TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science

Service management metrics for a large-scale Engineering Services Team

Master Thesis

Student: Märten Ester
Student code: 090202IAPM
Supervisor: Deniss Kumlander

Tallinn
2015

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

(*kuupäev*) (*allkiri*)

Abstract

The thesis studies how data and metrics can be used to drive improvements in a large scale Engineering Services Team. Decision making based on metrics is examined. The main goals of the thesis are the following:

- To understand the role of data and metrics in the service management and software development process.
- To establish Key Performance indicators and other metrics for Skype Engineering Services Team. One of the purposes of the metrics is to support the planning process. The second purpose is to understand the progress made towards the goals set by the team and the company.
- To find the best means for gathering, surfacing and using data to manage and support an Engineering System.

The thesis starts by examining the role of metrics in the software development and service management process in general. The practices and benefits of incorporating metrics into daily workflows are explored. The introductory part finishes by giving and overview of the Skype Engineering Services Team and how data and metrics are used.

Rest of the thesis continues by examining possibilities for improvements to Skype Engineering Services Team. Based on the business goals of the Skype Engineering Services Team, several metrics to measure the impact of the daily work are proposed. By analysing the data around the incident management process proposals for improvements are suggested. Key Performance indicators are established for the Build and Third Party Software area.

The outcomes of the work are suggestions and actual implementations in the following areas

- Service instrumentation.
- Processes and tools for data gathering, storing, visualisation and analytic.s
- Metrics for planning and understanding the impact of the work done in the Skype Engineering Service Team.

In addition to the outcomes listed above the work reveals side effects that a data and metrics project can have. The owner's lack of deeper understanding or misconceptions of the domain can be viewed as the biggest risk to the success of establishing Key Performance Indicators. In the other hand the exercise of measuring the impact of ones actions provides a good opportunity to get deeper insights in to business domain resulting in actions that otherwise would not have taken place.

Annotatsioon

Magistritöö uurib andmete ja meetrikate kasutamist suuremahulises tarkvaraarenduse tugiteenuste meeskonnas. Vaadeldakse meetrikate poolt toetatavaid otsustusprotsesse. Magistritöö põhieesmärgid on järgnevad:

- Mõista andmete ja meetriakte rolli teenuse opereerimisel ja arendusprotsessis.
- Välja töötada Skype arenduse tugiteenuste meeskonna jaoks tulemusindikaatorid. Selliste meetrikate üheks eesmärgiks on planeerimisprotsessi toetamine. Teiseks eesmärgiks on meetrikate abil mõista meeskonna ja ettevõtte eesmärkide täitmise ulatust
- Leida parimad vahendid andmete kogumiseks ja presenteerimiseks.

Magistritöö algab meetrikate rolli uurimisega tarkvara arenduses ja teenuste opereerimises. Vaadeldakse meetrikate kasutamise praktikaid ja sellest saadavat kasu. Sissejuhatav osa lõppeb ülevaatega Skype tarkvaraarenduse tugiteenuste meeskonnast ja kirjeldusega sellest, kuidas seal andmeid ja meetrikaid kasutatakse.
Järgnevalt keskendub magistritöö võimalustele Skype'i Tarkvaraarenduse tugiteenuste meeskonna töös erinevaid parandusi sisse viia. Lähtuvalt meeskonna ärilistest eesmärkidele pakutakse välja mitmed meetrikad igapäeva töö mõõtmiseks. Intsidentide haldus protsessi analüüsi käigus pakutakse välja mitmed parandused. Tulemusindikaatorid töötatakse välja *Buildi* ja Kolmanda osapoole tarkvara haldamise protsessi kohta.

Töö tulemusteks on ettepanekud ja rakendused järgnevates valdkondades:

- Teenuste instrumenteerimine.
- Protsessid ja vahendid andmete kogumiseks, salvestamiseks, visualiseerimiseks ja analüüsiks.
- Meetrikad Skype'i Tarkvaraarenduse tugiteenuste meeskonna töö tulemuste hindamiseks.

Lisaks ülal mainitud tulemustele toob magistritöö esile andmete ja meetrikate projekti kõrvalmõjud. Omanike sügavams arusaama puudumine või valearusaamad tegevusvaldkonnast on ühed suurimad riskid tulemusindikaatorite sisseviimisel. Teisalt pakub enda tegevuse tulemuste mõõtmine võimaluse saada sügavamaid teadmisi ärivaldkonna kohta, mille tulemuseks võivad olla teod, mis muul juhul ei oleks toimunud.

Definitions and Abbreviations

Third-party software (TPS) - A Third Party Software component is a software product developed by an entity other than the original vendor of the development platform. A Third Party Software component can be sold or distributed without a fee.

Key Performance Indicator (KPI) - Key Performance Indicators measure the success of a company in the most crucial areas of their business.

Service Level Agreement (SLA) - A service level agreement is a contract between the service provider and the user of the service. Service level agreement defines the requirements that the service provider must fill including those regarding performance and reliability. A service level Agreement might also include functional definitions of the service

Jira ticket – Also referred as Jira issue**.** Jira[1] is a service developed by Atlassian meant for carrying out product planning and development related tasks including requirements and defect tracking. A Jira ticket is a work item created in the system by any user. A Jira ticket has an associated workflow with a pre-defined lifecycle including start and end states.

(Software) Build - A Build is the end product of a process that turns source code into a working component. It includes preparing the right environments, managing and fetching the correct source code and dependencies, compiling the source code, running a set of tests needed to verify the build, and making the component and its documentation available for the users.

Software Process Improvement (SPI) – In this thesis Software Process Improvement is referenced as the initiative that introduces mature development process in Phillips.

Balanced Scorecard (BSC) – Balanced Scorecard is a widely used tool for strategic planning and performance management. In addition to the financial perspective, learning and growth, internal business processes and the customer perspective is taken into account.

Capability Maturity Model (CMM) – A model used to assess the level of predictability and reliability of a software development process.

[1] https://www.atlassian.com/software/jira

Table of figures

FIGURE 1 - ES CONCEPTUAL ARCHITECTURE 22
FIGURE 2 - TRACKED INCIDENTS 31
FIGURE 3 - CHANGE IN ALERT COUNT 32
FIGURE 4 - ALERTS, INCIDENTS AND COMMUNICATION HANDLED IN SEPARATE THREADS 33
FIGURE 5 - WORKFLOW BASED ON JIRA TICKETS 34
FIGURE 6 - THE % OF TICKETS APPROVED IN LESS THAN 14 DAYS 38
FIGURE 7 – THE NUMBER OF TICKETS APPROVED AND WAITING FOR APPROVAL 40
FIGURE 8 - RELIABILITY TOTAL PER DAY 46
FIGURE 9 - RELIABILITY BREAKDOWN PER MONTH 46
FIGURE 10 - SPEED TOTAL PER DAY 47
FIGURE 11 - SPEED BREAKDOWN BY MONTH 47
FIGURE 12 - CI EXPERIENCE ON 09.01.2015 48
FIGURE 13 – METRICS INFRASTRUCTURE 52
FIGURE 14 - ETL MODEL 60
FIGURE 15 - RESULTS FOR IMPROVEMENT DRIVES FROM PHILIPS [1] 61
FIGURE 16 - THE % OF INTERNAL TPS TICKETS APPROVED IN LESS THAN 14 DAYS 62
FIGURE 17 - THE NUMBER OF INTERNAL TPS TICKETS APPROVED AND WAITING FOR APPROVAL 63
FIGURE 18 - EXAMPLE OF FIXED VS ROLLING SLA 64
FIGURE 19 - ALERT AND INCIDENT REVIEW PROCESS 65
FIGURE 20 - RELIABILTY BREADOWN BY DAY 66
FIGURE 21 - RELIABILITY TOTAL PER MONTH 66
FIGURE 22 - BUILD SPEED TOTAL PER MONTH 67
FIGURE 23 - SPEED BREAKDOWN BY DAY 67

Table of tables

TABLE 1 - METRICS: LEVEL OF ACTIVITY IN PHILIPS 16
TABLE 2 - TOTAL NUMBER OF ALERTS IN ES 30
TABLE 3 - CRITICAL ALERTS PER SERVICE 30
TABLE 4 - CRITICAL ALERTS PER ITEM 30
TABLE 5 - EXPLANATIONS FOR TPS KPI COLUMNS 36
TABLE 6 - THE PERCENTAGE OF TICKETS APPROVED IN LESS THAN 14 DAYS 37
TABLE 8 - THE TOP RESOURCES ORDERED BY THEIR TOTAL USAGE TIME 44
TABLE 9 - THE % OF INTERNAL TPS TICKETS APPROVED IN 14 DAYS OR LESS 62
TABLE 10 - AUGUST AND SEPTEMBER FOR DISTRIBUTED TPS TICKETS BROKEN DOWN BY DAY 63
TABLE 11 - DATA FOR THE EXAMPLE OF USING A FIXED PERIOD VS USING A ROLLING AVERAGE SLA 64

Table of contents

Autorideklaratsioon 2
Abstract 3
Annotatsioon 5
Definitions and Abbreviations 7
Table of figures 8
Table of tables 9
Table of contents 10
1. Introduction 12
1.1 Scope and motivation 12
1.2 Goals 13
1.3 Methodology 13
1.4 Overview 14
2. Best practices for implementing metrics 15
2.1 The characteristic and number of metrics 15
2.2 Success Factors 17
2.3 Technical infrastructure 18
2.4 Benefits of using KPIs and metrics 18
2.5 Summary 19
3. Overview of the Skype Engineering Services 21
3.1 Description of Skype Engineering Services Team 21
3.2 Engineering Services data management and usage 24
3.2.1 AW stats 24
3.2.2 Log Stash 25
3.2.3 Nagios 25
3.2.4 Other 26
3.3 Summary 26
4. Metrics for alerts and Incidents 28
4.1 Background 28
4.2 Alerts and Incident Data 29
4.3 Learnings and improvements 31

4.4 Summary 34
5. A Key Performance Indicator for Third Party Software management process 35
5.1 Approval time as a KPI 36
5.2 Implementation of the KPI 38
5.3 Usage of the KPI 40
6. Key Performance Indicators for Build Systems 42
6.1 Methodology for Build KPI 43
6.2 Build reliability KPI 45
6.3 Build speed KPI 46
6.4 Learnings 48
7. Technical infrastructure 50
8. Conclusion 53
What can be done next? 55
Kokkuvõte 56
Järgmised sammud 58
Bibliography 59
Appendixes 60
Appendix A: Extraction, Transformation, Loading 60
Appendix B: Full table for improvement drivers 61
Appendix C: Tables and charts for TPS tickets 62
Appendix D: Additional tables and charts for alerts and incident management process 65
Appendix E: Build Reliability Charts 66
Appendix F: Build speed charts 67

1. Introduction

1.1 Scope and motivation

Huge amount of data is generated when people use online services. The data exist in the form of log files, text documents, reports from finished tasks either by people or machines. Are these records used? Most companies do store a lot of the available information. However rather often not much attention is paid to how the data is managed. As the activities around data are not very well thought through, gathered information will not offer much support to the business needs of the company. Only a small amount of the stored data gets structured and used to support decision making process. *"Understanding the whole process helps to structure data mining projects so they are closer to systematic analyses rather than heroic endeavours driven by chance and individual acumen"* (F. Provost and T. Fawsett, 2013,p.19). The systematic usage of data and metrics to describe teams' goals improves the quality of the service and increases the benefits that the system can offer.

The thesis is based on the example of Skype Engineering Services Team. The author himself is a member of that unit. The data generated in the Skype Engineering Systems can be separated into two groups.

- The data that can be used to understand and improve the service by its owners. This includes the statistics regarding the usage of the systems: who and when use the system? The performance of the system: how long does it take for the system to respond to customers' requests? The reliability of the system: how often does the system have unpredictable outages that render the functionality unusable?

- Skype Engineering Systems contain a huge amount of data describing the work process and quality of the products being developed. The available data includes various test reports, descriptions of dependencies to other products, overviews of different workflows etc. This kind of data would allow the customers to optimize their workflows and improve the quality of their products.

The thesis focuses mainly on the data from the first category. The work producing the thesis is carried out during the time span of almost one year from June 2014 until April 2015.

1.2 Goals

The purpose of the thesis is to make the data generated in the Skype Engineering systems useful for the owners and the users of the services. The main goals of the thesis are the following.

- To understand the role of data and metrics in the service management and software development process.
- To establish Key Performance indicators and other metrics for Skype Engineering Services Team. One of the purposes of the metrics is to support the planning process. The second purpose is to understand the progress made towards the goals set by the team and the company.
- To find the best means for gathering, surfacing and using data to manage and support an Engineering System.

Based on the work carried out in the thesis, Skype Engineering Services Team will be able to objectively measure the impact of their everyday efforts. The metrics can be used for unambiguous conversations within the team. Also the metrics can be communicated out the user of the Skype Engineering Services.

1.3 Methodology

The main goal of the thesis is to suggest and implement improvements in the Skype Engineering Services Team. Getting to understand the best practices and the current system support that goal. Therefor several combined methodologies are used in the thesis.

- An **analysis** is carried out to understand the data usage in Skype Engineering Services Team.
- **Research** is done regarding the role of metrics in software development lifecycle.
- Different tools and methods for gathering and analysing data are **compared.**

- Real live sets of data are **analysed.**
- New process and tools are **implemented.**

1.4 Overview

1. Chapter – Introduction to the thesis. Establishes the scope and goals. Gives an overview of the used methodology.
2. Chapter – Examines best practises regarding the usage of data and metrics in different organizations. Advice and benefits regarding the implementation of KPIs is studied.
3. Chapter - Gives an overview of the role and setup of the Skype Engineering services. Provides an summary how data was gathered and used in Skype Engineering Services Team when the work on the thesis was started in June 2014.
4. Chapter – Examines the monitoring and incident management process in Skype. The relevant data and metrics to measure the effectiveness of the incident management process are gathered. Suggestions for improvements are made and their implementation is described.
5. Chapter – Gives an overview of the Third Party Software management process in Skype. Metrics for understanding the status of the process are established.
6. Chapter – Gives an overview of the Build Systems in Skype. KPIs and other metrics for understanding the status of the process are established.
7. Chapter – Gives an overview of the technical infrastructure supporting the metrics in the Skype Engineering Services Team.
8. Chapter - Conclusion.

Kokkuvõte

Bibliography

Appendixes

2. Best practices for implementing metrics

Computer science and software development industry are growing in a very rapid pace. The problems that modern large scale software development companies face did not exist 10 or even 5 years ago. A lot of research, methodologies, practices and models are produced to turn software development and maintenance process into a more predictable and manageable process. There are various best practices and tools that support these goals. This chapter examines the recommendations for implementing KPIs and other metrics. Naturally no perfect solution exists that fits the needs of all businesses. Balanced Scorecard is one of the most widely known and used tools. It provides a holistic view of the organization. Implementing a whole strategic vision and performance monitoring system is too large undertaking to fit the scope of this thesis. The target of this chapter is to establish viable goals and strategy for the rest of the thesis.

2.1 The characteristic and number of metrics

The first thing to do when establishing KPIs is to understand the real purpose of the proposed metrics. Establishing a few characteristics while forming the metrics is useful for making a correct choice.

"From extensive analysis and from discussions with over 1,500 participants in my KPI workshops, covering most organization types in the public and private sectors, I define seven KPI characteristics:

1. *Nonfinancial measures (not expressed in dollars, yen, pounds, euros, etc.)*
2. *Measured frequently (e.g., daily or 24/7)*
3. *Acted on by the CEO and senior management team*
4. *Understanding of the measure and the corrective action required by all staff*
5. *Ties responsibility to the individual or team*
6. *Significant impact (e.g., affects most of the core critical success factors [CSFs] and*

more than one BSC[2] perspective)

7. *Positive impact (e.g., affects all other performance measures in a positive way)"* (D. Parmenter, 2007, p.5)

These characteristics will be taken under consideration when implementing KPIs for the Skype Engineering Services Team.

One of the papers used as a reference is a survey conducted in Philips. *„The paper presents and discusses improvement targets, improvement drivers, and metrics, and the degree to that they are being recognized in the software groups. "*(J. J. Trienekens et al, 2007,p.135). The study looks at *the level of metrics activity* and *the usage of resulting data* presented in "Table 1 - Metrics: level of activity in Philips"

Table 9 Metrics: level of activity

Question	Score					*t*-test		
	All	CMM1	CMM2	CMM3	ANOVA	1 2	2 3	1 3
% Groups with formal metrics program	48.0%	26.3%	38.5%	100.0%	**0.00**	0.48	**0.00**	**0.00**
Number of metrics	7.37	6.16	6.31	11.50	**0.00**	0.92	**0.00**	**0.00**
% of projects with evaluation*	3.06	2.45	3.31	3.90	**0.00**	**0.05**	**0.05**	**0.00**
% of projects report quantitative data*	2.62	1.90	2.62	4.00	**0.00**	0.12	**0.01**	**0.00**

*Please note that these questions were scored on a 4-point scale: (1 = 0–25%; 2 = 25–50%; 3 = 50–75%; 4 = 75–100%).

