University of Economics, Prague

The Faculty of Informatics and Statistics

The Department of Information Technologies

Study program: Applicated informatics

Domain: Information systems and technologies

Improvement of digital support process using ITIL best practices, Kanban and TOC in multicultural environment

MASTER‘S THESIS

Student	:	Bc. Miroslav Králik
Supervisor:		Ing. Tomáš Bruckner, Ph.D.
Opponent	:	Kaveh Kalantar, MBA

2015

Declaration:

I hereby declare that I am the sole author of the thesis entitled “Improvement of digital support process using ITIL best practices, Kanban and TOC in multicultural environment” I duly marked out all quotations.

The used literature and sources are stated in the attached list of references.

In Prague on 29.04.2015 . . . .

Miroslav Králik

Acknowledgements

I would like to express my gratitude to my supervisor Ing. Tomáš Bruckner, Ph.D. for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Kaveh Kalantar, MBA for the support on the way.

Abstrakt

Mnoho společností dnes musí čelit vylepšování podnikových procesů, neustálému odstraňování systémových překážek a identifikaci úzkých míst v systému, které bráni efektivnímu dosahování stanovených cílů.

V této diplomové práci popíšu, jak jsme se v společnosti s podobnými problémy vypořádali, v našem případě v oddělení správy a podpory webových aplikací pro jednoho mezinárodního klienta.

V první kapitole představím základní informace o historii, prostředí, v kterém pracujeme, dále jednotlivé aktivity, které jsou každodenní součástí zmíněného procesu.

Následně popíšu všechny překážky a úzká místa systému identifikovaná v původním procesu a návrh řešení a vylepšení pro jejich efektivní odstranění.

V poslední kapitole shrnu dosažené výsledky po průběžné implementaci navržených vylepšení, zda byla plánovaná očekávání naplněna a jaký pozitivní nebo negativní vliv dané změny přinesly.

Klíčová slova

Správa a podpora aplikácií, optimalizácia procesov, systémové prekážky, úzke miesta, teória obmedzení, Kanban, procesné KPI, reporty.

Abstract

Many companies today have to deal with business processes optimization, ongoing removal of system obstacles and identifying any bottlenecks laying on the way which are preventing us to reach our goals.

I would like to show how we have dealt with similar problems in our case applied in the department of web application maintenance.

In the first chapter I will introduce the basic information about the history, environment the company operate in and maintenance activities which are part of the process.

Following with description all of the system obstacles and bottlenecks identified in the process and possible solutions or improvements for their elimination.

In the last chapter I will present the achieved results after my executing of the proposed improvements and if the expectations have been met or not and what positive or negative influence it has brought.

Keywords

Application maintenance, process improvement, system obstacles, bottlenecks, theory of constraints, Kanban, process KPIs, reports.

Contents

1 Introduction 1
1.1 Topic Definition and Description 1
1.2 Motivation for Selecting the Topic 1
1.3 Objectives 2
1.4 Methods of Achieving the Objectives 3
1.5 Assumptions, Limitations and Delimitations 4

2 Introduction of the current Application Maintenance Process and Environment 5
2.1 Maintenance Process 5
2.2 Technology Platforms and Applications 8
2.3 Maintenance Tools 11
2.4 Maintenance Activities 12
2.5 Resources and Planning 14
2.5.1 Human Resources 15
2.5.2 Allocated Budget 16

3 Identifying Obstacles and Bottlenecks 18
3.1 Uncontrollable Work in Progress 20
3.1.1 Lack of request ordering and prioritization process 20
3.1.2 My request is more important than anybody else type of culture 20
3.1.3 No Single Point of Contact 21
3.1.4 Solution 21
3.2 Waiting for client feedback and approval 25
3.2.1 Know-how of using support tools and the process 25
3.2.2 No deadline for client on providing feedback or approval 26
3.2.3 Lack of Proper Follow-up Process 27
3.2.4 Solution 27
3.3 Resource Contention 32
3.3.1 Shared Resources between Projects and Maintenance 32
3.3.2 Limited resources and bottlenecks 33
3.3.3 Solution 33
3.4 Lack of System Access 36
3.4.1 Solution 37
3.5 Inefficient Deployment Process 38
3.5.1 Solution 39
3.6 Waste and Duplicated Tasks 40
3.6.1 Too many support tools 40
3.6.2 No integrated knowledge base 41

3.6.3 Solution 42
3.7 Definition what is and what is not a maintenance request 43
3.7.1 Solution 44
3.8 Lack of Measurements and Reporting 45
3.8.1 Solution 47
4 Process Evaluation and Achievements 51
4.1 Decreasing the Number of not Resolved Requests 51
4.2 Retainining Down of not Resolved Requests 53
4.3 Reducing Resolution Time of the Requests 56
4.4 Improving Clients Satisfaction 61
4.5 Evaluation of KPIs 63
5 Conclusion 69
Bibliography 79
List of Figures and Tables 81
List of Figures 81
List of Tables 81

1 Introduction

1.1 Topic Definition and Description

As William Edwards Deming said [3] *„If you can't describe what you are doing as a process, you don't know what you're doing"*, can the following question be answered: Do I follow the same principles or steps when processing a regular task?

There are many companies out there which serve their customers and many processes that work or are more effective in one environment, but opposite in other one, and we should continuously evaluate it, identify the obstacles that are preventing to reach the goals, move forward and not to fall asleep at the switch. The environment (both internal and external) is always changing, we have different expectations and goals can be different in time and per party (company employees, business partners, third parties etc.). We work with different key factors and resources which influence the results, quality and amount of work we can complete in time.

What is the right balance of these factors, how to be more productive, how to achieve better results with the same resources (people, money, time and quality) and how to measure it? What are the obstacles and tool that may boost the efficiency?

And this is the moment, when it comes to ongoing process evaluation and improvement. According to John Arthur Ricketts [1, page 151]*:*

"A common pitfall in practice, however, is to attempt to improve anything and everything that can be improved. That's disruptive and expensive, so a more targeted approach may prevail. Specifically, what to change and how to change it are key questions."

1.2 Motivation for Selecting the Topic

There are several reasons that I took into the account before I had decided for this topic. But the most important one was to come up with something useful and with a value related to my professional working experience gained from a big digital agency with several offices worldwide that I had a chance to become a member. I have been leading a small team of 12 shared technical people (Java, PHP, Ruby, .NET, System Administrators, Database Architects, Quality Assurance, Interactive Web Developers etc.) for several years, with an ownership and responsibility for the overall technical maintenance performance, serving and primary client liaison contact for any incident or request tickets reported by EMEA and CEEME regions (more than 50 stakeholders, 20 countries and 5 web applications). Approximately 1200-1500 requests is processed annually and the company has experienced different types of problems during my time on board causing performance inefficiency, team overhead or complaints from several stakeholders. I would like to go into more details and show how we as the company deal with them, what effort has to be allocated to reach our targets and goals which may be taken as an inspiration or reference guide for a similar case.

1.3 Objectives

The main part of this thesis is dedicated to describing the environment and processes we work in and with, try to find the obstacles and bottlenecks and find a way how to deal with them to improve the process efficiency. This includes also a different set of used methods and practices for gathering the information or applying them to practice.

The last one and not less important is to apply the agreed solutions to the current process, executing and measuring the progress and results (including evaluations and training all of the stakeholders).

As an output of the maintenance process improvement, I expect the following positive effects which I will also summarize for the years 2013 and 2014:

- Decrease the number of open / not resolved client requests under 50 within the three months period.
- Keep the open / not resolved requests trend around 50-70 maximum as an average.
- Be more efficient in resolving requests within first, second and third weeks.
- Be able to predict a request resolution better.
- Improve the client’s trust and relationship.

1.4 Methods of Achieving the Objectives

The thesis is written based on professional experience supported by theoretical explanations embedded in the specific (sub) chapters for better understanding the context and terms used across the sections. Some of the actuals have been interpreted by visual charts and schemes to provide an easy and comprehensive way to get the message.

At the beginning I will describe the environment and background of the maintenance department and for that I will use my personal experience since I have led the department from the beginning and have all necessary details.

In the main part where I will be describing the obstacles and bottlenecks I had gathered the information from internal documentations, tools and from discussions with internal teams and clients. In the subchapter *3.1 Uncontrollable Work in Progress*, moreover I have used a Kanban mechanism called „Class of Service“ to differentiate the business value of every maintenance request. The initial idea for prioritizing maintenance requests was from one of my colleagues and my role has been to create the formula (SPR Index), build and improve it over time when needed.

In the subchapter *3.2 Waiting for client feedback and approval*, the solutions have been proposed and agreed based on the trends of not resolved requests in time and discussion with different stakeholders.

In the subchapter *3.3 Resource Contention*, the solutions have been proposed and agreed based on theory of constraints, Scrum methodology, professional experience and discussion with different stakeholders.

In the subchapter *3.4 Lack of System Access*, the solutions have been proposed and agreed based on trend and categorization of the requests according to an expertise needed for their completion and comparison how many of the requests will be needed to be routed to technical teams before and after the access to specific systems and tools is granted to support leader.

In the subchapter *3.5 Inefficient Deployment Process*, the solutions have been proposed and agreed based on trend of delayed requests due to long lead time in QA and deployment department and discussion with different stakeholders.

In the subchapter *3.6 Waste and Duplicated Tasks*, the solutions have been proposed and agreed based on ROI mechanism, where are calculated the cost of recommended tools and savings made when eliminating the waste when using the current tools.

In the subchapter *3.7 Definition what is and what is not a maintenance request*, the solutions have been proposed and agreed based on internal discussion between different stakeholders and professional experience.

In the subchapter *3.8 Lack of Measurements and Reporting*, the solutions have been proposed and agreed based on the discussion with different stakeholders and professional experience.

All the proposed changes and improvements are a result of team work of different individuals. My main responsibility and a merit was dedicated mainly to SPR Index, KPIs and reporting, new support tools proposal, proposal for granting more access to different systems and tools and executing the maintenance by following up the new rules and policy.

1.5 Assumptions, Limitations and Delimitations

My assumptions are that the proposed solutions for eliminating some of the process obstacles will not be easy to implement due to the complexity of the environment and limited time and resources. Therefore the results and achievements will be reached step by step from the beginning of implementing a first change in middle of December 2012, following next one on the go continuously until the end of 2014.

Some of the proposed KPIs and measures will also be available only for 2014 and not 2013 or 2012.

Due to company’s privacy policy all the names have been removed and changed, so there shouldn’t be any evident connection to any of the parties.

2 Introduction of the current Application Maintenance Process and Environment

Application Maintenance provides the ability to change functions, screen appearance, and other aspects of your application's performance and capabilities. [9]

2.1 Maintenance Process

Since my application was accepted and I and the company agreed on terms for the job position of support leader, from the day one (around April 2011) most of the coming duties were new for me, but I was very excited to take the challenge. Except some technical knowledge I had to learn everything from scratch as someone would say, and it was not only about colleagues, company philosophy, policy and internal processes, but also what is more important, get to know the clients, their expectations and applications my team is going to support. Not everything was so complex from the beginning, actually everything was very transparent and quite easy to manage. There has been only one team yet and one web application under our support. In that time there were still localizations for other countries being developed, so the scope of work was slightly increasing, but nothing that could be dealt with any bigger problems. The agreement for delivering the maintenance services (Service Level Agreement or shorter SLA) between the provider (us as the agency) and customer had been on place as well.

Definition of Sevice Level Agreement (SLA) according ITILv2:

“A formal, negotiated document that defines (or attempts to define) in quantitative (and perhaps qualitative) terms the service being offered to a Customer. Confusion must be avoided over whether the quantitative definitions constitute thresholds for an acceptable service, targets to which the supplier should aspire or expectations that the supplier would strive to exceed. Any metrics included in a Service Level Agreement (SLA) should be capable of being measured on a regular basis and the SLA should record by whom. Typically it will cover: service hours, service availability. Customer support levels, throughputs and responsiveness, restrictions, functionality and the service levels to be provided in a contingency. It may also include information on security, charges and terminology.” [7]

I will not go in more details, but I see a good point to mention that the response time for every reported incident or request was maximum of 4 hours. Resolution time was with no limitation. Mantis had been used as a system for tracking issues and requests and the amount of work the technical team spent on monthly bases was approximately around 70h as average.

Definition of Mantis:

"MantisBT is a full-featured bug-tracking system that not only keeps track of bugs, but includes a user system so that multiple users can interact and multiple projects can be tracked." [8]

But the company is not the only one that developes any kind of web applications for this client. There are other third parties across the globe and after several months a decision came from the top management that the company is going to be a partner with other big US digital agency that is the main digital and technical partner for the client on a global scale. What did that mean? Many things have changed since that day, mainly the complexity of the environment. Specifically the following areas went through changes:

- The local Mantis system has migrated to global version used by all regions and teams (both projects and maintenance).
- The scope of the maintenance has changed, from one smaller web application localized in few of the languages, now supporting 9 web applications for EMEA and CEEME regions, which is more than 20 countries each.
- Number of not resolved incidents and requests across all the web application reached 100 in that time.
- Technical teams were spread out across different continents (Czech Republic, US and Bangladesh).
- There was not only one client, but 2-3 representatives from each of the countries in which the web applications were localized (in total as far as I remember it was more than 50!)
- There were other service level agreements with different structures and points, so a consistency was missing, including different agreement on resolution and response times

At the end of the 2012 the number of open / not resolved incidents and requests crossed the border of 300 and it fluctuated between 250 and 290 in the last quarter. The annual maintenance budget had been spent within 6 months. This was a signal something did not work properly and the team had to come up with an idea how to get this number down as soon as possible and improve the overall maintenance service due to very low client satisfaction. There was also lots of pressure mainly from the client top management side and it was justified, because country representatives did not know an expected resolution time of their requests, there was a huge mess, lots of not resolved incidents and requests and annual budget had been spent within the half of the year. In many cases it took several weeks, sometimes months to get a request resolved and most importantly there was no prioritization model and no resource planning in place. The support leader had chosen a request (many times just randomly) and get it resolved, which was not the best approach. I will go deeply into every problem in the next chapter one by one, also what actions have been taken.

In the figure below can be seen the trend of not resolved maintenance requests for all EMEA and CEEME applications. The data showed for the next figures are from September 24, 2012 as a starting point and December 2, 2012 as an ending point.

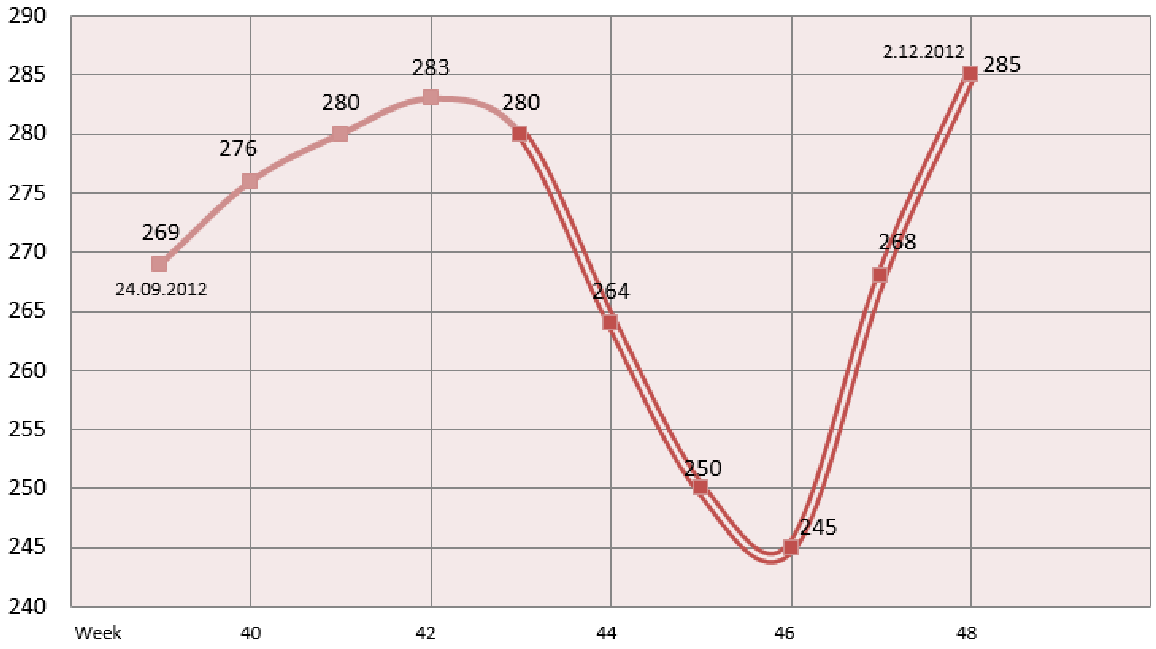


Figure 1:
Open / Not Resolved Requests Trend (Source: Author)

The following figure shows three mini charts. At the first one, the number of newly reported requests on weekly bases. In the second one, the number of resolved requests and incidents on weekly bases and in the last one, the difference between those two charts, which says whether the total number of not resolved requests was decreased (negative number) or increased the total number of not resolved requests (positive number), again on weekly basis.

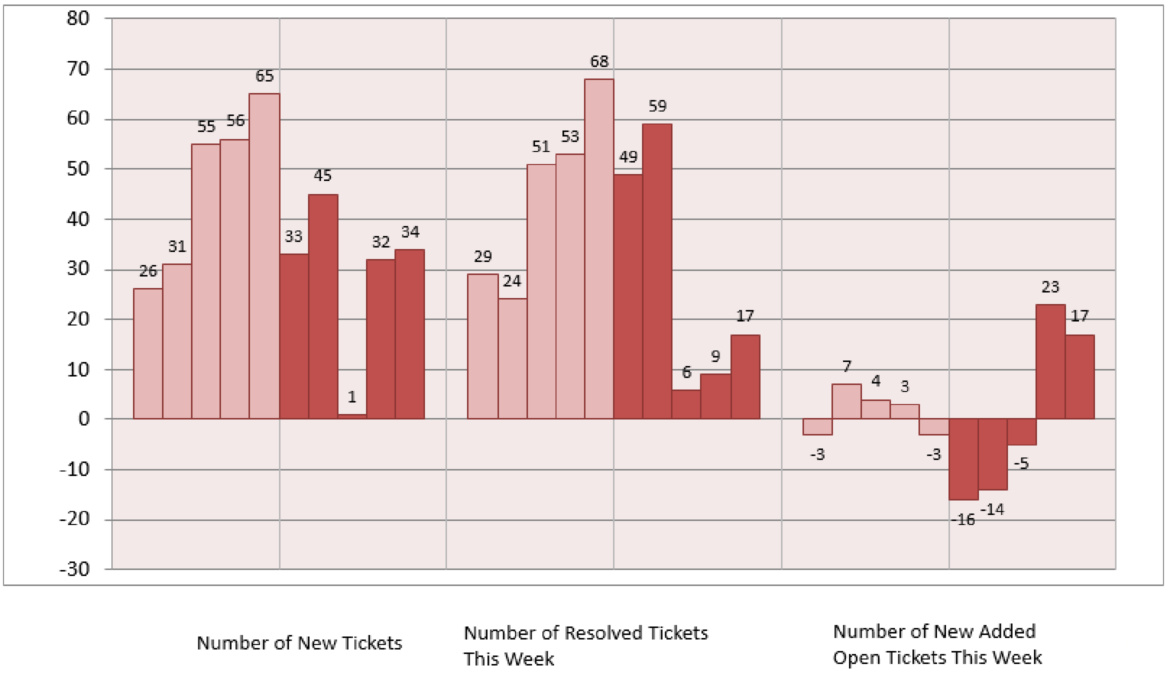


Figure 2:
Last Ten Weeks Trend (Source: Author)

2.2 Technology Platforms and Applications

I have led the support for several applications (some of them do not include development package but only coordination with third party providers) and they are built upon different platforms and CMSs (Content Management Systems). The purpose of every CMS in general is to provide an easy way and a user interface for non-technical people to make changes to a website application (mostly only content changes; usually when a code / application functionality change is required, the whole application must be re-built into a new application package and re-deployed to the appropriate application server).

Definition of Content Management System:

"A content management system (CMS) is a system used to manage the content of a Web site. Typically, a CMS consists of two elements: the content management application (CMA)

and the content delivery application (CDA). The CMA element allows the content manager or author, who may not know Hypertext Markup Language (HTML), to manage the creation, modification, and removal of content from a Web site without needing the expertise of a Webmaster. The CDA element uses and compiles that information to update the Web site. The features of a CMS system vary, but most include Web-based publishing, format management, revision control, and indexing, search, and retrieval." [10]

Below can be seen the applications structure according to programming languages they are built upon using a different technology. Every technology is supported by different technical teams and requires different knowledge.

