CHALMERS

Identifying Behavioral Patterns in Using Software Licenses

Master of Science Thesis in the Programme Software Engineering and Technology

HANS ANDERSSON
MARCUS SUNDBERG

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2011

Identifying Behavioral Patterns in Using Software Licenses

HANS ANDERSSON
MARCUS SUNDBERG

Examiner: MIROSLAW STARON
Supervisor: PER ZARING

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, June 2011

Sammanfattning

Idag brottas många företag med dyra licenskostnader. Hantering av mjukvarulicenser blir allt viktigare och allt eftersom antalet licenser växer blir det mer svårhanterligt för företagen. Om företagen kan hitta och identifiera onödiga licenskostnader kan de spara mycket pengar.

Problemet ligger i att hitta licenser som betalas för men ej används eller där företaget betalar en dyrare licens än behovet kräver.

Det här examensarbetet ser över möjligheterna att identifiera mjukvaruanvändande inom företag genom att titta på användarnas aktivitet när de använder program, för att upptäcka onödiga licenser. Detta görs genom att hitta mönster i användardatan för givna program och sedan jämföra detta med hur enskilda individer använder programmet. Arbetet utförs i en miljö som använder sig av Microsofts System Center Configuration Manager för att dra nytta av dess användardata. Mer detaljerad användardata samlas in via ett bakgrundsprogram som installeras på användarnas datorer.

En prototyp för att samla och visualisera datan skapas. Denna prototyp analyseras och sedan skrivs en kravspecifikation för ett förslag till en ny produkt som slutresultat.

Abstract

Today, a lot of companies struggle with expensive license costs. Managing said licenses is getting increasingly important to make business more profitable in a highly competitive market. Without paying unnecessary licenses companies can save a lot of expenses and increase their profit.

The problem is how to reduce the license cost and how to find unnecessary licenses which are underused or not used at all.

This master thesis looks into identifying and defining software usage within companies by looking at user activity when using programs, in order to discover unnecessary licenses. This is done by trying to find a pattern in the usage data for the given program and then comparing with individual user results. It is done in an environment with Microsoft's System Center Configuration Manager in order to make use of its usage data. More detailed usage data is gathered via a background program installed on all users computers.

A prototype is made to gather and visualize data. This prototype is analysed and as an end result, a Software Requirement Specification is made for a suggestion on a final product.

Acknowledgements

We would like to thank our supervisor Per Zaring for stearing us in the right direction as well as helping us with the structure of this master thesis report.

We would also like to thank Atea Spintop for providing us with a development and testing environment in their office in Göteborg and their employees. Special thanks to Joel Sanderi who has been our technical supervisor on the company and Kim Högström.

Contents

List of Figures

1 Introduction

This section will cover the background, purpose and scope of the Master's Thesis, as well as a glossary and a list of abbreviations.

1.1 Background

Software licenses stand for a significant part of a company's IT-budget and many companies have difficulties with maintaining correct licensing. There is also an uncertainty in how the software is used in practice and there is potential for saving money by reducing the number of licenses by uninstalling programs that are not used. A lot of software companies also have several different versions of programs, e.g. Standard, Enterprise and Professional Edition. In these cases it is preferable to make sure to not pay for more expensive versions than needed.

The master thesis will be done on Atea Spintop, situated in central Gothenburg. Atea Spintop is a subsidiary of Atea, responsible for creating solutions and developing products which are designed to automate advanced IT processes in a Microsoft infrastructure[3].

One of the things Atea Spintop offers is an evaluation tool that helps to save money by efficient licensing, mainly by finding unused licenses. Up until now Atea Spintop have mainly focused on checking whether programs have been started within the last 3, 6 or 9 months. Further development of this would be evaluating in detail how each program is used.

1.2 Problem

The problem focus will be in license management within organisations and unnecessary license costs. Is there a way to know if there are more licenses than users or users having the wrong type of license, i.e. using a more expensive program when a cheaper alternative is sufficient?

1.3 Purpose

The purpose of the Master Thesis is to investigate the licensing problem and the branch revolving around it to reduce overall cost for companies. This means finding out what solutions exist today for metering and reporting about used licenses as well as developing possible solutions that does not exist today.

1.4 Scope

During this Master's Thesis, it will be investigated what information about software usage and corresponding patterns can be gained from users in a network. Different methods to gather such information will be examined during the project to find a good approach on how to apply said information in an organisation. If a good solution or partial solution is found, the end result of the project will render specified requirements for the development of a tool which reflects upon this solution. The project will also investigate the possibilities to draw conclusions about software usage patterns and how such knowledge can be useful in an organisation.

A prototype will be made that can visualize the analysis in a user friendly way and where an end user shall be able to make conclusions on eventual actions to take. To support the

analysing part of the prototype, an agent that does software metering on client computers will also be developed during the project.

Construction of algorithms which automates and suggests how to reduce the license cost for the end user will not be part of the project. This means that the prototype will not be able to notice where improvements can be made. The problem would be that the prototype might not be accurate in every case, making it inefficient or hard to implement.

1.5 Glossary and Abbreviations

- Active Window - The window currently in the foreground on a computer.
- Agent - Program run in the background on a computer.
- AWL - ActiveWindowLogger. The agent that was developed during the project which gathers software usage data and stores it in a database.
- CM - Compliance Manager. An Atea Spintop product which keeps track of license usage within a company.
- Concurrent users - Users using the same programs simultaneously in the same network.
- End user - Person who will use the prototype.
- FileUsageSummary - Table in the SCCM database consisting of summaries of software usage presented as concurrent users.
- FileID - An identifier for specific versions of programs. Used in MeterData and FileUsageSummary.
- IM - Information Manager. An Atea Spintop product, which is installed on client computers, allows software to be administrated and sent out as plug-ins.
- MeterData - Table in the SCCM database consisting of explicit software usage information.
- Plugin - An extension program to a host application.
- Resource - Computer with a unique name.
- SAM - Software Asset Management. The field of managing software assets and licenses.
- SCCM - System Center Configuration Manager. A Microsoft product which manage Windows based networks and stores information in a huge database.
- SMS - System Management Server. The predecessor to SCCM, changed name in 2007.
- User - Person whose normal software usage is metered.
- VM - Virtual Machine.
- VMM - Virtual Machine Monitor.

2 Method

This section will cover which methods will be used during the thesis project.

2.1 Case Study

Since the task is to identify usage and finding patterns, this thesis will be done as a case study where a prototype will be developed and evaluated by end users and the end result will be a requirement specification. Prototyping serves to clarify problems and is useful when the requirements are unclear. A prototype is meant to be developed fast and serve to elicitate requirements [14].

Some projects have used the prototype to build upon when further developing the real product [15]. The prototype in this thesis will not be meant to be used that way. It is built as a throwaway. Since it is a prototype mainly built to investigate the possibilities and evaluating its usefulness, the prototype will ignore some important issues like security.

2.2 Literature studies

To be able to fully classify users and their behavior, literature studies will be made to find relevant methods of creating user profiles. This is done to strengthen the understanding of behavioral patterns as well as providing the basics to create structured profiles.

The main sources for literature will be Web of Science, Scopus, ACM Digital Library, IEEE Xplore and Google Scholar.

The software metering database tables from the Microsoft System Center Configuration Manager (SCCM) database, which holds usage data for users in a Microsoft Windows network, will be analysed. A case study for evaluating and using the data will be done to increase the understanding of the tools that will be used.

2.3 Development Environment and Prototype

Since the development will be done in a pure Microsoft environment, the prototype will be written in C# using Microsoft Visual Studio. The prototype will mainly be focused to produce a summary of the data found in the SCCM. The first step will be to analyse different ways to present the data to avoid showing numbers which can be hard to interpret. The prototype should give a good overview on different time spans, user profiles and programs. It will also summarise general user behavior. Graphs will be helpful in the presentation of when and how long a program is used, and how many users have it running at the same time. The main goal with the prototype is, as mentioned earlier, to find out where the company can reduce the total licence costs.

The prototype will be developed using an agile approach, with iterations each week. An agile approach is more fitting since the project is not completely defined from the beginning and what needs to be done is going to be affected by the results of previous weeks. It will be easier to expand and change the prototype as the project goes along.

The prototype will be developed as a stand-alone program with as few integrations as possible in Atea Spintop's current products. However, if the analysis of users are considered to satisfy Atea Spintop's customers, a plan for implementing it in future versions of Atea Spintop's products will be created.

2.3.1 Virtual Environment

The prototype will be developed on a workstation which is set up as several virtual machines (VM). ATEA Spintop has provided two workstations with identical setups. The workstations, or the virtual machines monitors (VMM), run on two different domains and have three VMs each:

- The first VM is the domain controller which handles authentication requests and other security related matters from the other VMs and the VMM within the Windows Server domain.
- The second VM is the machine where the Windows Server is installed. The SCCM is installed with all the administrative tools like the Console Manager together with SQL Server and Active Directory.
- The third VM is a client set up to report user information to the SCCM server. This is also the development environment with Visual Studio installed.

3 Theoretical Background

This section will contain information about Software Asset Management, Software Metering, Microsoft System Center Configuration Manager, Data Evaluation and Tools.

3.1 Software Asset Management

Software Asset Management (SAM), also known as Compliance Management or Software License Management, is a relatively new discipline that focuses on managing software assets and their licenses throughout an organisation or system. The discipline is there to help with things such as distribution, deployment, disposal and recycling of software assets. This is done with the help of various tools, processes and user policies. To what extent it is done differs widely in different environments. In smaller systems and networks it is easier for the administrator (software asset manager) to keep track of all users, computers and licenses while it becomes increasingly harder to do as the size of the system grows.

3.1.1 Why is it important?

Keeping track of licenses, software usage and user needs is increasingly important since software assets should be treated as any other asset in the company. A software asset manager should think of three things: legislation, strategic management and financial management[6]. These three points are closely related. Strategic management covers if it is better to take cheap or free software instead of more expensive ones, and weighing pros and cons against each other. This will impact how much the software will cost for the company.

Rightly implemented, SAM can help to eliminate over-buying licenses and can also be used to recycle previously purchased software. It can also help to limit deployment so that only software that is actually used is deployed. This helps to limit maintenance costs and by re-deploying software that is installed but not used, less money will be spent on new licenses.

Legislation is important because it is the software asset manager who is responsible for the company's installed software and will, according to the criminal law, be accountable for misuse.

