0.06 | 0.36 | -0.05 | 0.36 |
| category | -0.17 | -0.14 | -0.06 | 1.00 | 0.12 | 0.24 | 0.15 |
| minutes | -0.35 | 0.05 | 0.36 | 0.12 | 1.00 | 0.09 | 0.69 |
| simlutaneousOpened | -0.81 | 0.07 | -0.05 | 0.24 | 0.09 | 1.00 | 0.01 |
| ticketUpdates | -0.13 | 0.01 | 0.36 | 0.15 | 0.69 | 0.01 | 1.00 |

*KMO test = .504

This relationship seemed interesting and intuitively accurate. Consequently, this provided a variable which I could analyze further to see its impact on the time that tickets remained opened. This is not an earth-shattering discovery; however, we discussed how we could reduce the amount of repeated correspondence with the

customer and among internal resources. The overhead of communication is high within a support group in a software company given that the ticket is often assigned across department to employees that have another primary responsibility and do not consider the resolution of the ticket as a priority.

For curiosity sake, I performed a hierarchical cluster analysis and a K-means analysis on these variables but after removing outliers, the two clusters that were derived did not seem useful to classifying tickets as the best scenario had one of the two clusters cluster only representing 5% of the population.

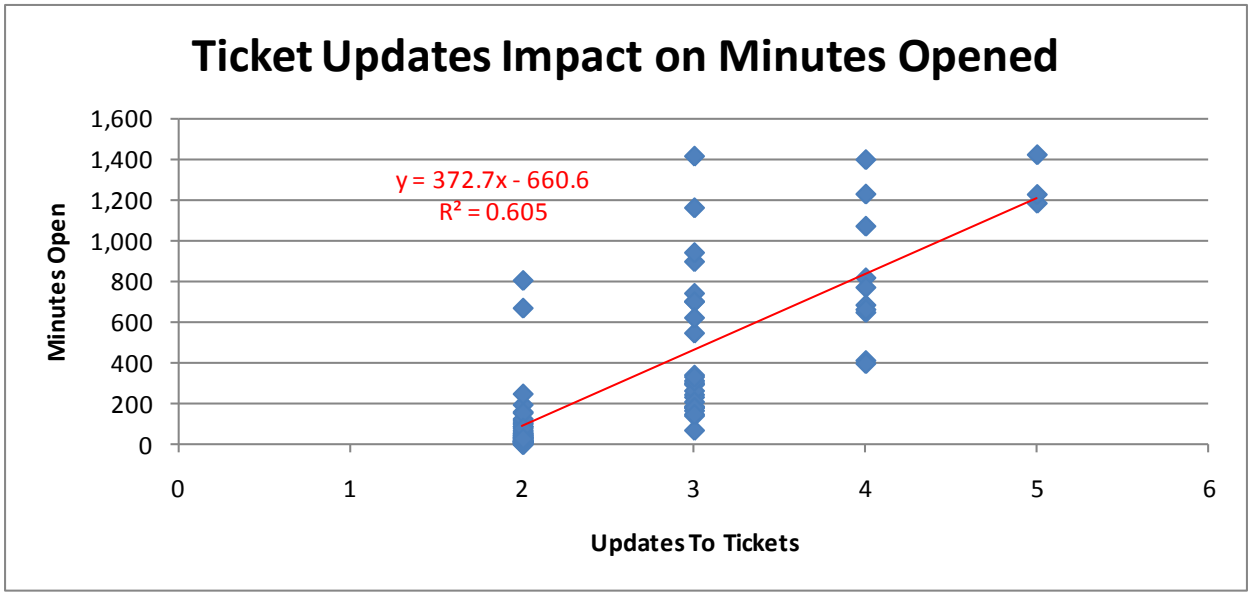
I ran regression observations against many combinations of variables. For example, I created dummy variables for the ticket categories and ran a regression analysis against the independent variables *MinutesOpened*, but the ANOVA proved these variables to have no significant impact.