In terms of CMSs used for easier content management as I have outlined briefly above, the company uses the following:

- Perseus - in-house solution built internally to support smaller projects built on PHP.
- OpenText - leading Microsoft solution enabling organizations to streamline the content publishing process and used for more complex web applications with more and advanced customization options. [4]
- YardStick LMS L2 - this one is not really a system for content management but rather learning management system LMS to maintain training kind of web applications.

Definition of Learning Management System (LMS):

"A learning management system (LMS) is a software application or Web-based technology sused to plan, implement, and assess a specific learning process." [11]

On the other hand, some of the web applications were built several years ago with old-fashioned (with an approach that is very difficult to adapt to the current environment and changes) programming ways and with no content management systems. The difference between using a CMS and changing the content in the application code directly is simple: where no CMS is present, the change needs to be done in the application code and the whole application has to be re-built and re-deployed again, which makes the process (even by doing a very simple content change) a time consuming process involving more people, resources and of course, money.

Definition of the term build:

“The term build may refer to the process by which source code is converted into a stand-alone form that can be run on a computer or to the form itself. One of the most important steps of a software build is the compilation process, where source code files are converted into executable code. The process of building software is usually managed by a build tool. Builds are created when a certain point in development has been reached or the code has been deemed ready for implementation, either for testing or outright release.” [12]

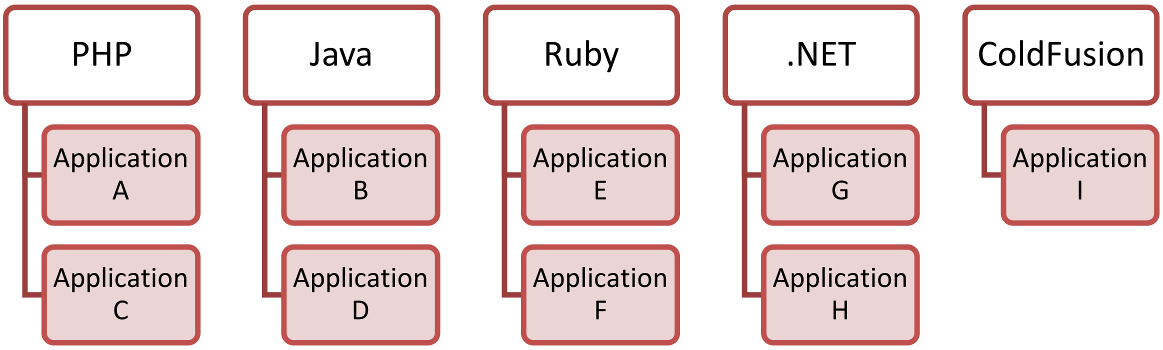


Figure 3:
List of maintained Applications (Source: Author)

To be familiar with the web applications which are part of the maintenance, I put together at least basics to understand their purpose:

- Application A - supports a group of users by providing resources and practice management services.
- Application B – supports a group of users with study materials and research including loyalty programs.
- Application C – a product database for searching information based on product code.
- Application D – a single place of training materials for a group of users, containing international researches and articles, live events and video archive.
- Application E – a central database where user data is stored.
- Application F – a tool for a group of users where they can create quick products recommendations for their customers.

- Application G – a web application presenting all products available in specific market for use by a group of users. It covers basic products information.
- Application H – an eCommerce website available in Germany where customers can order normal or prescripted products and employees are able to place orders for their customers.
- Application I – is a unique, on-line, educational experience, available at no cost to every member of the team. Through the application the customer will enhance his or her ability to understand, communicate and benefit from advocating proper products.

2.3 Maintenance Tools

Tools are not the most important key parts of a process however they can help significantly to increase a process efficiency. They support a process and team to make their work faster and usually with better results and higher quality.

Mantis Bug Tracker (Mantis) – used for reporting issues or requests; very simple and easy tool with a few customization options, not very advanced in comparison to other similar systems like Atlassian JIRA, however for many cases highly sufficient. It is used for both new projects and maintenance by all digital and client marketing, management teams across the whole region. Mantis is mostly used for development and quality assurance teams and does not cover the whole request lifecycle for all of the maintained applications but I will afterwards analyse this topic deeper. More information about Mantis can be found at: www.mantisbt.org.

Test Track Pro (TTP) – has similar purpose as Mantis - provides a way to track issues and requests. Few of the web applications have different deployment process and there TTP comes in. In general when a change is made on development environment and tested for these applications, a TTP deployment request is created and routed to an appropriate team in US who pushes the changes to stage and production environments. This happens only when all tests passed and are approved by responsible person (either client or management team). The process is described in more details in the next sub-chapter. More information about TTP can be found at http://www.seapine.com/testtrack/overview.

Zendesk – similar to previous systems, it is used for tracking issues and requests related to one of the web application maintained by a third party partner. More information about Zendesk can be found at www.zendesk.com.

Atlassian JIRA – used for tracking issues and requests for one of a Bangladesh team. The Jira is a very powerful system with many advanced customization options. More information about Jira can be found at www.atlassian.com/jira.

2.4 Maintenance Activities

As every process consists of more than one activity, in the maintenance process I have listed and highlighted the following main activities:

Monitoring and measuring the services

To ensure the applications are running properly and a warning system is set up in cases, any of the web application servers or databases go offline, which results to web application outage or its functionalities. The goal is to get a warning message and restore the services as soon as humanly possible.

Routing requests to team members

Who should work on a specific request? Is it the team A, B or C? Should it go to Java or PHP developer? Is it an application server issue or a database data manipulation is needed? To be able to determine the person, it is very important to know the applications and be able to identify the issue or request and which systems and platforms will it affect. The goal here is to assign the request to the right team member.

Communication to all involved parties

Communication with the internal teams, third-parties, internal and client management teams for the whole request lifecycle from reporting to resolving a request. When a new request is submitted, someone needs to go back to the client who reported it and say when they can expect the request to be completed or go back to him when more information is required to get the request done etc.

Evaluation of submitted requests

A new request comes in to the Mantis system and is assigned to support leader who needs to decide if all relevant information has been provided. Is it an issue, content change or user data problem? What is the impact on the system and priority in comparison to other work items in the queue? What is the budget the time spent on resolving the request will be paid from? Is the issue reproducible and what are the steps? Etc.

Development and Quality Assurance

This is the process of finding a resolution and putting in on the right place. It covers analysis and investigation, fixing a code or changing a content and testing of the changes made to the application to ensure no other parts of the web application were broken by that new change.

Deployment

Publishing the content changes or deploying code changes to specific web application environments. In general and as a best-practice there are used more than one web application environments. The following web application environments are used for this client [5]:

- Development
    - for internal purposes, most of the QA work should be done here before deploying anything to other environments,
    - usually there is no guarantee for the client it will work properly and it can contain issues (remember it’s called “development”), however in general no change should be presented to any client in this environment.
- Stage
    - on the other hand this is the release candidate (pre-production) and usually a mirror of the production environment,
    - it contains changes ready to be deployed or published to production environment after they are approved by the client or appropriate management team,
    - there are performed final testing scenarios mainly from the client side.
- Production
    - currently released version of the application, that usually contains only the approved content and functionality,

- it is live version available and to be used by public audience (clients or end users).

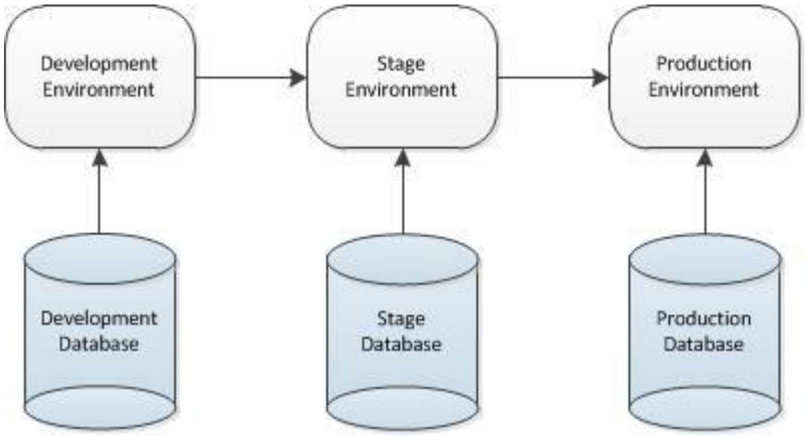


Figure 4:
Environment Architecture (Source: Author)

The deployment process is more complex than it seems to be from the first look, actually there are like four different deployment processes across the platforms (PHP, Java, .NET, Ruby on Rails). For application group A (Java, .NET, Ruby on Rails web applications), the Mantis request is routed to a specific application developer and he copies the changes to all environments as needed, either manually or by an automated process (testing the changes made to the application and passing a testing scenario is a pre-requisite). For application group B, (Java web applications) a new deployment request needs to be created in the system TTP. The TTP request goes to a deployment team and copies the changes to appropriate environments if it has passed through the testing process.

2.5 Resources and Planning

There is perhaps nothing more important than to have the best talented and loyal individuals on board who act on behalf of the company with full confidence, shared and common goal: be the best and be proud of what you are doing, and increase economic value of the company.

For our purposes it should be enough to define the resources into two categories: **People** (Human Resources) and **Money** (Allocated Budget).

2.5.1 Human Resources

The team consists of different groups of technical and management individuals who are part of various companies in different offices and continents worldwide however they feature as one team under one global company.

Development team is divided between US, Czech Republic, Poland and Bangladesh offices and all the tasks are usually managed and assigned through several maintenance tools like Mantis, TTP, Jira or ZenDesk (described more in the sub-chapter *2.3 Maintenance Tools*). When tasks are assigned, users get a notification via email and also they can view their current assignment list after logging into the tool. There is no integrated planning at all and that is also one of the reasons why sometimes a conflict happens, because there is no guarantee the individual will be available when necessary or needed due to other higher prioritized work items or projects which are running hot (by hot I mean, they are close to their deadline or already delayed). But the service level agreement (SLA) between the vendor and the client is given and signed up and should be strictly followed. Even if there are lots of other responsibilities or duties. Someone from the correspondent and qualified team has to find a gap in fully overloaded calendar even at the expense of his or her own free time and get the work done. This may be frustrating. In smaller amount it should not be a huge problem if there is kind of a compensation, but what developers hate is lack of organization in longer term and indecision, meaning too many changes in a short time. This may cause inefficiency at the team productivity or reduce the team performance rapidly.

From the management point of view there is no single client liaison contact and the requests go to different people from the Project, Support and Account Management team either through any of the maintenance tools or by email (in worse scenario via telephone call or Skype). Clients are mostly disoriented because they actually do not know who to contact primarily to help them with their request or issue. As a usual way they contact all from the management team. Not only clients have a mess within the system, but also internal team, because everyone has their own piece of information and it is very difficult to put it together in an easy way and timely manner.

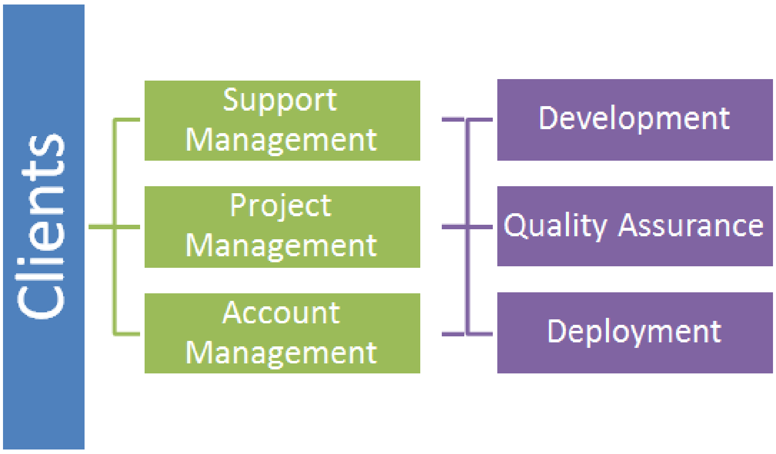


Figure 5:
Communication Matrix (Source: Author)

2.5.2 Allocated Budget

Maintenance budget is planned annually usually at the end or at the beginning of the year. It used to be separated for each of the regions EMEA, CEEME and most of the resources go to EMEA bucket due to larger amount of maintained applications available at the moment in that market area.

The estimation (the budget is calculated from) is done based on previous experience and the trend of used budget from previous year and from an increasing or decreasing part. This part depends on the situation whether there is an expectation the maintained applications will consume more hours and therefore more money (this can be due to other new projects in progress or planned to be completed in that year and thus the maintenance will need to take place for them) or less. On the other hand some of the applications may be turned off or cut off from the maintenance package, which will reduce the hours needed to keep them up-to-date and running smoothly.

One of the practices to control the budget is to divide it between the whole year per month and keep monitoring whether it is running hot or cold (over or below the average budget calculated per month). What may cause troubles is when it goes out of the control due to less or lack of continuous monitoring for any of the following reasons:

- Non-transparent or missing team timesheets *(if a responsible person is not able to see the actual time spent per task, individual or team, he is not able to control the budget at all as some pieces will miss).*

- Out of scope tasks *(when tasks from different projects are being paired into the Maintenance, even if they should be officially covered from a different budget)*.

Then suddenly the whole budget that was planned for the year had been spent in few months as it happened to the company in 2012. What will follow? There are two options: First, to go to the client and ask for more money. The other one, it needs to be covered from a different source, mostly from the internal budget or from the one where some savings have been made.

In the company's situation there is no actual budget monitoring performed continuously, only once per a longer period (half a year approximately) and it also usually causes problems I mentioned in the previous paragraphs.

3 Identifying Obstacles and Bottlenecks

Theory of Constraints (TOC) gets its name from the fact that all enterprises are constrained by something. If they weren't, they could grow as large and as fast as they wanted. But one has to monitor Wall Street or Main Street for only a moment to know that for the vast majority of enterprises growth is really hard. Constraints are why. [1, page 8]

The first thing we need to do before making any decision about changes, is identifying of the obstacles and bottlenecks. For this purpose I will use the process of ongoing improvement (POOGI) that is based on theory of constraint. In POOGI, the constraint is a reality of the system, it is not good or bad. Sometimes there can be a benefit in having a constraint in the system and sometimes not. In other cases it also depends on the goals what we want to get from the system. [6]

In this case continually check the process on X bases (X - can be weeks, months or years as example), try to find new constraints and check the already identified constrains. For each of the system constraint there is a 4 step process – the result of this process will define how we handle the constraint.

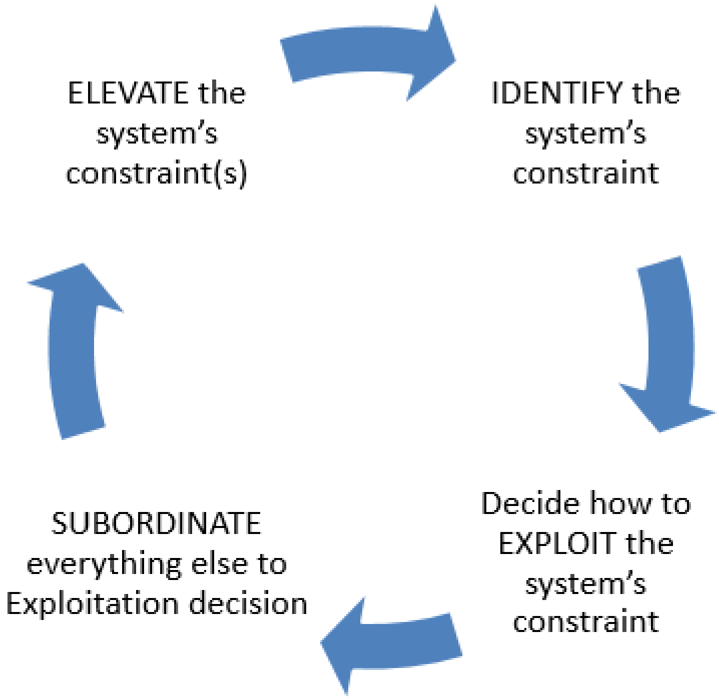

ELEVATE the system's constraint(s)
IDENTIFY the system's constraint
Decide how to EXPLOIT the system's constraint
SUBORDINATE everything else to Exploitation decision

Figure 6:
POOGI Process (Source: Author)

Because of the constraints in the system, we have the following undesirable effects (UDEs):

- Long lead time.
- Unpredictable support and maintenance process.
- Inability to balance resource requirements between project and maintenance requests.
- Inability to monitor maintenance performance, no KPIs.
- Inability to control and monitor budget (budget ran out, then we found out).

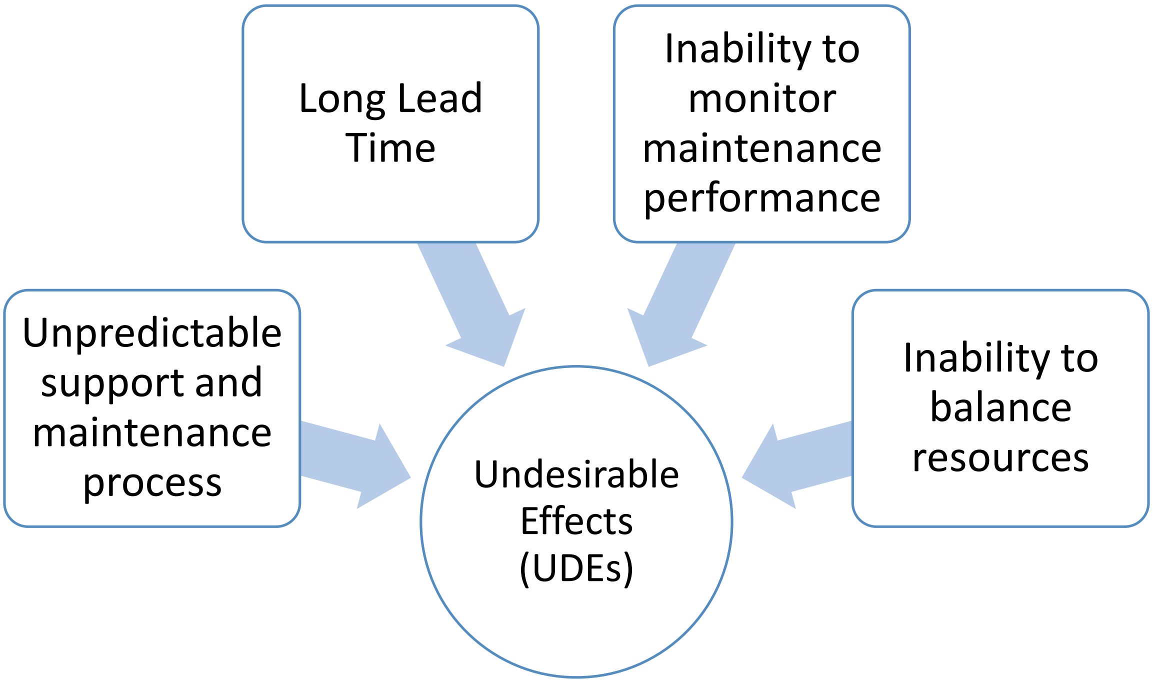

Long Lead Time
Inability to monitor maintenance performance
Unpredictable support and maintenance process
Undesirable Effects (UDEs)
Inability to balance resources

Figure 7:
Undesirable Effects (Source: Author)

As the result client satisfaction on maintenance was dramatically low in 2012. The number of requests were going up and down between 250 and 290 in the last quarter of 2012.

- 49% of requests were getting resolved in less than 3 weeks.
- 51% of requests were getting resolved in more than 3 weeks.
- 21% of the requests were getting resolved between 3-8 weeks.
- 30% of the requests were getting resolved in more than 2 months.

Additional goals to the ones I noted in the primary goals of this thesis:

- Shorter lead time – time from reporting a request to its resolution.

- Higher predictability – mainly the expectation when a request can be resolved.
- Better resource utilization.
- Budget monitoring.
- Reduce coordination and transaction costs.

3.1 Uncontrollable Work in Progress

3.1.1 Lack of request ordering and prioritization process

With the number of 250+ not resolved incidents and requests in the queue it is almost impossible to order or prioritize the work and as resources are limited there needs to be a control on top of them. There are two kinds of methods for prioritizing the requests. Most of the time the prioritization is made based on escalation process. In reality it means, that the country representative who escalated his or her request to top management, the request is prioritized and added immediately into work in progress. Based on continual escalating, client relationship cannot be built or improved and it is often very frustrating for both sides. Another model for prioritization is based on the impact on target web application. If the issue is critical or major (database or application server outage; user registration or login functionalities etc.) affecting a bigger number of end users who are target audience of the web applications and used it regularly, it is being worked on usually immediately.