Table 1 - Metrics: level of activity in Philips

The table shows that the average number of metrics used by a team varies from 7 to 11 for the groups that have a metrics program. This is in accordance with the suggestions and references by (D. Parmenter, 2007). Another interesting note from "Table 1 - Metrics: level of activity in Philips" is the percentage of teams on each CMM level that has a formal metrics program. According to (J. J. Trienekens et al) the average time required to move up a level is between one and a half and two years. This shows that the time it takes to implement up to 6 functioning KPIs in practice is measured in years rather than months. Based on these studies it is reasonable to set the target for Skype Engineering Services to establish 1-3 KPIs and up to 10 other useful metrics in the course of 1 year.

[2] Balanced Scorecard

2.2 Success Factors

Setting up a successful data and metrics project can be separated into 3 main Tiers. The most technical activities are carried out by BI experts. They set up an infrastructure to obtain and store the data. These activities are also known as Extract, Transform and Load. The second group of people are analysts who help to present the data in a meaningful way to the end-users. They create models and defined strategies how the data should be used. The last level is made up by the consumers of the data. They are the people who use the inforamtion to shape their decisions and strategies.

The activities carried out on the first 2 levels are complex and require expert knoweldge. However they are rather technical in their nature. Fairly standard procedures are applicable in majority of BI projects. The most valuable and also the most critical input is added on the consumer level. If the data is not used for decision making and the models are not adjusted to the companies needs the metrics initative is sure to fail. *"It has been argued that, perhaps due to the lack of integration of BI into the decision making process, more than 50% of BI implementations fail to influence the decision-making process in any meaningful way"* (A. Pourshahid et al, 2014, p.3)

Many interesting aspects to observe while carrying out a metrics projects were highlighted by (J. J. Trieneken et al.) when the question *"what are considered to be important improvement drivers for software groups*?" was asked. The survey established the following 7 drivers as the most of import for the success of the metrics program.

1. *"Commitment of engineering management*
2. *Commitment of development staff*
3. *Sense of urgency and perceived need to improve*
4. *Availability of engineers time for SPI*
5. *Commitment of business management*
6. *Availability of qualified SPI resources*
7. *Clear/quantifiable improvement targets"* (J. J. Trieneken et al.,2007,p.144)

The results received in their survey are in accordance with the argument by (D.Parmenter, 2007) presented in the previous section. Understanding risks in important to maximise the value delivered with the metrics program. As illustrated previously the most critical link in the

chain is the actual usage of the data to support the decision making. Based on the conclusions drawn from (J. J. Trieneken et al., 2007) and (D.Parmenter, 2007) a very big emphasis will be put on facilitating and monitoring the relevance and usage of the surfaced data in this thesis. The steps of gathering, storing and analysing records will be carried out iteratively to validate the relevance of the data and models by the customer.

2.3 Technical infrastructure

When a metrics project is carried out it is necessary to have the tools in place to gather, store and visualise the data. The focus of this thesis is on establishing and using the correct metrics rather than building a full scale data mining solution. *"The project team should promote the use of existing in-house applications for the collection and reporting of the performance measures for at least the first 12 months. Much can be done with standard applications such as Excel, PowerPoint, SharePoint Team Services, and Access."* (D. Parmenter, 2007, p.32)

Two principles regarding the technical infrastructure will be kept in mind while writing this thesis. For the first implementations of the ideas presented in this thesis very lightweight tools will be used. The main focus will be put on the interpretation of the metrics and getting fast feedback. Based on the feedback changes will be made to the metrics to fit the needs of the organization. Even thou currently the main focus is not on the technical implementation a scaling architecture is the second thing to keep in mind. Once we have established the metrics we want to use within the organisation we need to be able to provide a scalable, reliable and maintainable platform for everyday usage.

2.4 Benefits of using KPIs and metrics

There are numerous benefits of using data and metrics to understand your business. First of all it is very difficult to control a process that is not sufficiently understood. Without adequate information it is difficult to take decisions. Also the impact of the decisions will not be known. "*14% of the respondents answered the access to a common pool of accurate, timely information which allows decision makers to monitor progress and take corrective actions promptly. In this way, 13% of respondents argued that KPIs' mechanism contributes to the minimization of errors. Furthermore, 10% of respondents consider that KPIs are necessary tools for decision makers." (*K. Konsta, 2012, p.152*)* Therefor correctly using KPIs gives the

business manager the opportunity to exercise insightful control over the processes he is responsible for.

In addition, using metric creates a good platform for meaningful and unambiguous communications. *"By using KPIs the company's objectives are translated into, and measured by, a set of targets for the manager to be achieved. Moreover, 6% of respondents consider that KPIs contribute to the proper implementation of company's programmes and 12% of respondents think that KPI measurements conduce to the improvement of internal organization." (*K. Konsta, 2012, p.152*).* KPIs work as powerful means for sharing vision and goals with the organisation. Influence of properly shared vision on Financial Performance, Staff satisfaction, Customer satisfaction, Productivity and Staff/Manager Tenure were studied by (F. F. Jing et al., 2013). The research found positive impacts to all of the mentioned attributes. *"Vision-communication and -sharing were significantly related to retaining both managers and staff, which in turn enhances the bottom line, not only through direct savings, but also by retaining an understanding of the organization and its customers. Performance and productivity increase under both long-term managers and staff"* by (F. F. Jing et al., 2013.) KPIs help to reduce the misunderstandings and increase the cohesion within the organisation. When communicated properly KPIs also help to improve the communication with external stakeholders. *"Furthermore, 12% of respondents think that the KPI measurements conduce to the minimisation of disputes as well as to the improvement of competitiveness. In addition, 10% of respondents argued that KPIs lead to improvement of customer relationships."* (K. Konsta, 2012, p.152).

2.5 Summary

The following list will summarise the main learnings that will be followed during the rest of the thesis.

- A chosen KPI needs to be something that is understandable and approved by the whole team. This includes the management, developers and the people that operate the services. One of the success criteria for the KPI is whether people can act based on it.

- It takes time to introduce functioning KPIs. The number of KPIs used in a team should be smaller rather than bigger. In the context of Skype Engineering Services it is reasonable to introduce 3-5 in an iterative process during a 12-18 month period.

- While introducing new metrics and KPIs the main focus should be on the actual usage of the data. Tools used for data mining and Business Intelligence are important. Also the scalability of the infrastructure should be kept in mind. However the main effort should be applied to defining the correct metrics.

3. Overview of the Skype Engineering Services

This chapter gives and overview of the role and structure of the Skype Engineering Services Team. The provided description is from June 2014 when the work on the metrics was started. In this chapter the word “current” refers to the same interval when used in this chapter.

3.1 Description of Skype Engineering Services Team

Skype Engineering Services is a team supporting Skype and Lync Engineering organisation within Microsoft. There are close to 4000 users for the services located in 10 bigger and numerous smaller offices around the world on many different continents.

The size of the Skype Engineering Services Team is around 40 people divided into smaller sub teams. Some sub teams have specialized functions. The functions include system administration, providing level 1 support to end users, developing or maintaining some specific service. The following list describes a conceptual overview of the services by their function.

- **Agile workflow tools** support work item and defect tracking and reporting. These services help developers, testers, engineering managers, product managers and other interested parties to plan and track product development and maintenance.

- **Source code management tools** help developers and testers to maintain and share their source code.

- **Source code analyses tools** support quality assurance activities on the source code and object code level. Examples of such tools are Static code analyses and code review tools.

- **Artefact management** provides the framework for producing and consuming binaries and executables. Artefacts are the outcomes of a software build. The end result of one team is often used as the input for another team. Artefact management is also a key starting point for live deployments.

- **Build Services** provide an automatic infrastructure for developers to share their work results with their team and with the rest of the company by building and sharing their software. They also use the artefacts from other developers and teams in their build process. Technically the Build Services consist of a Build orchestrator and of more than 500 build machines. The Build orchestrator helps to set up and schedule the necessary Builds machines which carry out the build tasks. Source code management and artefact management tools are also very tightly integrated into the Build infrastructure.

- **Test tools** provide a unified set of tools for quality engineers to maintain test cases, run some of the generic tests and gather feedback from the beta testing process.

- **Other.** Not all the tools fit exactly into the current classification. One of those tools Third Part Software management tool, which deserves to be mentioned. The aim of the tool is to support developers' efforts to be compliant with the legal and technical requirements associated with the usage of third party software.

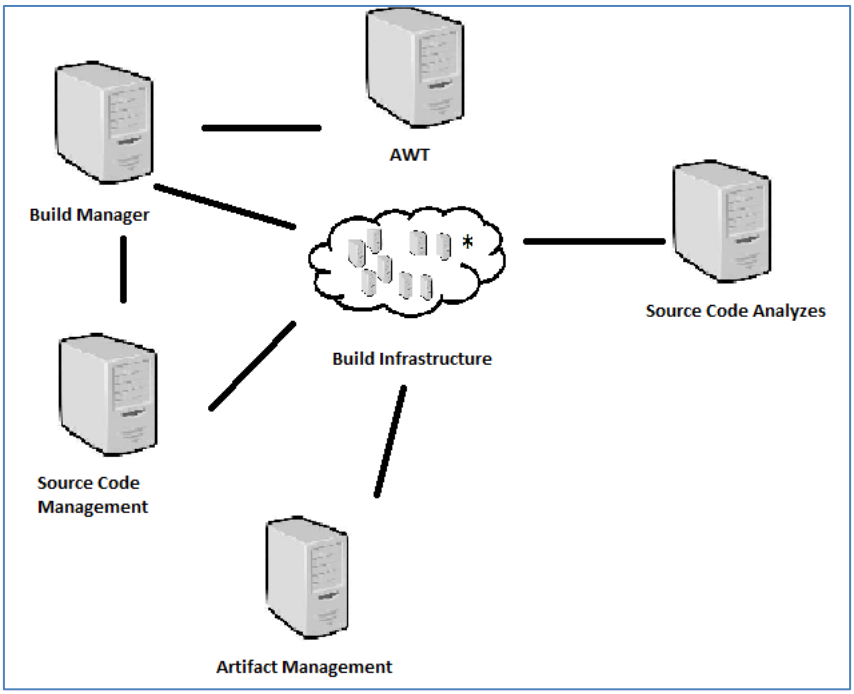

Figure 1 - ES conceptual architecture

Skype engineering services has evolved over the past 10 years. The requirements for the services have changed and will continue to change rapidly in the future. As technology

progresses and various legacy systems have to be merged there is a need to support a large selection of different technologies. The main reasons for this complexity are the following:

- A very wide variety of technologies that need to be supported due to the nature of developing multiplatform software.
- Merger of two Engineering organizations (Skype and Microsoft Lync) with legacy systems that cannot be directly replaced by one another.
- The need to implement new technologies with a limited negative impact to the organization during the adoption period.
- The need to unify different services and processes with a limited negative impact to the organization while the changes are being made.

Due to the reasons mentioned above there are several duplicated functions that need to be supported in parallel. Also considering the size of the organization the implementation of the model depicted on "Figure 2 - ES conceptual architecture" is rather complex.. The following list of similar services gives a better understanding of the complexity of the implementation of the architecture.

- There are currently 5 source code systems that are being used.
- There are 5 bigger artefact types that need their own management tool (or at least a very advanced management capability within another tool).
- There is a transition happening to adopt a new build Management system. How long the full system migration takes is not yet known. It can be up to several years. Until then both systems need to be supported.
- The same applies to Agile Workflow tools. There is a plan to start a migration to a new system, but the timelines are not clear and there will be a long period during which several systems need to be supported in parallel.

The need to integrate different systems with one another adds also adds a new layer of complexity.

3.2 Engineering Services data management and usage

The data management and usage inside the Skype Engineering Services Team was analysed. Two groups of persons were interviewed: the people operating the services on a daily bases and the people responsible for planning the future developments. The tools for gathering and presenting the data were also examined. The goal was to find an answer to the following questions:

- What tools are used for gathering data from engineering services?
- How is the data currently used?
- What are the obvious gaps and problems of the data usage?
- What improvements should be done to instrumentation and data usage to support the decision making for engineers and managers?

The next section will give an overview of the main data gathering and analytic systems that are currently used within the Skype Engineering Services Team. The general purpose of the tool and the actual usage by the team members is described.

3.2.1 AW stats

AWStats[3] is one of the most widely used Web analytics tools in the world. It is distributed under the GNU General Public License (GPL)[4]. AWstats generates reports based on application log files. The main functionality includes the usage of different web resources broken down by criteria such as dates, regions etc.

Most of the tools in Skype Engineering services have AWstats enabled. The systems provide the information of their usage which can be accessed and viewed online by any interested party. However the data from AWstats is not used very often. An example use case is the identification of accounts (usually automated service accounts) with abnormally large activity when the systems in questions have performance issues.

When talking to the Product Management team there was no indication that the data from AWstats is used to support the planning of the future development and maintenance efforts of

[3] http://www.awstats.org/.
[4] http://www.gnu.org/copyleft/gpl.html

the services or to monitor and understand the impact of developed functionality. In addition some of the data shown in AWstat is very obviously incorrect. For an example for some periods of time the records indicated no activity at all. Second example is that in some cases the numbers in AWstats showed very high level system usage that clearly could not have happened.

3.2.2 Log Stash

For storing and analysing logs the Skype Engineering Services Team has just started to use 2 tools: Logstash[5] and Kibana[6]. Both of the applications are third party software and distributed under Apache 2.0 license.[7] Logstash is a tool for storing and analysing log files. Logstash is designed to be easily integrated with other tools. One of those tools is Kibana, an application used for visualising the data output from Logstash. Some of the sub teams in Skype Engineering Services Team use Kibana and Logstash for searching logs. Some rules and alerts are created based on anomaly detection. Current rules are not very reliable when it comes to defining the overall status of the application. Logstash and Kibana are mostly used for resolving some very specific problem with a special search created to address a particular question.

3.2.3 Nagios

Nagios[8] is one of the most widely used monitoring and alerting systems in the world. The components of the software are distributed under different licenses. These licenses include Nagios Open Software License[9], Nagios Software License[10] and GPL[11]. Nagios provides functionality to sample the state of an application or its component. Based on the results, Nagios can send alerts to interested parties. The checks can be done either by a Nagios standard function or by a custom script written by the service owner. The functions under monitoring can be roughly divided into two categories.

- Infrastructure related checks such as CPU, disk, bond and network access.

[5] http://logsatsh.org
[6] https://www.elastic.co/products/kibana
[7] http://www.apache.org/licenses/LICENSE-2.0.html
[8] http://www.nagios.org/about/overview
[9] http://assets.nagios.com/licenses/nagios_open_software_license.txt
[10] http://assets.nagios.com/licenses/nagios_software_license.txt
[11] http://www.gnu.org/licenses/gpl.html

- Accessing the application in the same fashion as regular user would. Sending several http requests to the services and analysing the received answer.

The main working principle of Nagios is simple. An agent runs one of the checks and reports back the results. Nagios can be configured to run many checks before reporting that the system has problems.

Nagios is enabled for a large majority of tools in Engineering Services. The alerts are delivered by e-mail. System administrators also check Nagios dashboards on daily bases to get an overview across the services under their supervision.

The Nagios data is used differently across the sub teams in Skype Engineering Services. Some teams make use of it; some teams take practically no benefit. The volume of the alerts is in hundreds per day, suggesting that majority is being ignored. Some teams do react to some of the alerts. None of the teams are able to use the data to prevent problems. The data generated by Nagios and its usage across the Skype Engineering Services Team will be covered in more depth in Chapter 4.

3.2.4 Other

.All of the system managed by Skype Engineering Services Teams store data related to the functions carried out in the application by the end users. Often this data is can be viewed within the tool or retrieved by API. Applications provide easy ways to create dashboards and heat maps with data specific to the given application. These reports are very helpful but they are mostly used to provide information on very specific problem and each report is used by a small amount of people. In addition several sub teams of Skype Engineering Services Team have tried to implement dashboards for getting quick status update across the systems under their control. These efforts have so far either failed or have no significant usage.