| minutes | Dum_2 | Dum_3 | Dum_6 | Dum_9 | Dum_10 | Dum_11 | Dum_12 | Dum_13 | Dum_14 | Dum_15 |
|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1526 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1637 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 331 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 15253 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 81 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2727 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 8656 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9322 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

| Model | Unstandardized Coefficients | | Standardized Coefficients | t | Sig. |
|---|---|---|---|---|---|
| | B | Std. Error | Beta | | |
| (Constant) | 6337.07 | 4347.66 | | 1.46 | 0.15 |
| Dum_2 | -4976.27 | 8362.88 | -0.05 | -0.60 | 0.55 |
| Dum_3 | 4010.26 | 6573.04 | 0.06 | 0.61 | 0.54 |
| Dum_6 | -612.59 | 6270.28 | -0.01 | -0.10 | 0.92 |
| Dum_9 | -4128.89 | 8080.73 | -0.05 | -0.51 | 0.61 |
| Dum_10 | 11181.43 | 10196.16 | 0.10 | 1.10 | 0.27 |
| Dum_12 | 5004.93 | 12103.36 | 0.04 | 0.41 | 0.68 |
| Dum_13 | 28759.23 | 8362.88 | 0.32 | 3.44 | 0.00 |
| Dum_14 | -4034.07 | 7837.85 | -0.05 | -0.51 | 0.61 |
| Dum_15 | 7955.21 | 7440.17 | 0.10 | 1.07 | 0.29 |

Then I further scrubbed the data identifying extreme values with an SPSS *missing value analysis* to only include tickets that were both critical issues and were closed within one day. The logic behind this scrubbing was that the tickets that were open for more than a day were not the type of ticket that we were trying to simulate considering that we were trying to see the impact of ticket updates during the early stages of a tickets life. In a sense, those observations were outliers relative to the analysis.

### a5. Regression Analysis Ticket Updates on Minutes Tickets Remained Open



| ANOVA | | | | | |
|---|---|---|---|---|---|
| Model | Sum of Squares | df | Mean Square | F | Sig. |
| Regression | 7717355.45 | 1 | 7717355.45 | 119.45 | 0.00 |
| Residual | 5039420.10 | 78 | 64607.95 | | |
| Total | 12756775.55 | 79 | | | |

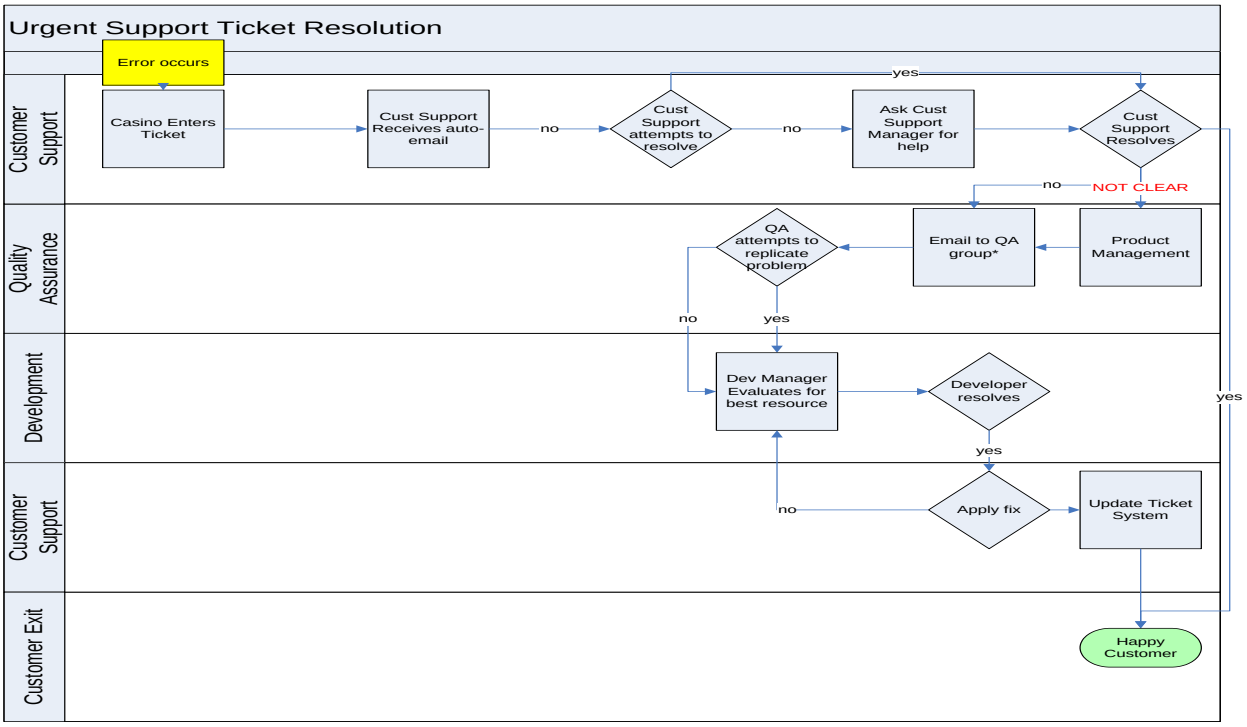
| Model | Unstandardized Coefficients | | Standardized Coefficients | t | Sig. |
|---|---|---|---|---|---|
| | B | Std. Error | Beta | | |
| (Constant) | -660.6228623 | 95.5511189 | | -6.9138161 | 1.1487E-09 |
| ticketUpdates | 372.7281728 | 34.1036568 | 0.777792578 | 10.9292729 | 2.1336E-17 |