According to Jakubička [6], the most common ways of software piracy includes several distinct cases. These can be categorised in to three different categories:

- Intended piracy is when a software or product is used without considering buying the license.
- Unintended piracy is when the software usage exceeds the number of licenses bought by the company. This is the most relevant category in this master thesis, since the purpose is to find functionality that can narrow the license costs down as much as possible without crossing the fine line of having too few licenses.
- User piracy is when a user downloads and/or installs a software without having the correct licensing for it. This can be intended or unintended, but it is still the software asset manager who is responsible.

3.1.2 Challenges with SAM

While the potential benefits for SAM are great, there are factors that makes it a bit challenging to be successful with the discipline. One of them is the different licensing models; There is capacity based, concurrent usage, modular based, etc. Other challenges are identification of installed software, use of browsers and generic clients, i.e. cloud computing.

This Master's Thesis will mainly focus on the concurrent usage license model.

3.2 Software Metering

Software metering is the practice of monitoring and maintaining software licenses by measuring, and sometimes controlling, its usage. An example of software metering is checking how many people have used a certain program within a given time interval or measuring concurrent usage in order to help the manager decide on how many licenses are needed.[8]

Software metering can also be done in a more active way by controlling the concurrent usage in real time. When a user wants to start a metered software, the software metering tool can actively stop the user if the amount of concurrent users have already reached the license limit. Other types of controlling software usage with software metering is prohibiting certain programs to be run or controlling, and stopping the usage of unlicensed programs.

This kind of active software metering demands a lot more network traffic and for large companies with slow network connections, this can often be unfeasible.

3.3 System Center Configuration Manager

System Center Configuration Manager, SCCM, is a Microsoft product which is used to assess, deploy and update servers, client computers and devices across physical, virtual, and distributed environments. It helps with managing the IT infrastucture in an organisation and gives enhanced insight and control [5].

3.3.1 History

SCCM was originally known as Systems Management Server (SMS) and was released 1994. This was the first out of three different iterations of SMS, called 1.x versions. The second iteration was 2.x versions and the one after that is called SMS 2003. Between SMS 2.0 and SMS 2003 some major functionality was introduced to the product, mainly the number of users the server could handle and new supported architecture [5].

The solution was intended to solve the growing problem with increasing number of laptops within organizations. Even though a user logged on to multiple locations the same programs should be downloaded and installed on the resource. However, it should support different download locations. The data gathered by the resource was also supposed to receive policies from its own site. All this was achieved with the introduction of Advanced Client and the Management Point [5].

The first version of SCCM was released in late 2007 [5].

3.3.2 SCCM Collections

With SCCM being the administrative tool it is, usability comes down to how well an administrator can control large groups of users and resources. Within an organization there are typically a lot of different levels of information sharing, as well as what software should be installed on each resource and for which users these should be available. Active Directory, which is also a Microsoft service, provides support for system hierarchy and will be explained in Section 3.8. SCCM is able to use discovery methods to find the information in Active Directory and import it. It is then up to the administrator to choose which groups of users should have which programs installed. It is also possible to create custom made collections of users within SCCM.

3.3.3 SCCM Reporting

SCCM generates an extensive amount of information, which is maintained in a database. Since the database is large and stores a wide range of information, it is a challenge for the administrator to get relevant information out of it. SCCM Reporting helps in this task by having a set of Reports defined beforehand that can be extracted into views from the database. These reports are defined to meet regular needs of businesses and include areas such as Software Distribution, Operating System Deployment, Software Updates, Software Metering and more [4].

3.3.4 SCCM Software Metering

Software Metering is an administrative tool used in SCCM that describes software usage within the organisation where SCCM is used. The administrator can enable what should be monitored based on a list of software that is generated by SCCM or create own metering rules and specifications. These results hold information, such as when a program is started and when it terminated. It also holds information on which product the executable file belongs to and from what resource it was run. Everything is stored in the SCCM database and is easily accessed through the different views and tables.

Active software metering was a part of SCCM but was removed due to network overloads when users had to contact the server before even opening a program.

3.4 Compliance Manager

Atea Compliance Manager is a software tool for managing and surveying software installations and licenses in an organisation. It uses Microsoft's SCCM reports to find information such as how many instances are installed and how many licenses the company has bought of each product. It also finds how many of these instances that has been used during the last 3 months, the last 6 months and the last 9 months. With this information, Compliance Manager can help with surveying licenses and uninstall unused instances of products. Compliance Manager provides an easy interface compared to the SCCM interface for working with the SCCM database and utilizing the information stored there.

3.5 SAM Products

Improving SAM is constantly worked on and many companies develops products which helps in this matter. Often they work as a layer on top of Microsoft SCCM which makes it easier to access the important information for the software asset manager. The thing that most of the developed software do is summarize information into to user friendly environments to remove the overwhelming feeling of data as seen in SCCM.

One of the products examined is LicenseWatch, in which the administrator can through a Web interface access and control assets within the company. It supports both active and passive metering. However, active metering only works if the user has the LicenseWatch agent (LQClient) installed on the computer. If the user then tries to open a program which has reached its limit on concurrent users, the program can't start. The reports that can be given is how many licenses are installed, how many are available and how many that are unused.

Another product is the Snow License Manager which have a similar solution.

3.6 Requirement Elicitation Methods

There are a lot of requirements elicitation techniques which is used when doing a software project, but only a few will be mentioned here. Many of them require much involvement from stakeholders, like customers, developers and end users. Due to this Master's Thesis being done in a researching purpose on what can be done rather than developing a product, stakeholders in this project are restricted. The choice of elicitation method will be prototyping, because the project is experimental in the sense that no prior information about what could be gained from it were known.

3.6.1 Prototyping

In order to find requirements and to get an understanding of a problem, prototyping can be used. It can be done for a whole system or smaller parts of it. Often the prototype being produced is a throwaway and is just built to get unambiguous and correct requirements [14]. There are a few different kinds of prototypes [15]:

- Presentation - Used to present ideas and convince the client that the future application is useful.
- Exploratory - Used when needing to clarify initial ideas and the developers can get insight on the problems they face.
- Evolutionary - Continuously developing the prototype. Developers work in close contact with end users to get a good product. This can often lead the prototype being used as a final product.

3.6.2 Brainstorming

In an initial phase of a project, brainstorming to find requirements are widely used [14]. This requires stakeholders to meet up and come up with innovative ideas during a set meeting time. It is very important that no ideas are shut down during the meeting; every stakeholder should

feel comfortable to voice her opinion. The brainstorming technique is a quick and effective way of initialising requirements. Typically, the meeting is divided into four smaller sessions: First unstructured ideas are listed, followed by a session to group related ideas. After that comes the part to categorise the ideas and the last session is to expand and prioritise the ideas.

3.6.3 Questionnaires and Interviews

Questionnaires and interviews are very much alike, in a sense, but the effectiveness and correctness of the result differs a lot [16]. The difference is that interviews are interactive, and the person who is asking the question is able follow the dialogue and clear uncertainties directly. The questionnaires on the other hand is an asynchronous way of communicating, and ambiguousness can occur. The benefit of questionnaires, however, is that it is possible to ask the same questions to a lot of people.

3.7 .NET and Visual Studio

Microsoft .NET is a software framework developed for Windows operating systems. It consists of two parts: a class library which supports several different programming languages and the common language runtime (CLR). The CLR works in runtime and provides memory management, thread management and other services. It also enforces strict type safety which in turn makes the execution very robust. The latest version, and the version used in the development described in this Master's Thesis, is .NET 4.0 [11].

Microsoft Visual Studio is a software development environment which is supported in the .NET Framework. Included in the software is live debugging, code editor and code refactoring, version handling (Team Foundation Server, TFS) and other tools. The latest version is Visual Studio 2010 and was used when developing during the Master's Thesis [12].

3.8 Active Directory

Microsoft Active Directory keeps track of every user, group, resource, and other objects in a domain. It is a directory service with everything accessible at the same place, simplifying administrative tasks within the domain. These tasks include password policies, auditing, different tools to help restore or restart the service if something goes wrong, etc.

User groups can be defined and administrated from Active Directory which simplifies setting up different permissions and levels of authentication for different users. Active Directory is a part of Windows Server [13].

3.9 Information Manager

Information Manager is a relatively new product from Atea Spintop. This application is installed on all clients in a network and is a tool that can further help with management. The plug-in architecture allows for a variety of possible functionality. It's appearance can be seen in Figure 1.

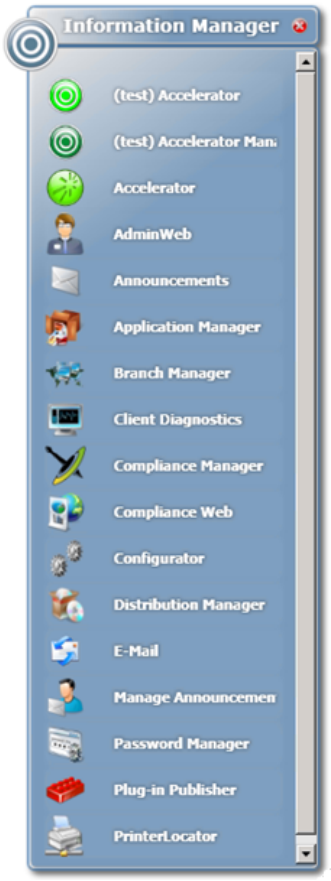

Figure 1: Information Manager

Taken from the Information Manager product page at Atea Spintop's homepage [2]:

Information Manager enables you to:

- *Reach out to your end users with important information.*
- *Provide your organization with customized and role-based functionality.*
- *Reduce IT Service Desk workload.*
- *Role-based distribution of functionality*
- *The distribution engine enables you to distribute the features of Information Manager according to user roles. This Atea-developed approach is thereby restricting system features to authorized users.*

Run-time extensibility without end user intervention:
Since Information Manager is built and based upon a plug-in based architecture the system functionality can easily be extended without the need of reinstalling the software. The plug-in architecture not only enables the client solution functionality to be extended it also enables deployment of new functionality during run-time. All without any intervention by the end users.

4 Prototype

This section will describe the prototype that has been developed. The prototype has two parts: collecting usage data and showing usage data with evaluation tools. The first section explains the data collection agent, and the following sections describes the different kinds of data evaluation tools which have been developed. Each subsection will have an explanation of its functionality, how it helps and opinions given by license experts. The opinions come from two different sources who are both end users of Compliance Manager (CM):

- The first one is a group consisting of two license experts at Atea's main branch who help customers with CM and evaluation of the given data within CM. These people have 10 years and 5 years experience respectively in license management, but do not have any technical background which would help in understanding SCCM better. They have worked with several big Swedish companies as consultants during their time at Atea [10].
- The second is the responsible license manager at a danish company with approximately 22000 employees worldwide. He uses CM as it is today and has over 10 years of experience within the field. He started out as a consultant from Atea, helping his current company with licenses. He, too, has no technical background [9].