3.3 Summary

Two diverse points characterise the data usage in the Skype Engineering Services Team. Firstly, several systems have been set up to gather data. The aim of these systems is to provide insights and awareness. On the other hand, the data initiatives are rather incoherent. The data usage usually takes place based on very specific needs by a small group of people disconnected from the rest of the team. Therefor the impact and benefits obtained are very

limited. The following list summarises the findings around the data usage in Skype Engineering Services Team:

- A fair amount of data is generated and stored by the Skype Engineering Services Team. However the data is mostly unstructured and under used.
- Data is mainly used to support operational decision making and not product planning.
- Relevant data is mostly obtained when the specific need arises.
- Using data is expensive and the return on investment is considered to be too low to depend more on data during the planning process.

As the previous section pointed out, there are very obvious gaps in data management and usage. However the current situation in the Skype Engineering Services Team does provide a platform for building a more structured and beneficial working model. By providing a more structured approach to data usage and management a lot more operational and managerial decisions can be based on data in a lot more coherent fashion.

4. Metrics for alerts and Incidents

The purpose of KPIs and other metrics is to provide support for decision making. During the time of writing the implementation of new Incident Management process was in focus for the Skype Engineering Services Team. The reasonable thing to do was to align the work done on metrics with the efforts from the rest of the team. Therefor the author gathered, structured, and analysed records related to incidents. The purpose of the data was to quantify the impact of the new Incident Management process. Two sources of data were available:

- Records of manually tracked occurrences of incidents. This data was manually entered and kept in the issue tracking tool Jira.

- Data saved by automatic monitoring service Nagios. The monitoring and alerting had been set up by the members of Skype Engineering Services Team.

Current chapter will give an overview of the incident management process and automatic service monitoring in Skype Engineering Services Team. Several views of the captured data are presented. Some of the problem areas are surfaced and several proposals for improvements are made. Also the results from the implemented improvements are shown.

4.1 Background

In the beginning of May 2014 a new Incident Management process was implemented within the Skype Engineering Services Team. The new process had two main goals:

- Assure that customers are informed of ongoing incidents. Process was sending out updates on recovery progress was put in place.

- Formally track and analyse existing incidents. The goal was to understand the root cause of the problems and improve the service based on the learnings.

According to the new process, an event is considered to be an incident if the service is affected in such a way that it is visible to the end-user and prevents them from carrying out their tasks within the system. Alternative definitions do exists. One of them states that an incident is "a*n unplanned interruption to an IT service or reduction in the quality of an IT*

service. Failure of a configuration item that has not yet impacted service is also an incident, for example failure of one disk from a mirror set." (ItilFoundations, 2014). In this chapter the word *incident* is used as it is defined in the Incident Management Process by the Skype Engineering Services Team. The alternative definition is presented to demonstrate a strong link between incident management and application monitoring. "*In the fields of information technology and systems management, Application Performance Management (APM) is the monitoring and management of performance and availability of software applications. APM strives to detect and diagnose application performance problems to maintain an expected level of service"* (Wikipedia). Alerting based on monitoring is an automatic and objective activity carried out by another application. It is not based on human perception, but an actual measured response by the system under monitoring. The results, both positive and negative are automatically captured and stored.

4.2 Alerts and Incident Data

There is no finite and strict list of service definitions in the Skype Engineering Services Team. In order to support the incident management process a table listing the services was assembled. In addition to listing the services the following questions were answered regarding each area:

- What is considered to be the normal state of the service?
- How to understand if service is experiencing an incident? What priority does the incident have?
- How is the performance of different services measured?

The list is operational but not complete. Not all the services provided by Skype Engineering service team are correctly defined. Also some of the associated data is missing. However the table can be used as a starting point for understanding how services are monitored. Based on the list it can be said that there a roughly 20 different services. In this chapter 5 bigger and most important services are analysed.

The incident management process was put in place in the beginning of May 2014. The data gathered during the first 2 months of the process was examined. "Table 2 - Total number of alerts in ES" gives a broad overview of the data captured and sent out by Nagios.

Period	SERVICE CRITICAL (HARD)	SERVICE UNKNOWN (HARD)	SERVICE WARNING (HARD)	Grand Total
2014	2466	20	1960	4446
May	1343	8	990	2341
June	1123	12	970	2105
Grand Total	2466	20	1960	4446

Table 2 - Total number of alerts in ES

First of all the great number of alerts strikes out from the table. The total number of alerts is incredible large. The big number of critical alerts is even more interesting. 1343 alerts per month equals roughly 65 critical alerts per each working day. The number indicates either a very unusable service or a fact that service monitoring is configured in such a way that it provides a lot of noise. While talking to the service owners it is agreed that the second hypothesis is in fact true. The services were not abnormally unstable during those 2 months. The misconfiguration of alerting was creating a constant stream of noise irrelevant of the actual service level. "Figure 2 - Tracked incidents" suggests the same. It is visible that the number of manually tracked Incident was between 17 and 27 which differ greatly from the number of alerts.

Period	Gitorious	Jira	Nexus	Pam	Quickbuild	SVN	Grand Total
2014	117	886	59	4	64	1336	2466
May	96	382	17	3	44	801	1343
June	21	504	42	1	20	535	1123
Grand Total	117	886	59	4	64	1336	2466

Table 3 - Critical alerts per service

Period	cpu	createissue	disk	https	java	load	ntp	qb_queue	quicksearch	Grand Total
2014	2161	142	44	41	2	7	4	51	14	2466
May	1255	11	14	21	2		3	31	6	1343
June	906	131	30	20		7	1	20	8	1123
Grand Total	2161	142	44	41	2	7	4	51	14	2466

Table 4 - Critical alerts per item

"Table 3 - Critical alerts per service and "Table 4 - Critical alerts per item" give a more detailed overview of the alerts. The tables indicate that 88% of the alerts are caused by CPUs. When talking to service owners it came out that the current CPU alerts are not used at all. Single CPUs are being monitored on multicore machines with more than 20 CPUs. In this setup a single CPU alone does not define the performance of the application. Therefor the

alerts do not give any information at all. Instead of being redefined the CPU checks are currently simply being ignored.

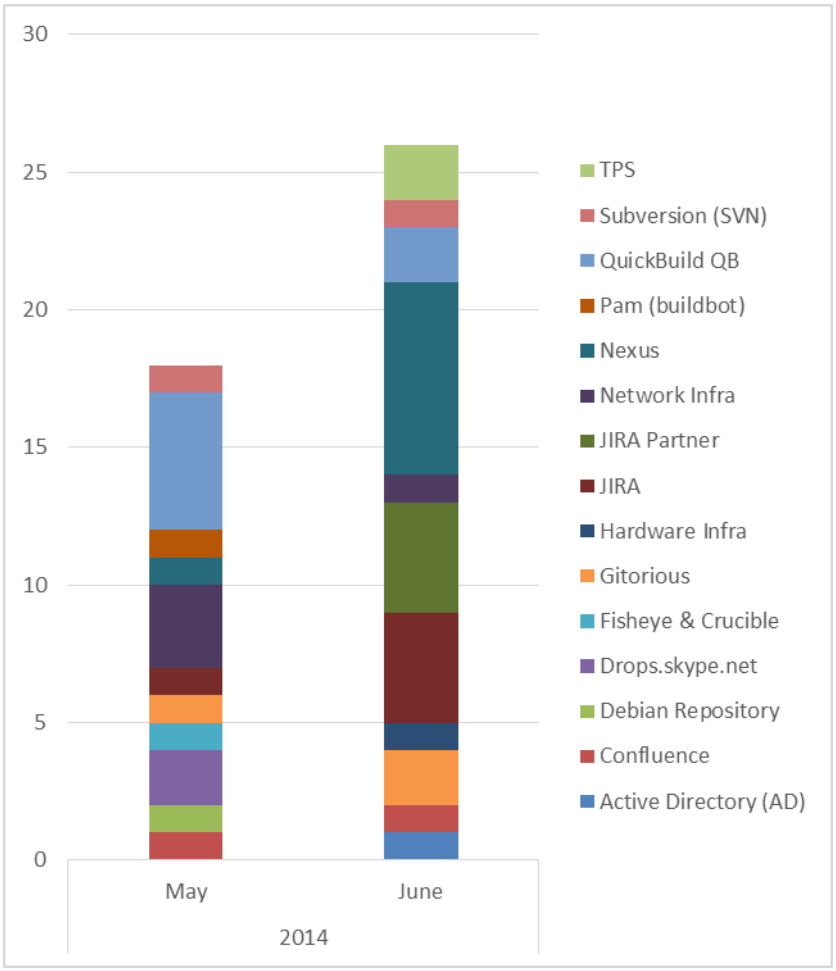

Figure 2 - Tracked incidents

4.3 Learnings and improvements

The initial objective for analysing the alerts and incident data was to understand the impact of the incident management process. While gathering and analysing the first set of monitoring data, it became evident that the objective cannot be met. There are too many alerts that a meaningless and do not indicate the level of system health. The definitions of useless "Critical alerts" are not being changed, they are simply being ignored. The number of alerts that are being ignored has created such amount of noise that the useful alerts simply get lost.[12]

[12] The level of relevance and usefulness of alerts differs as the alerting is not set up and used by one group of people. Alerts are managed and used by different sub teams in Skype Engineering Services Team.

Even though the main objective was not met, the visualisation of the monitoring data gave the following results.

- Awareness was created in the Skype Engineering Services Team of what and how is currently being monitored.
- Main source of noise was discovered.

Based on these insights the following actions need to be taken in order to make better use of the automatic monitoring

- The amount of noise needs to be reduced.
- The process of managing alerts and alert definitions needs to be included into the daily workflow of the service teams.

The data indicated that around 88% of alerts came from monitoring CPUs. The service owners indicated that none of the alerts associated with CPU monitoring was used. The solution was to either stop monitoring CPUs altogether or improve the process to minimize

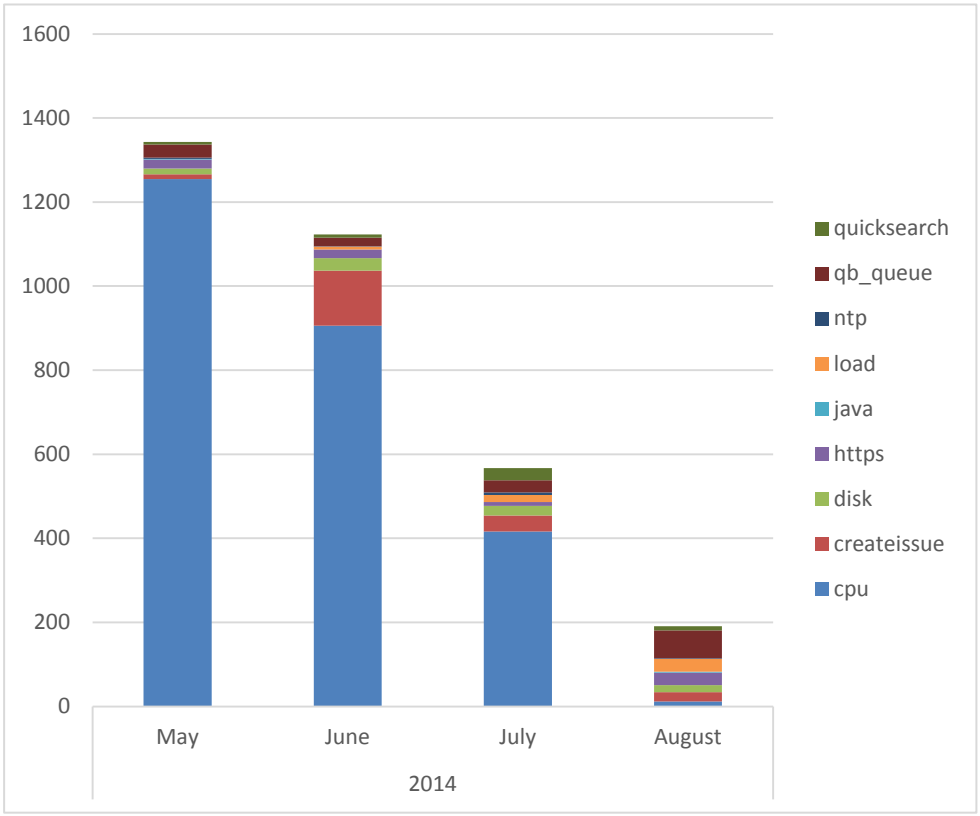

Figure 3 - Change in alert count

the number of false positive alerts. The task of improving the CPU monitoring turned out to be a lot easier than thought. The improvements were implemented during the month of July. The resulting change in overall number of alerts can be seen on "Figure 3 - Change in alert count" The number of alerts in May has dropped by 86% compared to the number in May. While the reduction of noise does not directly improve the service quality it does provide the following positive outcomes:

- Increases drastically the percentage of relevant alerts. This makes it a lot more probable that alerts are used.
- Makes it a lot easier in the future to analyse the incoming alerts due to the reduction of their volume.

"Figure 4 - Alerts, incidents and communication handled in separate threads" describe the work flow where Nagios alerting, incident management in Jira, and communication to the end users is handled in three parallel threads. According to this model alerting and incident

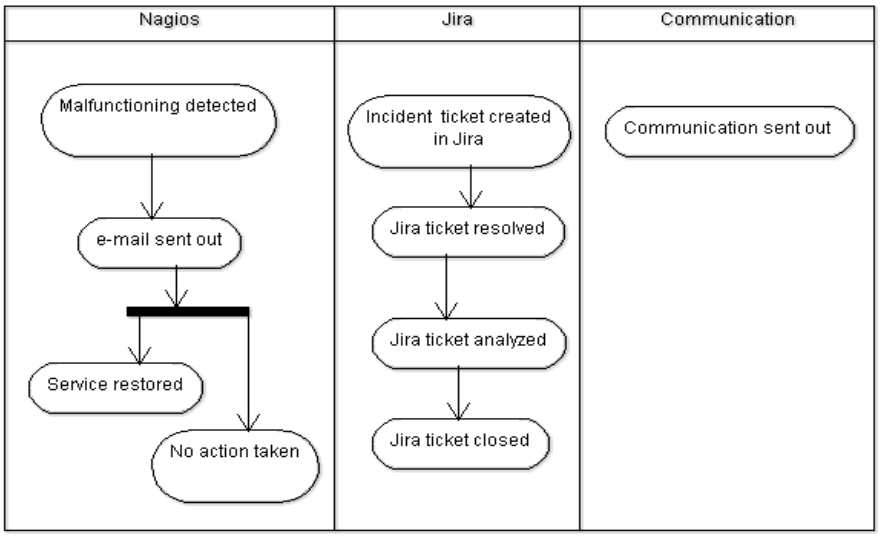

Figure 4 - Alerts, incidents and communication handled in separate threads

management is not integrated in way. Efforts are duplicated but alerting has no impact on the incident management process. The author proposed a model depicted on "Figure 6 - Workflow based on Jira tickets". The purpose of the model is to tie the three processes together. Incidents should be managed based on automatic alerts. Communication should be automatically generated based on Incident reports and status changes in Jira. During the incident review process, monitoring and alerting should be improved. False positive alerts

should be amended or removed. Incidents that were not created by automatic monitoring help to point out the areas were monitoring should be enhanced. During the writing of the thesis the integration between alerting and incident management process was implemented and piloted for a few services.

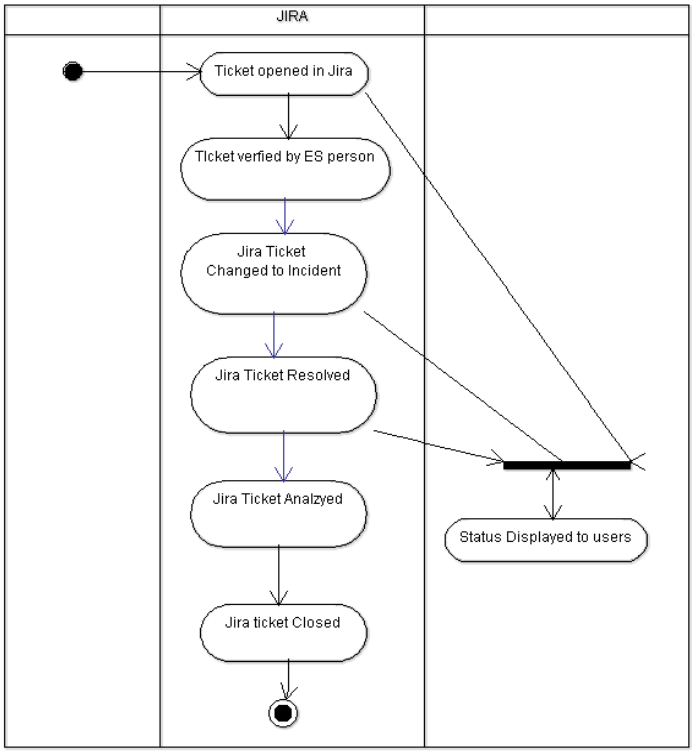

Figure 5 - Workflow based on Jira tickets

4.4 Summary

The objective of this chapter was to quantify the impact that the new Incident Management process had on the performance of the services. This goal was not met. The assumption had been that the change in the number of alerts is an indication of the change in the application performance. This assumption proved to be invalid as the monitoring and alerting system was not used properly. The monitoring and alerting needed to be changed so that the data would reflect the actual state of the systems. Therefor the focus shifted to improving the alerting process. The solution consisted of two parts. Firstly, the alerts that obviously were the biggest source of the noise were removed. In addition a process was implemented that tied the alerting into the incident management process. Having these processes coupled assured that alerts were used to get notifications about occurring incidents. In addition the change enforced that alerting was constantly reviewed during the incident review sessions.