This analysis did render some interesting information. Although this confirmed what we knew intuitively, it further drove home the fact that we need to solve the problems by getting the right ticket in the hands of the right person, quickly. The model that represents the impact of multiple updates to tickets is

*Minutes Open=372.2 (Number of Updates to Ticket)-660*

### a6. Process Flow Map

After analyzing the more quantitative side of the process, we moved to the more tangible portion of the analysis. This was probably the most entertaining as we exposed what we all thought was a pretty straightforward efficient process!




The swim lanes represent the departments that are touched during the ticket answering process. Obviously, most tickets do not require this complete flow. I believe this is why we all thought that the process was streamlined. However, an *urgent* ticket of any complexity could very likely follow this entire flow. In fact, we used a recent

ticket as an example to validate this flow. The points of communication alone, ignoring the actual solution to the problem, would cause our process goal to be out of control.

## IMPROVE

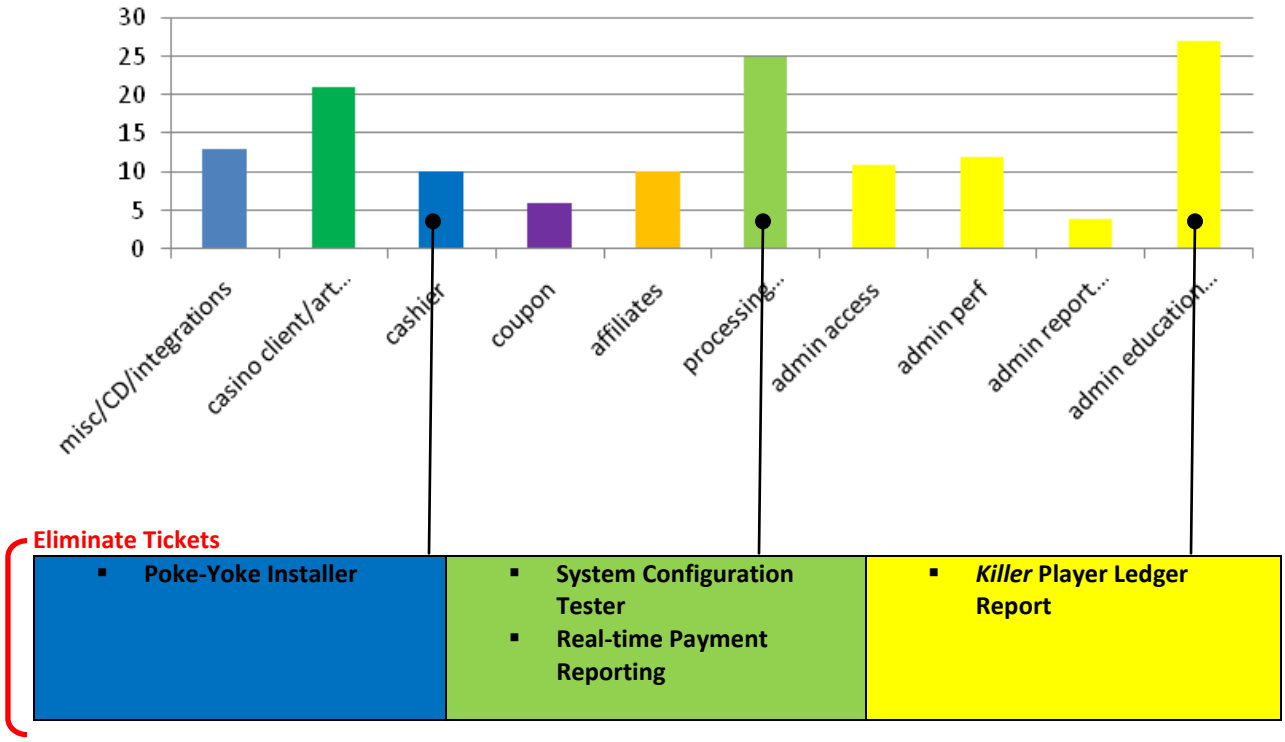
Taking in all of our findings from the *analyze* phase, there were some obvious opportunities. Although the quantitative analysis was certainly Six Sigma methodology, Lean fundamentals seem to be effective in addressing the problems that were exposed. This can be simply categorized as

1. Reduce tickets that need to be addressed
2. Reducing complexity beforehand to eliminate defects and then reduce complexity in troubleshooting and solving the defect when it is reported.
3. Giving the customer what they *really* need to circumvent defects.
4. Put the person who can solve the problem closer to the process.
5. Identify high-risk periods and allocate resources accordingly

Worth noting is that although this list seems vague, everyone involved in the process completely understands what is meant by even this brief definition. Each improvement category was then addressed by specific *improvement* projects that we defined (i1-i3).