3.1.2 My request is more important than anybody else type of culture

Sometimes I hear someone saying “I am more important than anybody else in the world”. Some country representatives feel they are the most important in the world and therefore we are supposed to focus primarily on their requests and get them resolved faster and as soon as possible. If I imagine to hear this from more representatives for the same client at the same time, all of them want to push their requests, what should I do? Put other client’s requests aside? This is not the right and fair way.

3.1.3 No Single Point of Contact

Everybody who is in communication with the client regularly from account management team, through project management and top management gives promises to follow up on specific requests and most of the time they follow it up on their behalf. I can say, it is essential to have more people engaged. Partially I am right, but sometimes it causes more mess and undesirable effects than I would normally wanted to accept. It is on daily basis that technical team is overhelmed, not able to determine what is more important, whose request has bigger priority, because it is coming from different people from low to top management. And what about the transparency? Many requests are coming through all communication channels (emails, Mantis system, phone calls, and personal meetings) to different people. From that point, me as a person who is supposed to be in charge of the maintenance is losing the control over all the work that is supposed to be part of the maintenance. To be able to find out the latest status of a request I have to contact all these people and ask them if someone is already working on the request, what has been explained and agreed with the client, was there a promise the request will be resolved in a certain date? These and many other questions and problems usually show up.

3.1.4 Solution

a) Not everything has the same level of value or time pressure. Some items should be moved faster than the other either because they are needed to meet client expectations or because they have a major impact on the business side. Some have to be completed on an exact date, because they are tightly connected to marketing campaigns and they start on an exact day. Some of them if they are related to end users, users who use the application should be resolved as soon as possible if we do not want to lose the users and so client's customers. Others can be completed in the normal course of flow. It means that not everything should be treated as a same item. The goal is to create an easy and transparent way of prioritizing the work to address any inherent variability. And for this purpose a Kanban mechanism called "Class of Service" looks to be as an ideal solution. [13]

   Every Class of Service has different target resolution time, which is the maximum period of time the request should be resolved. In the following table there are more details about each of the Class of Service:

***Table 1**: List of Class of Service*

Class of Service	Target Resolution	Description
Expedite	2 days	Any request evaluated as Expedite class of service must be resolved immediately. It can interrupt any other assigned tasks across maintenance or any projects for the client. Example: Web application server or database outage Issues affecting more than one user (registration or login issues etc.) Legal issues
Fixed Date Delivery	Fixed	Needs to be done on certain date. The support manager needs to be notified 3 weeks in advance. Example: Code or content change related to a marketing campaign with precise starting date. It cannot be launched neither sooner, nor later.
End User Request	14 days	Reported requests by end user who use the web applications. Example: Issues with registration or login Changing the personal data Resetting the password
Standard	21 days	All other incidents / requests that do not fit to Expedite, Fixed Deliver Date, End User Request or Intangible classes Example: Content updates Styling or minor functionality issues Generating reports
Intangible	3-6 months	Fixing of problem root causes, system upgrades or anything that can be done later and will not impact business at the present time.

b) Creation of a new prioritization model called SPR Index, which determines a final importance and value of requests and incidents in queue. It consists of the combination of several factors:

- Severity (impact on the web application = determined by the support leader).
- Priority (priority in comparison to other country representative requests = determined by the country representative).
- Country (every country has a different value and importance and it needs to be taken into the account when calculating the final priority).
- Lead Time (calculated as Due Date – Today in days; due date is set based on the Class of Service).

Formula of the SPR Index is as follows:

SPR Index = Part 1 + Part 2

Part 1 = if (Expedite; +1000), if (Fixed Date Delivery; +500), if (End User Request; +250), if (Lead Time < 0; +300 + ((-5) * Lead Time)

Part 2 = Priority Value * Severity Value * Country Value * Lead Time Value (if not <0)

Table 2: *List of Priorities*

Priority	Value	Description
Urgent	5	High business value request.
High	4	Defect, legal related or campaign based.
Medium	3	Normal request.
Low	2	Can be done at any time.

Table 3: *List of Severities*

Severity	Value	Description
Critical	5	All systems impacted.
Major	4	System impacts major number of users impacted.

Severity	Value	Description
Minor	3	A defect in the system, everything functional, minimum number of users impacted).
Cosmetic	2	Normal request or incident.

Table 4: *List of Country Values*

Value of importance	Country or division
5	Central Division, Agency, Ireland, United Kingdom, France, Germany, Italy, Spain
4	Netherlands, Belgium, Luxemburg
3	South Africa, Sweden, Noway, Finland, Denmark, Russia, Switzerland
2	Other

Table 5: *List of Lead Time Values*

Lead Time (Due Date – Today) in days	Value
< 0	300 + (5 * Lead Time)
>=0 and <=7	4
>7 and <=14	3
>14	2

c) Introducing a single point of contact for any maintenance related request. There will be no one else who can promise, decide or lead maintenance incidents and requests except the support leader who will take the full responsibility for the whole process and results and will be the primary person for any communication with all the parties. He will own the process and know all the statuses for all the maintenance requests and will be also the only one who can communicate and prioritize any work for internal technical teams. Those are the starting points.

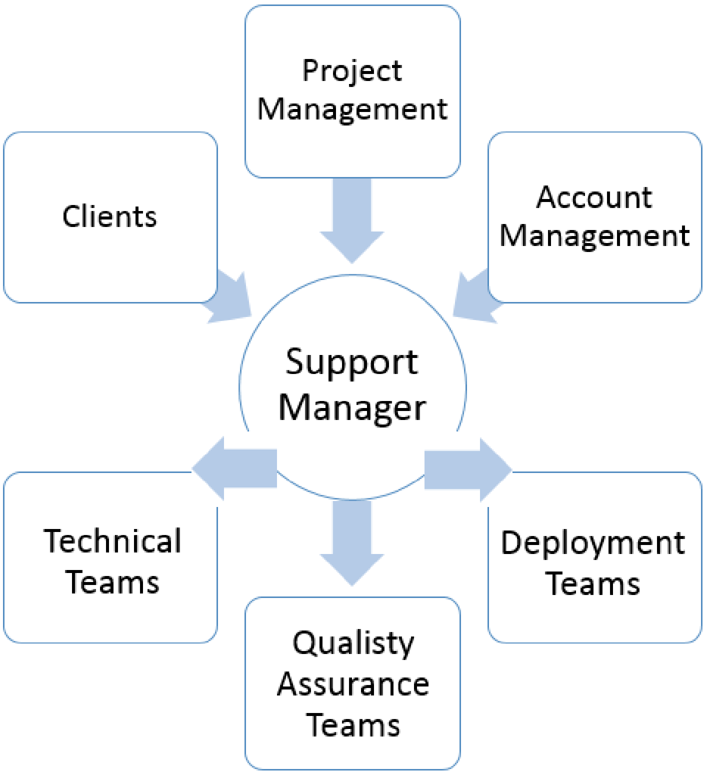


Figure 8:
Communication Matrix (Source: Author)

3.2 Waiting for client feedback and approval

3.2.1 Know-how of using support tools and the process

Usually when a new client / country representative join the team, they need an access to support tools like Mantis, so we grant the access and let them discover the insides with just basic instructions how to create incidents or requests. The problem is that some of the clients are not new generation, meaning they are not so IT experienced people and are a kind of afraid of using any new tools. They rather choose and use standard ways as emails and phone calls are. But as I already mentioned in the previous chapters, this makes the process and environment messier and the requests are untraceable and not transparent to all parties.

On the other hand, we actually do not have a proper process in place and there are no strict guidelines what can be done, what cannot be done, how it can be done and when some outputs can be expected. Not to mention there are no intro trainings for them. One person (either client or internal person) follows a way he or she thinks is appropriate, another one differently based on his expectations. Some of the requests are just sitting there and waiting for their owner to provide required information that is important to get it completed or just a confirmation and approval the request is resolved and resolution is accepted.

As the result there is an inconsistency in the way of receiving the requests and many of them are sitting like in a black hole with no reaction from their owners. Some of them are even reported in a wrong project bucket and lost in the Mantis universe from an action for a long time until they are escalated by their owners.

3.2.2 No deadline for client on providing feedback or approval

It is a common trend that the number of open (not closed) requests have an increasing tendency. We can resolve a request from development perspective, but a final confirmation and approval has to be made by its owner, the client. Therefore the resolved requests have been sitting there still opened for days, weeks or even months. This situation has several undesirable effects:

a) Inability to track the final resolution performance, so answers to the following type of question: what is the final resolution time as an average? The current options which are available do not take into the account the long period requests are waiting for a resolution confirmation.

b) Resolved (but not closed) and not resolved requests are mixed together. When I imagine 250–290 requests, it is very difficult to see what has been completed, and what has not been even started.

c) A pre-requisity to resolve a request is to have all needed information from the reporter. If any piece of information is missing, it can block the other phases of the request lifecycle and with no responding on the comments from clients, the resolution is delayed.

3.2.3 Lack of Proper Follow-up Process

With the high number of not resolved and not closed incidents and requests, it is quite impossible to have control over all of them. A usual daily activity is to go through as many of them as possible and update their status if something has changed, the clients are eager for any update and a movement forward. A common situation is that the clients ask for a status in specific requests, but nobody get back to them and answer their question. Simple there are lots of them and limited time and resources to open every single request and keep it updated regularly.

3.2.4 Solution

As David J. Anderson cites Dr. Eliyahu M. Goldratt who put it in the following way [2, page 103]: *"often reducing batch size is all it takes to bring a system back into control."*

In this case the batch size represents the number of work items (incidents or requests) planned for a specific week. Rather to pick few and complete them in high quality and with focus than let the system go out of the control, which I will explain in more details below. We intuitively know that things are easier to control when they move slow rather than fast and this fits to this case as well.

After analyzing the situation and given problems, the following solutions should help to deal with it:

a) Set up proper trainings through conference calls for all the existing and new clients including internal employees (majority of the clients had been interviewed and they see a lack of training on the process and support tools). The main focus of the training will be dedicated to support tools like Mantis and other web application administrations which clients can get into a contact. During the calls process improvements and changes will be covered as well.

b) Preparation of offline training materials to explain all the systems, tools and process itself (this was actually one of the demands from the clients and they will really appreciate it and I can understand the point).

c) If no feedback or confirmation will be provided within the two weeks from the last client reaction into the request, it will be suspended and closed. It will be possible to reopen it again when needed and the required information is available. In case of final confirmation / approval of the resolution, the full responsibility will go to QA department or support leader and only occasionally to the client if the type of a request requires it.

d) To prepare and send a maintenance weekly work plan with list of incidents or requests. To be able to imagine how it would work, the support leader will have a backlog of all not resolved requests and choose the first top 40-50 requests sorted / prioritized based on the SPR Index. If the SPR Index is bigger, it has higher priority. This list will be shared with internal teams and also with all clients every week. The support leader will prepare the list every Thursday and share it with technical teams and they will be responsible for having a quick look at the requests they will be supposed to work on. On Friday each of internal support team members will have a conference call to go mainly through complicated requests and discuss either a possible solution or making an estimation how much time they will need for completion of those requests. The meeting should take 30 minutes as a maximum and a benefit in having everyone on the call will be, that each of the team members can discuss how and if a change they will do, affect other web applications which other teams maintain to prevent from causing other issues or complications in delaying a resolution of a request. And finally there will be a commitment the planned requests will be completed in the upcoming week. In case a longer discussion will be needed among only few of the members, they will do it after this conference call separately. After the meeting the support leader will make a final update of the weekly plan and share it with all of the teams and this time also with all clients. An inspiration has been taken from the Scrum, agile project management methodology where all team members are supposed to have a daily stand up meetings. In such meetings they discuss what they worked on the day before, what is their working plan for current day and what obstacles they are experiencing with the tasks if any.

Definition of Scrum [14]:

“Scrum is an agile methodology that can be applied to nearly any project; however, the Scrum methodology is most commonly used in software development. The Scrum process is suited for projects with rapidly changing or highly emergent requirements. Scrum software development progresses via a series of iterations called sprints, which last from

one to four weeks. The Scrum model suggests each sprint begins with a brief planning meeting and concludes with a review. These are the basics of Scrum project management."

The expected benefits:

- Technical teams – they will know exactly what they are supposed to work on every week.
- Clients – they will know when they can approximately expect to have their requests being resolved.
- Support leader – focusing only on a short list of requests instead of all requests in the queue.

Example of a weekly plan to be shared with clients.

Table 6: *Example of the client weekly plan (Source: Author)*

Ticket No.	Application	Summary	Reporter	Date Submitted	Planned Lead Time (days)	INDEX
19997	Application C	Learning Center - access not possible due to automatic redirect to application	Radomil	21.8.2014	-184	1375
21025	Application D	Clinic cannot prolong prescription	Martina	26.1.2015	-33	1275
20891	Application D	Address update in shop does not work properly	Martina	29.12.2014	-61	1200
20859	Application F	Reco print outs need units of weight adding	Tim	6.2.2015	-15	1200
20899	Application D	Shop codes not working	Martina	28.1.2015	-24	1100
21160	Application E	Products - info in French	Stefan	11.2.2015	-10	980
21126	Application F	New product A is not available in the list product	Cristiana	10.2.2015	-11	950
21225	Application B	nl-nl/app/dkl deventer	Eliza	19.2.2015	-2	860
20990	Application E	Products - Transmit Dutch translation BEN --> NL	Stefan	16.1.2015	-36	860
20919	Application G	CRM data Greece	Anna	6.1.2015	-46	850
21106	Application E	EST translation update	Anna	5.2.2015	-16	710
21247	Application E	Change language causes crash	Michal	24.2.2015	3	480
21292	Application E	Extract for eCommerce Q1	Cesar	2.3.2015	9	270
21326	Application A	CRO Where to buy section data	Anna	9.3.2015	16	112

Example of a weekly plan to be shared with technical teams.

Table 7: *Example of the team weekly plan (Source: Author)*

Ticket No.	Application	Summary	Solver	INDEX	Development	Testing	Deployment	Verifying
19997	Application C	Learning Center - access not possible due to automatic redirect to application	Jiri	**1375**	🟩	🟩	🟨	🟨
21025	Application D	User cannot prolong prescription	Lonnie	**1275**	🟨	🟥	🟥	🟥
20891	Application D	Address update in shop does not work properly	Lonnie	**1200**	🟨	🟥	🟥	🟥
20859	Application F	Reco print outs need units of weight adding	Marcin	**1200**	🟨	🟥	🟥	🟥
20899	Application D	Shop codes not working	Lonnie	**1100**	🟨	🟥	🟥	🟥
20657	Application F	QR - FR No mix ration available 1_12_14	Nicolas	**1100**	🟨	🟥	🟥	🟥
20990	Application E	Products - Transmit Dutch translation BEN --> NL	Lonnie	**860**	🟥	🟥	🟥	🟥
20919	Application G	CRM data Greece	Miroslav	**850**	🟨	🟥	🟥	🟥
21106	Application E	EST translation update	Lonnie	**710**	🟥	🟥	🟥	🟥
17074	Application F	choosable conditions wrong in application when language is french in switzerland	Nicolas	**710**	🟨	🟥	🟥	🟥
20212	Application E	DELETE DOUBLED PRODUCTS	Lucas	**650**	🟨	🟨	🟨	🟨

Table 8: *Action steps (Source: Author)*

Undesirable effects	Solutions
Lack of know-how of using support tools and the process	Conference call trainings. Offline training materials.
No deadline for client on providing a feedback or approval	QA or support manager can make a final resolution approval. If no feedback is provided within the two weeks from the last reaction, the request is closed.
Lack of proper follow-up process	Weekly Maintenance Meetings Weekly Work Plans Limitation of Work in progress to 40-50 requests per week maximum

3.3 Resource Contention

3.3.1 Shared Resources between Projects and Maintenance

All the resources have been shared between projects and maintenance for all the teams in different locations, but usually the main focus and priority are put on new projects where deadlines are tight and connected to marketing campaigns scheduled to be launched on a precise date. With all of this pressure the maintenance focus is put aside as a second priority which is also one of the reasons the number of not resolved requests have started growing so fast and suddenly is so high at the end of 2012. But the resources are not shared only between projects and maintenance for the same client, they are being actually shared across multiple clients and even among various digital partner agencies. Due to no integrated planning and visibility for other teams there is no guarantee that a specific resource will be available in a certain day by a certain time.

Having this way of mind set and process in place, the maintenance cannot be successful at all and of course the client happy.

3.3.2 Limited resources and bottlenecks

Me as a support leader I am not only overseeing the maintenance process and work, but also participating in several projects as needed and it usually consumes a lot of my working time (around 20%) at the expense of less focus on my primary role which is maintenance of the projects. This have usually resulted in lack of communication with the clients about status of their requests, proper evaluation of new requests which are coming every day and lots of back and forth with the clients and technical teams as proper requests analysis has not been done before having passed it to technical teams. I can consider myself as a bottleneck of the process who has limited working hours and has to cover all the maintenance requests and also take care of some projects activities which is not manageable based on the current situation and trend of open / not resolved client requests.

Another bottleneck identified in the team is a ruby developer who has maintained and developed new projects for critical business website applications. There are two problems. On one side this person has being contacted every day by different managers from account management, through project management and top management people and all of them want to complete their request the same day and communicate it as an urgent case. As a result this person is unable to identify the priority of the requests and almost nothing is done as planned and on time. On the other side most of the focus is on new projects instead of maintaining the existed ones and resolving incidents or requests that are being reported.

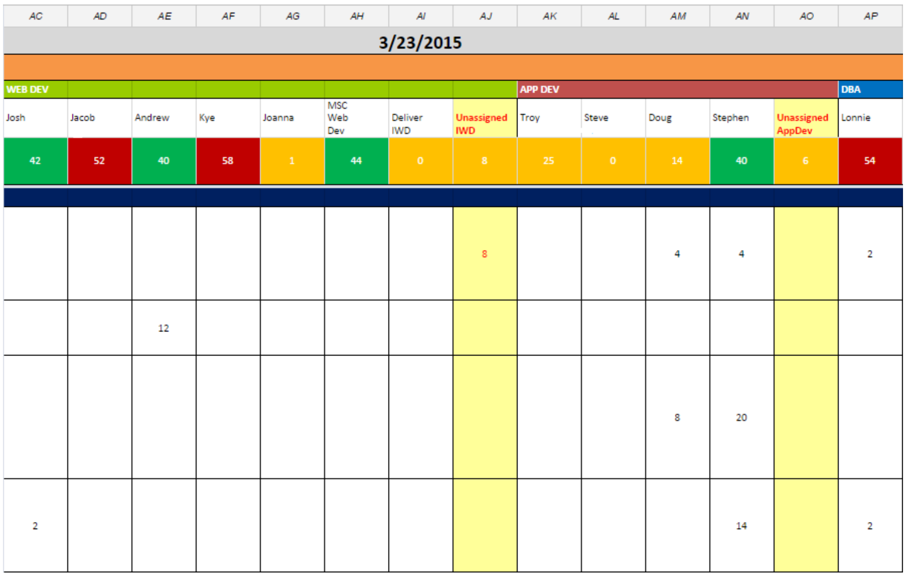
Same situation is with QA lead. There is only one QA lead in US covering all the regions EMEA and US. He is under pressure by both of the teams in US and Prague and often the requests have been completed and ready for testing, but it takes days the QA lead is able to look at them which is slowing the process down rapidly and delaying the fixes and changes to be published to production environment.

3.3.3 Solution

Resources Planning - As the environment is so complex, the resource planning has to be improved, integrated and visible for competent team members. The following points need to be implemented and accepted both internally and globally (again the inspiration has been taken from Scrum methodology):

a) Resources shared with US and Prague offices (global resources) – creation of a shared document with list of projects and resources to be shared between the projects and maintenance in US and Prague. Every Friday there will be an integrated planning conference call with all the project managers who manage any of the projects in the list. Before the call starts each of the project managers will be supposed to put the specific resource allocations into the spread sheet for any resource they will need for upcoming 2-3 weeks in advance for their projects. During the call, each of the project managers will provide a basic overview and status of his or her projects and highlight critical milestones or resource needs for the upcoming week. The same day the project director who manages the projects portfolio and resource allocations will send a meeting minutes including hot project items to be prioritized over other items in the queue in the next week.

The global resource planning spreadsheet will be used and shared across various digital agencies and offices worldwide.



AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP
3/23/2015													
WEB DEV								APP DEV					DBA
Josh	Jacob	Andrew	Kye	Joanna	MSC Web Dev	Deliver IWD	Unassigned IWD	Troy	Steve	Doug	Stephen	Unassigned AppDev	Lonnie
42	52	40	58	1	44	0	8	25	0	14	40	6	54
							8			4	4		2
		12											
										8	20		
2											14		2

Figure 9:
Resources Planning Sheet (Source: Author)

b) Resources shared with other clients (internal resources in Prague only) – Creation of an internal planning spreadsheet similar to the one above and discussed only internally with project managers and delivery manager who will own and manage the document on one side and coordinate the resources and needs on the other side.