5. A Key Performance Indicator for Third Party Software management process

When software is produced, a lot more components are used than just the source code developers themselves write. Third party libraries are used from the first party code. Tools written by other companies are used to test, analyse or otherwise manipulate first party components. Code snippets implementing some useful algorithms written by other people get embedded to source code.

Third party components are accompanied by a license describing how the component can or cannot be used. There are tens of different communal licenses used for software components. In addition everyone is allowed to make up their own proprietary licence as they see fit. There are still many caveats using a software component that comes with a free license such or when a fee is paid for using the component. Even though a seemingly free or already paid for license is used, negative legal and financial consequences might follow from the misuse of the component. The use of a licensed component might have restrictions, not covered by the obtained rights. Utilisation of the licensed component might set demands to how the first party product itself can be licensed.

In addition quite often the due diligence that has to be done for using a third party component is not limited to understanding and correctly handling just the license of the component under question. A component obtained by a dependency management tool such as ivy or maven might be in turn using other dependencies that are hidden from the maintainer of the original product. Also the license for one version of the component might not be valid for the next version of the same component, while the upgrade process might be seamless and go unnoticed by the involved parties.

Third party software management is complex and risky process. Especially for a company the size and prominence of Microsoft. In order to reduce the effort to correctly attribute Third party software in Skype and to increase the compliance to various legal requirements a special TPS process has been put in place. In addition, a custom developed tool called iTPS has been implemented to accommodate the process. The tool offers an easy entry point for a user who wishes to declare his TPS usage. The tool facilitates the component impounding and license

review process. The system returns an answer in case the component in question is already fit for use or a new impounding process has to be started. iTPS is integrated with other tools, one of them being Jira where the tickets for new TPS requests are handled.

One of most important attributes of the TPS process is the time it takes for the requester to get his ticket approved. The SLA is provided by the TPS process manager and the head of Skype Engineering Services. At least 85 % of the TPS tickets need to be approved in less than 14 days. The next sections will describe how such metric can be obtained. The learnings from the statistics will also be reviewed.

5.1 Approval time as a KPI

The Service Level Agreement "*at least 85 % of the TPS tickets to be approved in less than 14 days*" is a straightforward metric to measure. The data can be obtained straight from the issue tracking system Jira where the tickets are stored. Jira does have detailed history of the tickets and provides relatively easy means to query the information. It is possible to get the ticket *created date* and the ticket *approved date* for each of the tickets.

"Table 6 - The percentage of tickets approved in less than 14 days" and "Figure 6 - The % of tickets approved in less than 14 days" show the data regarding the TPS KPI. Both the table and the graph have the same data depicted in a different format. The columns for the table are explained in "Table 5 - Explanations for TPS KPI columns".

% approved under 14 days	This column shows the % of the tickets closed in less than 14 days. It takes into account the tickets that were closed on a given day or during a given period.
Status	The colour indicates whether less than 80% of the tickets were closed in 14 days or not.
Total Approved	The total number of issues closed on a given day or period
Open over 14 days	The average number of tickets open on a given day or a period and already open for more than 14 days

Table 5 - Explanations for TPS KPI columns

The tables presented in this chapter do not show the data for all the KPI tickets. The requests are classified by TPS, to be either used *internally* or to be distributed. The reasoning for such a separation is the different nature of their use.

Tickets for Distributed TPS				
	% approved <14 Days	Status	Total Approved	Open over 14 d
2014	62.4		359	52
Q1 2014	71.4		63	
January	66.7		9	
February	81.8		22	
March	65.6		32	
Q2 2014	27.3		55	
April	29.2		24	
May	40.0		5	
June	23.1		26	
Q3 2014	71.1		180	
July	72.2		18	
August	84.4		77	
September	58.8		85	
Q4 2014	59.0		61	52
October	73.3		45	
November	9.1		11	
December	40.0		5	52
2015	54.5		11	80
Q1 2015	54.5		11	80
January	54.5		11	80
Grand Total	62.2		370	71

Table 6 - The percentage of tickets approved in less than 14 days

For an example the internal consumption of TPS is less complex and risky to handle. Therefor obtaining the approval can be a faster process. The data sets for internal and distributed TPS are rather similar in nature and the same conclusions can be drawn from both. Therefor only the data for Distributed TPS is presented in this chapter. The tables and graphs with the statistics for internal TPS is presented in "Appendix C: Tables and charts for TPS tickets"

It is visible from the charts that the SLA set for the TPS program is not met. The percentage of the tickets approved in less than 14 days rose from 23 % in June to 72% in July and 84% in August but declined to 58 % in September. Therefor the service level did not show a sustainable improvement over a longer period of time. The next chapter will give a deeper insight of the learnings obtained from compiling and interpreting the TPS statistics.

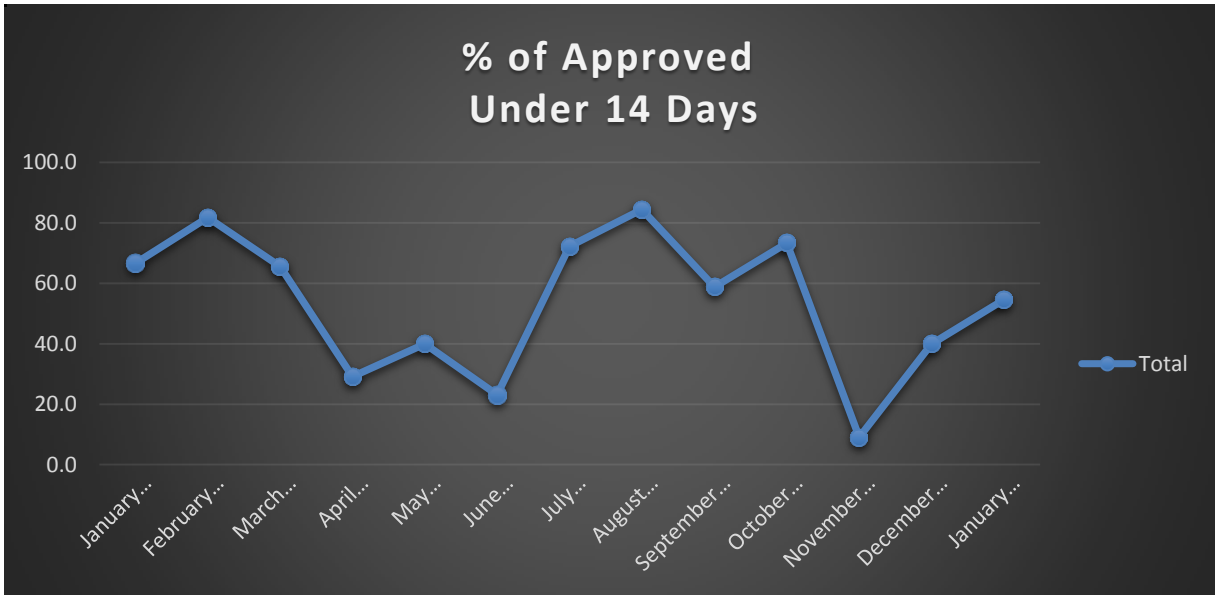

Figure 6 - The % of tickets approved in less than 14 days

5.2 Implementation of the KPI

The purpose of the TPS KPI data is to understand the performance of the service during a given period of time. This aspect raised 2 interesting question. How long should the time period be? Should the tickets be chosen based on the start or the end date? The SLA definition given by the ES management: "*at least 85 % of the TPS tickets to be approved in less than 14 days"* did not define the time range that the KPI should target. In general there are 2 ways to specify the time period. Firstly, it can be a fixed period in the Calendar such as the month of January or the third Quarter of the year 2014. The second option is to use a rolling period of time. So the SLA becomes "*at least 85 % of the TPS tickets to be approved in less than 14 days for any given consecutive 90 days"* where the start date can be arbitrarily chosen and the end date is simply 90 days after the start date. As the time range for the SLA was not defined by the management it left a lot of room for interpretation. While discussing the SLA with the people involved in the TPS management process it came out that there was a will to measure

the SLA for all the TPS tickets starting from the very first request made. However such methodology does not give a very good insight on the impact of the improvements made during a particular time period. In addition it does not show the true state of the service for the present time, preventing the service owners to make informed decisions regarding the investments that the service might need. The tables presented in the "Appendix C: Tables and charts for TPS tickets" shows the SLA for the time range of 1 day in addition to the monthly, quarterly and yearly numbers shown in the charts and tables included in this chapter. The "Figure 17 - Example of fixed vs rolling SLA" presents an illustrative example of a SLA measured over a fixed period of time versus using a rolling average from starting with the very first ticket. It is clear from the chart that the SLA has been met for the past quarter while the rolling average for the SLA is below 70 %. The author is in the opinion that a quarterly range should be used to measure the overall service level of the TPS process. Three months is a period long enough to minimize the impact of random events that might take place. On the other hand a quarter is short enough range to enable the service owners to understand and react to problems and changes in business requirements in a timely manner. In addition one quarter is the cadence for planning in the Skype Engineering Services Team.

Each ticket has 2 important dates being used for the SLA: the date of creation and the date of approval. When we are looking at a given date or time period, the question is which tickets should be presented? One option is to present the tickets created during a given period. In this case there is a possibility that the numbers will change in the future as more tickets from that period get approved A better option is to present the tickets approved during the given period.

The previous question raises a new problem. We do know how fast the closed tickets were approved, but we have no way of knowing how fast the tickets, that are still open, will be closed. This statement might seem obvious, but it presents a need to understand the context of the SLA numbers. An additional metric shown on "Figure 7 – The number of tickets approved and waiting for approval" can be used. The red line represents the number of tickets in the open state that have not been approved in more than 14 days. This metric is shown as a red line on the chart. It is not trivial to obtain historic data from Jira for this metric. Therefor the number is shown since December, when the process of storing such data started. Having this metric next to the number of approved tickets gives us an understanding how sustainable the current pace of approval is.

The SLA "*at least 85 % of the TPS tickets to be approved in less than 14 days"* suggests that the speed of getting a TPS request approved is important for the requestor. It might also be important to get a negative answer back in timely manner. In this case the requestor can seek other alternatives to resolving his problem. This aspect is not covered by the SLA.

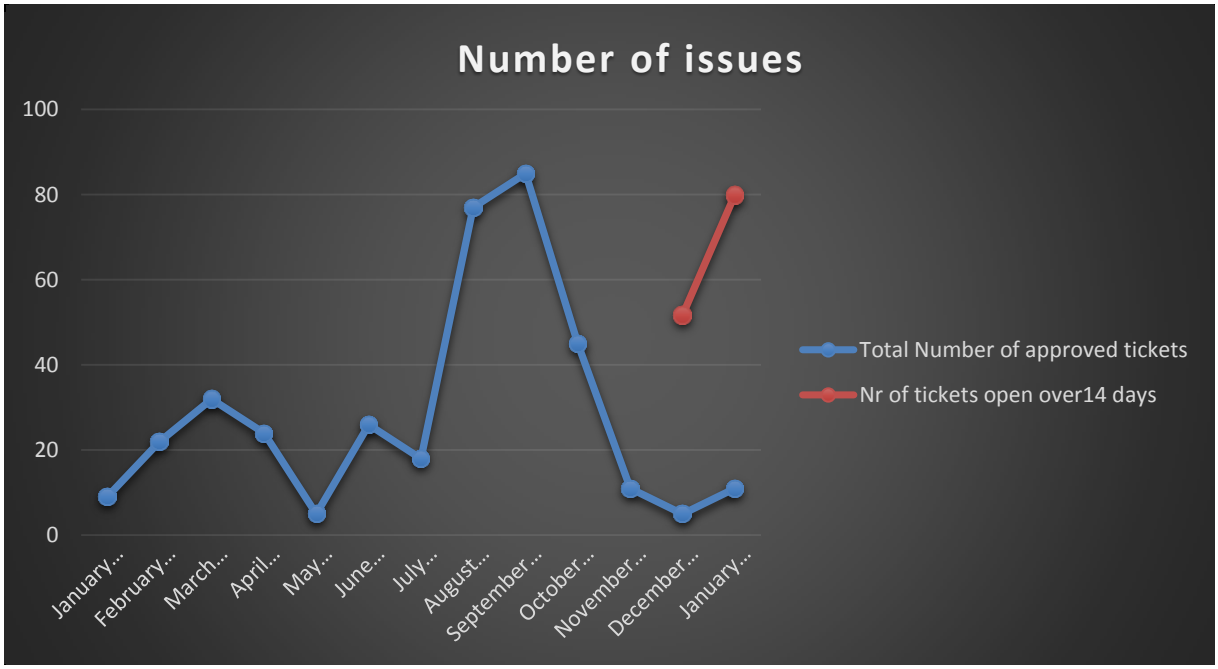

Figure 7 – The number of tickets approved and waiting for approval

5.3 Usage of the KPI

The KPI tables and graphs charts clearly and unambiguously show that the SLA has not been met. From the 4 quarters of the year 2014 the highest percentage of approved tickets in less than 14 days, was in the 1st quarter with the rate of 71.4 %. The lowest rate was 27.3% in the 2nd quarter. And the rates for the 3rd and 4th quarter were 71.1% and 59% respectively. None of the quarters meet the SLA criteria, nor is there a significant trend towards an improved approval rate. In addition, the amount of requests that have not yet been approved, but have been waiting in the queue for more than 14 days is rather big in comparison to the amount of tickets that is usually approved in a month. The median amount of tickets approved in a month was 22 while the number of tickets that have been waiting for approval for more than 14 days was 80 in the month of January.

The metrics were reviewed with the people responsible for TPS management process in the Skype Engineering Services Team. The fact that the team is so not close to meeting the SLA

and the trend is also negative was treated as a surprise. It was believed that the implementation of the automatic impounding tool iTPS would speed up the impounding and approval process. To an extent that was the case. Adding automation to the process, removed a lot of semi-manual work from the members of the Engineering Service team. Automation also helped to discover a lot of hidden dependencies turning the TPS management process.. The increased amount of impounded dependencies during July and August 2014 is also visible from table. Even though the process has been made more efficient and reliable there are still some bottlenecks. Most of the blockers come from dependencies to other teams, such as the legal department. Therefor no immediate action can be taken to influence the SLA metric. The main conclusions from reviewing the usage of the KPI are the following.

- Having a well-defined and presented metric helps to surface the problems and supports discussions that can lead to further improvements.
- Meeting the SLA is important. The ROI for fully meeting the SLA is not big enough compared to other improvements and functionality that can be delivered to the company related to the TPS process.

The proposed SLA is a useful metric to follow. It also seems like a natural candidate for a KPI. Unfortunately this is not enough to treat the SLA as a KPI.

6. Key Performance Indicators for Build Systems

A build process turns source code into a working software component. The build process can be divided into 6 bigger steps. Firstly, the right environments for the build process need to be prepared. Secondly, the correct source code needs to be checked out. The third step is fetching the dependencies, other components and specific tools needed for the build process. Then the code is compiled. Compilation is followed by quick unit and verification tests. The last step is the publishing of the components and the documentation.

Skype uses a central build model. Most of the builds are run on centrally managed build infrastructure. The benefits of this type of setup include the possibility to retain the integrity of the produced artefacts, reduction of time and effort it takes for the developers to set up their build environments and the possibility to easily combine together end products of many different workflows.

On a conceptual level the build infrastructure consists of a central build orchestrator and over 500 build machines which carry out the actual builds. The source code management systems and dependency management systems are very heavily integrated into the build services. In addition, different build processes need to interact with various additional services such as Jira, testing and code analyses tools and others.

Skype develops components that run on various platforms. The environments for compiling different products have to be also rather assorted. Skype has been acquired by Microsoft and the development organizations of the two companies need to be aligned. Therefor the Build Infrastructure developed and maintained by the Skype Engineering Services Team needs to be aligned with the engineering needs of rest of Microsoft.