### i1. Reducing Tickets per Category (eliminating defects)

This is the chart from the *analyze* phase (a2). Below the chart are the actual projects that we decided would address the worst categories relative to urgent ticket submission. Also, the improvements listed are easily defined and implemented. Obviously, iterative analysis will be required as each solution is implemented to then reprioritize our problem categories, but these are inarguably areas that need addressing and the corresponding solutions can be developed and implemented independently. The color-coding and line pointers represent the ticket category that the corresponding *improvement* is attempting to reduce. The color-coded items are the *ticket elimination* projects. They will serve to simply processes, eliminate steps, and provide the customer with real-time reporting and information to resolve perceived issues that should never even be submitted to our support staff. The remaining *improvements* (grey) are projects that will generally reduce the time a ticket remains unresolved.




**Poke-Yoke Installer** is an automated installation wrapper for the cashier application. There are several tedious steps when installing this application including linking to external processors, administrative applications and web site configuration that are easily missed and are eventually the origin of most support tickets involving this category. Obviously, this is a critical application as this is the interface for players to transfer money into the casino. I mention poke-yoke because this installer will allow for step-by-step testing of all of the configuration settings for this system as the support representative is stepping through the install.

**System Configuration Tester** also relates to the cashier and payment processing function of the software platform. This would allow the installer and support staffs to test the payment processing process and identify the point in the transaction process that was failing. Many times the problem is the responsibility of the third party payment processor. So, time is wasted troubleshooting a problem on our side that is out of our control. The obvious next step is to coordinate this with the third party to improve their portion of the process.