The two meetings were conducted as follows: A video conference was set up with between the parties. The different parts of the prototype were presented, and they had time to give their opinions on the different concepts; if and how it could work in a real situation and in that case, to what extent they found it interesting and usable. The license experts were guided through the functionality of the prototype and were not able to test anything for themselves by any other means than asking the developers to do it. Since the goal of the meetings was to evaluate the functionality and have an open discussion around them, the meeting only had the prepared demonstration and some general questions.

The data shown in figures in this section are from two different sources and has been used for testing the prototype and all its evaluation functionality:

- The first is a database with SCCM meter data that holds usage information for approximately 1000 users for a selected few programs over 2-3 months.
- The second is a database from the collected ActiveWindowLogger (see Section 4.1) data that has 10-15 users over 2 months for all used programs.

4.1 ActiveWindowLogger

Since SCCM does not support Active Software Metering because of too much network traffic, an agent was needed that records information on each client on how much each program is used. This agent has been developed and will in the text be referred to as ActiveWindowLogger (AWL).

Atea Spintop has their client based product, Information Manager (IM), which handles things like automatic updates during runtime on plug-ins and possibilities to have different functionality for different user roles. Due to IM being well suited for an agent, AWL was developed as a plugin to this product.

4.1.1 Active Window and Process Logging

The AWL gathers information on which window is in the foreground and which process this window belongs to. This is done by checking the foreground window within a given time interval. The usage information is gathered in a database on a server. Further information sent to the server is how much of the total time was counted as inactive, i.e. where the user had not moved the mouse or pressed a key on the keyboard for a certain amount of time. While the information does not necessary reflect the usage completely for all programs, some programs are meant to run mostly in the background, it does add information about the usage of opened programs.

Since the AWL checks the foreground window with given time intervals, its process is set to a low priority in order to make sure it does not take away needed processing power from other applications.

In addition to logging the active window, AWL also supports logging all active processes on a resource. This means it checks which programs are running, when they are started and when they are terminated. While this is something done by SCCM too, AWL does it in real time. This could also service smaller companies with software metering that does not need to have something as vast as SCCM installed.

The AWL also takes into account when it does not log any data and this information is stored in a database table consisting only of log-intervals. This is done to be able to separate processes which have been running when active window has not been logging any data, because the server only consists of data with start time and end time and does not actively check for computer hibernation or other interruptions. Such interruptions in logging needs to be stored as well to be able to know which of the logged processes can be matched with the active foreground window logging.

4.1.2 Local Storage and Network Traffic Load

In order to not cause too heavy traffic on the network, AWL stores usage information locally in log files on the client computer and then reports once every hour to the server. The hour report is a calculation of the logs created by checking the active window where the total time for each process is calculated.

When the report is made, the more information heavy logs are removed from the client computer and when the report is sent to the server, the report file is removed. This way, the AWL will not take up a lot of unnecessary space and it also makes sure to send all existing reports each time it connects to the server. If the client is disconnected from the server for a while, it will still report for the hours it was disconnected upon reconnecting.

Internal tests showed that one week's worth of saved active window reports for one user make up for roughly 2 MB space on the local computer. For a company with 10000 users, this would mean a 20GB weekly network traffic load for storing active window information. This may not be much for some companies while it may be a burden for others.

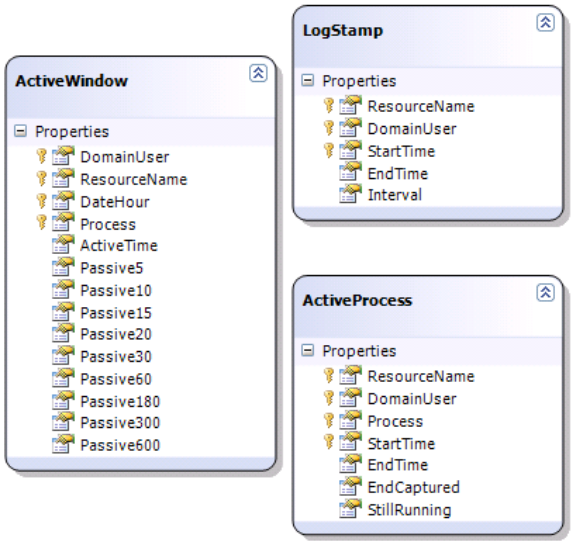

Figure 2: Database tables for ActiveWindowLogger

One issue is the question how the server will be able to handle the data load and how the database structure will be needed to be set up. There need to be possibilities to split up the work between multiple servers and the data may need to be split up into several smaller databases. Another aspect is scheduling. The report sending from the users to the server needs to be scheduled to make sure there is a limit on users trying to send in reports at the same time. One issue that can be seen with Microsoft's SCCM is storage space. The database will grow large quite fast so there needs to be a limit on how far back the gathered data is stored and there should also be a limit to a selected amount of programs metered. SCCM limits the amount of data by doing summaries of its metered data and the administrator is responsible to select which programs should be stored in the summary. This is described in section 4.3.

4.1.3 Server Database

There are three database tables that holds the information logged and reported by the plugin: ActiveWindows, ActiveProcesses and LogStamps. See Figure 2.

- ActiveWindows has a row for each unique combination of user, resource, time and process.
- ActiveProcesses has a row for each unique combination of user, resource, process and start time for given process.
- LogStamps has a row for each unique combination of user, resource and start time.

4.2 Stages of Passivity

As seen in the ActiveWindow table in Figure 2, the agent logs different stages of inactivity. The ActiveTime field holds the total time in seconds that the process has been the foreground

window. Passive5 shows how much time of the ActiveTime has been where the user has been passive for 5 to 10 seconds. Passive10 shows the same but for 10 to 15 seconds. This goes on until Passive600 where it holds all inactive time over 600 seconds. Passive is when the user has not moved the mouse or pressed a key on the keyboard.

An example: A user sets the Notepad process as a foreground window and then takes lunch for 30 minutes with the computer on. This would give the data:

- ActiveWindow: 1800
- Passive5: 5
- Passive10: 5
- Passive15: 5
- Passive20: 10
- Passive30: 30
- Passive60: 120
- Passive180: 120
- Passive300: 300
- Passive600: 1200

The intervals at the early stages of passivity (5, 10 and 15 seconds), were set as 5 second intervals and not shorter, since the set intervals of when the agent checks is 2 seconds and there needed to be some room for inaccuracies. Some tests run in the development environment showed that these inaccuracies can come from the agent process having a low priority on the computer, which can cause it to have a time interval of up to 4 seconds between checks. If the intervals would be 3 seconds when measuring inactivity stages, it could potentially skip an interval. Setting the interval at 5 seconds made sure that that no intervals would be missed. At the later stages of passivity, it was not crucial to divide it into as many intervals. This is because the later stages of passivity contains less information about the actual usage, since when a user uses a program actively, mouse movement and key presses happens more frequently.

4.3 Concurrent Users

Concurrent users has two different data sources, one is from an SCCM database and the other from the database with data collected by the agent Active Window Logger.

SCCM has two tables that is of interest when examining concurrent users: MeterData and FileUsageSummary. MeterData stores all information about software usage that is needed to find concurrent users, like start- and endtime, which user on which resource, etc. With this, the exact number of concurrent users can be found and the table can also reference which users are using the program at this time. This table is filled with thousands of rows

in a very short amount of time due to every start time for a process on every machine being stored. To solve this growing problem SCCM makes summaries of the data from MeterData into FileUsageSummary and then removes the original data from MeterData. SCCM also does not meter data automatically, an administrator needs to start metering of a number of selected programs that are of interest.

FileUsageSummary stores the data differently from MeterData. Instead of having all the detailed information, the data is divided into intervals of either 15 or 60 minutes. It is grouped together with a fileID and then counts the amount of unique instances running at during said interval from MeterData. The table will have the number of concurrent users for that fileID for each interval. A fileID represents a specific version of a program that is used, which means that several fileIDs can exist for the same program and when calculating concurrent users it is possible to simply add these together.

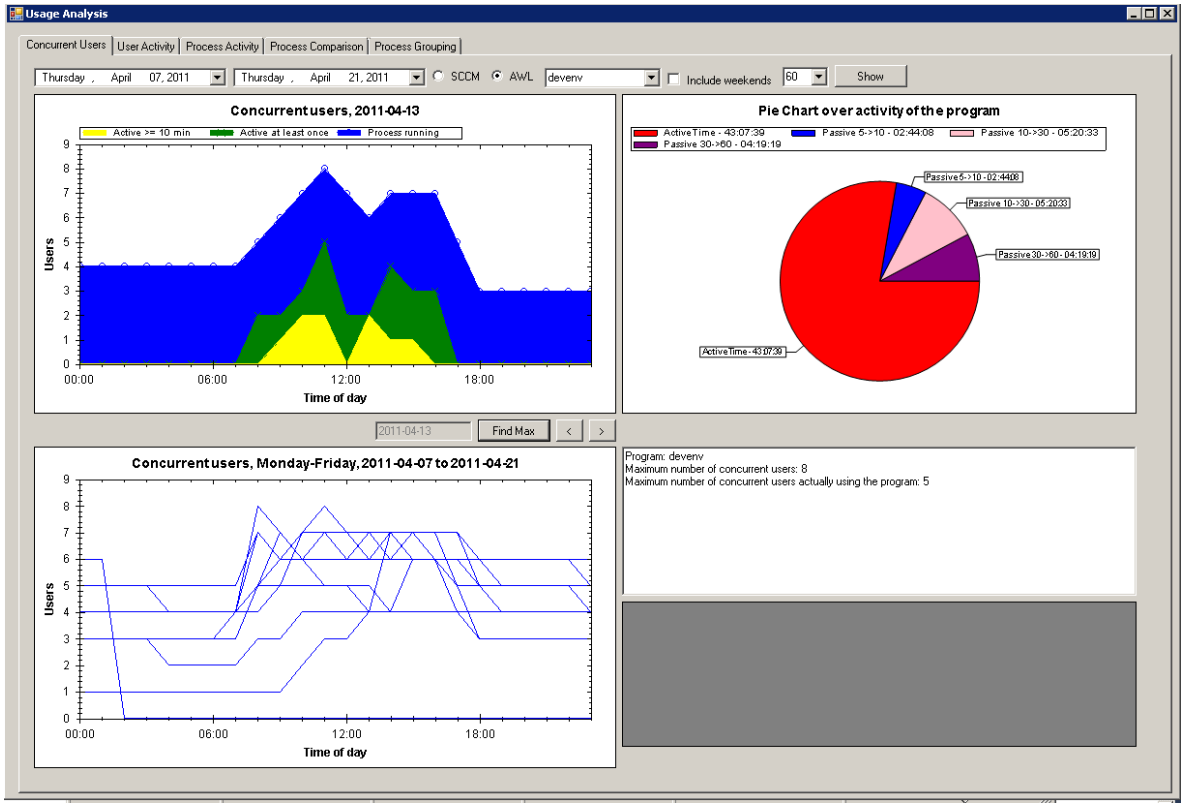

Figure 3: Concurrent users part of the prototype with data from the AWL database. In this figure, the interval is set to 60 minutes and devenv (Visual Studio) is examined.