Reliability and the speed of the Build feedback loop were established as the most important attributes of the Build Systems. This conclusion was reached after several discussions with the key stakeholders. The people involved were managers of the Skype Engineering Services, the developers of the Build Systems and the end-users of the Build System.

6.1 Methodology for Build KPI

Getting a clear overview of the characteristics related to the build feedback loop is not a straightforward process. There are more than 20 000 build jobs running inside the Build Infrastructure every day. Two main aspects make it difficult to measure the speed of the build correctly. Firstly there are lots of different types of build jobs. A common Continuous Integration Build aims to give the developer fast feedback. He wants to know whether the change he made was good or it broke the build. In addition to traditional CI Builds developers have set up jobs to carry out different tasks. These tasks include running builds that do various tests on top of building and verifying the change. These tests might be thorough and getting fast feedback might not be a priority at all. The second aspect that makes it difficult to correctly measure the change of the speed of the build process is the alternations made to the Build configurations. There might be a well-founded need to add new steps to the Build or to add content that make the current Builds longer. A developer might want to check out more code or different dependencies, run more verification tests on top of the build or do other tasks that add to the Build time. The two reasons mentioned in this paragraph make it difficult to measure the impact that the changes made to Build Infrastructure have on the speed of the feedback loop.

Changes that are not controlled by the Engineering Services Team make it difficult to collect meaningful statistics about reliability. A developer is interested in knowing if a change made by him caused the Build to break. A Build failure might also happen due to infrastructure problems. In that case the information about a build failure is noise for the developer wasting his time.

In order to get objective statistics about the performance of the Build infrastructure the author made a proposal to use Benchmark Builds as a proxy to get information about the Build Infrastructure. The Build Operations team set up 5 Build configurations that mimic real live Builds. The Build configurations are under the control of Engineering Services team and the changes made to the configurations are kept to minimal. Both the performance and the reliability of the Build will be only influenced by the Build Infrastructure. There are 5 configurations producing 3 builds per hour each. Altogether 360 builds get produced every day.

There are altogether more than 500 machines where the Build jobs are run. These machines are divided into resources. Different types of builds run on different types of resources. This causes variation in performance and reliability for different Benchmark builds. How heavily the resource is used in real life varies greatly. "Table 8 - The top resources ordered by their total usage time" presents the top of the most heavily used resources. The table is ordered by the 3rd column showing the total time the specific resource was in use during January. The rows for the 5 resources used to run the 5 Benchmark builds are marked in bold. In total 22.3

Resource Count	Build Count	Total time	% of total time
flow-controllers	97085	1176399	24.9
w81-universal	**50873**	**580861**	**12.3**
tll-uxsts-qb1:8811	9447	244303	**5.2**
skylibwin	**19771**	**239370**	**5.1**
c2c-atm-cntrl	762	165401	3.5
w8-rtm-vs2012-update1	14385	151270	3.2
linux-skypekit	26480	126389	2.7
mac-mountainlion-5-xcode-5.0.1	31862	111587	2.4
linux-wheezy64-universal	**28350**	**107849**	**2.3**
azure_testers_2.3	14571	104667	2.2
azure_testers_2.2	18592	94545	2.0
azure_deployers_2.3	**10368**	**92669**	**2.0**
w8-universal	7173	82517	1.7
calling-skytest-win32	2392	68373	1.4
w81-experimental-staging	5623	65340	1.4
azure_deployers_2.2	3791	63755	1.4
calling-skytest-win32-ng	1454	56589	1.2
azure_testers_2.4	12444	49175	1.0
w2k8_universal	5497	41535	0.9
qik-backend-z3	3680	41479	0.9
linux-webapp	4664	36707	0.8
mac	6844	33459	0.7
linux-squeeze32-universal	6471	32961	0.7
skypechat-android-ubuntu64	635	30486	0.6
lync-build	1482	29949	0.6
w81-experimental	2554	29131	0.6
mc-win732-01:8811	1992	27170	0.6
mac-mavericks-2-xcode-5.1.1	**4950**	**26106**	**0.6**

Table 7 - The top resources ordered by their total usage time

% of the build time was spent on these resources. The resources ranking 2nd and 4th are covered by Benchmark Builds. The most heavily used resource *flow-controllers* is only used to run jobs that provide help in orchestrating more complex Builds. The time spent on these resources is considered irrelevant from the performance and reliability perspective. In

addition there are many resources which are used for very specific configurations. There is a lot of room for improving the representation of the user experience. In the other hand the current set-up is satisfying to start measuring the performance and reliability of the Build Infrastructure.

6.2 Build reliability KPI

The aim of a good Build System is to provide fast feedback to users. Developers want to know if their commit was successful or it broke the build. The feedback that the developer gets has to be reliable. This means that a build failure must indicate a problem in the change made by the developer. On some occasions also problems in the Build Infrastructure can cause build failures. There might be problems in the network, random malfunctions of the Build machines or other problems. False failure reports are a huge distraction for the developers using the Build Systems. The source code and dependencies used for the Benchmark builds are fixed to known revisions. Therefor all failures can be attributed to infrastructure problems. Keeping infrastructure relate failures to a minimum helps to improve the quality of the feedback.

The reliability of the Build Infrastructure is measured by using the Benchmark builds. The calculations on the charts depicted on “Figure 8 - Reliability total per day” and “Figure 9 - Reliability breakdown per month” are simple. The lines show the percentage of builds that finished in the “Successful” state.. The first chart shows the combined reliability for all the 5 Benchmark Builds broken down by day. The second chart has 5 lines, each of the lines representing one of the Benchmark Builds. The values shown on the chart are monthly averages. Two more views for Build Reliabilty are presented in the Chapter 10.5 Build Reliability Charts.

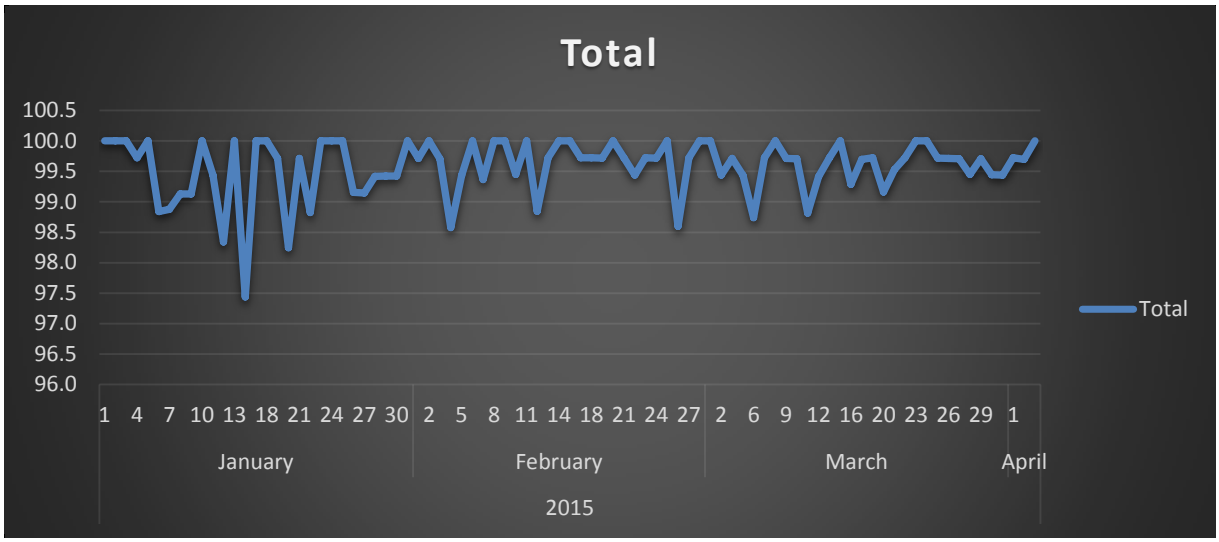

Figure 8 - Reliability total per day

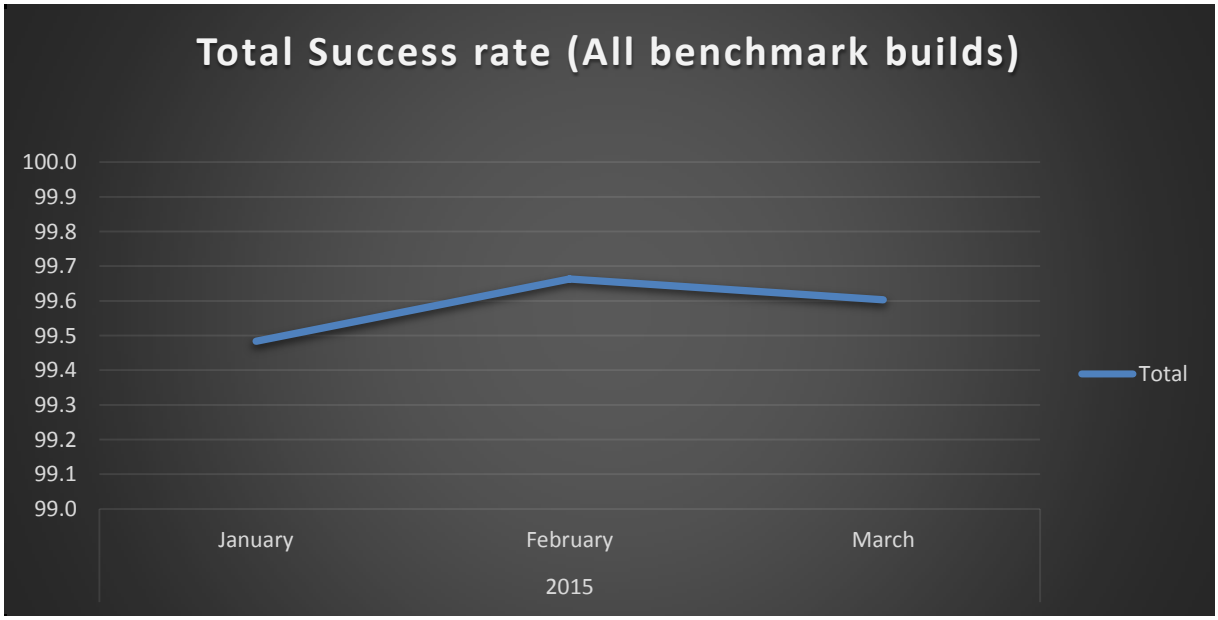

Figure 9 - Reliability breakdown per month

While putting together the proposals for KPIs, measuring the reliability of the Build Infrstrucure was accepted by all the stakeholders.

6.3 Build speed KPI

The owners of the Build Infrastructure established 15 minutes to be a reasonable time that a developer has to wait to get feedback on his commit. The charts on “Figure 10 - Speed Total per day” and “Figure 11 - Speed breakdown by month” show the percentage of builds finishing under 15 minutes. Only the successful builds are counted on this chart.

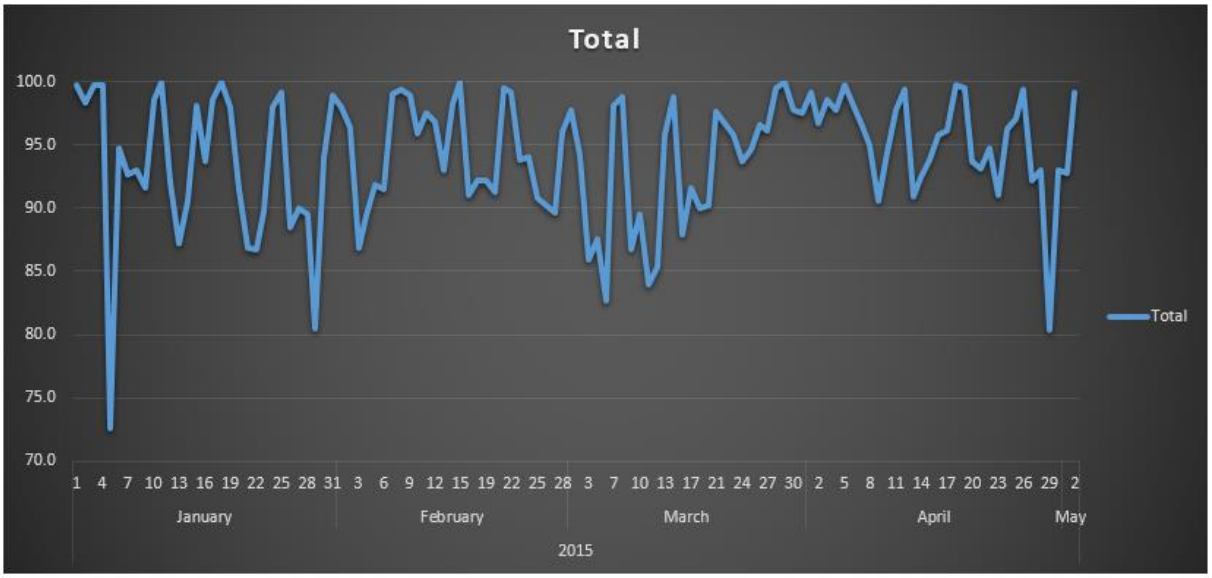

Figure 10 - Speed Total per day

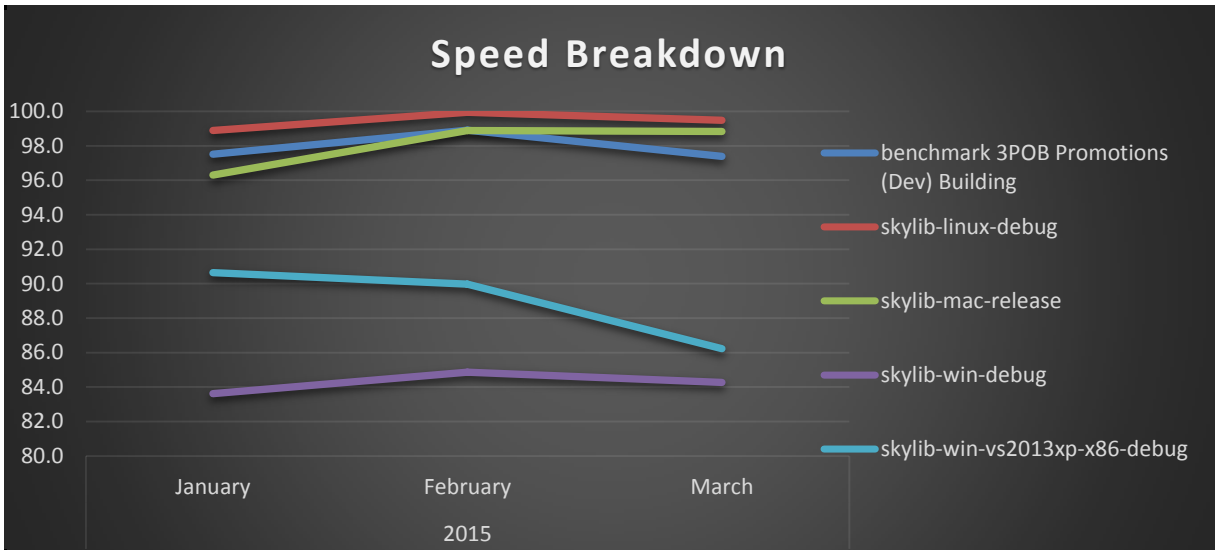

Figure 11 - Speed breakdown by month

The percentage of Builds finishing under 15 minutes is considered to be a Key Performance Indicator for the build infrastructure. Similar information can be depicted in a bit different fashion. “Figure 12 - CI experience on 09.01.2015” shows the duration of each individual Benchmark Build. The time it took for the build to wait in the queue and the actual build time are stacked. The total amount of time shows how long the developer had to wait from the time the build request was made until he received the feedback about his Build. Only the successful Builds are counted. Depicting data in the fashion that has been done on the Figure 12 shows that the time a build has to wait in the queue is a major bottleneck. Figure 12 also points out the hour of the day when it takes longest to get feedback from the system.