**Real-Time Payment Reporting** will proactively enable the customers to see the streaming funds transactions from their customers as well as failed attempts and reasons for failures. We will then be able to automate the testing so that customers can test the payment processors themselves thus decreasing this burden on the support staff even further.

***Killer* Player Ledger Report** is information that the customer has needed for a long time. In reality, they have access to all of the information in this conceptual report, but it is in five different places in the administration application. This has been identified as a need for almost a year on our project plan, but we did not know the impact of the absence of the information on our support staff. The information in this report would allow the customer to easily identify perceived discrepancies in a players account and track all activity by the player so that our support staff does not need to manually research this issue by writing SQL code against production databases. This is an exciting feature in that it will solve so many problems while actually increasing the marketability of our administration application.

### i2. Reducing Tickets per Category (increasing responsiveness)

(from chart i1)

**Developer Assisted Initial Rollouts**
**Higher Level Support or On-Call Developer**
**Add "category" to Support Tickets**
**Automated Problem Type Notification to Best Resource**

To reduce the responsiveness we have decided to take some other measures. While these are more labor intense and not as technically exciting as the previous solutions. However, there implementation can happen immediately and have immediate impact, especially given that the aforementioned *ticket elimination* projects will take time to implement. As we identified with the time series chart, our spikes in defect tickets occur during the initial stages of software "roll-outs" to customers. These steps will serve as a stop-gap in the short-term. As we continuously improve, we will hopefully make these initiatives obsolete.

**Developer Assisted Initial Rollouts** is a simple appropriation of resources to the right place at the right time. The concept is that we will have a developer in the support facility that was responsible for the major features that are included in the software version release. From our observation, we can see that the early stages are the most critical. This will only distract the resource for a couple of days, but as we have seen, that resource is distracted anyway by the defects. The benefit here is two-fold. The response time and resolution will be much shorter as we eliminate levels of communication. Also, the developer is able to witness the problems of the support staff and learn new ways of improving or simplifying applications so that applications can be refactored to reduce defects in the future.

**Higher Level Support or On-Call Developer** is a related solution that calls for a developer to always be "on-call" for urgent issues. This will solve the big issue we have with support identifying the correct resource to resolve the defect. While developers tend to be resistant to support, they are willing to share the responsibility so that more accurate evaluations of problem can be made immediately and the correct resource identified. In interviewing the developers, we found that they were primarily frustrated when they investigated a problem for hours just to find that they were the incorrect resource to resolve the problem.

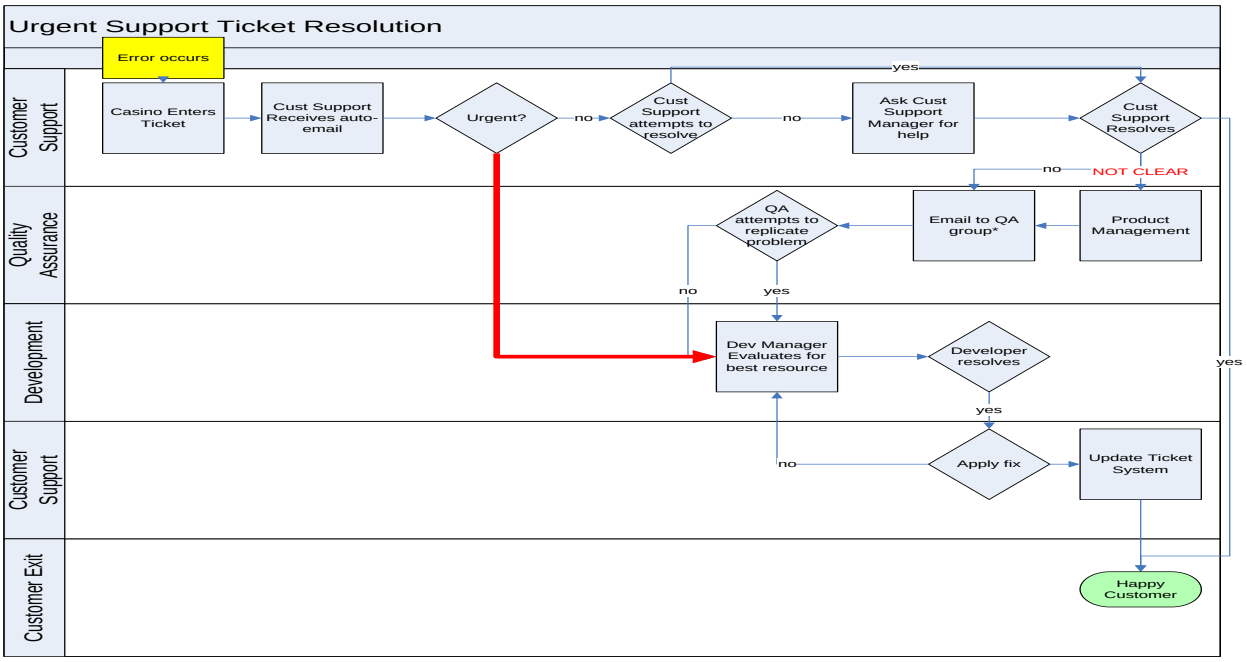
**Add "category" to Support Tickets** is a frighteningly simple feature that we will add to the ticket management system that will be a simple drop list for categorizing tickets as they are created and initially answered. Of course, we will need to educate customers and support staff to know how to categorize issues correctly, but based on the categories we have identified in our earlier analysis, the categories are explicit enough to make the ticket association relatively easy.