4.3.1 Functionality Description

The current version of the concurrent users part of the prototype uses two different sets of data. One being SCCM and the other being the data collected by AWL. The AWL data gives the possibility to add more detailed information, such as active usage of the program and not only start time and end time.

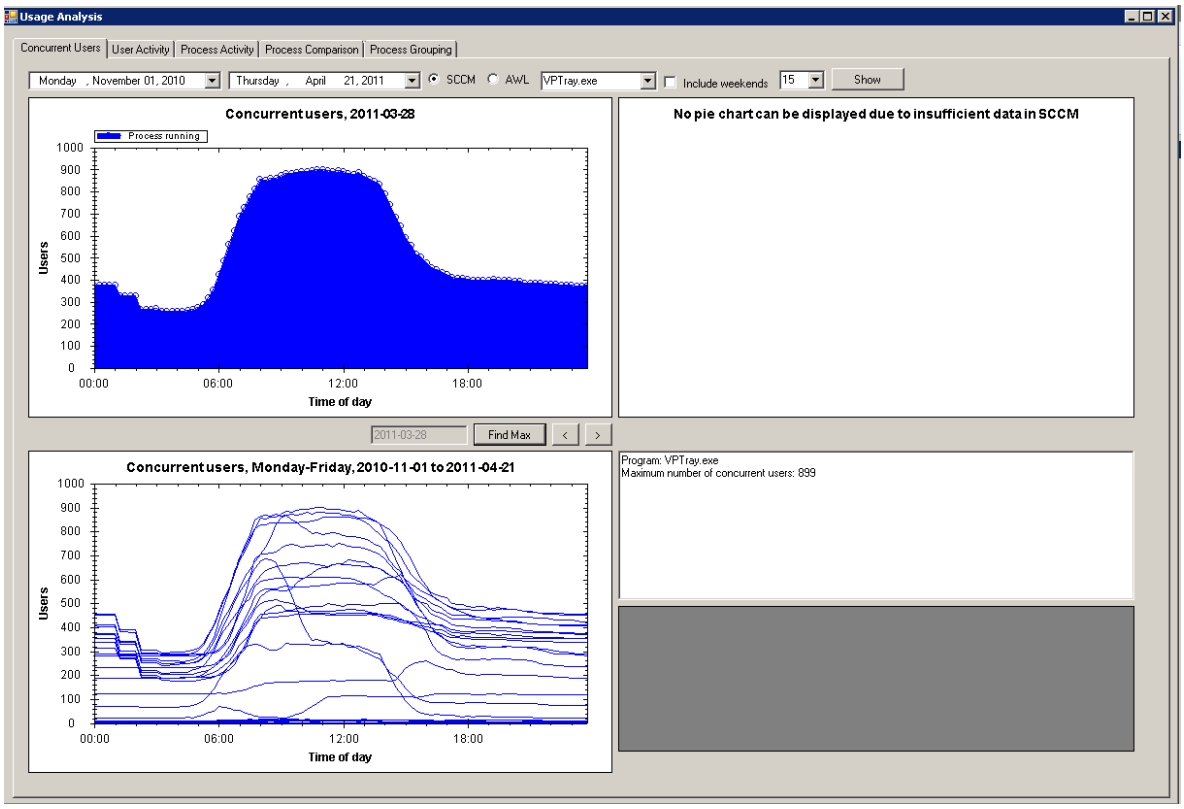

Figure 4: Concurrent users part of the prototype with data from the SCCM database. Here, the interval is set to 15 minutes and the program examined is VPTray.exe (Tray bar for Norton Antivirus).

With SCCM data, the prototype shows each day as a graph with amount of users running the chosen program. When using the AWL data, the additional information the prototype can show is the amount of users which are actually using the program, i.e. having the window active at least once during the time period, and using it more than 10 minutes during that time period. This is shown in the top left graph in Figure 3. There are three different colors which represents one group each: concurrent users are blue, active users are green and at least 10 minutes active are yellow. The pie chart in the same figure displays activity of the chosen program for the chosen time period. The pie slices are divided into four segments:

- Active Time (red), which represents all time the program is used where the user don't go passive at all.
- Passive from 5 to 10 seconds (blue), which is the time the user spends being passive between 5 to 10 seconds.
- Passive from 10 to 30 seconds (pink).
- Passive from 30 to 60 seconds (purple).

Figure 4 represents the SCCM part of the prototype. The pie chart is unavailable due to the database not having precise information about the users' active windows and the fields

which represent actual user count and actual active user count in the graph. The third graph in the figure, seen in the bottom left, represents the concurrent users for all days during the specified time interval in the same graph. This is the same in the AWL data but it is harder to see what Figure 3 visualises due to the low amount of concurrent users.

4.3.2 Prospected Use

As mentioned in the description of the prototype, the data can give information about concurrent users, with high precision, based on activity within the program. The prototype supports finding the day with maximum number of concurrent users which can be used to see the 'worst case' scenario. Licenses for some programs are based on a concurrent users basis and this way, the company can get information on how many licenses they actually need and only have to pay for the number of users running a program simultaneously. The day with the most concurrent users is sufficient to get that information.

The third graph can be used to identify strongly deviated days which would help to determine why some days might have spikes in their graphs and why others don't. This can help when analysing how it normally is and how often it deviates. If there are few deviations, a scheduling solution may be possible in order to avoid having a lot of extra licenses.

The AWL data has more detailed data to do an analysis on than SCCM. Since the active usage is known during any point in time, better conclusions can be drawn about the concurrent users. The usage patterns of different types of programs can be seen. An example of this is which programs are run a lot but not used. It helps in identifying if there are users who have the program running without using it. Being able to identify and eliminate such behavior can save a company money since the concurrent users will go down.

4.3.3 Evaluation

The Atea license experts thought the graph for day by day and finding the day with the highest amount of concurrent users was useful but they also wanted to see information about which the specific users are. They wanted this functionality to be connected with the process activity which will be mentioned in Section 4.5. A suggested connection was that when clicking on a given time interval in the graph, the process activity information would be shown for that interval and program.

The license manager thought the graph was excellent. He concluded that the company he works for has a lot of programs where they pay for concurrent usage and where it is very useful to see if the people actually uses the program actively, or only has it running in the background.

The graph that shows all days at the same time had a varied response and the experts from Atea were a bit neutral at its potential usefulness. They said it could be useful but it looked like the cluster of graphs could make it hard to separate single days. The license manager did not think he would use it much.

The pie chart was not seen as useful at all. It did not give much information of value.

4.4 User Activity

This part of the prototype is a tool for analysing user activity and how much of each program is used, in comparison to other programs, within a certain time frame. The data is gathered by the AWL agent and is based on the time the program is actively used when a user has the window in the foreground. This information is not available at all using the SCCM data. Activity is measured by checking whether the user has pressed the keyboard or moved the mouse.

The left pie chart seen in Figure 5 shows the most used programs measured by active time. The different stages of active time ranges from 0 to 5 minutes since the user had any activity. The 'Other' slice in the chart is there to represent programs which have been used too little to show independently. The limit for a program to be shown independently goes at 2% of the whole chart. This was mainly an aesthetical point of view due to the pie slices being too small otherwise and making the pie chart unreadable. The 'Idle' slice shows how much time the user has spent as inactive. A user counts as idle after 5 minutes of inactivity.

The right pie chart in the figure shows how active the specified the user has been during the chosen time period and time frame. The different pie slices here has the same limits as in the concurrent users part, namely 0-5 seconds, 5-10 seconds, 10-30 seconds and 30-60 seconds. The difference between this pie chart and the concurrent user pie chart seen in Figure 3 is that this pie chart is showing the activity of a user where the concurrent user version shows the information bound to a specific program and all users.

It is also possible to show the two mentioned pie charts for all users.

4.4.1 Functionality Description

There are some options which can be changed to customise the choice of data for the pie charts. First off is the ability to set the times of the day that should be considered work time. Employees might have their computer running when they leave work for the night or for the weekend. By only looking at work hours, the data will be more accurate since it otherwise will show a lot of hours with inactivity. Because of this, there is also a possibility to filter away weekends.

Another option is that a specific user can be chosen for the analysis. The functionality is the same as when observing all the users together, but it will give more precise information about which programs the specified user make use of during the time period. When a user is selected, it is possible to specify which resource to show the data from, which would show an even more narrow usage analysis.