The internal resource planning spreadsheet will be used and shared only internally for Prague team.

c) Resources shared between projects and maintenance in Prague (both internally and globally shared resources) – To be able to prevent from the situations like “only projects are important”, there needs to be an agreement between Prague project managers and support leader that there will be always a precise time dedicated to maintenance for any of the resources used on weekly basis. The amount of hours each of them may spend on maintenance will be 8h which is twenty percent of their working week.

Example: *Franta is a Java developer who may spend 32h on projects and 8h on maintenance every week with an assumption there is 40 working hours per week.*

In situations there is no work item in maintenance to be worked on, the person will not just sit on a chair and be looking out of a window, but will again continue with project items.

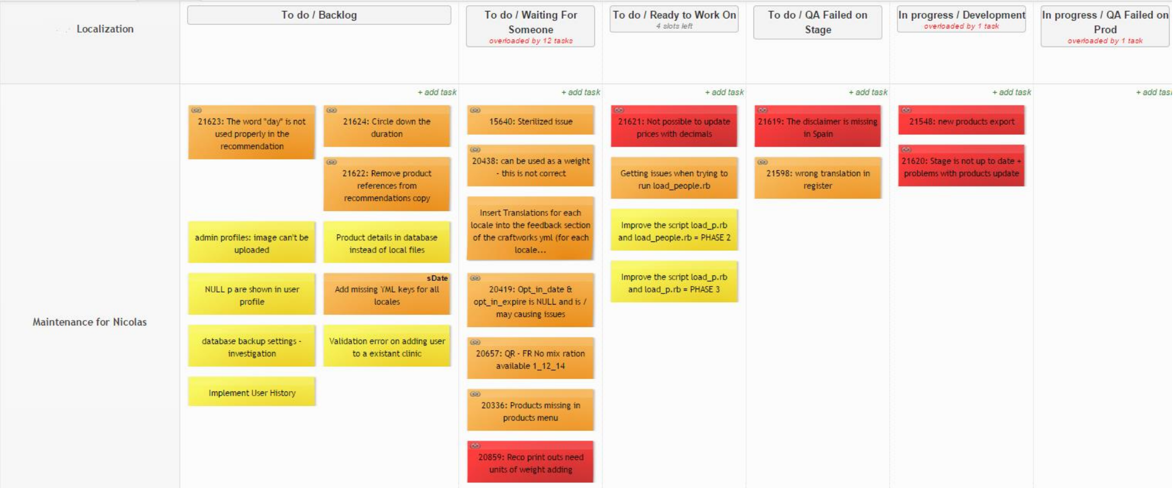
And why 8 hours? The estimation of all the maintenance requests in queue and with the objective to decrease the number of open incidents and requests under 50 within the next three months, will require that each of the team members who will participate on resolving the requests, will need to spend around 8h on maintenance every week. Of course there is not the same number of requests per each of the technology and there is not same number of team members programming in a language, so some of them may reach the target sooner, other later. However to keep a long term standard the team have decided to keep it same across all technical team members.

Addressing the Bottlenecks

For the support leader is the primary key to have the maintenance under control and proper communication in place by limiting the work from projects and maintenance to only maintenance, so the full focus will be dedicated only on maintaining the current projects.

To be able to react faster when a request is resolved in development and to not hold the resolution to be pushed in production environment, the company will hire a second QA lead dedicated only for Prague team.

The case with Ruby developer will need more attention and some of the solutions I have already outlined in the previous sections like only project and support management can plan the work for technical teams and the plan cannot be interrupted by adding any other items before agreeing by responsible project / support leader. Also as most of the open / not resolved requests need to be resolved by this Ruby developer, to reach the goals of decreasing not resolved requests will require to hire at least an additional external Ruby developer for one month dedicated to work on maintenance requests only. As these team members are bottlenecks, it is important to increase the visibility over the work what is currently in progress and planned for every week and manage their tasks closely.



Localization	To do / Backlog	To do / Waiting For Someone overloaded by 12 tasks	To do / Ready to Work On 4 slots left	To do / QA Failed on Stage	In progress / Development overloaded by 1 task	In progress / QA Failed on Prod overloaded by 1 task
Maintenance for Nicolas	+ add task 21623: The word "day" is not used properly in the recommendation 21624: Circle down the duration 21622: Remove product references from recommendations copy admin profiles: image can't be uploaded Product details in database instead of local files NULL p are shown in user profile sDate Add missing YML keys for all locales database backup settings - investigation Validation error on adding user to a existant clinic Implement User History	+ add task 15640: Sterilized issue 20438: can be used as a weight - this is not correct Insert Translations for each locale into the feedback section of the craftworks yml (for each locale... 20419: Opt_in_date & opt_in_expire is NULL and is / may causing issues 20657: QR - FR No mix ration available 1_12_14 20336: Products missing in products menu 20859: Reco print outs need units of weight adding	+ add task 21621: Not possible to update prices with decimals Getting issues when trying to run load_people.rb Improve the script load_p.rb and load_people.rb = PHASE 2 Improve the script load_p.rb and load_p.rb = PHASE 3	+ add task 21619: The disclaimer is missing in Spain 21598: wrong translation in register	+ add task 21548: new products export 21620: Stage is not up to date + problems with products update	+ add tas

Figure 10:
Kanban Board Planning (Source: Author)

3.4 Lack of System Access

Within the maintenance, there are processed annually around 1200 incidents and requests and around 50 % of these requests can be completed by a less technical experienced person (at the moment it is only around 20%). If the support leader is capable of doing such stuff, the capacity of the team can be increased effectively instead of passing it to developers or other technical teams who should focus rather on more complex issues and requests. Here is an example of such requests:

- User data management (changing password to access a user account, changing other personal information).
- Login, registration and other user related issues troubleshooting.
- End user database management and administration.
- Generate ad-hoc reports from database and prepare segments for emailing.

- Manual user data imports / exports.
- More accurate issue analysis from the log files before passing an issue to technical teams.
- Grant access and permission to different applications.
- Act as a backup for content management when needed.

3.4.1 Solution

When we open few more “doors” to allow the support team lead to access different application administrations, database and servers and I will assume this knowledge is adequate to be able to work with them with a minimum of training, 50 % of the requests which represent around 600 requests per year can be resolved by this role. And more technical people will have more capacity to focus on bigger and more complex tasks. This would bring a significant team and process effectivity with no or minimum cost.

It would have also other side benefits. When having an opportunity to work with different tools and be able to see the applications from different sides, it should provide a deeper knowledge of supported applications. This would be something very helpful when communicating with the clients and internal teams or recommending improvements and solutions.

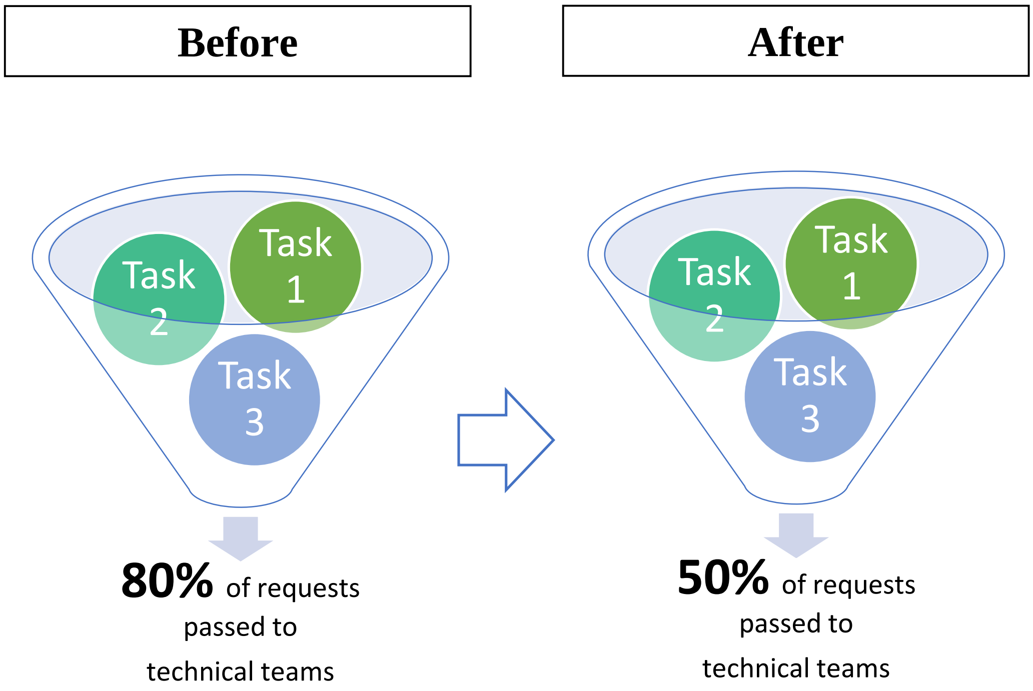

Before
After
Task 1
Task 2
Task 3
80% of requests passed to technical teams
Task 1
Task 2
Task 3
50% of requests passed to technical teams

Figure 11:
Reducing the number of requests passed to Technical team after granting more access to Support Leader

3.5 Inefficient Deployment Process

At the moment one of the supported web applications for which we receive majority of the client requests is built on Java and shares the code base between Europe and US regions. The projects are separate in each of the regions and there is no visibility for Europe team to see what US projects are in the plan or progress and vice-versa. However both of the teams maintain, develop and improve the application separately and it makes the deployment process more complicated.

There have to be done detailed code reviews for any code change (excluding content changes done via CMS) before deploying them among any of the environments (dev, stage or production). The reason of these code reviews is to ensure that the change will not break the application in other region and vice-versa and are completed by a developer who was not involved in the code change. As such reviews are time consuming and take from minutes to hours (each case is individual), they cannot be performed often. They are usually planned once a week, which is also one of the reasons why many client requests are delayed and in queue due to the deployment process complexity.

But the team also has to ensure that all the changes are ready to be deployed to specific environment, which means they are fully tested by QA department and hence do not contain any bugs and all the changes are approved (either by project/support manager or client).

In urgent cases an ad-hoc deployment is done also out of this scheduled window, but this happens rarely.

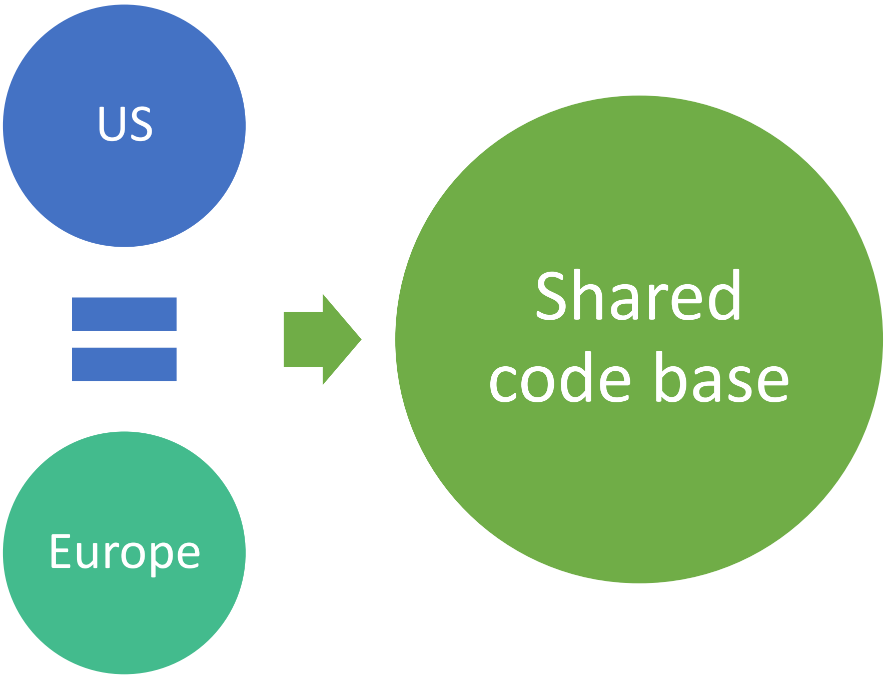


Figure 12:
Shared code base (Source: Author)

3.5.1 Solution

The ideal solution would be to separate the code base to allow US and Europe to manage their application code separately with no connection to each other. This would eliminate the existence of code reviews as the primary purpose of them will no longer be valid. By simplifying the deployment process and reducing the effort needed to deploy changes among the environments, there can be two or three deployments every week instead of only one as it is right now. In case any issues are discovered after deploying the changes it will not take another week to fix them and re-deploy again, but the team should be able to do it the same week.

This should reduce the overall request lifecycle significantly.

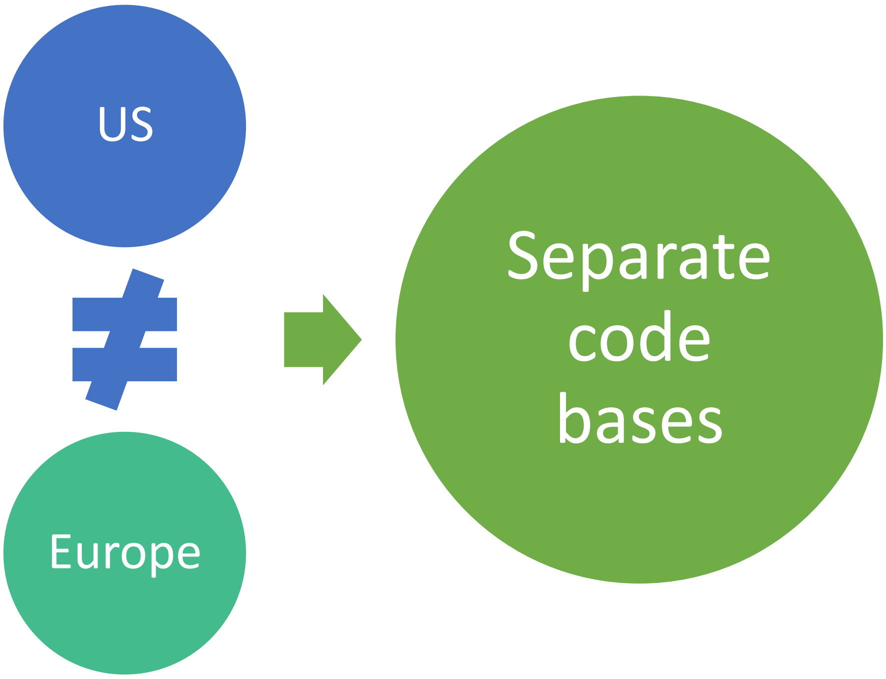


Figure 13:
Separate code bases [Source: Author]

3.6 Waste and Duplicated Tasks

3.6.1 Too many support tools

As noted across the previous chapters, there are used several tools for maintenance. It is JIRA, Mantis, Zendesk, TTP and other standard Microsoft Office applications like Excel, Word for planning and sharing reports and information.

This leads to a complex and messy environment as most of these tools have been build to aim the same or similar goal, which is to track issues and bugs within an organization or for project management purposes. It is not unusual, that we re-create some requests from one tool to another one and thus waste our internal time and client's money for no additional value. As example, when a new request is submitted in Mantis and is related to specific web application, the deployment process is going to take place in TTP tool that is used by deployment and QA team in US. It means most of the information from Mantis needs to be just copied into a new TTP request. And there are many similar cases where such wasting is obvious.

Also most of the tools do not support separate workflows for different projects or there is no way or very difficult to report on important metrics.

When I will make regularly weekly plans and reports as suggested in the subchapter *3.2.3 Lack of Proper Follow-up Process*, there is currently no way to automatize some parts of the process and it will have to be done manually. To reduce the time spent on such activities may be reduced by a proper tool with needed setup in place.

I have identified and summarized a list of problems:

- No integration with other systems or tools (project management tools, web application tracking tools, documentation systems, git source storage, release and build management etc.).
- Difficult or unable to report on various metrics.
- Difficult or unable to customize project workflows.
- Difficult to prepare weekly work plans or make updates.
- Difficult or unable to plan maintenance items based on agile methodologies (scrum, kanban).
- Planning and updating plans manually using excel (weekly planning and reporting).
- Difficult or unable to see/export reports in a minute.

3.6.2 No integrated knowledge base

There was already a situation that someone from the team had decided to accept a new professional challenge and leave the team. Yes, there is usually a plan to train and transfer as much knowledge and practises as possible to a new replacement, but it is not enough or possible to cover all cases. And therefore the learning curve starts almost like from scratch because there is no single up-to-date documentation or knowledge base where the application parts are described also from technical perspective. It causes very slow progress when resolving either application issues or developing new functionalities in the next weeks or even months.

There are few documentations but spread out in various tools and local files in different locations which makes it very difficult to go through and find relevant information.

A knowledge base is also important for documenting known errors that had been resolved and can be reused in the future again. Or for clients to learn how the systems works and how to work with some of the web application where an administration exists and is supposed to be used.

3.6.3 Solution

To reduce the number of support tools which as described in most cases just duplicate the work, we will keep only one that will be used across all the partners and regions. I have prepared some rough calculations how much money only the team in Prague inefficiently or absolutely unnecessarily consumes. These are also client's money that may be used far more effective.

There are around 50 team members in our team who use the current tools and have the same experience of inefficient wasting. When each of them spends 5 minutes of their work on duplicating the requests from one to another system every day, it is 21 hours per person yearly. The blended rate is approximately 60 USD per hour of work as average. On the other side, all the licenses and trainings are estimated to around 40,000 USD (primary licenses = 32,000 USD, other licenses = 3,000 USD, trainings = 5,000 USD) as total per year. There are included also licenses and trainings for an integrated knowledge base.

When deducting the cost for the licenses and trainings from the savings I expect to achieve, I should even recognize a positive balance of 23,000 USD every year after implementation of the tools.

I am not taking into the account the licenses for the current support tools as they are being owned and used by different agencies and partners in the pool for their other clients.

63,000 USD
−
40,000 USD
=
+23,000 USD

Figure 14:
Expected savings after reducing the number of support tools (Source: Author)

3.7 Definition what is and what is not a maintenance request

Every single project or a task should have a definition of the scope, even it is small or big. It may be defined as maximum amount of something per unit. The unit can represent in this context a single project or maintenance request. To replace the term "something", I can use as example amount of hours needed for completion of a request limited only for specific web application, or only group of web pages and functionalities etc.

Within the maintenance process there is a limitation what kind of requests can be processed and which web applications are under the support, however one of the important piece is missing and it is time. It represents how many hours the team can spend on a request during its lifecycle from initiation until deploying or publishing to production environment. In case I see a request is going to take more than 100 hours, the team does not continue and stop working on the request and rather discuss with the client to consider it as out of scope request and include it into a separate, incremental project. What does an incremental project mean? It has clear scope definition, work estimation for all roles who will participate on the request and a separate approved budget out of the maintenance budget.

Often there are processed simultaneously few requests which take tens of hours and as there are limited resources and time they can spend on maintenance, it is slowing down the progress on decreasing the open requests and react promptly on newly reported issues or changes. As a result, the trend of open requests looks as follows:

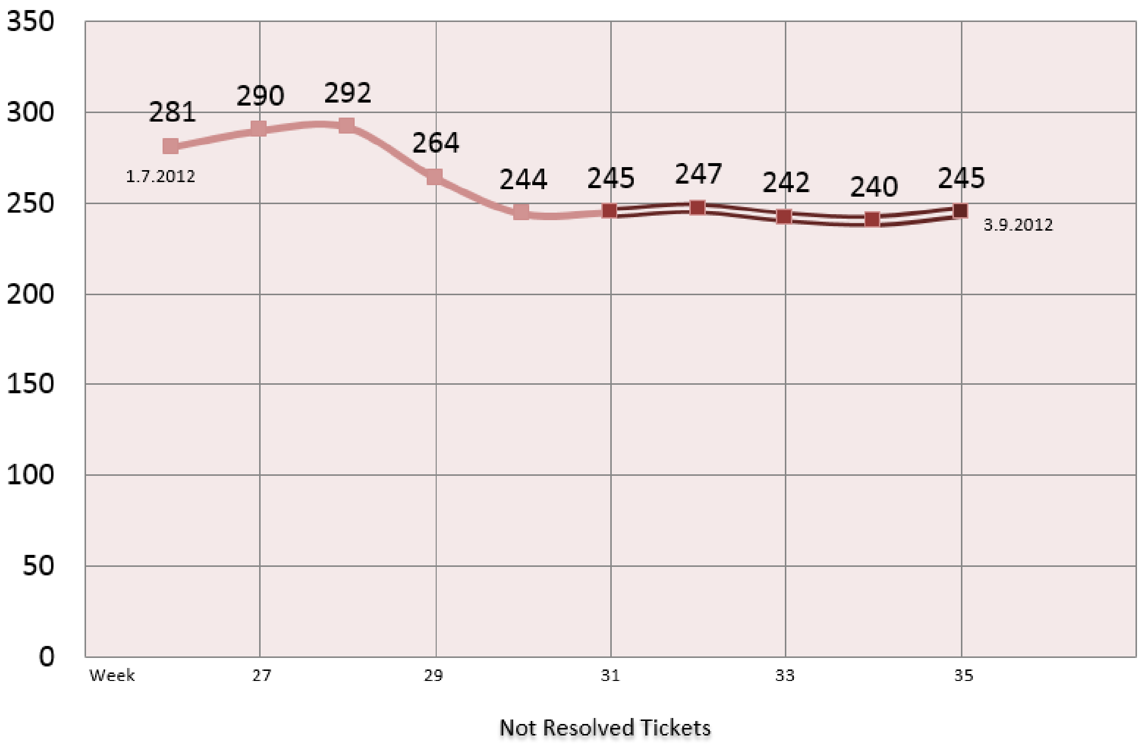


Figure 15:
Open / Not Resolved Requests Trend Jul-Sep 2012 (Source: Author)

But the issue is not only with the fact that there is a very slow progress in decreasing the number of not resolved requests. The bigger problem is that the actual budget spent on maintenance at the end of 2012 is almost 50% over the plan. This cannot be acceptable in the next year, not with the results like in the Figure 11 above.