**Automated Problem Type Notification to Best Resource** is the ultimate situation for managing urgent tickets. As we reduce the amount of urgent defects that occur. I would like for the urgent ticket to be routed directly to the highest level resource that is responsible for the defect type. In this case, we will take most of the investigative burden of urgent tickets off the support staff and put them directly on the developer or artist that produced the defect or has been established as a functional expert.

Critical to all of these improvements is a general effort to educate the customer so that they provide more information when submitting a ticket. As we develop the ticket categories we will be able to provide some sort of expert system features that will instruct the customer to provide specific information about the issue they are submitting.

### i3. Process Flow Map Improvement

After analyzing the process flow map, the unnecessary complexities in the process became obvious, especially for an urgent ticket. A huge improvement that is relatively simple to implement immediately was to add a "decision" (Urgent?) subsequent to the receipt of the ticket and a corresponding flow (in red) to a higher level resource. This will be made even simpler when the aforementioned *improvements* are implemented, tickets are categorized and higher level resources are assigned to receive these urgent issues.

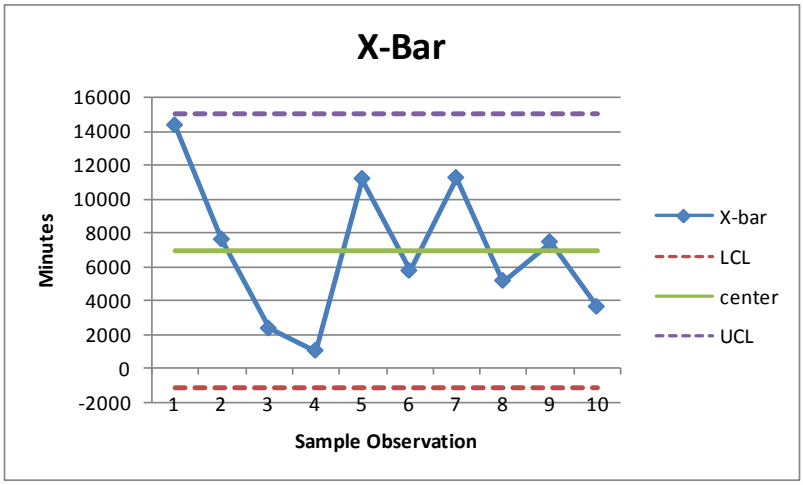



This will circumvent non-value added steps. These were not obvious until now because these fit into our internal processes but really do not contribute to customer quality considering urgent requests. These are certainly not steps for which the customer is willing pay when their entire business is down!