4.4.2 Prospected Use

The danger when analysing the results of this part of the prototype is that every user uses his or her software differently. The information gained can not be directly connected with how active particular users behave, only which programs they actually use the most. It can however be used as a tool to check up on specific users to see more information about software usage and which programs they use the most. A problem software companies have is that when asking the employees which programs they use the most, and which are needed, a lot

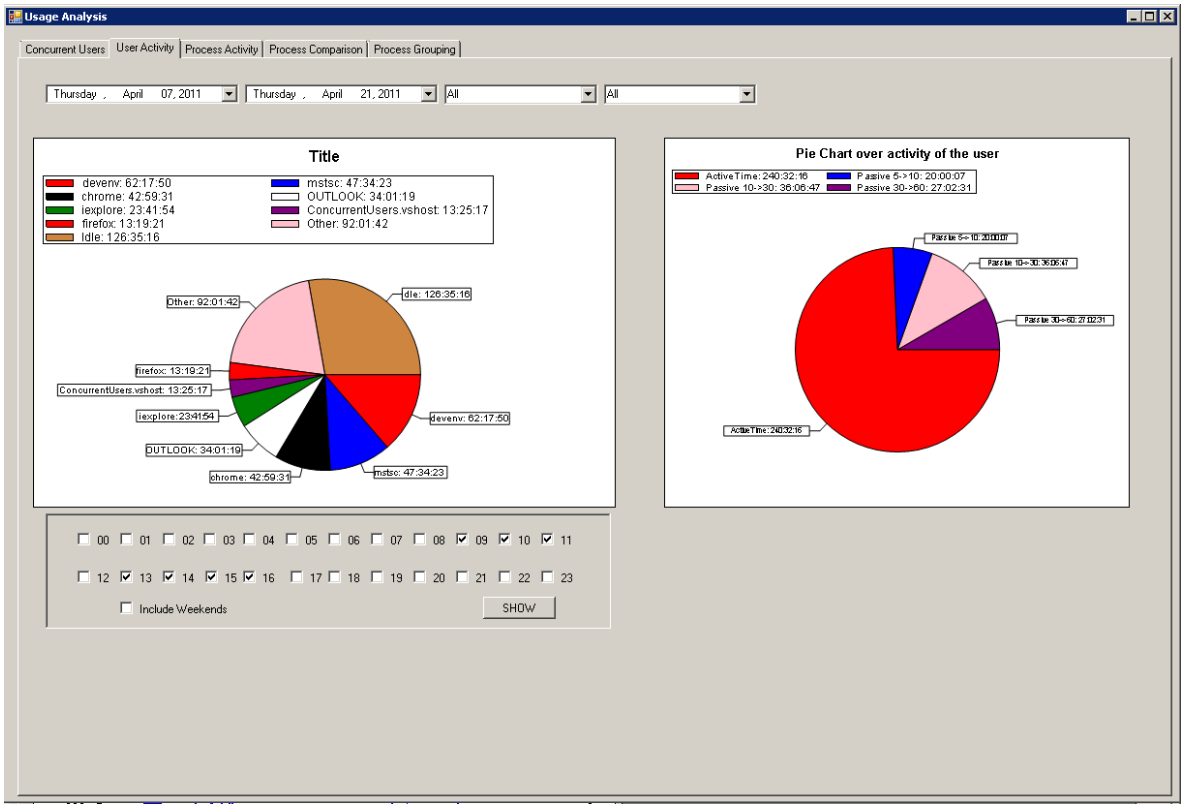

Figure 5: User activity for all users between 9:00 and 17:00 for weekdays, where lunch is considered between 12:00 and 13:00, hence not included.

of them answers what they perceive is correct, when the truth can be very different [9]. Even if this is done unintentionally, it can cost a lot for the company when having licenses which are not being used with full potential.

With the option to choose which hours that should be checked during the day and if weekends should be included, information about work hours can be extracted from the information as well. When people stop working their idle time increases. Still, this is no tool to find specific users but rather to examine them further when needing detailed information about them.

4.4.3 Evaluation

The Atea experts wanted to be able to divide the statistics in groups of users from Active Directory groups rather than single users. Showing what programs are used actively in a group could help in seeing needs in departments and groups. Showing a single user did not give much information of value at all.

The license manager saw a value in also showing charts for single users, and mentioned the difference between what programs were used and how it measured against what were perceived to be used. He also said that, as an addition to the functionality in the user activity part of the prototype, support for the SCCM data could be added. This will not have the exact same

functionality since the pie chart would have to be based on data like a programs total time, median time or mean time running instead of active time running. The difference in results would be that the pie chart would consist of which programs has the longest running time, without any information about software usage patterns. However, it might still be interesting information for some, but in reality, no real conclusions can be drawn from such a result alone.

4.5 Process Activity

Process activity lists users and their values, which measure their activities in chosen programs. All the data is bound to a given time interval. The columns in the table are listed below and can also be seen in Figure 6:

- User: The user connected to the specified data.
- Times Used: The amount of times the program is started and ended within the specified time period.
- Total Time: The total time the program has been the running.
- Median Time: The median time the program is run.
- Mean Time: The mean time the program is run.
- Active: The summarized time when the user has the window active.
- Active/Total: Percentage of how long the user has been active over how much total time has been logged.
- Last Used: When the program was closed the last time it was used.

There are differences in the results depending on the data source. When using the data from SCCM, the Active and Active/Total can not be measured at all. due to the information being non-existent in the database. Another advantage with the AWL data source is that it also holds information about when the agent is logging data on each client - which means that events like hibernation of a computer, where no programs are running but no programs are recorded as shut down, is caught. This gives more accurate values of measurements such as Times Used, Total Time, Median Time and Mean Time.

Since SCCM does not catch hibernation in its software metering, it will look like the programs have been running from the time they were first started until they are actively shut down or the computer is shut down, which makes the usage information misleading.

The information needed to make reports can be found in the SCCM database but, there are some problems which needs to be attended to first. The table MeterData has everything needed to make extended reports about usage like mean and median time a program is run. However, due to MeterData's extensive information, the size of the database grows fast in a company with a larger user base, and takes up a considerable amount of space on the server. Since SCCM is made to work for all sorts of companies, where some have very limited computer resources, the size of the database needs to be limited as much as possible.

Because of this the information in MeterData is summarized, simplified and put into the table FileUsageSummary. As this happens, the data is gradually being deleted from MeterData.

These summarizations takes away some valuable information when analysing software usage and for most companies, storage space is not that big of an issue. In order to save the information in MeterData and make process activity measuring possible, parts of MeterData had to be copied into a new table daily.

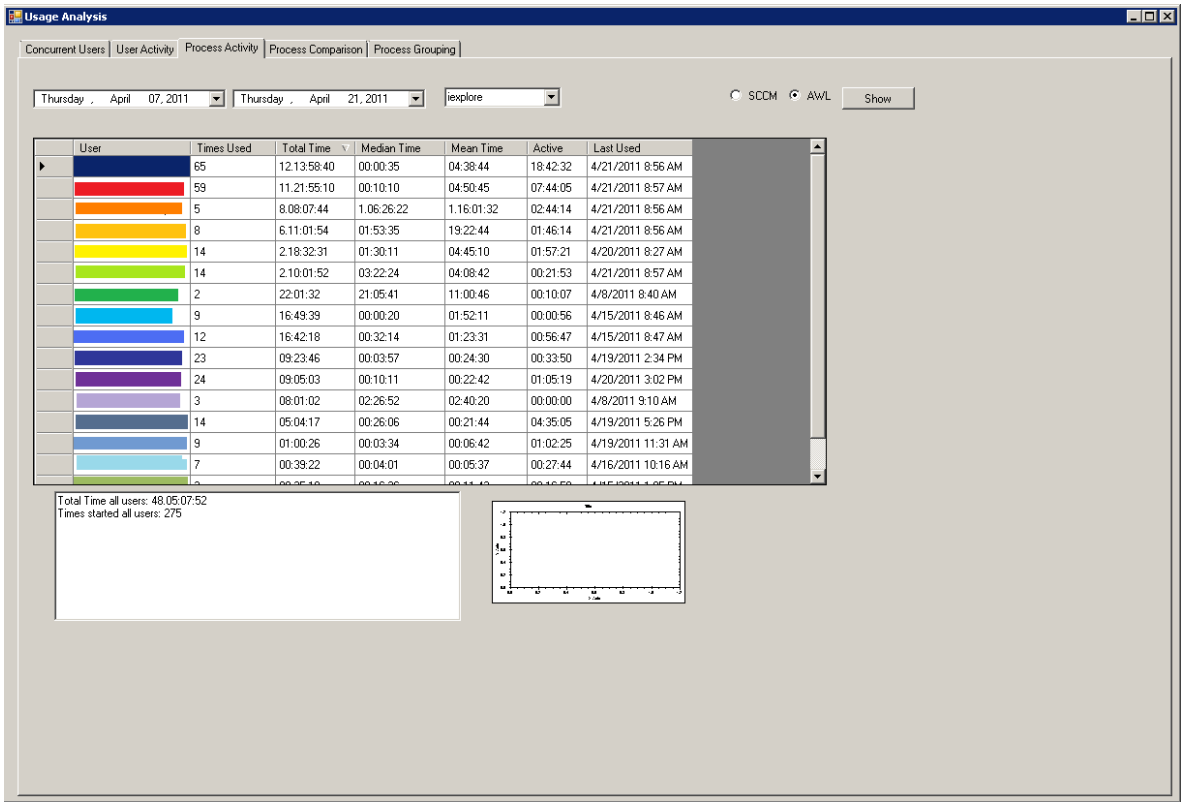

Figure 6: Process activity part of the prototype showing iexplore (Internet Explorer) with data from the AWL database. Users have been covered in different colors to ensure anonymity in this report.

4.5.1 Functionality Description

The two options for what to display are the selection of data source, SCCM or the AWL agent, and the program to be analysed. When the list is displayed, the rows can be sorted based on each of the columns' values. This is shown in Figure 6.

4.5.2 Prospected Use

The information gained from the list is extensive and has a lot of data in it and because of that it is an excellent way of finding users with low activity within certain software. Obviously the information has to be weighed based on the situation, but programs with expensive licenses should also be used. With this tool rough usage patterns of the users can be seen as well as

how much they use the programs.

Since the information can be sorted it is easy to use the categories with most value within specific organisations. If a person is very active when running the program it means she is probably utilising the that program in her work. If the opposite is discovered, precautions could be taken about whether or not she actually needs that program and might in turn save money on licenses for it.

4.5.3 Evaluation

The license experts from Atea really liked this information. It was easy to handle and export to Excel, which they said was very important. The process activity part of the prototype held useful information, with both the SCCM and the AWL as data source. The fact that the data held concrete and easily understandable and sortable information was appreciated. One useful thing they wanted was to have filtering options added. This would make it easier to find groups of users which have unwanted behavior or stands out in other ways. It could be useful to add average values for the program being evaluated and also some graphs that could add general information to compare with the specific numbers given.

While the Atea license experts liked the information given, the license manager did not find much value with it. He felt that the data was given without context. Median and mean time was not that interesting to him.

4.6 Process Comparison

In process comparison, two sets of programs can be evaluated and put up against each other in order to get an overview in how they compare when it comes to how they are used. The analysis is based on the same values as user activity (see Section 4.4) where it is measured how actively the programs are used when they are used.

4.6.1 Functionality Description

The end user adds programs in to two different groups and chooses a time interval. The displayed results are two pie charts that show how actively these groups of programs are used (see Figure 7). The pie charts follows the same formulas as user activity, which means that it shows how much time that has been spent in different stages of activity. Each pie chart is made from data of its given group of programs for all users in the given time interval.

4.6.2 Prospected Use

This functionality makes it easy to compare different types of programs and see how these are used. One example is Web browsers. They should follow a similar pattern which can then be used to compare with other programs to see if the activity data is alike.