3.7.1 Solution

I have agreed internally and with the client that a request to be considered as a maintenance request may take up to 8h for its completion. These 8 hours represent time for all team members who will participate on its resolution including the management team.

I have categorized and identified that around 30 % of the current not yet resolved requests will consume more than 8h of work for its completion. So 30 % of them which is approximately 85 requests will be routed as a separate incremental project and will be excluded from maintenance as it is supposed to be so far (categorization as out of scope requests).

For better budget monitoring and controlling, there will be performed an internal review on the actuals on monthly bases. This should provide the needed visibility on the spending regularly and take necessary actions when needed in advance.

This should bring several positive outputs:

- The number of maintenance requests will decrease down to around 200 (when throwing out 30 % of the incremental requests).
- Team will not spend days on focusing on one or two requests, but they will be able to process more requests with less effort needed.
- Better and transparent budget monitoring and control.

3.8 Lack of Measurements and Reporting

Some of the metrics and reporting are shared with management and clients on weekly bases however it does not cover all metrics useful and needed to be tracked and measured. Also a consolidation, revision and a pattern of the reports is needed to be created.

At the moment are measured:

- Total number of reported requests (weekly bases).
- Total number of resolved and closed requests (weekly bases).
- Total number of not resolved requests (weekly bases).
- Total number of reported requests (weekly bases).
- Total number of not resolved requests per status (weekly bases).
- Total number of not resolved requests per project (weekly bases).

Moreover there have not been done any proper evaluation of the maintenance process and its achievements yet at the end of any year. This is very important to identify and review the results for every year and make necessary decision when and where needed. Example of the reports we do at the moment is as follows:

Not resolved requests trend (number of requests vs. number of weeks in 2012):

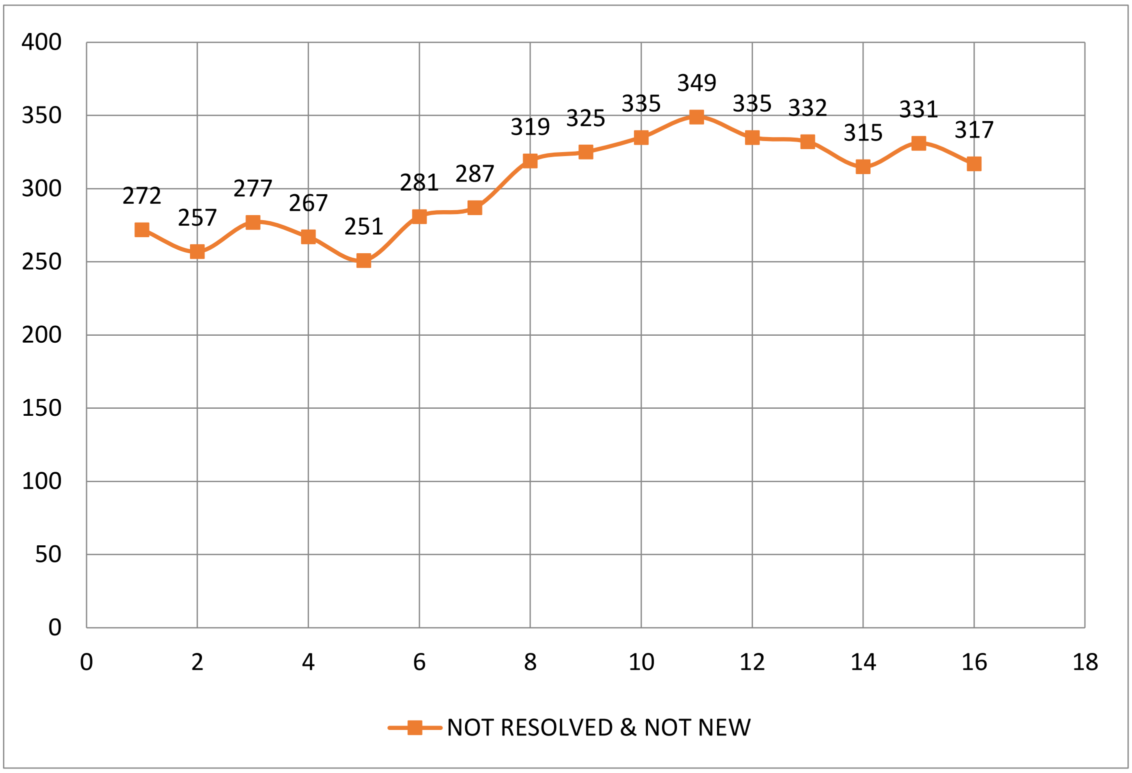


Figure 16:
Trend of open incidents and requests in 2012

Total number of reported and resolved requests (number of requests vs. number of weeks in 2012):

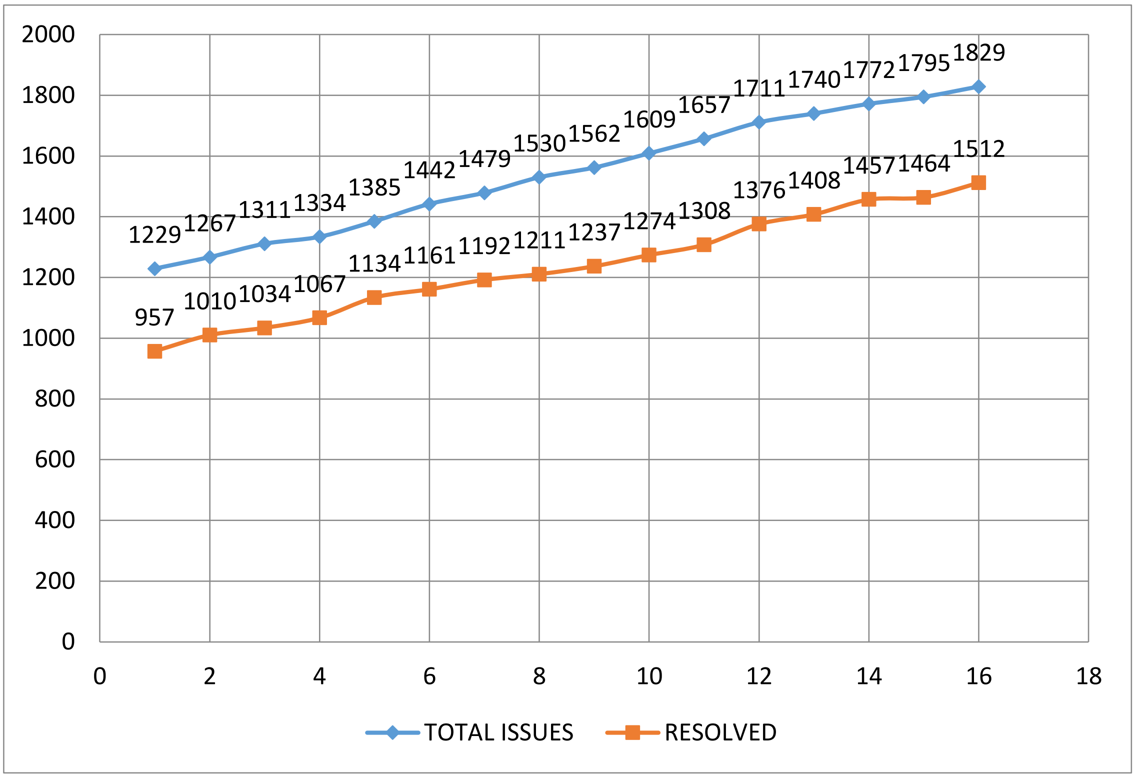


Figure 17

Trend of open and total reported incidents and requests in 2012

3.8.1 Solution

I have put together a list and overview of the measures that should be tracked and will need to be set up. They have been agreed together with internal teams and clients to provide requested overview on the performance of the maintenance as a whole and teams on weekly bases. They are separated between weekly measures and annual measures:

- Weekly measures – measure weekly performance and provide an overview of the work completed in the previous week and work in queue.
- Quarter and annual KPIs – provide an overview on the KPIs.

Weekly Measures[1]

- **Measure A1**[2]: Number of not resolved requests per country and number of requests where more information is needed from the client (country will be calculated based on the request reporter).

  <u>Goals:</u>

  - To see which countries are more active in terms of submitting new requests and/or which countries are identified with high number of issues.
  - To see which country representatives are supposed to provide a feedback and see a trend in comparison to previous week(s).

- **Measure A2**: Number of not resolved requests per class of service (total vs. in queue vs. waiting for client).

  <u>Goals:</u>

  - To see the number of requests in each of the class of service categories as each of them represents different importance that needs a separate focus.
  - To see the number of requests in queue, so we can identify and track a lack of resources and take necessary actions if needed.
  - To see the number of requests where a client feedback is needed.

- **Measure A3**: Number of not resolved requests per application and type.

  <u>Goals:</u>

  - To see and identify the defectiveness of the particular applications.
  - To see the proportion of the submitted requests based on the nature of requests (issue vs. change request vs. end user request).

[1] The visual of the reports can be found in the appendix.

[2] A1-5 is related to not resolved requests.

- **Measure A4**: Number of not resolved requests per application and lead time.

  Goals:

  - To see how many of not resolved requests already exceeded the planned lead time and how many of them are still within the planned range per application.

- **Measure A5**: Number of submitted requests on weekly bases.

  Goals:

  - To see and track the trend of requests submission.
  - To see the trend of request types being submitted.

- **Measure B1**[3]: Number of work in progress (WIP) per lead time and status.

  Goals:

  - To see how many of requests have been planned for the current week in each of the class of service categories.
  - To see how many of requests planned for the current week have been already resolved and the planned lead time has been met or exceeded.

- **Measure B2**: Number of work in progress (WIP) per status in each of the request lifecycle.

  Goals:

  - To see the status in each of the request lifecycle and measure the performance of the teams (development, QA and deployment).

- **Measure B3**: Number of work in progress (WIP) per team members.

  Goals:

  - To see how many of requests are planned for specific team members.
  - To see and measure the performance of specific team members.

[3] B1-3 is related to work in progress.

Quarter and annual KPIs

- **KPI 1**: How many of the requests had met or exceeded the planned lead time per class of service (in %).
- **KPI 2**: What is the minimum, maximum and average resolution of the requests per class of service (in days).
- **KPI 3**: How many of the requests had met or exceeded the planned lead time per standard class of service and specific technical teams (in %).
- **KPI 4**: How many of the requests had been resolved by specific technical teams (count).
- **KPI 5**: How many of the requests had been resolved per country (count).
- **KPI 6**: How many of the requests had been resolved per application (count).
- **KPI 7**: What is the average of not resolved requests at the end of each month (count).
- **KPI 8**: What is the proportion of the resolved requests between “change request”, “issue” and “end user request” (count + %).

4 Process Evaluation and Achievements

4.1 Decreasing the Number of not Resolved Requests

Goal: To reduce not resolved requests under 50 within the first three months.

The implementation of the process improvements had started as suggested sometime around December 2 in 2012 and within the first four months of my execution, the number of not resolved requests went rapidly down from 285 to 54 in April 1 in 2013.

The initial goal was to get not resolved requests down under 50 within the three months which was not met as planned, however this result was far more beyond the expectations of both internal management and the client. It was a great achievement as it took only few months in comparison to previous years where the number had just been growing.

The critical parts to this result were to categorize the requests based on the scope of maintenance (in scope vs. out of scope), have the team members available every week as agreed for 20% of their working time as minimum and of course following the rules and process policy and be strict with a minimum of exceptions.

To sum it up, the results were classified as successful and bringing satisfaction to everybody. In July 2013 the number of not resolved requests went historically down to 29 only!

Currently (April 2015) the number increased slightly to an amount of around 60 due to other obstacles that will be analysed and addressed in a next round of improvements. The process of ongoing improvements as I can see never ends, and it requires continual process evaluation and addressing new root causes.

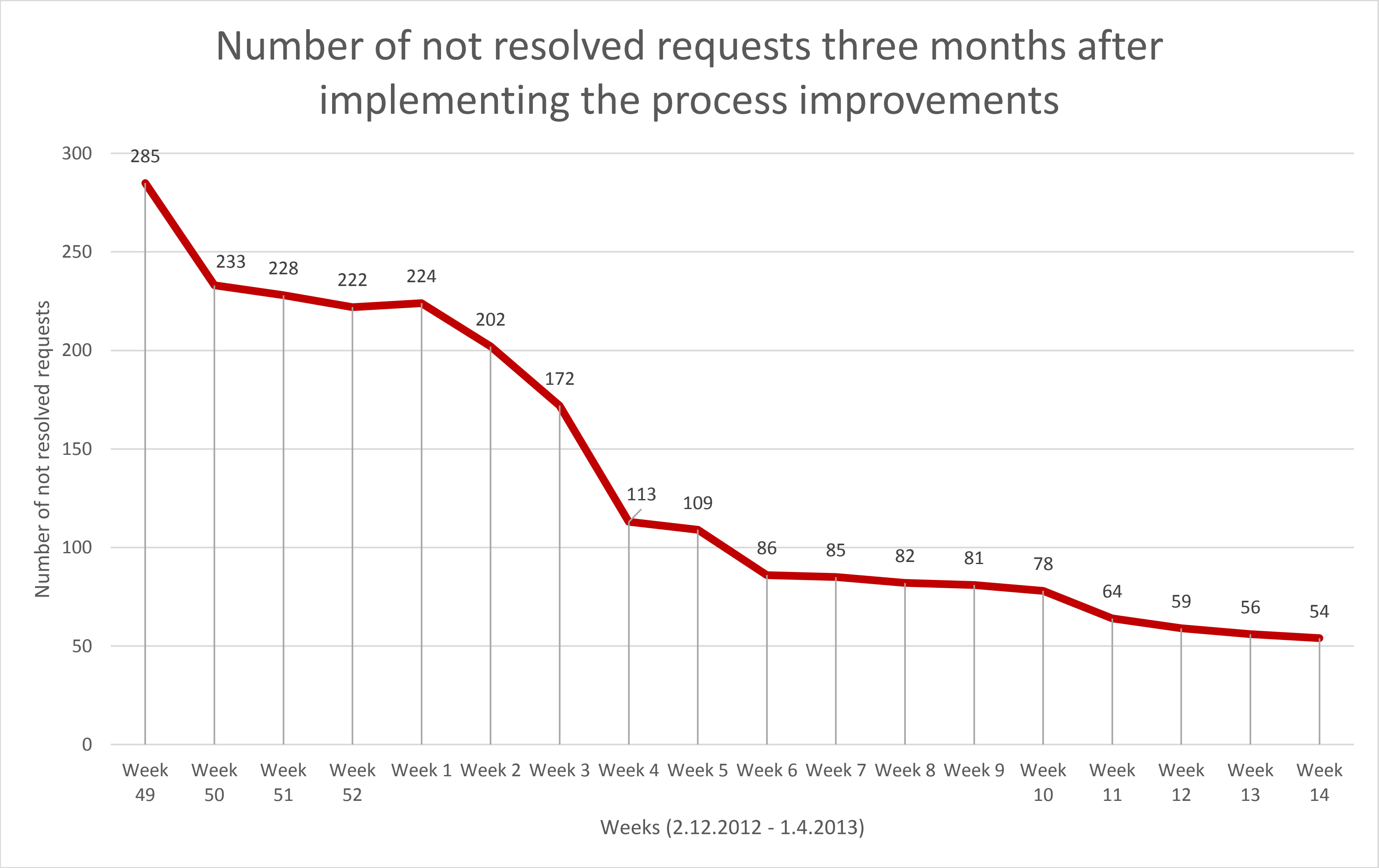


Figure 18:

Number of not resolved requests three months after implementing the process improvements (Source: Author)

4.2 Retainining Down of not Resolved Requests

Goal: To retain the number of not resolved requests around up to 70 at the end of every month.

I have put together an overview for the last two years (2013 and 2014) after the process improvements and the trends can be found on the following slides. In 2013, the average number of not resolved requests was 63 and in 2014 the number slightly increased to 71. 50 % of these requests are in queue, 25 % are waiting for a client feedback and the last 25 % are in development, testing or deployment phase.

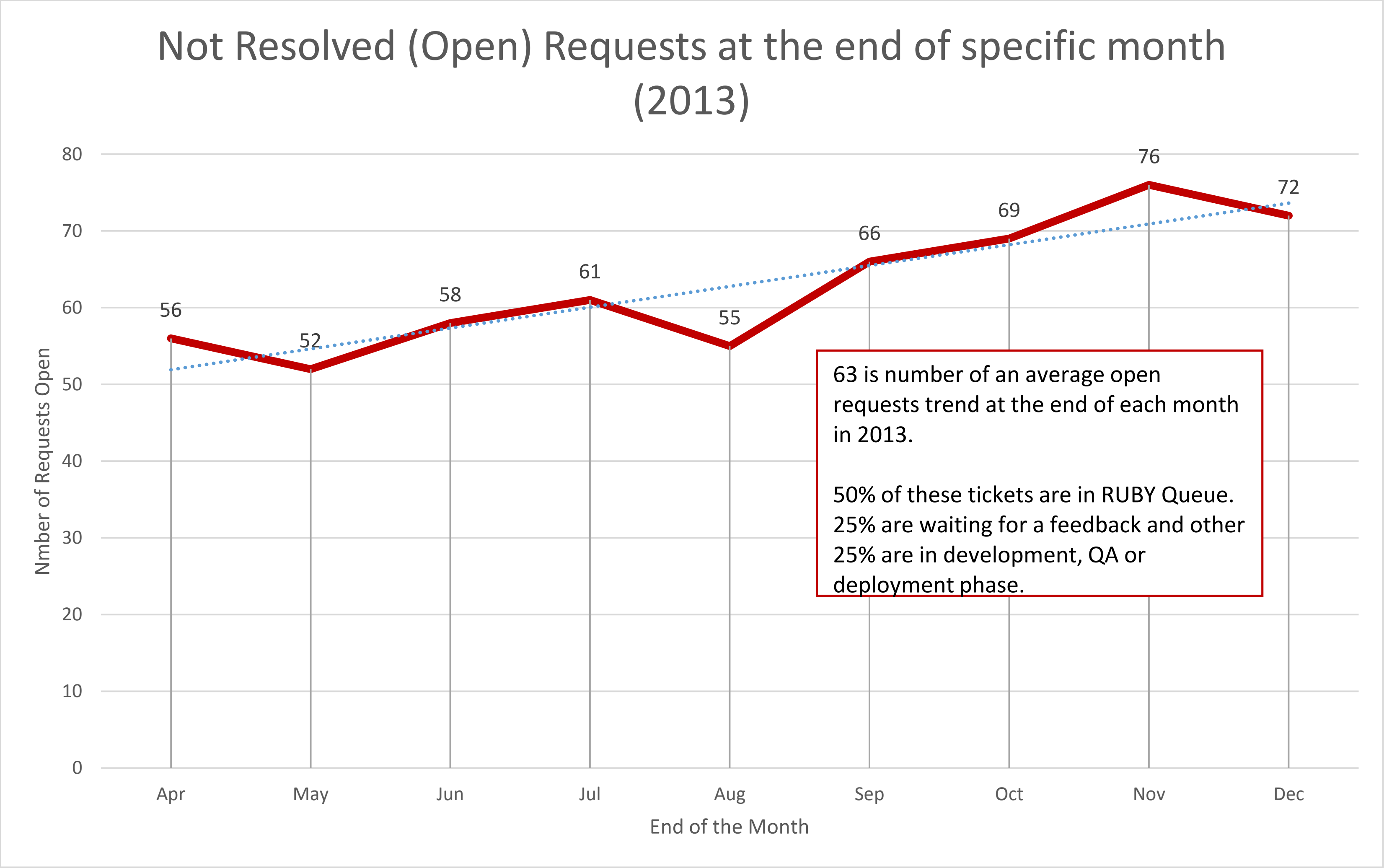
There is still a scope around 75 % where these numbers can still go even down, but it requires dealing with another problems which have occurred, but I will not go in more details as it is not part of this thesis.