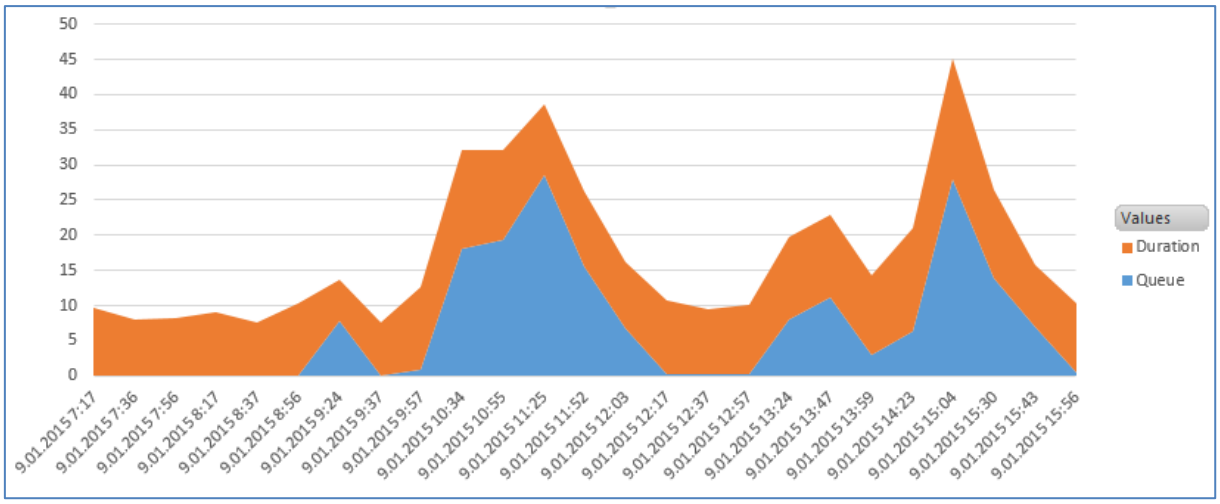

Figure 12 - CI experience on 09.01.2015

6.4 Learnings

The aim of the Skype Engineering Services team is to improve the service and provide faster feedback for the users. The KPIs show if a change made to the system had the desired impact. On 21st of March and the 7th of April significant changes were made to the Build Infrastructure aiming to make the Builds faster. "Figure 10 - Speed Total per day" shows a distinct change after both of the changes. The first change was a success and the KPI for the Build speed showed a significant improvement. The second change, made on 7th of April had the opposite effect[13] and actually caused an increase to the Build feedback time. Build KPIs gave a very precise indication of the changes made to the Infrastructure. In addition the feedback received from the users confirmed the information shown by the KPIs.

One purpose of a KPI is to provide a common understanding for the whole team about the important characteristics of the system. KPIs help to measure and communicate information about the most significant aspects. One of the main difficulties while establishing the Build KPIs was to get different stakeholders to agree on common KPIs. While people work towards their targets a deeper understanding of a common goal is rather weak. In the beginning it was even difficult for people to interpret one specific metric in the same way. In addition, while discussing the KPIs with stakeholders, on many occasions the conversations would drift into

[13] A number of virtual machines were added to increase the number of build machine. The aim was to reduce the time a Build had to wait in a queue before the work was started. The addition of the VMs had a signicant negative impact on the performance of the machines.

details. Instead of discussing the importance of a given metric, the causes for specific values were discussed. Therefor it took many iterations to implement metrics that are understood and accepted by most of the team.

The KPIs were worked out in groups with a limited amount of active team members. In a presetnation to a larger audience an interesting question was brought up. A system administrator wanted to know the events that correlate to the drops in reliabilty and the speed of the feedback loop. He believed this to be important for making future improvement. The Head of Erngineering Services pointed out that the numbers indicating problems are not a pathology. The KPI values are the results of the total amount of work done by the team. The fact that this question was brought up demonstrates the gap between personal targets and the the impact on companys goals.

7. Technical infrastructure

The main focus of the thesis is on the metrics with the right characteristics. However, there has to be a technical infrastructure that supports the gathering, storing and presenting of the metrics. *"The project team should promote the use of existing in-house applications for the collection and reporting of the performance measures for at least the first 12 months. Much can be done with standard applications such as Excel, PowerPoint, SharePoint Team Services, and Access."* (D. Parmenter, 2007, p.32). In the beginning, the technical setup was kept very simple and flexible, so that changes could be implemented as fast as possible. The correct metrics were worked out in iterations. Data was gathered and presented to the end users. Based on the feedback, changes were introduced. This cycle was repeated several times. In the other hand the usage of metrics should be as low-cost as possible. The infrastructure for storing the data should be reliable and require minimal maintenance. It should be scalable and the data should be easily accessible. Therefor the technical infrastructure went through many iterations as well. Starting off with flexibility in mind and moving in the direction of providing a scalable platform

In the first iteration of the metrics project, different systems were accessed by scripts running on one laptop. The data was gathered in CSV files. Based on the files excel charts were created. The charts were presented in meetings and distribute by e-mail. Such operating model is highly manual and does not scale at all. However, this approach allowed to get very fast feedback. It was possible to iterate very rapidly to find out useful metrics and candidates for KPIs.

The second iteration of the technical infrastructure focused on the automatic presentation of the metrics. One of the drivers was still flexibility. There was the need to get fast feedback and make changes. However, the usability and scalability were now taken into account. A simple SQL database was set up. Scripts that pulled data from different services were written. The database and the scripts were set up on a central server. The data was pulled into Excel using Power Query[14]. Data models and the presentation of the metrics were created using

[14] https://support.office.com/en-in/article/Introduction-to-Microsoft-Power-Query-for-Excel-6e92e2f4-2079-4e1f-bad5-89f6269cd605

Power Pivot[15]. Charts were presented to the end user using a SharePoint[16] site. This approach allowed the end user to consume the KPIs in a very straightforward way. The setup was fully automatic and the data got refreshed on a daily bases. The metrics could be used and propagated in the Skype Engineering Services Team and also to their customers. The infrastructure had been broken into several components. The components were mostly under the control of the producers of the metrics. This allowed to make quick changes to data models and left the possibility to easily switch components in the stack.

The infrastructure mentioned above had two flaws. Firstly it required a proprietary SQL database. Someone had to maintain the database. He had to make sure that there is enough storage; the database is backed up etc. In addition, every time a new event was added, a new table had to be created. Every time an event was changed, the existing tables had to be altered.

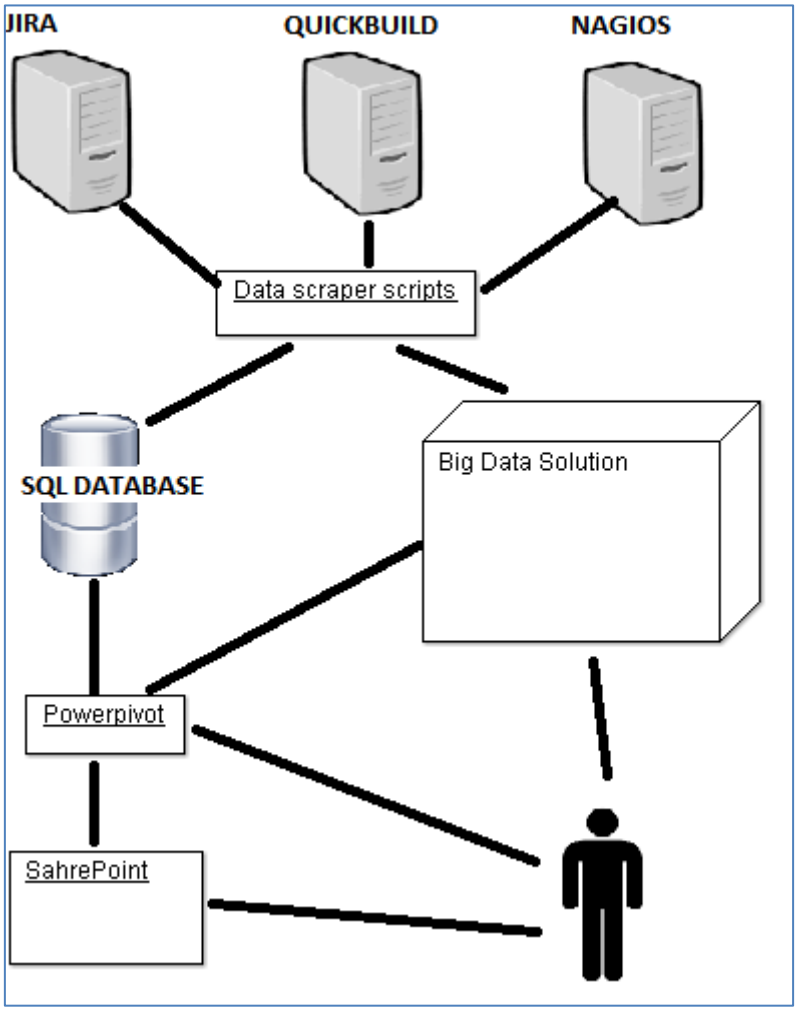

Figure 13 – Metrics Infrastructure

[15] https://support.office.com/en-nz/article/Power-Pivot-Add-in-a9c2c6e2-cc49-4976-a7d7-40896795d045
[16] https://products.office.com/en-us/sharepoint/collaboration

Secondly, if someone wanted to access the raw data, he needed to have tools, such as PowerPivot, installed on his laptop. Because of the reasons mentioned above, alternative solutions were researched. An Internal Microsoft Big Data Solution[17] was tried out. It is a service that accepts and stores events. It allows querying of the events and the creation of dashboards. It does have some limitation. There is a learning curve for the people who start sending the events. The functionality of the dashboards is limited. The whole solution is optimized towards bigger amount of data than the Skype Engineering Services Team has. The interfaces for obtaining aggregated data are more advanced than the ones that provide raw data.

The current infrastructure is depicted on "Figure 13 – Metrics Infrastructure". It uses the SQL database and the Big Data Solution in parallel. The next iteration will try to make more use of the Big Data Solution as it is more scalable. There is a plan to deprecate the SQL database.

[17] Because the solution is propietort, it will not be covered in more detail

8. Conclusion

The thesis studies how data and metrics can be used to drive improvements in the Skype Engineering Services Team. Three main objectives were set in the beginning of the thesis. The first goal was to understand the role of data and metrics in the service management and software development process. This objective was fulfilled in three parts. Firstly, existing studies on the usage of metrics in different organisations, including large software development companies, were examined. Based on the studies, it was decided that the reasonable amount of Key Performance Indicators to be implemented was 3. Those KPIs should be developed during 1-2 years in an iterative process. Following the recommendation the most attention was put on agreeing on the metrics with the correct characteristics. Secondly, the operational model of the Skype Engineering Services Team was analysed. It came out that several efforts had already been made to use data and metrics to support everyday operations and planning. However, the efforts had been disconnected from each other. It was recognized that more systematic approach to metrics usage could help in improving the service. The rest of the thesis describes and analyses the improvements implemented based on the findings in Chapter 2 and Chapter 3.

The second bigger objective of the thesis was to find the best means to gather, surface and use data to manage and support an Engineering System. The author followed the aspects described in Chapter 2 to keep the initial technical infrastructure as simple as possible by using known tools. This approach allowed keeping the focus on the meaning and usage of the data. Nevertheless having the right tooling in place for gathering and presenting the metrics is crucial. Chapter 7 gives an overview of the technical infrastructure used to gather and present the metrics developed in the scope of the thesis. The possibilities for a more scalable architecture are also described.

The third and the most important objective of the thesis, was to establish Key Performance Indicators and other metrics for the Skype Engineering Services. The author worked on three different fields within the Skype Engineering Services Team. The results turned out to be very diverse for all of the three areas.

The first task was to understand how the new incident management process affects the reliability of the services. It came out that the data gathered regarding the downtime of the

services was not valid. Therefor the data was unusable for creating an understanding of the quality of the services and the objective could not be met. However, based on the learnings the author proposed a new process to manage service alerts. The new process was implemented for some of the services and these insights proved to be helpful in the management of the systems. In addition the new process enabled the team to acquire reliable data regarding the service health.

Secondly, a KPI was implemented for The Third Party Software Management Process. The owners of the service had a clear understanding of what should be measured. Figuring out the implementation details and doing a technical setup was needed. After the implemented metrics were reviewed with the stakeholders, it came out that the metric was useful but could not be considered a Key Performance Indicator. The stakeholders agreed that it was important to follow the trend shown by the metric. Also they wanted to influence the metric as much as possible. The downside was that there were significant aspects of the metric that were not under the direct control of the Engineering Services Team. Therefor the metric was taken into use, but not as a KPI.

The third sets of metrics that the author worked on were the KPIs for the Build Services. After many iterations and discussions with different stakeholders two KPIs were established. Measuring the percentage of the Builds that fail because of the problems in the Build Infrastructure was recognized as the KPI for reliability. The second KPI was about measuring the feedback loop speed of the Build process. The metric gives information how fast developers get feedback on their commits. Benchmark Builds were implemented and used as proxies to measure both the speed and the reliability. Today both of the Build KPIs are used to understand the effect that different improvements, changes and other factors have on the experience of the Build Infrastructure usage. The KPIs are used as bases of conversation inside the team. The KPIs are also communicated to customers in order to illustrate the impact that the work from the Skype Engineering Services Team has.

The following are the main tangible outcomes produced by the author:

- A Key Performance Indicator to describe the speed of the feedback loop.
- A Key Performance Indicator to describe the reliability of the Build Infrastructure.

- A process and tooling for automatically capturing and analysing data regarding incidents related to Skype Engineering Services.
- Useful metrics for describing the TPS management process.
- Technical infrastructure for gathering and presenting the data and metrics.

The main objectives of the thesis were met. Of course not everything went as planned. Some of the results differed from what was expected in the beginning. Some targets were not achieved. Nevertheless the overall result was useful and satisfactory for the Skype Engineering Services Team.

What can be done next?

The process of working out meaningful metrics takes years and keeping data up to date is an ongoing work. This thesis focused on the first implementation of KPIs for the Skype Engineering Services Team. The following list provides some suggestions for future improvements.

- Work out new metrics and KPIs for the Skype Engineering Services Team.
- Improve the current KPIs for Build Infrastructure.
- Propagate the usage of metrics within Skype Engineering Services Team and the rest of the Company.
- Improve the reliability and scalability of the technical infrastructure used to capture and present the metrics.
- Find other ways to make use of the data available in the Skype Engineering Services infrastructure.

Kokkuvõte

Magistritöö uurib andmete ja meetrikate kasutamist paranduste tegemiste juhtimisel Skype'i tarkvaraarenduse tugiteenuste meeskonnas. Magistritöö alguses püsitati kolm põhilist eesmärki. Esimeseks eesmärgiks oli andmete ja meetrikate rollist arusaamine tarkvarateenuste opereerimis-ja arendusprotsessis. See eesmärk täideti kolmes osas. Esiteks uuriti meetrikate kasutamise kohta läbiviidud töid erinevates organisatsioonides, sealhulgas suurtes tarkvaraarendus ettevõtetes. Vastavalt uuringutes leitule otsustati, et 3 tulemuslikkuse indikaatorit on mõistlik hulk, mille väljatöötamisele 1-2 aasta jooksul iteratiivses protsessis keskenduda. Lähtudes soovitustest suunati enamus tähelepanu õigete karakteristikutega meetrikate leidmisele. Teiseks uuriti Skype'i tarkvaraarenduse tugiteenuste meeskonna toimimismudelit. Ilmnes et eelnevalt oli tehtud mitmeid jõupingutusi, et kasutada andmeid ja meetrikaid igapäevaste toimingute ja planeerimise läbiviimiseks. Paraku olid need jõupingutused olnud üksteisest eraldatud. Tuvastati, et süsteemsem lähenemine meetrikate kasutamisele oleks abiks teenuste parandamisel. Ülejäänud töö kirjeldab ja analüüsib Peatüki 2 ja Peatüki 3 põhjal tehtud ettepanekute juurutamist. Magistritöö teine suurem eesmärk oli parimate meetodite leidmine tarkvaraarenduse tugiteenuste meeskonda toetavate andmete kogumiseks, esiletoomiseks ja kasutamiseks. Autor lähtus Peatükis 2 välja toodud aspektidest kasutada esimestes iteratsioonides juba tuttavaid vahendeid ja hoida esialgne taristu tehniliselt võimalikult lihtne. Õigete vahendite kasutamine andmete kogumiseks ja presenteerimiseks on siiski ülioluline. Peatükk 7 annab ülevaate tehnilisest taristust, mida kasutati magistritöö käigus väljatöötatud andmete kogumiseks ja presenteerimiseks. Samuti kirjeldatakse võimalusi mastaabiga kohaneva arhitektuuri loomiseks.

Magisitiröö kolmas ja kõige tähtsam eesmärk oli Skype'i tarkvaraarenduse tugiteenuste meeskonna jaoks tulemuslikkuse indikaatorite ja teiste meetrikate väljatöötamine. Autor tegeles Skype'i tarkvaraarenduse tugiteenuste meeskonnas kolme eri valdkonnaga. Tulemused olid kõigis kolme teenus puhul vägagi erinevad.

Esimeseks uuritavaks valdkonnaks oli uue intsidentide haldusprotsessi mõju teenuste töökindlusele. Ilmnes, et andmed, mida oli kogutud süsteemide seisakute mõõtmiseks, ei olnud paikapidavad. Seega ei olnud võimalik neid andmeid kasutada, et kirjeldada teenuste kvaliteeti ja püstitatud eesmärk jäi täitama. Lähtudes omandatud teadmistest pakkus autor

välja uue protsessi, kuidas hallata teenuste häireteateid. Uus protsess võeti mõnede teenuste puhul kasutusse. Uuest protsessist saadud kaemused osutusid kasulikeks süsteemide haldamisel. Lisaks pakkus uus protsess võimaluse saada täpset infot teenuste seisukorra kohta.