4.6.3 Evaluation

The Atea license experts wanted to be able to base it on groups of users from Active Directory groups in order to see how the usage compared between different sets of programs. One

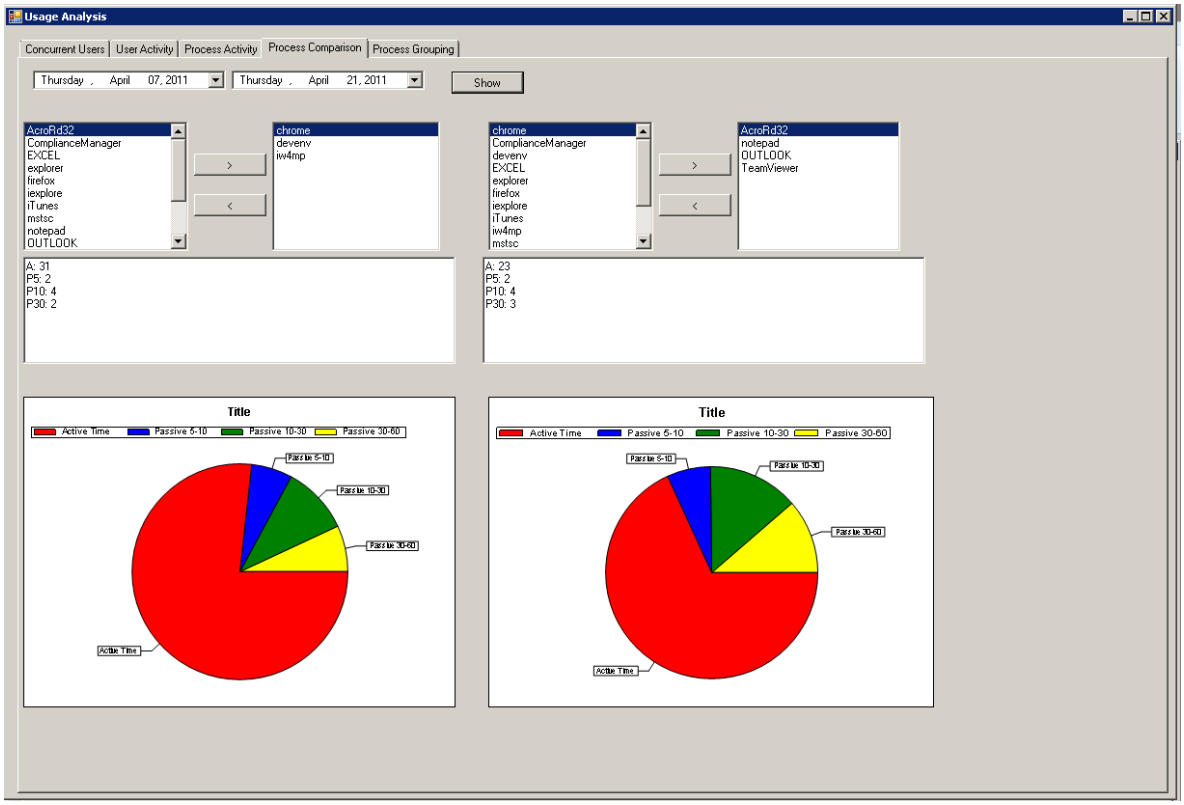

Figure 7: Process comparison part of the prototype where activity within groups of programs can be compared and analysed.

example is if a certain department used Adobe Professional in the same way as the metered standard for Adobe Reader. It could then be investigated if the free Adobe Reader is sufficient for that department instead of Adobe Professional.

The license manager only had time for a quick glance at the process comparison part, so he did not have very much to say about its usefulness. Also, he was not very interested in the functionality which only included data that required the AWL database, because he was unsure about whether or not his company could install such an agent at all.

With more substantial data, an automatic grouping of processes should be possible, opening up for companies to find software usage patterns. These can then be used to potentially identify programs which can be switched to a cheaper alternative.

4.7 Process Grouping

This section will explain the process grouping part of the prototype. This only works with the AWL database, since it uses software activity data when grouping programs.

In process grouping, the programs or users get spread out on a graph depending on a value they get from a formula based on its activity data. It mainly uses the same data as user

activity (see Section 4.4) but presents it in an alternative way. The resulting graphs can be changed to find patterns in software activity within the company based on the different formulas implemented.

As can be seen in Figure 8, it has two graphs, each showing percentage values. The lower graph shows the value for each program based on data for all users, the upper graph shows the value for each user on a chosen program. The formula used is displayed in the top left corner.

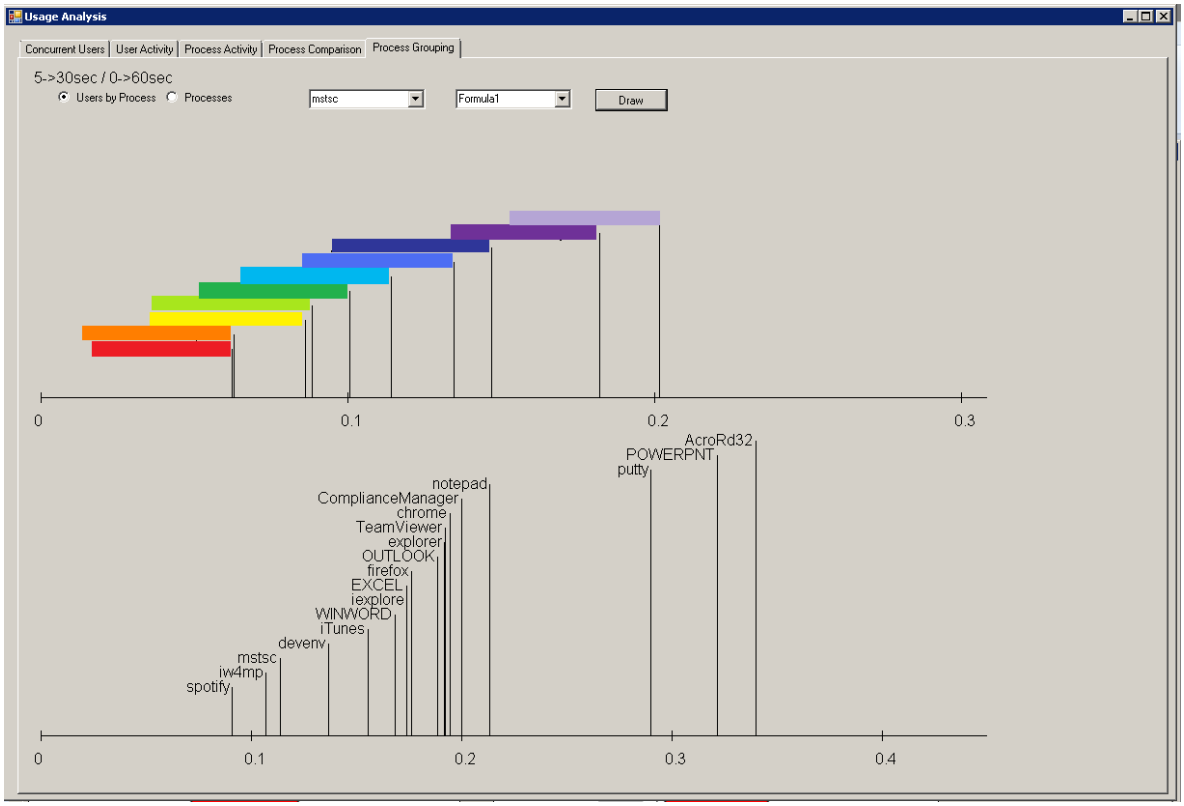

Figure 8: Process grouping part of the prototype. In this picture, activity within mstsc (Remote Desktop Connection) and overall for every program is displayed. Each user have been colorized to ensure anonymity in this report.

4.7.1 Functionality Description

There are two options when running process grouping. The first option is to choose which program the top graph with all the users gathered together should display and the second option is to select which formula should be used for the graphs. The graphs are then drawn and depending on which formula is used, different activity percentages will be shown.

4.7.2 Prospected Use

Looking at the specific formula used in Figure 8, one can see the measuring of the time spent in a passivity stage where there has been no activity from a user for 5 to 30 seconds. This

is divided by the total time the users have been relatively active where total active time is defined by at most 60 seconds since last keyboard input or mouse movement. This particular result shows a higher passivity when looking at AcroRd32 (Acrobat Reader) than it is in mstsc (Remote Desktop Connection in Windows). With a higher user count to base this on, conclusions could be drawn that when using Acrobat Reader, users are more passive than when using Remote Desktop Connection.

It is the same when looking at the first graph in the figure, some users are more active than others. Even if this might not mean anything in particular per user, comparison between different programs can give some information about the activity norms within the organisation. With a larger user base this can give the distribution over one program about activity.

4.7.3 Evaluation

The Atea license experts though the information shown is hard to interpret in graphs and wanted to implement a way of showing the numbers generated from the formula instead. They had an idea to connect it with the functionality of process activity (see Section 4.5). The most important feature for them is to be able to export the data easily and show it to other people which might not be as familiar with software usage as they are. The data exported needs to be understandable by all, and they thought numbers were better than graphs in this matter.

The license expert did not have time to look on the functionality of this part, but as mentioned earlier he was not very interested in parts that only included the AWL database.

In addition, the functionality to create own formulas for the values available in the database is needed. This would possibly make it more diverse since customers might not be looking for the same results that can be gained from grouping processes. Also, the first graph in Figure 8 would likely be changed to a histogram in the future when more users are in the database. It would not be interesting to see individual users in this particular graph, but rather the activity distribution between them.

5 Results

In this section the results of the project will be presented.

5.1 License Management

License management is about everything that has to do with licensing within a company. Examples of this are keeping track if how many licenses a company needs to support all their users, making sure that as many licenses as possible are used, and distributing correct software to users to reduce overall costs. This is a very hard task as it is today, and many companies have realised their need to save money on licensing, which is getting increasingly costly due to more expensive licenses.

There are a variety of products that helps with this, mainly through software metering. Some products only have passive software metering, where usage data is measured but no intervention with the usage happens, while others have active software metering. Examples of active sotware metering is stopping a user from starting a program when a given limit has been reached for running instances of that program in the network. LicenseWatch and Snow License Manager are two products who both use a solution where an agent is installed on each computer that helps with active software metering.

Compliance Manager (CM) from Atea Spintop only uses passive software metering. The biggest strength of Compliance Manager is that it does not need an agent installed on every computer. CM uses System Center Configuration Manager (SCCM) to get usage data instead. SCCM is a system management software product from Microsoft that is already used by a lot of companies around the world. Instead of installing an agent on each computer, an administrator will only need to start metering the programs that are interesting in SCCM.