In 2013, there was an upward trend which had been caused due to an unexpected staffing changes within the team and there was not enough capacity to cover the amount of work as used to in timely manner.

It had been continuing also till 2014 and in March, not resolved requests increased up to 97 as a new critical project went live and many issues occurred.

Based on the initial goal, it had not been met in this case either, however at least the average is very close to the goal.

Also in this case, new obstacles occurred which affected and slowed down the progress in reaching the initial target goals.

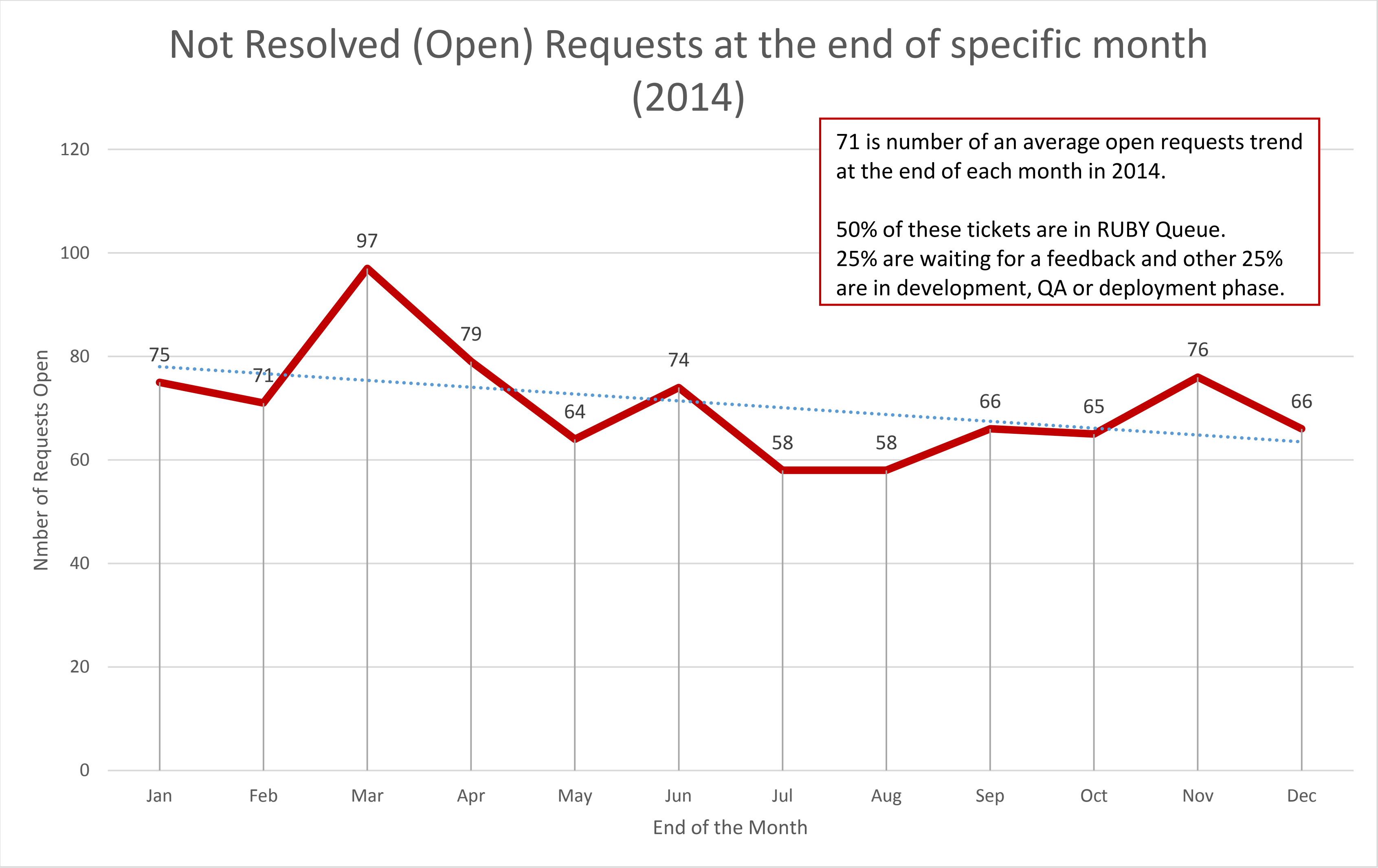


63 is number of an average open requests trend at the end of each month in 2013.

50% of these tickets are in RUBY Queue. 25% are waiting for a feedback and other 25% are in development, QA or deployment phase.

Figure 19:

Not resolved requests at the end of specific month (2013) (Source: Author)



71 is number of an average open requests trend at the end of each month in 2014.

50% of these tickets are in RUBY Queue.
25% are waiting for a feedback and other 25% are in development, QA or deployment phase.

Figure 20:
Not resolved requests at the end of specific month (2014) (Source: Author)

4.3 Reducing Resolution Time of the Requests

Goal: To reduce the resolution time.

I have noted in one of the chapters that 49% of the requests were getting resolved in less than 3 weeks and 51 % in more than 3 weeks. As I have introduced the new classification based on class of service, I will interpret the results separately for the two most common classes (99% of the requests) and these are **standard** and **end user request**.

In the table below there is a huge resolution time reduction in every measured time frame every year from 2012. When I take into account the planned lead time for Standard (3 weeks) and End User Request (2 weeks) and rate the years 2013 and 2014 (in 2012 this classification had not been measured, therefore I will take the maximum of 3 weeks) according university grading system it would end up as follows:

2012 – 49% => 4

2013 – Standard (67% => 3), End User Request (77% => 2)

2014 - Standard (78% => 2), End User Request (94% => 1)

Table 9: *Percentage of resolved incidents and requests in time*

Class of Service	Less than 24 hours	Less than 1 week	Less than 2 weeks	Less than 3 weeks	More than 3 weeks	More than 8 weeks
2012						
All	4%	23%	31%	**49%**	51%	30%
2013						
Standard	22%	45%	59%	**67%**	33%	17%
End User Request	31%	64%	**77%**	81%	19%	6%
2014						
Standard	28%	56%	70%	**78%**	22%	6%
End User Request	61%	90%	**94%**	95%	5%	2%

This goal was very successful even if there is still a need of improvements to reach at least 90–95% in both of the classes of services at their planned lead time (for End User Request class of service I can consider also this goal as accomplished).

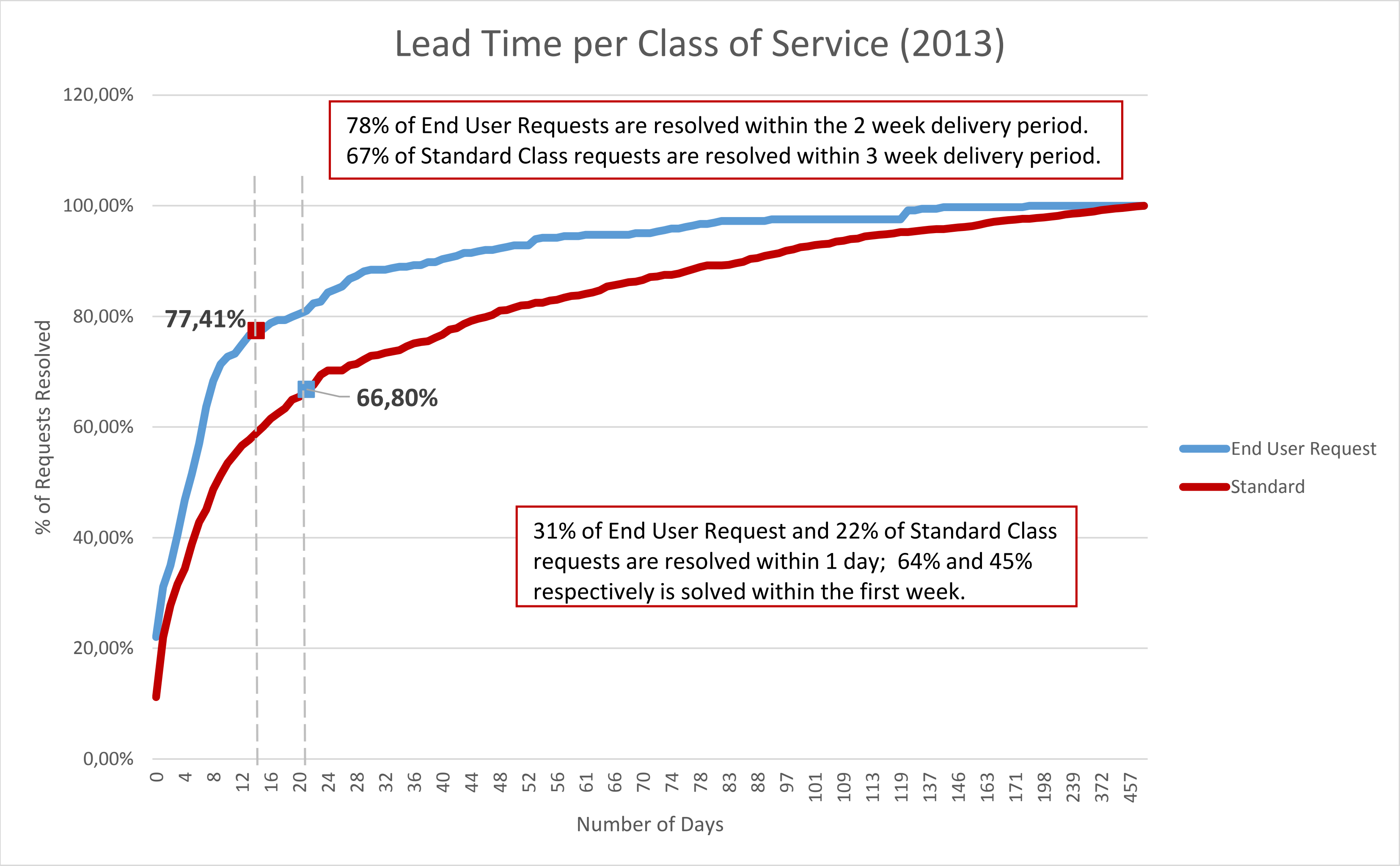
I have also evaluated how the specific technical teams had been doing while resolving the client requests in 2014 (I have chosen only Standard class of service as there are still needed improvements based on the data interpretation in the table above). I will use again the university grading system and three weeks planned lead time:

- System Administrators (100%) and Interactive Web Developers (97%) = 1
- Java Developers (84%) and Support Leader (86%) = 2
- PHP Developers (71%), .NET Developers (67%) and Database Architects (68%) = 3
- Ruby Developers (41%) = 4

Based on these results, I can clearly see there are some significant obstacles in the Ruby technical team as they were able to resolve only 41% of the requests as planned. This is caused due to lack of resources in the team on one side and high workload on the other side.

System administrators and IW developers were doing very well and there is not much to deal with. Java developers and support leader were doing also well, but there seems to be some obstacles and after taking a closer look, the main problems were with complicated deployment process which takes long time and it caused this slight inefficiency (the improvement of the deployment process has been however already addressed in April 2014).

PHP, .NET developers and database architects were also not so effective in resolving the maintenance requests, it was caused due to several factors: lack of resources and inefficient planning. Some of these have been already addressed, some are still being working on.



78% of End User Requests are resolved within the 2 week delivery period.
67% of Standard Class requests are resolved within 3 week delivery period.

31% of End User Request and 22% of Standard Class requests are resolved within 1 day; 64% and 45% respectively is solved within the first week.

Figure 21:
Lead Time per Class of Service (2013) (Source: Author)

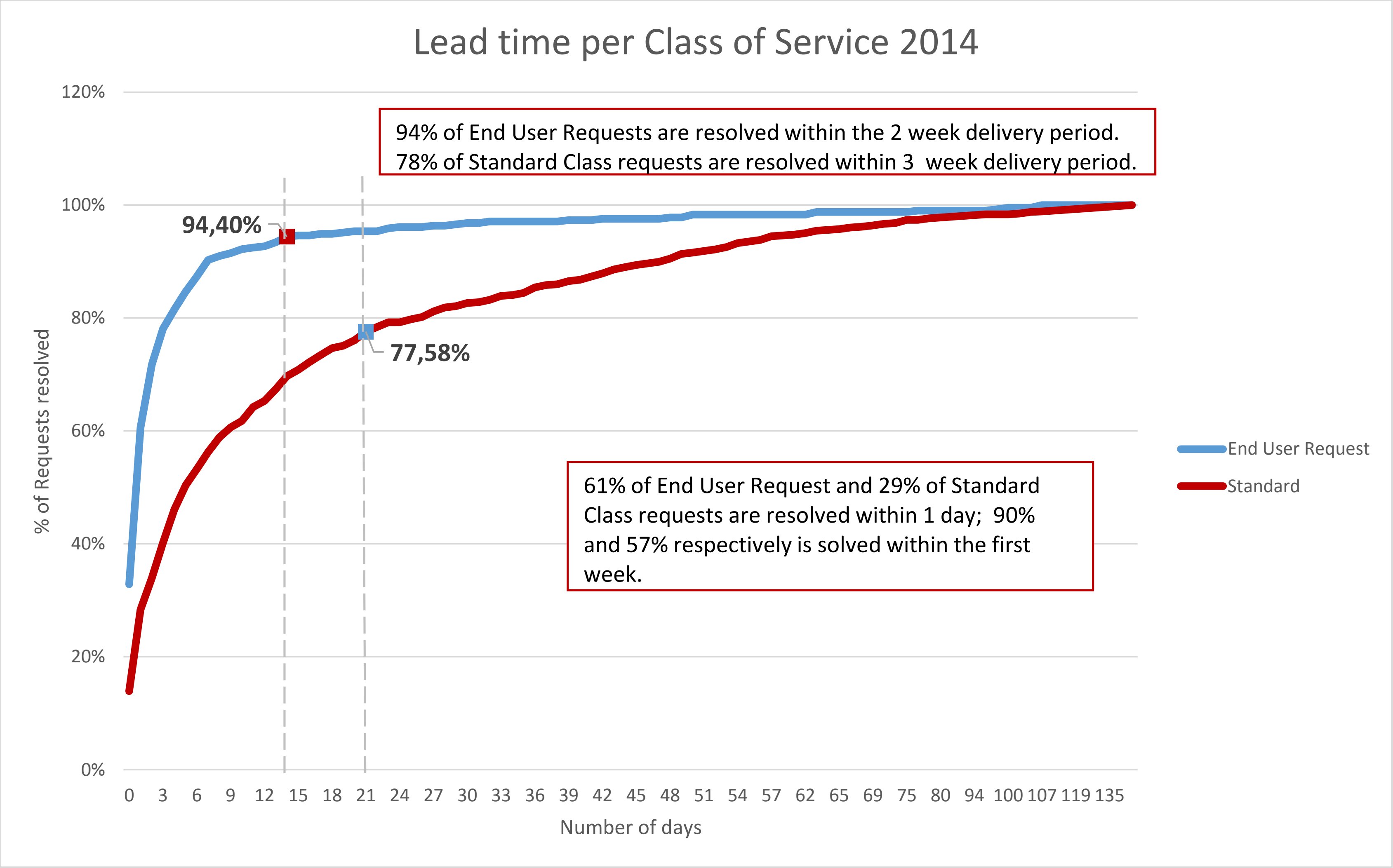


Figure 22:

Lead Time per Class of Service (2014) (Source: Author)

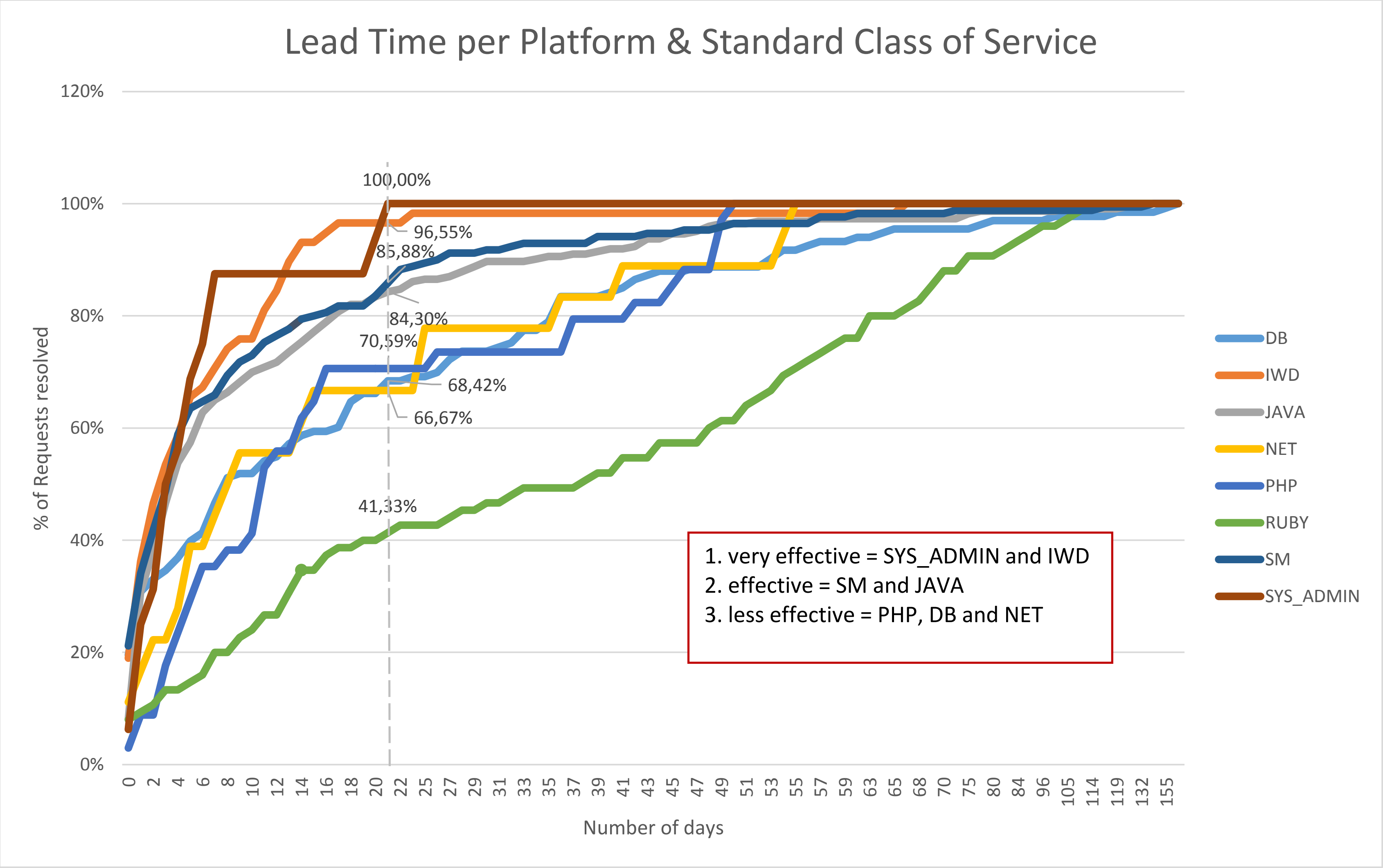


Figure 23:

Lead Time per Platform & Standard Class of Service (Source: Author)

4.4 Improving Clients Satisfaction

Goals:

- The requests resolution will be more predictable.
- The requests resolution will be faster.
- The new process will improve the maintenance service.

I have collected a feedback from some of the country representatives after three months of implementing the new process in 2013 and 75% of respondents indicated they agree or strongly agree that the requests resolution is now more predictable. 67% of respondents agree that the new process changes improved the maintenance service as a whole and 57% of respondents agree that the requests resolution is now faster.