## CONTROL

Although our process is far from "in control," we thought it was interesting to build an X-bar chart of our initial situation.

### c1. Control Chart




This chart is not very valuable, but it did provide a prototype for the developers to construct a real-time control chart in our ticket management system. The initial mean in our process was greater than a day and a half for a ticket. The samples for the chart were based on ticket category, but we will adjust this to track every individual urgent ticket when the system is up and running. Needless to say this excited the customer support manager as he has been managing "by gut" for 3 years.

As we apply these improvements, we are iteratively reconsidering our measures and analysis to see that our improvements remain relevant and prioritized relative to the goal. We have decided that we must always consider a few key concepts as we seek to control our process and continually improve.

- Always Look to Eliminate Defect First!
- Admin System Performance→ more than meets the eye
  - Some customers will not commit to adequate platform and will not "allow" us to meet control limits
  - Not worth software re-write
- Customer Tiers → Lower tier = disproportionate urgent requests
- Rollout Time→Developer "On Call"
- Keep fine tuning our control tools

These points are critical as we get people excited about the new techniques. The organization has never made an explicit commitment to support (other than hiring bodies). So, we want to set internal expectations about what we are actually trying to improve right now. Every person involved has their own opinion on our "real problem," so if we are not seen to be addressing their concern immediately then we could be frustrating employees that are crucial to our long-term success. These points stress that we want to

**Eliminate defects first** as some employees have made a career of simply fixing things repeatedly to get the proverbial "pat on the back."
**Admin application performance** cannot be "fixed" for customers that are not willing to make an investment in hardware. We will identify these customers. This is a customer account management issue not a support issue.
**Customer Tiers** are important for many of the same reasons. The premium customers pay more for premium support. We strive to meet expectations of all customers, but in the short term this is strategic in meeting our goals.
**Rollout Time** means that we need to have a developer dedicated. This is the Support Team's responsibility to coordinate with the development group.
**Keep giving feedback on our control tools** since this is the only way that we have for measuring our progress and real-time customer satisfaction.

*Most of all…stay focused on urgent tickets!*

Applying these techniques to all tickets will happen naturally. But, everyone needs to measure ourselves by urgent response. This is what undermines customer satisfaction and directly costs money and impacts revenue.

Given our current situation, the goal of eliminating 50% of the urgent issues seems very feasible. Furthermore, the 90% reduction in cycle time is actually too simple given the horribly long durations we have connected to some tickets. The team agreed that a better metric will be to keep urgent ticket resolution below one hour. This does not alter our strategy and it will give everyone involved a more tangible measurement.

## CONCLUSION

During this project, we identified other areas to extend the methodologies.

- Process Improvement Cycle
  - Prevent Repeated Mistakes
  - Stop playing with puzzles
- Application Performance
  - Microsoft Laboratory Engagement
  - Replicate Urgent Scenarios
  - Load Testing Tools
  - Six Sigma within the application
- Customer Satisfaction Surveys
  - Integrate the customer
  - See how we are doing
- Simplify Development Cycle with Scrum and Agile

Obviously, we want to extend the improvement and control techniques to the entire support **process improvement cycle**. However, we also want to apply it within the application platform by implementing continuous **application performance** improvement. We have already reserved Microsoft Laboratory engagements for testing as well as building monitoring tools to simulate customer experience in the field so that we know they are having problems *before* they submit a ticket, and we can proactive deal with performance tickets as performance on outdated hardware begins to diminish. Customer Support will request customer feedback in a formal survey periodically “see how we are doing.” Finally, we want to incorporate Six Sigma and Lean within our software development process. There are several recent movements in the software industry to combine these techniques with Agile development. We restrained the urge to broaden our scope during the project and kept these projects separate.

This project itself provided a good basis for showcasing Lean Six Sigma in the organization. This was a worthwhile project that all participants were committed to solving from the Finance to the Development department. In this case, improving external customer service will improve every internal process also and this iterative evolving learning experience provided a great model going forward.