An important aspect of license management that is hard to manage is knowing whether or not a software is used and if it is used, how much, and by how many users. In SCCM there is information about software usage like start time and end time for each program. Although the information exists on a per user basis, it gets summarized into tables and removes detailed data for each user that could be interesting from a license manager's perspective. The usable information for the chosen metered programs that are left after the summaries is mainly concurrent users over time and when each user started it last.

5.2 Prototype

This section will provide a quick summary of the prototype developed during the project without any technical information.

The prototype which was developed consists of two different parts. The first one being an agent, ActiveWindowLogger (AWL), which is installed on client computers and collects data about software usage on that particular machine which is sent to a server. The second part is a usage analysis tool which shows possible extensions of displaying usage data in Compliance Manager. It extends the usage of the data given by SCCM and also makes use of the more detailed information provided by AWL. It presents the data in various ways, like visualising

with graphs and having sortable data tables.

The agent logs the amount of time that each program is the active window on the client computer as well as how actively the computer is used by measuring different stages of passivity. These stages of passivity are defined by how long ago a user touched the keyboard or the mouse. It also has the possibility to log all the processes that are currently running if there is no SCCM data available to give such information. It then reports it to a server which collects all the data from all clients and stores it in a database.

The Usage Analysis prototype consists of five different parts, each displaying their own type of information. With only the SCCM database available, concurrent users and process activity can be viewed and analysed. In order to be able to show the more detailed information from SCCM, the data that is removed automatically by SCCM when summaries are made is stored in a new table before removal. With the AWL agent available as well, user activity, process comparison and process grouping is also viewable. Each of these parts of the prototype shows usage data in different ways which can be useful for a License Manager.

5.3 Evaluation and Analysis

For evaluating purposes the prototype and its functionality was presented to license experts within Atea as well as a license manager from a company with over 22000 employees who voiced their opinions about it. This allowed for the prototype to be evaluated by potential customers and future users. They found the results interesting overall and they could see the potential benefits of having access to information gained through the prototype.

Their opinions differed a little when it came to what part was the most interesting. The license manager thought that every functionality that could be found using only SCCM data was better since it would not require installing and managing an agent, which he thought would be hard to convince his supervisors to invest in. The license experts from Atea found the process activity part of the prototype the most interesting because it handles raw data and numbers which is easily exported to a spreadsheet in Microsoft Excel and can be used as a reference when presenting an analysis.

5.4 Requirements

The result consists of two Software Requirement Specifications (SRS) presented to Atea Spintop combining the functionality of the prototype together with their own products (see Appendix A). The products that were considered was mainly Information Manager (IM) and CM since these are the two which would probably be where the new functionality can be implemented. The reason the SRS is divided into two projects is because of the strong difference of the prototype with and without the support of an agent as well as the difference in difficulty to implement it and sell it.

The first SRS project is based only on SCCM data and how to use its full potential. This project will handle the additions which will be needed in order to improve CM and its functionality. Even if this only gives some sort of basic functionality compared to the analysis made in this master thesis with the help of an agent, this is a very important part and is also

easier to implement.

The SRS consists of two smaller parts, one which handles the SCCM database and makes sure that usage information is not deleted and another which visualises the stored information in the SCCM database. Since it has nothing to do with the agent, Atea Spintop's customers will not have to install any additional products to gain the improved functionality in CM.

The second SRS project suggests to implement an agent that is installed on machines which should be monitored for active software metering. It is not specified how this should be developed but it suits to be installed as a plugin for IM. This project is considered an addition to the first project.

In this project, a database needs to be constructed to store all the data the agent collects. A lot more information needs to be handled by the system and the development will be more time consuming due to planning and testing.

5.5 Switching to a cheaper alternative

A big part of the initial phase was to find alternative program when talking about software licensing to reduce the overall cost. This means installing a program which have all the functionality needed but does not cost as much as an expensive license program. This was considered a failure due to insignificant or nonexistent data where software usage patterns could be found.

To be able to know when the possibility exists to downgrade to a cheaper version of a program or to find another program which has the wanted functionality is very hard. It became problematic to find a solution since the data that was available during this project was not what was expected from the beginning. Due to this, no valid way of finding consistent software usage patterns could be found, and hence no results will be presented.

5.6 Software Usage Patterns

Due to the low quality of the SCCM data when it comes to active software metering, it was very hard to find any software usage patterns. The AWL data became the solution to that problem. However, during the project very little data was recorded. Nevertheless, some tendencies in the data collected by the AWL could be seen and will be presented here.

The first part is the concurrent users together with process activity. They fall under the same category, since what is shown in those two parts of the prototype are strongly connected. It was observed that there were always more users having the program running than actually using it, and active time within a program was often pretty far from the total time the program was running. This is due to users not terminating the program when they are not using it.

Another tendency towards software usage patterns was also discovered when examining the data displayed in the process comparison and the process grouping part of the prototype.

Here it could be seen that programs which involved more reading (Acrobat Reader, Outlook) than actively typing (Visual Studio) showed data leaning towards more passivity.

6 Discussion

When it comes to identifying software patterns, it is hard to know in detail what has been done before and how much of what has been done in this thesis is new. There is not much in form of scientific articles about the subject, and companies withholds information when developing own agents and software metering tools, which makes it hard to research.

6.1 Test Data

The lack of test data for this project can mainly be attributed to two reasons. The first one being that the SCCM data is not sufficient when it comes to mapping software usage and user behavior in different types of programs. The only thing that is actually in the SCCM database is the support for finding concurrent users for programs which are metered. Storing the table MeterData daily is also an alternative way of gaining more information about software usage, but it will not tell anything about detailed software usage. This is why the agent was made.

The second reason for the lack of test data is that no large system was available to run the developed agent on. The agent was only run in Atea Spintop's own office where 10-15 people sits. Because of this, no real conclusions could be drawn from the gathered data and the evaluation of the usefulness of some of the graphs became more speculative. Some tendencies for patterns in the data could be detected and with more test data, one could perhaps have refined them and actually be confident that there were useful patterns to find in those graphs.

6.2 Agent - Installation, Upkeep and Network Load

One of the strongest benefits of Compliance Manager (CM) is the fact that no agent is needed for the product to work, since it makes use of SCCM data. Because of this, the company does not need to install and handle updates for it on every client computer. Some functionality, like active software metering has been sacrificed in order to have the strength of not needing an agent. Developing an agent may seem to go against that particular strength, but in this case, the agent is there to add functionality rather than being the essential part for the software metering to work.

The benefit of having an optional agent for added functionality, that works together with CM, is that it does not need to be installed on every single client in the network. This is the biggest drawback for agent based products in larger companies. With an agent that only needs to be installed on the clients of interest, meaning the ones who actually use the particular programs that needs the extra functionality in software metering, it becomes much more manageable. The license manager for the company of 22000 employees gave an example where they had a program that was really expensive and where more active software metering would be very beneficiary. This program was not used by more than 100 people.

6.3 Comparison with Other Projects Using Prototyping

The prototype developed in this Master's Thesis is mainly considered an exploratory prototype, but can be used more as an evolutionary prototype since it can be built upon together with more end user interaction. Even if it is developed further as an evolutionary prototype,

it should still be a throwaway prototype. Other projects that were investigated were mainly of an evolutionary type. Some projects which are considered failures, like Process Control System and Document Management System, often fail due to too little contact with end users and customers [15]. To be able to strengthen the chance of the prototype developed during this thesis being used, more user involvement is required, especially when it is an evolutionary prototype.

7 Conclusions

The resulting software requirements specification (SRS) should be seen as a first draft. The task in the thesis was to find useful ways to identify software usage patterns and the SRS gives a more clear view on where the focus should be. Because of the limited end user interaction, the SRS should not be seen as a final version.

The limitation of the test data is also another factor that should be considered when analysing the two suggested projects in the SRS.

A way to further increase the strength of the SRS would be to get end users in on a more active role in the elicitation; after the first iteration of the prototype was made, and when the requirements were more clear. Structured interviews would be good to get an even more clear view on prioritisation. Active end user involvement should at least be an essential part during the development of the product.

7.1 Prototyping as a Requirement Elicitation Technique

Since the thesis was based on a vague idea about finding possibilities to identify software usage patterns, it would have been hard to get much useful information with other elicitation techniques that involved stakeholders, before finding out what was possible to do. It would have been hard to explain the ideas and deciding what could be useful without any concrete example to show.

The fact that issues like security and network load were, partly, ignored was necessary in order to get a prototype out as fast as possible. These issues, while important, were not needed in the prototype, since it was a base for discussions around the possibilities and new functionality for the Compliance Manager. Since these issues were ignored, it was vital that the prototype should be a throwaway, since these aspects need to be considered from the beginning when building the architecture of the program.

One thing that would have been useful would be to have another iteration of the prototype and another end user evaluation. There were evaluations with the in-house developers at Atea Spintop throughout the prototype of the development, but these were not from an end user perspective. Structured interviews with end users as well as brainstorming, with the prototype as background information, would also be useful when new product requirements should be elicited.

7.2 Further Work

During the master thesis, discussions came up about personal integrity. One approach to a different project would be to evaluate how much analysis can be made without violating the integrity of the users. This analysis can be extended to compare laws in different countries as well as how to work with a product similar to the prototype developed here.

Another project would be to find software usage patterns with good amount of data. This would include doing extensive analysis of software usage. The quality of the data is crucial to a project of this type since it is based solely on analysis. In addition to having better data, interviews can be done on the users of that network to do even deeper analyses. Connecting the two will render substantial analytic results.

Further development of the agent and its usefulness would be adding active software metering. The agent would then have more active functionality which can include stopping users from starting certain programs when a limit of concurrent users is reached. An administrator could flag certain programs and when a program has been idle for a while on a computer, a dialog box could come up to request the user to close it or even force the process to be closed. This could be done without sending data to any servers and the network traffic would not be affected.

References

[1] Atea Spintop (2011). *Compliance Manager*[Online]. Available from: http://www.ateaspintop.com/spintop_uk/products-spintop/compliance-manager.aspx [Accessed: May 31st 2011]

[2] Atea Spintop (2011). *Information Manager*[Online]. Available from: http://ateaspintop.com/spintop_uk/products-spintop/jumpstart/information_manager.aspx [Accessed: June 2nd 2011]

[3] Atea Spintop (2011). *About Atea Spintop*[Online]. Available from: http://www.ateaspintop.com/spintop_uk/about-spintop.aspx [Accessed: May 30th 2011]

[4] Microsoft (2011). *Reporting Overview*[Online]. Available from: http://technet.microsoft.com/en-us/library/bb632995.aspx [Accessed: May 31st 2011]

[5] Microsoft (2011). *System Center Configuration Manager 2007 - Configuration Management*[Online]. Available from: http://www.microsoft.com/systemcenter/en/us/configuration-manager.aspx [Accessed: May 31st 2011]

[6] Jakubička, M. (2010). *Software Asset Management*. Inst. of Comput. Sci., Masaryk Univ., Brno, Czech Republic.