When I look closely at these numbers, they do not really look so good than I would expect. After contacting few of the respondents, they indicated that the maintenance service had been improved, however they still feel it is not enough. I had tried to get as much details as possible and had identified the following areas that will require an improvement or change:

- Mantis – the problem with Mantis itself is, that when submitting a new request, it is being submitted under the project I am currently viewing. It means that if I submit a new request in a wrong project, it is lost within all the projects and most of the time only by escalating and providing the specific request ID, the support leader is able to find it. This leads to disappointing of the client. At the moment a new JIRA tool is being implementing that should eliminate such cases.
- High workload for Ruby developers – in that time the problem with high workload for ruby technology had not been addressed yet and the requests resolution for this particular area was not efficient enough. This was also one of the reasons why the respondents did not respond as positively as I was expecting. Today, I can say this obstacle has been addressed and the ruby team performance is high.

However when I look at the trends and results from previous sub-chapters, it is obvious the overall maintenance service has been significantly increased and generates better results every year.

To the occasion of reaching the historical number of not resolved requests in July 2013 (29 not resolved requests), the team was given the delicious cake as a gift below, which can be considered as a kind of the client satisfaction with the current maintenance service.

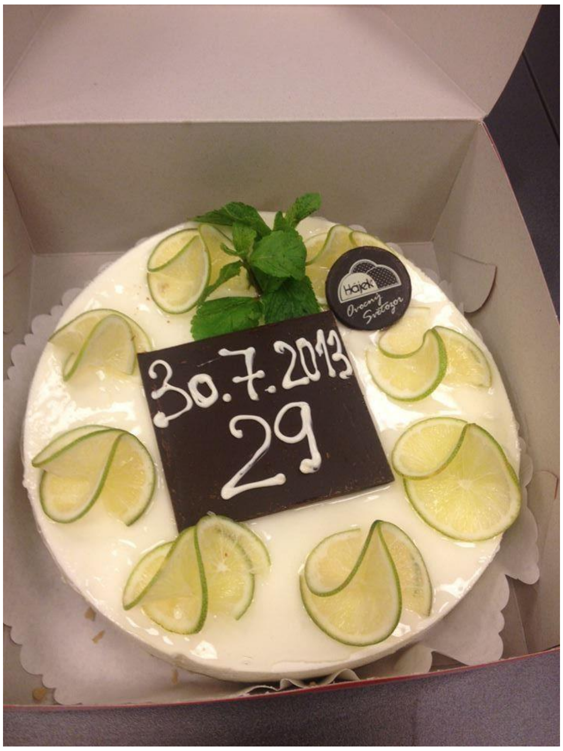


Figure 24:
A gift for the achievements in July 2013
(Source: Author)

4.5 Evaluation of KPIs

Table 10: *Evaluation of KPIs*

ID	KPI	2013	2014
#1	How many of the requests had met or exceeded the planned lead time per class of service (in %).	• End User Requests • Met (77%) • Exceeded (23%) • Standard • Met (67%) • Exceeded (33%) • Expedite • Met (76%) • Exceeded (24%) • Fixed Date Delivery • Met (91%) • Exceeded (9%)	• End User Requests • Met (94%) • Exceeded (6%) • Standard • Met (78%) • Exceeded (22%) • Expedite • Met (96%) • Exceeded (4%) • Fixed Date Delivery • Met (100%) • Exceeded (0%)
#2	What is the minimum, maximum and average resolution of the requests per class of service (in days).	• End User Requests • Min (1 day)	• End User Requests • Min (1 day)

ID	KPI	2013	2014
		- Max (190 days) - Average (14 days) - Standard - Min (1 day) - Max (588 days) - Average (31 days) - Expedite - Min (1 day) - Max (69 days) - Average (8 day)	- Max (108 days) - Average (5 days) - Standard - Min (1 day) - Max (155 days) - Average (21 days) - Expedite - Min (1 day) - Max (14 days) - Average (1 day)
#3	How many of the requests had met or exceeded the planned lead time per standard class of service and specific technical teams (in %).	This measure has been tracked since 2014.	- Database Architects - Met (68%) - Exceeded (32%) - Interactive Developers - Met (97%) - Exceeded (3%) - Java Developers - Met (84%) - Exceeded (16%)

ID	KPI	2013	2014
			• .NET Developers • Met (68%) • Exceeded (32%) • Ruby Developers • Met (41%) • Exceeded (59%) • Support Leader • Met (86%) • Exceeded (14%) • System Administrators • Met (100%) • Exceeded (0%)
#4	How many of the requests had been resolved by specific technical teams (count).	This measure has been tracked since 2014.	• Database Architects (152) • Interactive Developers (67) • Java Developers (253) • .NET Developers (21) • Ruby Developers (91) • Support Leader (579)

ID	KPI	2013	2014
			• System Administrators (17)
#5	How many of the requests had been resolved per country (count).	• Benelux (124) • Czech Republic (38) • England (117) • France (164) • Germany (79) • Italy (53) • Nordics (73) • Poland (13) • Russia (3) • South Africa (23) • Spain (34) • Switzerland (42) • Other (219)	• Benelux (127) • Czech Republic (21) • England (171) • France (157) • Germany (121) • Italy (34) • Nordics (120) • Poland (20) • Russia (64) • South Africa (37) • Spain (44) • Switzerland (21) • Other (216)
#6	How many of the requests had been resolved per application (count).	• Application C (0) • Application A (531) • Application B (123) • Application D (0)	• Application C (38) • Application A (413) • Application B (62) • Application D (57)

ID	KPI	2013	2014
		• Application G (56) • Application F (164) • Application E (204) • Application H (40) • Application I (32)	• Application G (128) • Application F (116) • Application E (329) • Application H (27) • Application I (50)
#7	What is the average of not resolved requests at the end of each month (count).	This measure has been tracked since 2014.	• January (75) • February (71) • March (97) • April (79) • May (64) • June (74) • July (58) • August (58) • September (66) • October (65) • November (76) • December (66)

ID	KPI	2013	2014
#8	What is the proportion of the resolved requests between “change request”, “issue” and “end user request” (count + %).	This measure has been tracked since 2014.	• Change Requests (593 = 49%) • Issues (218 = 18%) • End User Requests (407 = 33%)

5 Conclusion

To summarize the research and gained achievements from implementing the suggested maintenance process improvements, the expected results and goals have not been reached in that range as initially expected. One of the reason was that some of the key factors had not been considered:

- The changes will take some time until they are implemented globally and accepted by all the parties.
- Newly formed and unexpected system constraints and bottlenecks during the process of implementation and executing.

However when taking into account the mentioned constraints above, I can say, there have been a lot of improvements and step ahead from the old process to the new version and I evaluate the results as very successful.

I was able to reduce not resolved client requests with the team by more than 80% from 285 to 54 within three and half of a month and at the end of July 2013, even to 29 which was a historical number. Next months till the end of 2014 the number fluctuated and an average of open requests was 63 in 2013 and 71 in 2014. Upon closer examination, around 50% of these requests had been in queue due to other system bottlenecks that occurred in the meantime, 25% had been waiting for a client feedback (still within the agreed two weeks period) and 25% had been in progress (either development, testing or deployment phase).

Very interesting results have been achieved with reducing the lead time of the client requests. Before the process changes only 4% of the requests were resolved within 24 hours, 49% within three weeks and 30% in more than 2 months. After process improvements and introducing the Kanban mechanism “Class of Service”, in 2013, 22% of the Standard and 31% End User requests were resolved within 24 hours, 67% of the Standard and 81% of the End User requests were resolved within three weeks and 17% of Standard and 6% of End User requests in more than 2 months. Following by 2014, 28% of the Standard and 61% End User requests were resolved within 24 hours, 78% of the Standard and 95% of the End User requests were resolved within three weeks and 6% of Standard and 2% of End User requests in more than 2 months. There is a significant annual lead time reduction in all the periods and a great potential it will continue the following months and years.

I analyzed all the requests with lead time more than three weeks for Standard Class of Service and have discovered that 41% of the requests had been exceeded due to another system constraint and bottleneck within the team which has been addressed already a couple of months.

From the clients´ perspective, the overall satisfaction has been very positive and over the expectations from the headquarters. Though only 75% of the country representatives had agreed that the prediction of requests resolution is now more accurate, 67% had agreed the process changes improved the overall maintenance service and 57% agreed that the requests are getting resolved faster, which is mainly due to new bottleneck and constraint within the system which needs to be evaluated and addressed again.

Here I can see that the POOGI is a never ending process that requires ongoing evaluation of the process and addressing new obstacles present within the system.

Appendix

Weekly Reports

Figure 15 – Represents number of not resolved requests per country (total vs. requests where more details are needed from the client).

Figure 16 – Represents number of not resolved requests per class of service (total vs. requests where more details are needed from the client vs. requests in queue).

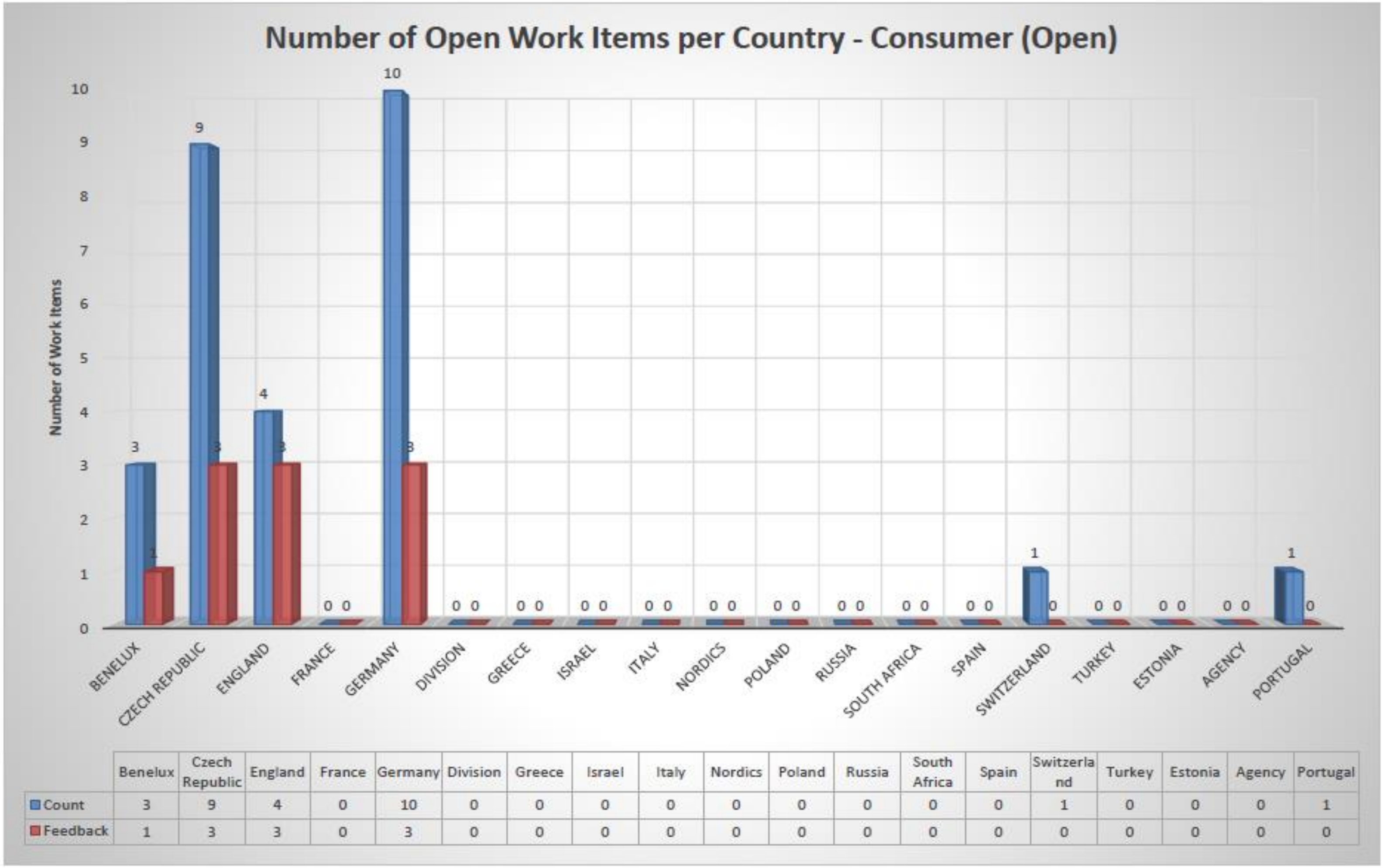
Figure 17 – Represents number of not resolved requests per application and type (issue vs. change request vs. end user request).

Figure 18 – Represents number of not resolved requests per application and lead time (total vs. agreed lead time has been met vs. exceeded).

Figure 19 – Represents number of work in progress per class of service and lead time (resolved vs. in progress).

Figure 20 – Represents number of work in progress per status (in progress vs. not started vs. completed).

Figure 21 – Represents number of requests reported on weekly bases (total vs. issues vs. change requests vs. end users requests)



	Benelux	Czech Republic	England	France	Germany	Division	Greece	Israel	Italy	Nordics	Poland	Russia	South Africa	Spain	Switzerland	Turkey	Estonia	Agency	Portugal
🟦 Count	3	9	4	0	10	0	0	0	0	0	0	0	0	0	1	0	0	0	1
🟥 Feedback	1	3	3	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25:

Weekly Reports - Number of Open Requests per Country (Measure A1) (Source: Author)

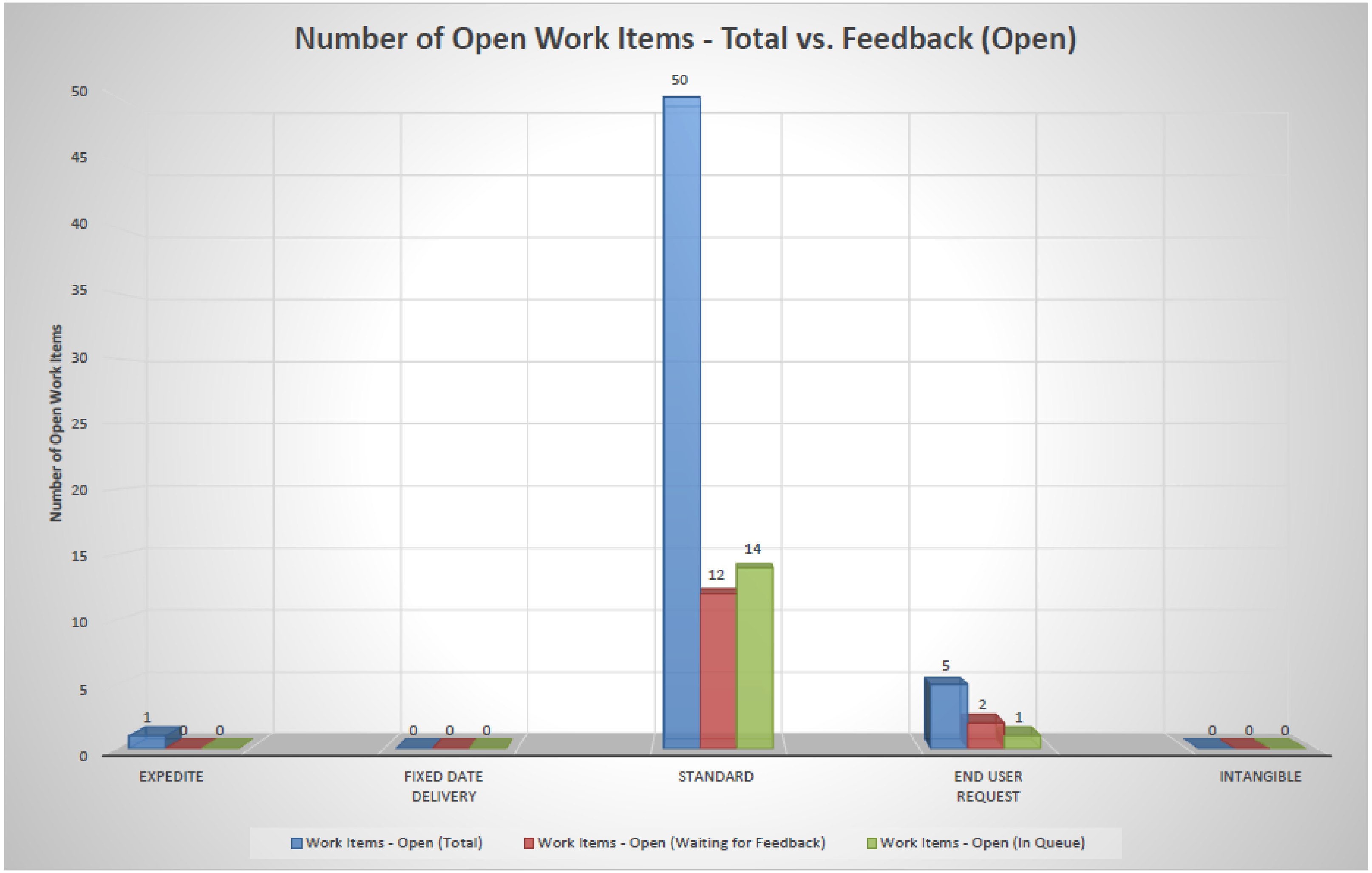


Figure 26:

Weekly Reports - Number of Open Requests per Class of Service (Measure A2) (Source: Author)