Teiseks juurutati tulemusindikaator kolmanda osapoole tarkvara jaoks. Teenuse omanikel oli selge arusaam sellest, mida tuli mõõta. Vaja oli välja töötada rakenduslikud detailid. Pärast väljatöötatud meetrika ülevaatamist koos huvitatud osapooltega, ilmnes, et meetrika oli kasulik, aga seda ei saanud pidada peamiseks tulemuslikkuse indikaatoriks. Nõustuti, et on tähtis antud meetrikat jälgida. Samuti oli soov meetrikat võimalikult palju mõjutada. Puuduseks osutus asjaolu, et antud meetrika juures oli mitmeid aspekte, mis ei olnud Skype'i tarkvaraarenduse tugiteenuste meeskonna otsese kontrolli all. Seetõttu võeti meetrika küll kasutusse, aga mitte tulemuslikkuse indikaatorina.

Teine grupp meetrikaid millega autor töötas, olid tulemuslikkuse indikaatorid *Build* teenuste jaoks. Pärast mitmeid iteratsioone ja arutelusid huvitatud osapooltega töötati välja kaks tulemuslikkuse indikaatorit. Veaga lõppenud tööde protsenti hakati kasutama töökindluse tulemuslikkuse indikaatorina. Teine tulemuslikkuse indikaator oli *Build* protsessi tagasiside tsükli kiiruse mõõtmise kohta. See meetrika annab näitab, kui kiiresti saavad arendajad tagasisidet tehtud koodimuudatuste kohta. Näidis *Build'e* kasutati nii kiiruse kui töökindluse mõõtmiseks. Antud hetkel kasutatakse mõlemat indikaatorit, et saada aru millist mõju avaldavad erinevad parandused, muudatused ja teised faktorid *Build* teenuste taristu kasutuskogemusele. Tulemuslikkuse indikaatoreid kasutatakse vestluse alusena meeskonna sees. Samuti kommunikeeritakse tulemuslikkuse indikaatoreid klientidele illustreerimaks, millist mõju Skype'i tarkvaraarenduse tugiteenuste meeskonna töö omab.

Järgnevalt on väljatoodud autori poolsed tulemused:

- Tulemusindikaator kirjeldamaks tagasiside tsükli kiirust.
- Tulemusindikaator kirjeldamaks tagasiside tsükli töökindlust.
- Protsess ja taristu Skype'i tarkvaraarenduse tugiteenuste meeskonna teenuseid puudutavate intsidentide kohta andmete automaatseks salvestamiseks ja analüüsiks.
- Kasulik meetrika kirjeldamaks kolmanda osapoole tarkvara haldusprotsessi.
- Tehniline taristu andmete ja meetrikate kogumiseks ja presenteerimiseks.

Magistritöö peamised eesmärgid saavutati. Muidugi ei läinud kõik täpselt plaani kohaselt. Mõned tulemused erinesid algsetest plaanidest. Mõned sihid jäid saavutamata. Siiski olid üleüldised tulemused kasulikud ja rahuldasid Skype'i tarkvaraarenduse tugiteenuste meeskonna vajadusi.

Järgmised sammud

Oluliste meetrikate väljatöötamine võtab aastaid ja andmete ajakohasena hoidmine on pidev töö. Magistritöö keskendus Skype'i tarkvaraarenduse tugiteenuste meeskonna jaoks esimeste tulemusindikaatorite väljatöötamisele. Järgnevalt on esitatud nimekiri võimalikust edasiarendustest.

- Töötada välja uusi meetrikaid ja tulemusindikaatoreid Skype'i tarkvaraarenduse tugiteenuste meeskonna jaoks.
- Täiustada olemasolevaid meetrikaid *Build'i* taristu jaoks.
- Propageerida meetrikate kasutamist Skype'i tarkvaraarenduse tugiteenuste meeskonnas ja kogu ettevõttes.
- Täiustada kasutuses oleva taristu töökindlust ja mastaabiga kohanemise võimet.
- Leida uusi viise kuidas Skype'i tarkvaraarenduse tugiteenuste meeskonna poolt hallatavas süsteemides eksisteerivaid andmeid ära kasutada.

Bibliography

[1] "Logstash," [Online]. Available: http://logsatsh.org. [Accessed 07 10 2014].

[2] "ItilFoundations," [Online]. Available: http://www.itilfoundations.com/processes/incident-management/definition/. [Accessed 07 October 2014].

[3] F. Provost and T. Fawsett, Data Sciense for business, O'Reilly, 2013.

[4] D. Parmenter, Key Performance Indicators Developing, Implementing, and Using Winning KPIs, John Wiley & Sons, Inc., 2007.

[5] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Application_performance_management. [Accessed 25 04 2015].

[6] K. Konsta and E. Plomaritou, "Key Performance Indicators (KPIs) and Shipping Companies Performance Evaluation: The Case of Greek Tanker Shipping Companies," 16 May 2012. [Online]. Available: http://www.ccsenet.org/journal/index.php/ijbm/article/viewFile/13633/11404. [Accessed 05 march 2015].

[7] F. F. Jing, G. C. Avery ja H. Bergsteiner, Enhancing performance in small professional firms through vision communication and sharing, New York: Springer Science+Business Media, 2013.

[8] J. J. Trienekens, R. J. Kusters, J. I. M. M. van Genuchten and H. Aerts, "Targets, drivers and metrics in software process improvement: Results of a survey in a multinational organization," *Software Quality Journal*, vol. 15, no. 2, pp. 135-153, 2007.

[9] A. Pourshahid, . I. Johari, G. Richards, D. Amyot and O. S. Akhigbe, "A goal-oriented, business intelligence-supported decision-making methodology," *Decision Analytics*, vol. 1:9, 2014.

Appendixes

Appendix A: Extraction, Transformation, Loading

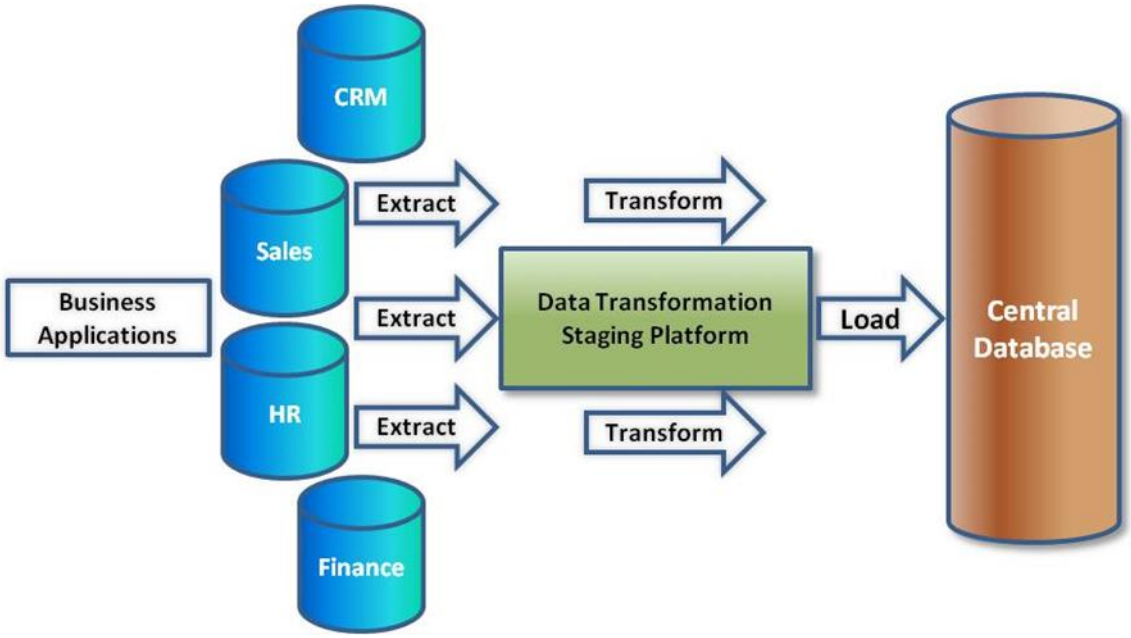

Figure 14 - ETL model[18]

[18] http://www.imc.com/services/enterprise-data-warehousing/etl-process-management

Appendix B: Full table for improvement drivers

Table 6 Results for improvement drivers

Improvement driver	Mean				ANOVA	*t*-test		
	All	CMM1	CMM2	CMM3		1 2	2 3	1 3
Commitment of engineering management	4.0	3.7	4.0	4.7	*0.08*	0.47	*0.08*	**0.04**
Commitment of development staff	3.8	3.6	3.8	4.0	0.54	0.55	0.63	0.31
Sense of urgency	3.5	3.6	3.2	3.5	0.63	0.38	0.58	0.75
Availability of engineers time for SPI	3.4	3.6	3.4	3.1	0.63	0.66	0.62	0.36
Commitment of business management	3.3	3.3	3.0	3.5	0.69	0.55	0.39	0.75
Availability of qualified SPI resources	3.3	3.2	3.0	3.9	0.24	0.69	*0.06*	0.17
Clear/quantifiable improvement targets	3.2	2.8	3.2	3.7	*0.08*	0.33	0.15	**0.04**
Use of accepted framework such as CMM	3.1	2.5	3.6	3.9	**0.00**	**0.01**	0.54	**0.01**
Clear relation between SPI/business goals	3.1	2.7	3.8	3.3	**0.02**	**0.01**	0.22	0.17
Confidence in SPI results	3.0	2.9	2.8	3.6	0.12	0.85	**0.02**	*0.09*
Visibility of intermediate results	2.9	2.8	2.7	3.4	0.26	0.88	*0.07*	0.16
Sufficient investment in SPI training	2.8	2.6	2.5	3.7	**0.03**	0.84	**0.01**	**0.02**
Proper tooling to support the processes	2.8	2.6	3.1	2.7	0.57	0.32	0.41	0.86
Cooperation other engineering disciplines	2.6	2.3	3.0	2.6	0.43	0.19	0.54	0.58
Integration SPI in general improvement actions	2.5	1.9	3.2	3.0	**0.02**	**0.01**	0.79	**0.02**

Figure 15 - Results for improvement drives from Philips

Appendix C: Tables and charts for TPS tickets

Tickets for Internal TPS				
	% approved <14 Days	Status	Total Approved	Open over 14 d
2014	51.1		5043	83
Q1 2014	86.3		656	
Q2 2014	73.4		612	
Q3 2014	31.0		2835	
Q4 2014	73.0		940	83
October	83.9		461	
November	55.6		248	
December	69.7		231	83
2015	81.9		166	140
Q1 2015	81.9		166	140
January	81.9		166	140
Grand Total	52.1		5209	122

Table 8 - The % of internal TPS tickets approved in 14 days or less

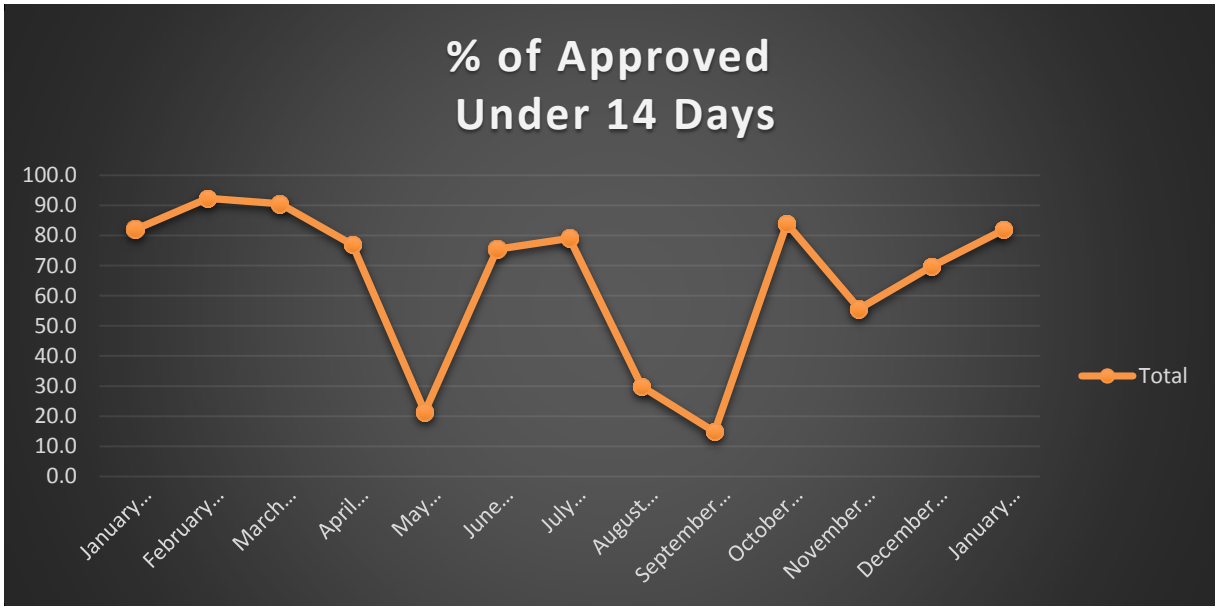

Figure 16 - The % of internal TPS tickets approved in less than 14 days

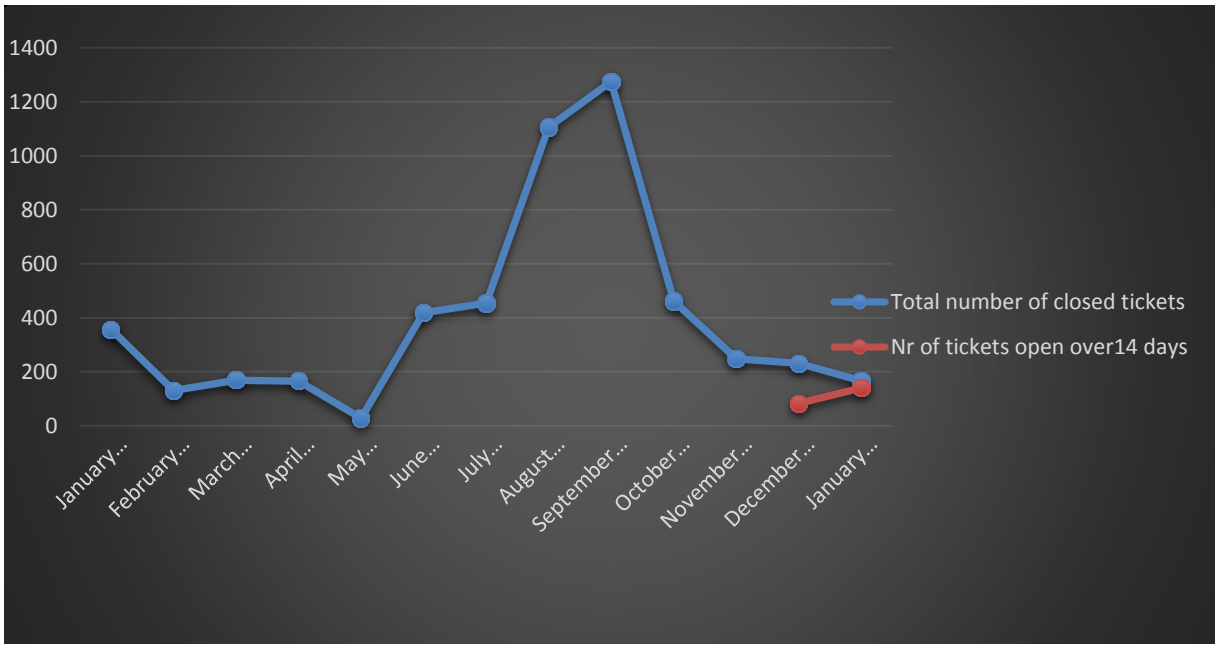

Figure 17 - The Number of internal TPS tickets approved and waiting for approval

August	**84.4**		**77**
1.08.2014	100.0		1
2.08.2014	100.0		1
4.08.2014	88.7		71
5.08.2014	0.0		1
13.08.2014	0.0		1
19.08.2014	0.0		1
20.08.2014	0.0		1
September	**58.8**		**85**
1.09.2014	10.0		10
4.09.2014	0.0		2
5.09.2014	0.0		1
9.09.2014	0.0		4
11.09.2014	40.0		15
15.09.2014	0.0		1
16.09.2014	100.0		1
17.09.2014	0.0		1
23.09.2014	85.4		48
29.09.2014	0.0		1
30.09.2014	100.0		1