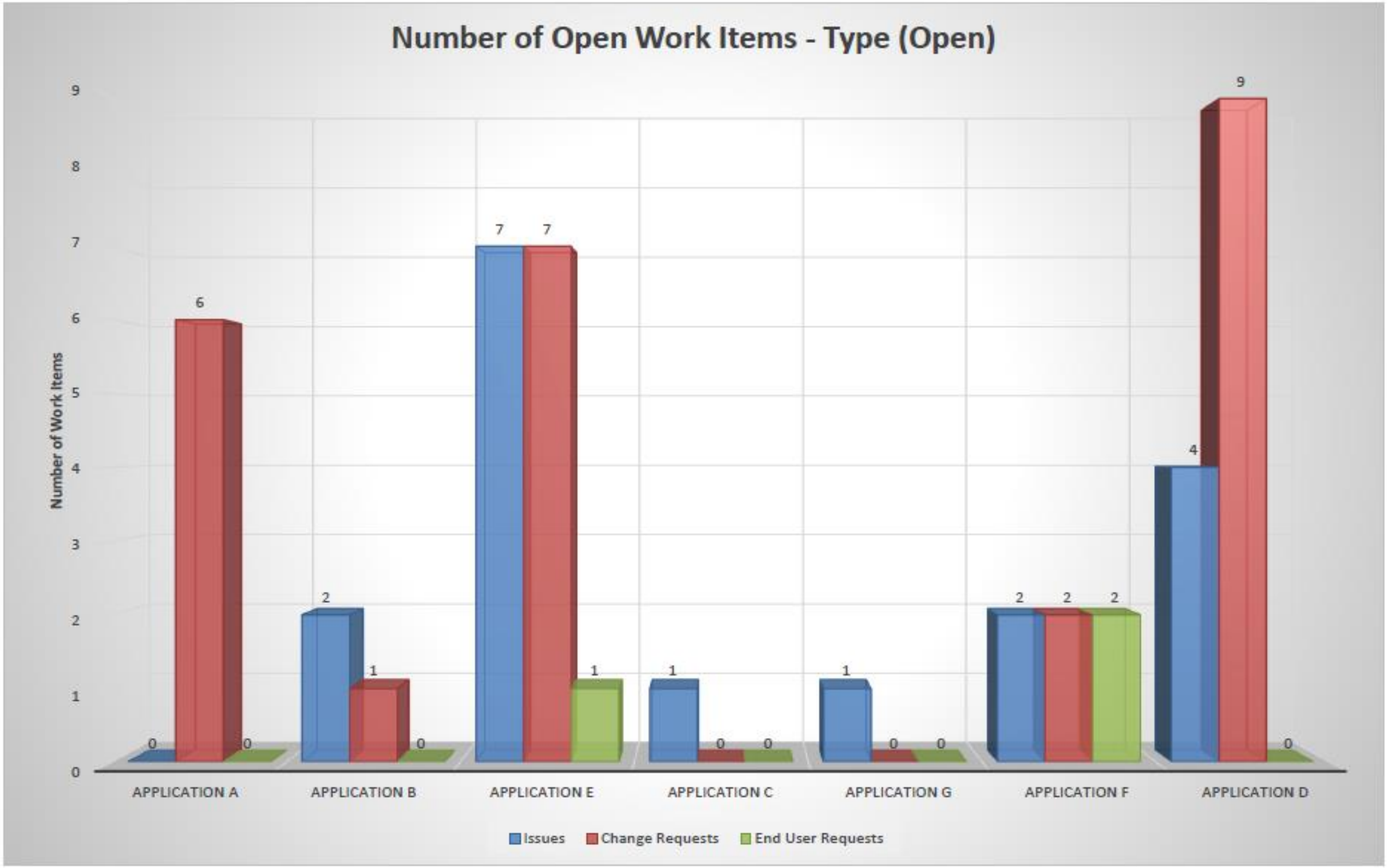
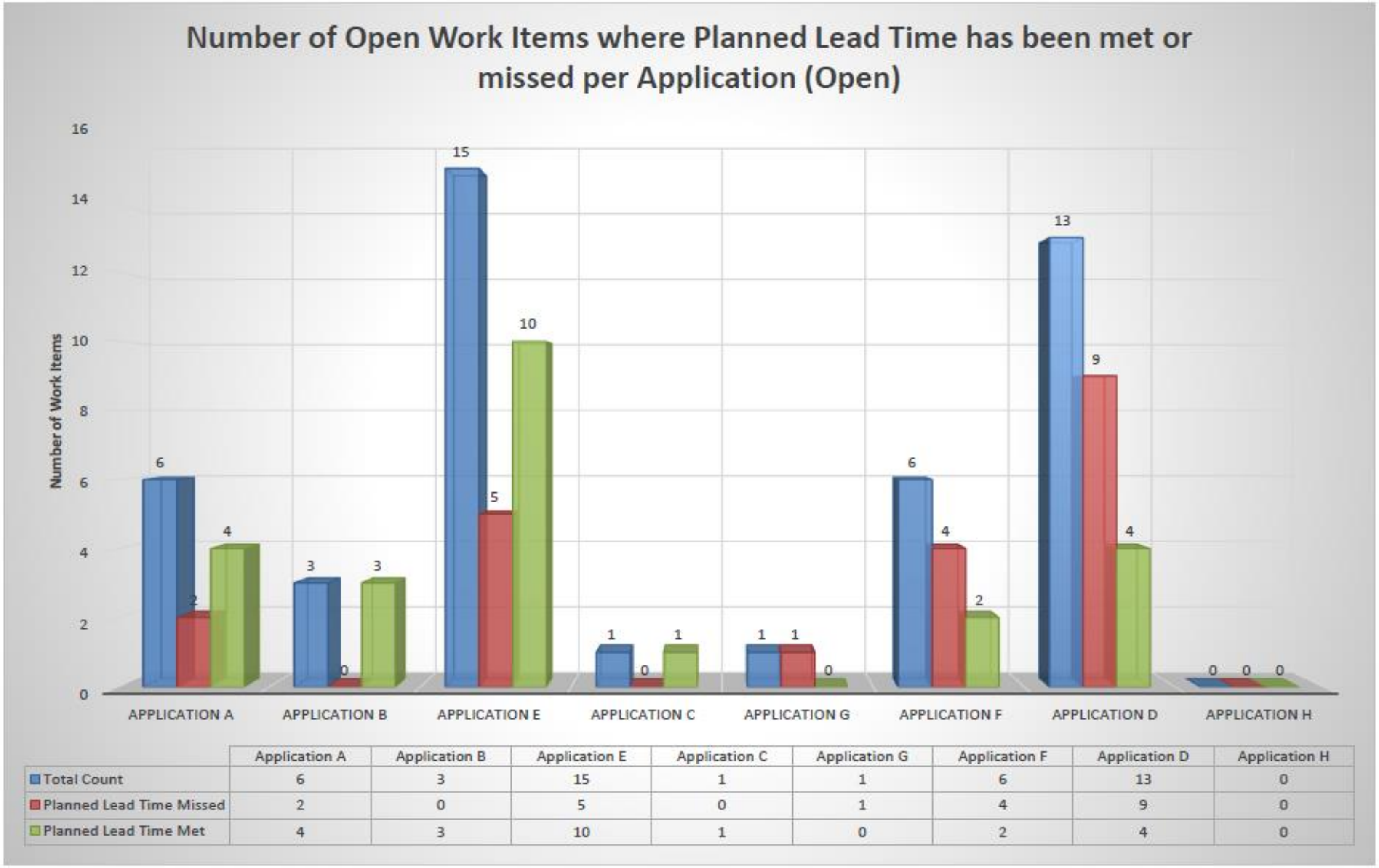


Figure 27:

Weekly Reports - Number of Open Requests per Application and Type (Measure A3) (Source: Author)



	Application A	Application B	Application E	Application C	Application G	Application F	Application D	Application H
🟦 Total Count	6	3	15	1	1	6	13	0
🟥 Planned Lead Time Missed	2	0	5	0	1	4	9	0
🟩 Planned Lead Time Met	4	3	10	1	0	2	4	0

Figure 28:

Weekly Reports - Number of Open Requests per Application and Lead Time (Measure A4) (Source: Author)

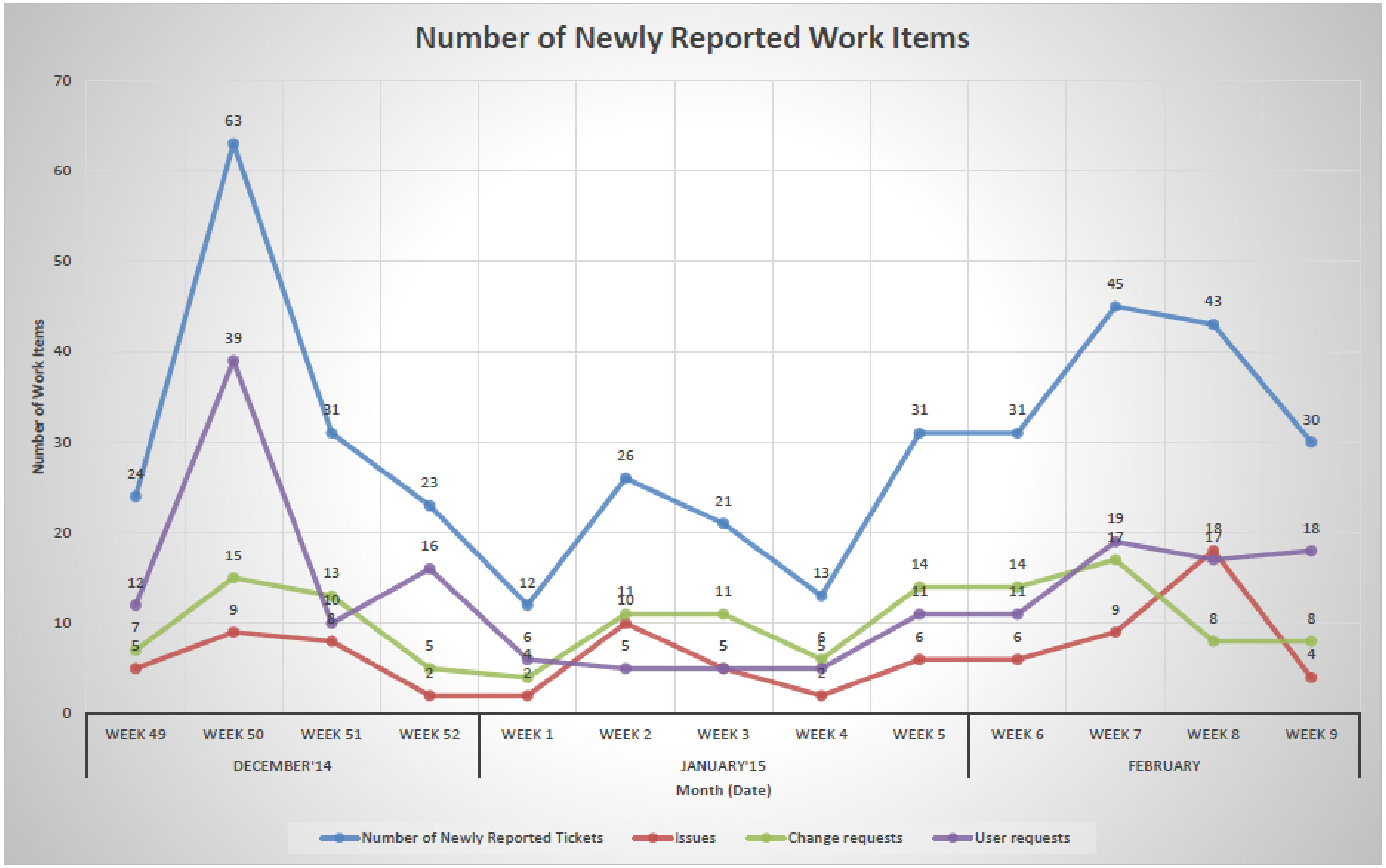


Figure 29:
Weekly Reports - New Requests Trend (Measure A5) (Source: Author)

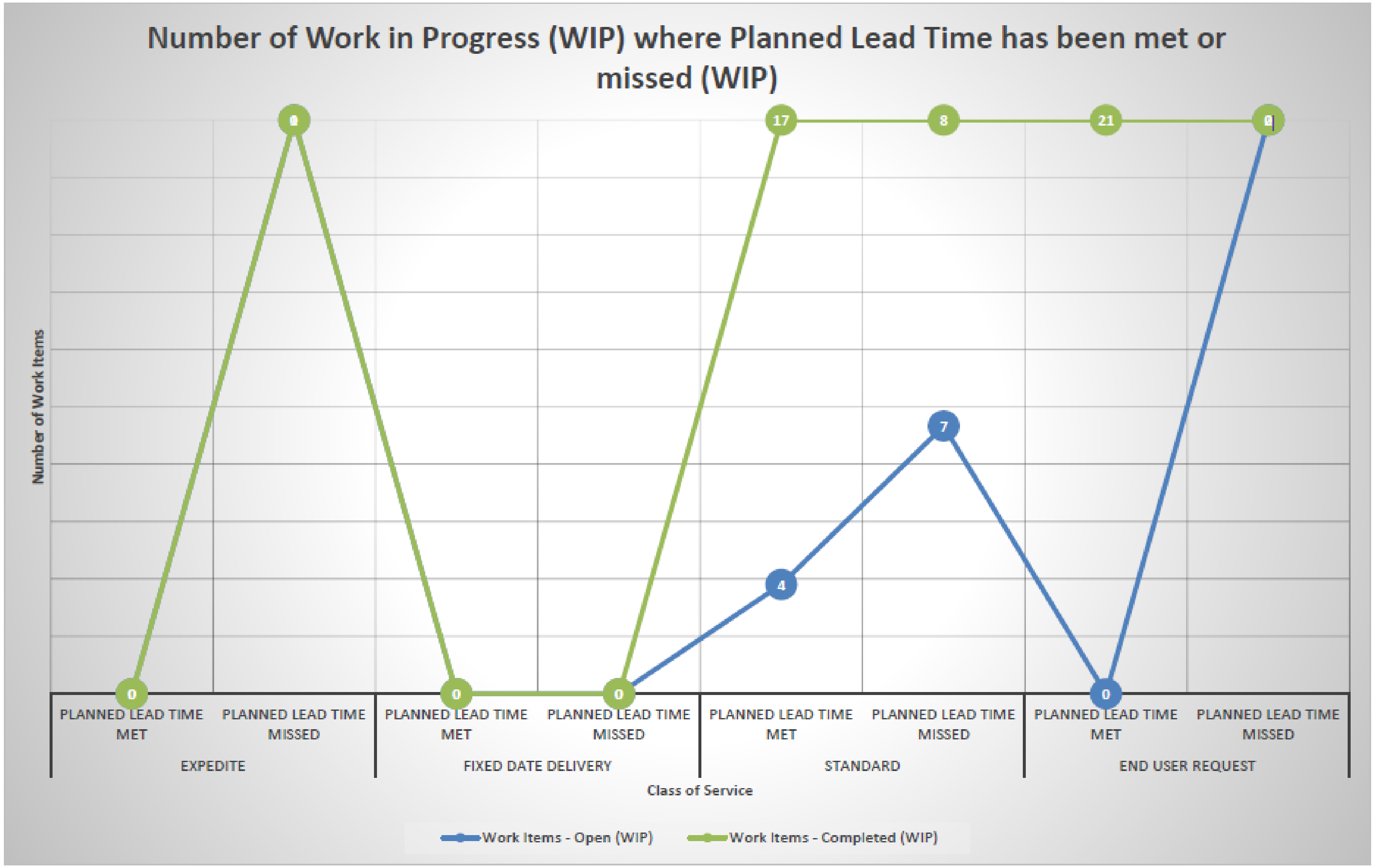
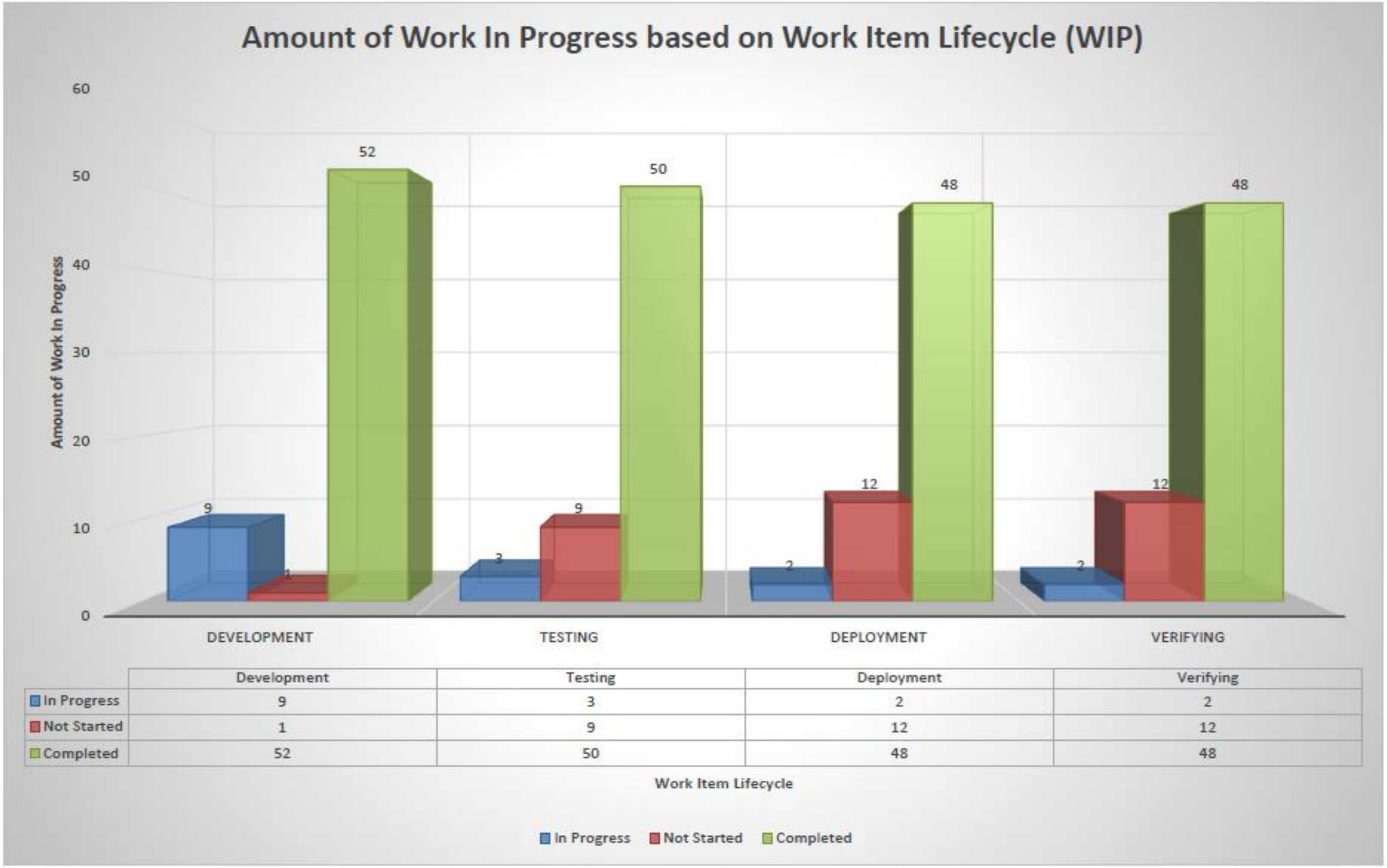


Figure 30:

Weekly Reports - Work in Progress per Class of Service and Lead Time (Measure B1) (Source: Author)



	Development	Testing	Deployment	Verifying
🟦 In Progress	9	3	2	2
🟥 Not Started	1	9	12	12
🟩 Completed	52	50	48	48

Figure 31:

Weekly Reports - Work in Progress per Status (Measure B2) (Source: Author

Bibliography

[1] Ricketts, John Arthur. *Reaching the Goal: How Managers Improve a Servicesbusiness Using Goldratt's Theory of Constraints.* [S.l.]: Ibm Press, 2007.

[2] Anderson, David J. *Lessons in Agile Management: On the Road to Kanban*. Sequim, WA: Blue Hole Pr., 2012.

[3] The Quotations Page: Quote from W. Edwards Deming. *The Quotations Page*. [online]. 2015 [cit. 2015-02-09]. Available at: http://www.quotationspage.com/quote/5154.html.

[4] RedDot CMS (now OpenText Web Site Management). CMS MATRIX. CMS Matrix [online]. 2015 [cit. 2015-04-06]. Available at: http://www.cmsmatrix.org/matrix/cms-matrix/reddot-cms-now-opentext-web-site-management-

[5] Planning the Development, Testing, Staging, and Production Environments. MICROSOFT. MSDN [online]. 2015 [cit. 2015-04-06]. Available at: https://msdn.microsoft.com/en-us/library/cc296714%28v=bts.10%29.aspx

[6] The Five Focusing Steps (POOGI). THEORY OF CONSTRAINTS INSTITUTE. Theory of Constraints [online]. 2014 [cit. 2015-04-06]. Available at: http://www.tocinstitute.org/five-focusing-steps.html

[7] Service Level Agreement. KNOWLEDGE TRANSFER. Knowledge Transfer [online]. 2008 [cit. 2015-04-20]. Available at: http://www.knowledgetransfer.net/dictionary/ITIL/en/Service_Level_Agreement.htm

[8] WALLEN, Jack. Review: Mantis Bug Tracker (MantisBT). CBS INTERACTIVE. Tech Republic [online]. 2009-08-20 [cit. 2015-04-20]. Available at: http://www.techrepublic.com/blog/product-spotlight/review-mantis-bug-tracker-mantisbt/

[9] Web Application Maintenance. ITISL TECHNOLOGIES. ITISL [online]. 2010 [cit. 2015-04-20]. Available at: http://www.itisl.com/maingroup/servwebappmnt.aspx

[10] Content management system (CMS). ROUSE, Margaret a Klaus SVARRE. TECHTARGET. TechTarget [online]. 2011-01-01 [cit. 2015-04-20]. Available at: http://searchsoa.techtarget.com/definition/content-management-system

[11] Learning management system (LMS). ROUSE, Margaret. TECHTARGET. TechTarget [online]. 2005-09-01 [cit. 2015-04-20]. Available at: http://searchcio.techtarget.com/definition/learning-management-system

[12] Build. JANALTA INTERACTIVE INC. TechoPedia [online]. 2015 [cit. 2015-04-20]. Available at: http://www.techopedia.com/definition/3759/build

[13] STEVENS, DENNIS. KANBAN: WHAT ARE CLASSES OF SERVICE AND WHY SHOULD YOU CARE?. DENNIS STEVENS [online]. 2009 [cit. 2015-04-20]. Available at: http://www.dennisstevens.com/2010/06/14/kanban-what-are-classes-of-service-and-why-should-you-care/

[14] Scrum. MOUNTAIN GOAT SOFTWARE. Mountain Goat Software [online]. 2015 [cit. 2015-04-20]. Available at: http://www.mountaingoatsoftware.com/agile/scrum

List of Figures and Tables

List of Figures

Figure 1: 7
Figure 2: 8
Figure 3: 10
Figure 4: 14
Figure 5: 16
Figure 6: 18
Figure 7: 19
Figure 8: 25
Figure 9: 34
Figure 10: 36
Figure 11: 38
Figure 12: 39
Figure 13: 40
Figure 14: 43
Figure 15: 44
Figure 16: 46
Figure 17 47
Figure 18: 52
Figure 19: 54
Figure 20: 55
Figure 21: 58
Figure 22: 59
Figure 23: 60
Figure 24: 62
Figure 25: 72
Figure 26: 73
Figure 27: 74
Figure 28: 75
Figure 29: 76
Figure 30: 77
Figure 31: 78

List of Tables

Table 1: List of Class of Service 22
Table 2: List of Priorities 23
Table 3: List of Severities 23
Table 4: List of Country Values 24
Table 5: List of Lead Time Values 24
Table 6: Example of the client weekly plan (Source: Author) 30
Table 7: Example of the team weekly plan (Source: Author) 31
Table 8: Action steps (Source: Author) 32

Table 9: Percentage of resolved incidents and requests in time 56
Table 10: Evaluation of KPIs 63