Table 9 - August and September for distributed TPS tickets broken down by day

Nr of uarter	1	2	3	4	5	6	7	8
Rolling average	15	20	23	26	32	41	48	53
Value in the quarter	15	25	30	35	55	85	89	93

Table 10 - Data for the example of using a fixed period vs using a rolling average SLA

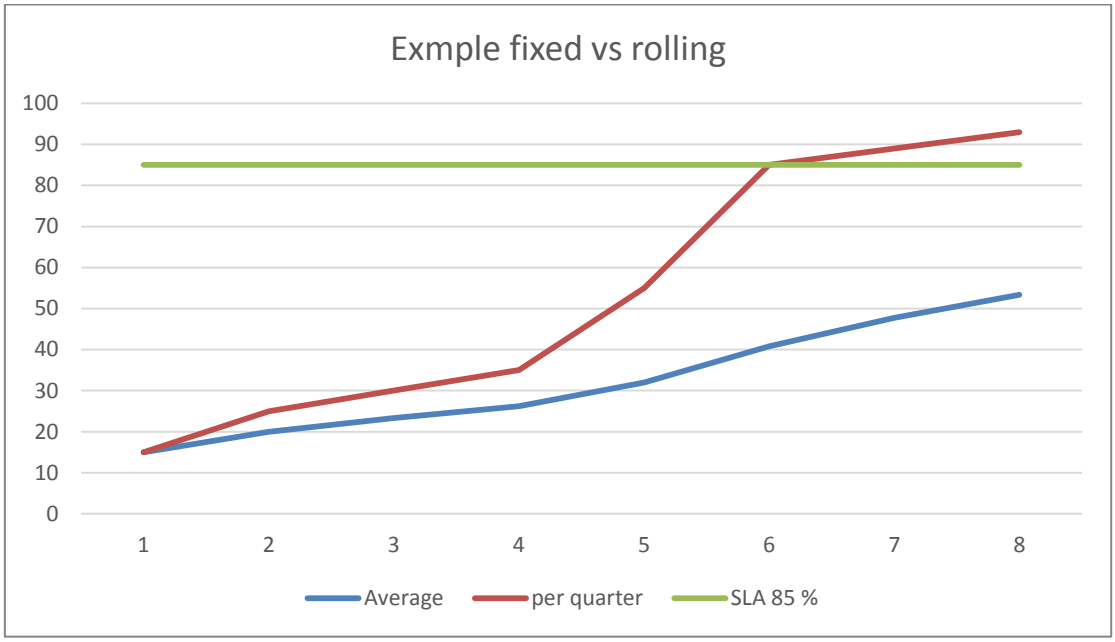

Figure 18 - Example of fixed vs rolling SLA

Appendix D: Additional tables and charts for alerts and incident management process

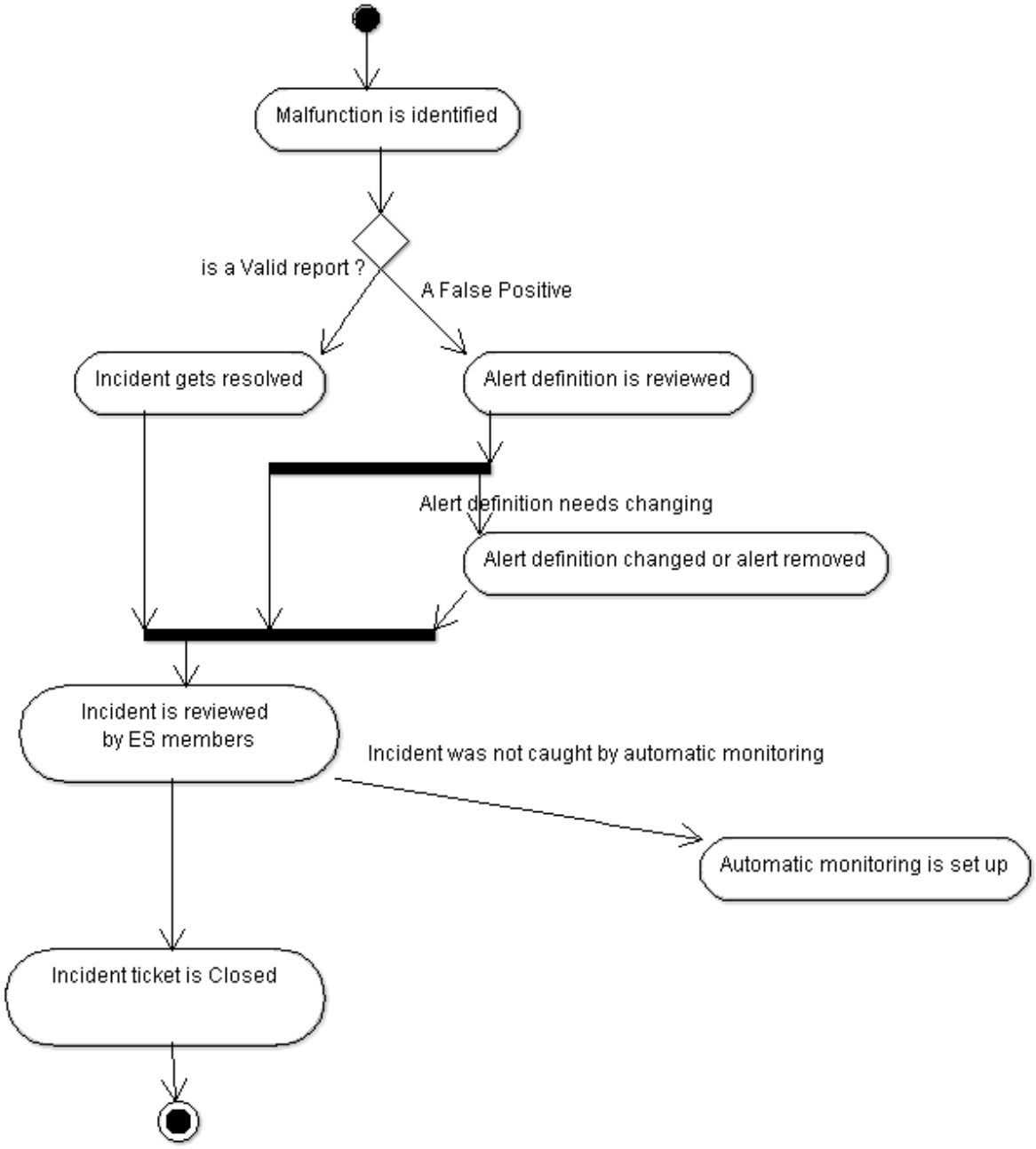

Figure 19 - Alert and Incident review process

Appendix E: Build Reliability Charts

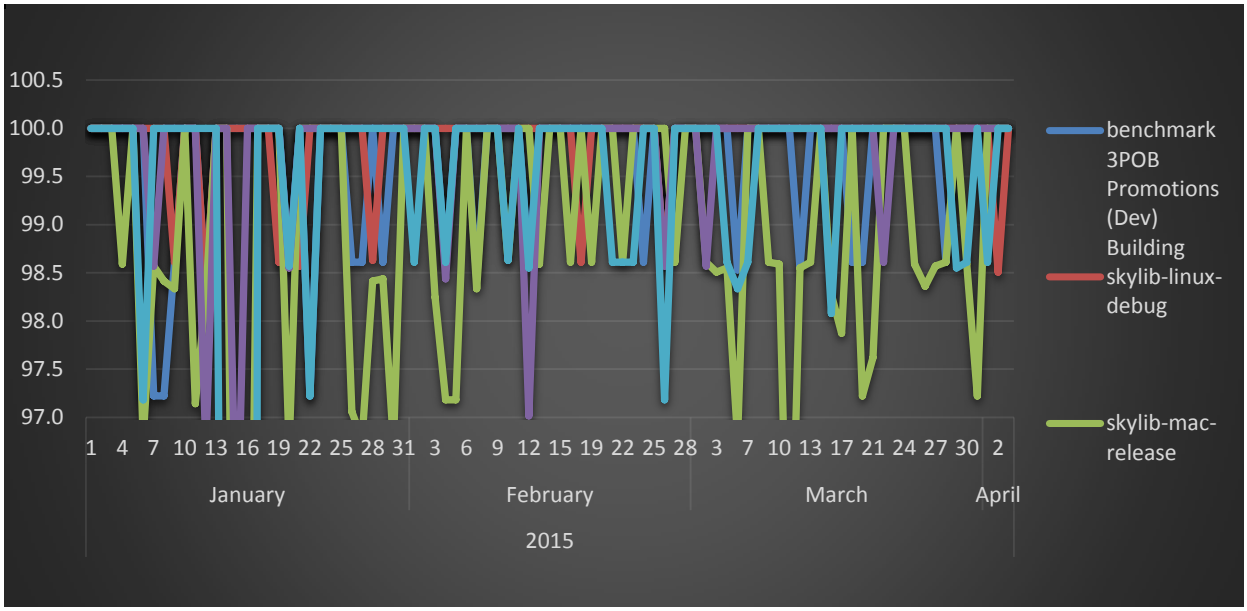

Figure 20 - Reliabilty breadown by day

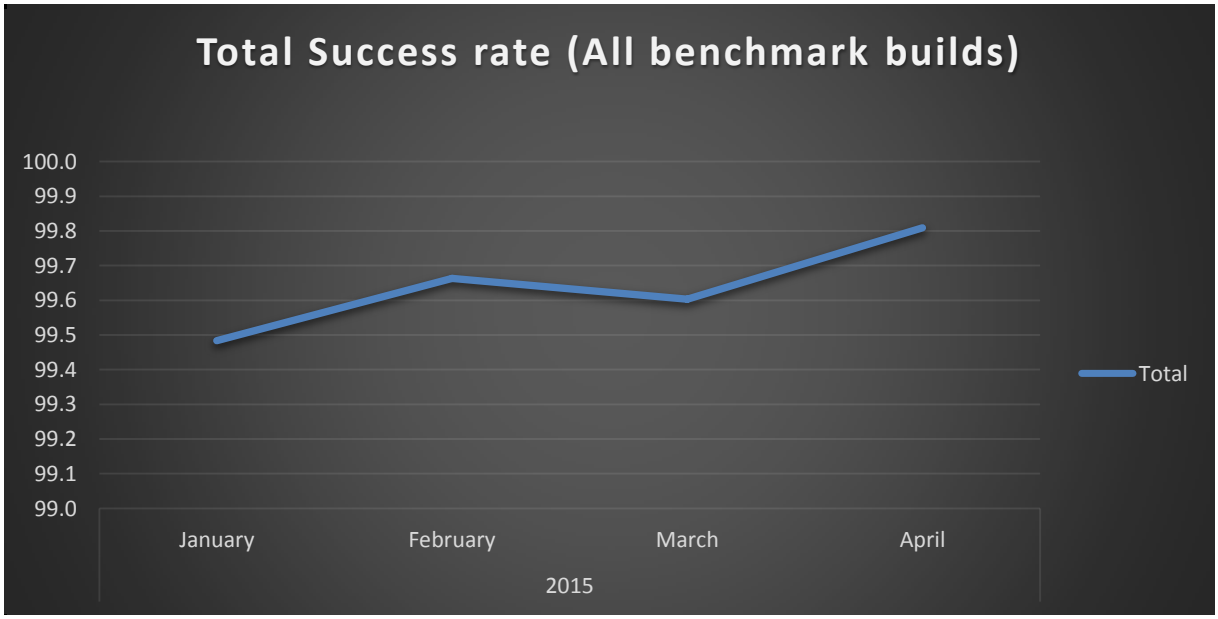

Figure 21 - Reliability total per month

Appendix F: Build speed charts

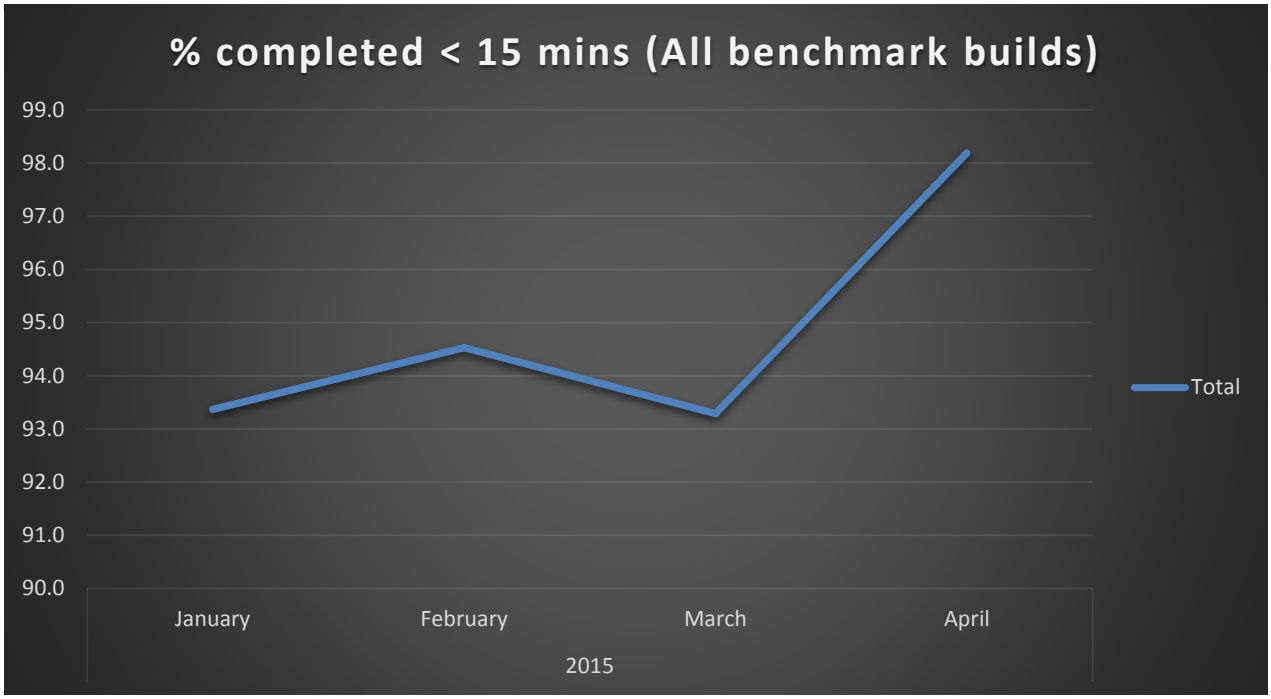

Figure 22 - Build Speed total per month

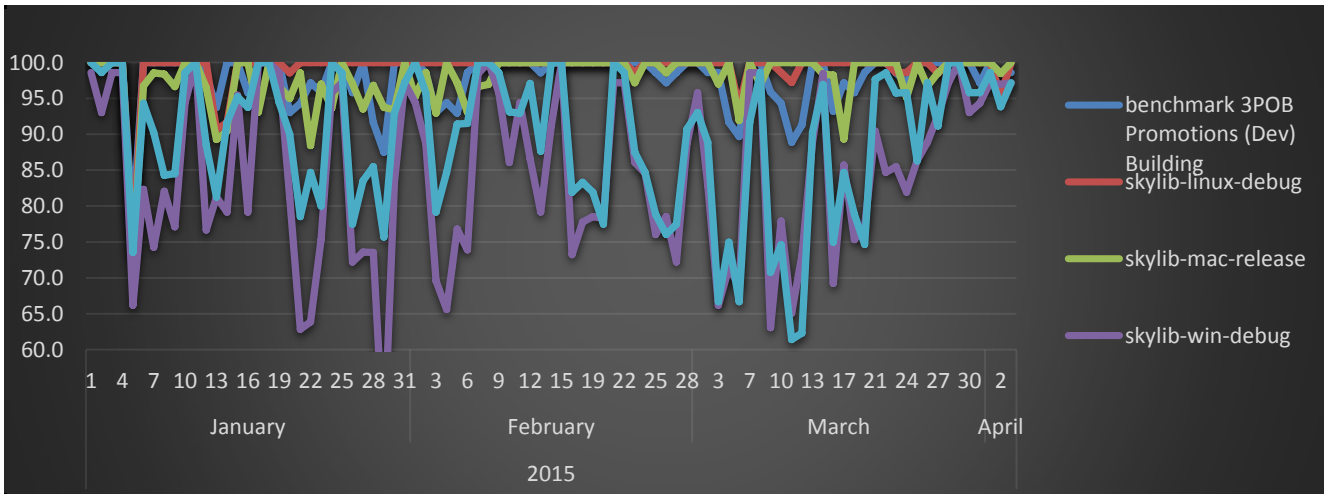

Figure 23 - Speed Breakdown by day