[7] Addy, R. (2007). *Software License Management*. EFFECTIVE IT SERVICE MANAGEMENT p263-273.,

[8] Kruetzfeld, R. (2003). *Pro SMS 2003*. Apress. ISBN 1590596986.

[9] Anon. *Evaluation of prototype with a License Manager from a company with 22000 employees*. [Interview] Atea Spintop Office with Joel Sanderi, Marcus Sundberg and Hans Andersson. May 11th 2011.

[10] Anon. *Evaluation of prototype with license experts from Atea Main Branch*. [Interview] Atea Spintop Office with Joel Sanderi, Marcus Sundberg and Hans Andersson. May 5th 2011.

[11] Microsoft (2011). *.NET Framework Conceptual Overview*[Online]. Available from: http://msdn.microsoft.com/en-us/library/zw4w595w.aspx [Accessed May 31st 2011]

[12] Microsoft (2011). *Visual Studio 2010 Product Highlights*[Online]. Available from: http://msdn.microsoft.com/en-us/library/dd547188.aspx [Accessed June 2nd 2011]

[13] Microsoft (2011). *Windows Server 2008 R2: Active Directory*[Online]. Available from: http://www.microsoft.com/windowsserver2008/en/us/ad-main.aspx [Accessed June 3rd 2011]

[14] Berenbach, B. et al. (2009). *Software & Systems Requirements Engineering: In Practice*. New York: McGraw-Hill

[15] Lichter, H. & Schneider-Hufschmidt, M. & Züllighoven, H. (1994). Prototyping in Industial Software Projects - Bridging the Gap Between Theory and Practice. *IEEE Transactions on Software Engineering* Vol. 20. (No. 11, November). P. 825-832.

[16] Lloyd, W. J. & Rosson, M. B. & Arthur, J. D. (2002). Effectiveness of Elicitation Techniques in Distributed Requirements Engineering. *Proc. of the IEEE Int. Req. Eng. Conf. (RE)*. P. 311–318.

Appendices

A Software Requirement Specification

The requirements that was provided to ATEA Spintop with the results from the master thesis. The document is divided into two projects due to the extensiveness to implement an agent which meters software usage where adding functionality to utilise SCCM is easier.

Software Requirement Specification

Hans Andersson
Marcus Sundberg

2011-06-03

Project 1 - SCCM additions

Introduction

In this section the purpose and the scope of the project is presented.

Purpose

The purpose of this project is to add functionality to Compliance Manager (CM) which results in additional information about software usage in the SCCM environment. This includes tools that helps with analysis of users and their software usage of chosen programs.

Scope

The project will focus on checking running processes for each user at given times. No information about if the programs are actually used will be given. There will be no functionality that makes its own assumptions and suggests uninstallations, downgrades or any other types of conclusions about software usage. The project only focuses on making tools for helping with usage analysis.

Definitions, acronyms and abbreviations

CM - Compliance Manager.
FileUsageSummary - Table in the SCCM database which stores information about concurrent software users seen in intervals of 15 or 60 seconds.
MeterData - Table in the SCCM database which stores information about software usage: users, computers, programs, start times and end times is some of the information stored which will be utilized by the result of this project.
SAM - Software Asset Management.
SCCM - System Center Configuration Manager.

Overall Description

The people and their knowledge about computer systems which are using Compliance Manager varies a lot. This needs to be taken into consideration when developing the system or adding functionality to CM. The different kinds of end users have been grouped and will be explained below.

In smaller companies, often the person responsible for license management also has other tasks assigned to him or her, which makes the license management a subtask. These people have a wider knowledge about computer systems and often has a more technical background which makes them more suitable to make changes and not as much support is needed for

them. An example of such a person would be an SCCM technician which is responsible for their network setup and has access to and knowledge about the company's SCCM distribution.

In bigger companies, the people responsible for license management and CM are SAM experts. These people are dedicated license consultants. The SAM experts are usually not people with technical backgrounds and are not experts on handling computers.

Knowing the difference between these two types of people which will use CM is important to make sure it can suit everyone. Especially when it comes to the installation and the potential additions that needs to be done in the SCCM database.

General stuff about the data analysis CM additions:
While graphic presentations are nice, they don't give much concrete information. Avoid only having graphic representation. When having graphic representation, make it possible for the user to get more concrete information.

Specific Requirements

General functional requirements which revolves around the SCCM database will have the identifier A and Compliance Manager specific requirements will have the identifier B. The priorities are set as High, Medium or Low.

A-1. Copy meterdata
Requirement: The SCCM database must copy information from meterdata each day.
Rationale: The SCCM database deletes data from meterdata when it is summarized into the table fileusagesummary. The additional information needed for extended software information is stored in meterdata which is the reason why it has to be stored individually.
Dependencies: None.
Priority: High

A-1.1. Store meterdata
Requirement: The modified meterdata must be stored for a set time which is defined by the user.
Rationale: Since the modified meterdata will be growing and take up a lot of space it is important that the user can restrict this and chose how long it should be stored. Data which is considered old and outdated should be deleted.
Dependencies: A-1.
Priority: Medium

A-1.2. Chose programs to store
Requirement: The user must be able to select which programs should be metered in detail.
Rationale: There is no point in taking up space in the modified meterdata with information that is not needed. Only chosen programs should be stored.
Dependencies: A-1.
Priority: Medium

A-1.2.1. Chose programs in CM
Requirement: The user should be able to decide which programs that should be metered directly in Compliance Manager.
Rationale: A user-friendly way to change which programs should be metered is a functionality which is appreciated and save time for a lot of users.
Dependencies: A-1.2.
Priority: Low

A-1.3. Merge meterdata rows
Requirement: The modified meterdata must merge rows which are overlapping.
Rationale: To make sure that all data in the modified meterdata are unique.
Dependencies: A-1.
Priority: Low

A-1.4. Remove short time usage data
Requirement: The user should be able to filter meterdata which are shorter than a specified time interval.
Rationale: Removing short time usage. If the program is only run for say 3 seconds, it should be removed since it is not actually used. The minimum time a program should be run to be a valid session should be specified.
Dependencies: A-1.
Priority: Low

A-1.4.1. Modified FileUsageSummary
Requirement: There must be a table in the database which is a file usage summary on the modified meterdata.
Rationale: If the requirement A-1.4 is fulfilled, a new file usage summary is needed so the correct information is stored.
Dependencies: A-1.4.
Priority: High

B-1. Concurrent users graph
Requirement: The application must be able to show concurrent users of a program in a graph during a specified date with intervals of 15 or 60 minutes.
Rationale: To be able to display a graph over a day is the essential addition to Compliance Manager and will provide a useful tool when investigating software usage. The different intervals is there to give the user the option for how detailed the information should be.
Dependencies: None.
Priority: High

B-1.1. Find maximum date
Requirement: The user must be able to find the date which maximum users are using the chosen program.
Rationale: To easily see which day when most people use the program.
Dependencies: B-1.
Priority: High

B-1.2. Show concurrent users
Requirement: When the graph of concurrent users is clicked, information about which users had the program running during that hour (or 15 minutes) should be displayed.
Rationale: To make the concurrent users graph more manageable and useful towards finding which users are actually using the program at any point in time.
Dependencies: A-1, B-1.
Priority: High

B-1.3. Cycle through dates
Requirement: The user must be able to cycle through the days of the concurrent users graph.
Rationale: To be able to see all days, one by one, as a graph over concurrent users would be necessary to compare different days with each other.
Dependencies: B-1.
Priority: Medium

B-1.4. Show all days
Requirement: The application should be able to show all days for concurrent users in the same graph.
Rationale: To see if the usage is even over a specified interval or if it changes from day to day. A graph with clotted lines would mean more even usage, where a more spread graph would mean that the usage is varied from day to day. It gives an overview over a specified time period.
Dependencies: B-1.
Priority: Low

B-2. Process activity information
Requirement: The application must be able to query the database to show mean time, median time, total time, amount of times used, last time used for each user and a chosen program.
Rationale: Seeing this information in raw form gives a good understanding to how users effectively use up their licences.
Dependencies: A-1.
Priority: High

B-2.1. Sort result data
Requirement: The application must be able to sort the data from the query on all values in the table.
Rationale: To easily find what information you are looking for and users which have certain behavior.
Dependencies: B-2.
Priority: High

B-2.2. Exportable data
Requirement: The data must be shown in a format which is easily exported to Microsoft Excel.
Rationale: There needs to be a quick way of managing the data to be able to summarize and send information to other parties which might be interested in the result.
Dependencies: B-2.
Priority: High

B-2.3. Filter data
Requirement: The user should be able to filter results of the query on several values.
Rationale: To find specific or groups of users who have a behavioural pattern which does not follow the wanted norm.
Dependencies: B-2.
Priority: High

B-3. Most used programs
Requirement: Show pie chart of most used programs based on times started or total time. Can be viewed based on groups of users.
Rationale: This will help show what programs are used most by the given group of users and gives an overview of the essential programs for those users.
Dependencies: A-1.
Priority: Medium

Project 2 - Agent development

Introduction

In this section the purpose and the scope of the project is presented.

Purpose

The purpose of this project is to further build upon the functionality implemented in Project 1 by adding more usage information. While Project 1 only focuses on running processes, this project will focus more on active usage by checking which window is active and logging it.

Scope

An agent needs to be created that logs how much the user actively uses the programs that are running. This will be done by logging active window and user activity. There will be no functionality that makes its own assumptions and suggests uninstallations, downgrades or any other types of conclusions about software usage. The project only focuses on making tools for helping with usage analysis.

Overall Description

For detailed specifics about different users, see Overall Description in Project 1.

To get the agent to work it needs to be installed on each client computer which should be monitored. Before developing such a tool, consideration needs to be taken about how patching, different versions, server connection problems and locally stored reports should be handled. It is also important to note the difference in complexity of this project and Project 1. While Project 1 only adds functionality to existing programs, here a different approach needs to be taken.

Specific Requirements

Functional requirements for the agent that needs to be developed will have the identifier C, server for storing data which the agent collects will have the identifier D and Compliance Manager specific requirements will have the identifier E. Some of the requirements have dependencies from Project 1. The priorities are set as High, Medium or Low.

C-1. Log active window
Requirement: The agent must log the active windows for the local user and create an hour report.
Rationale: In order to measure usage more thoroughly than just looking at which programs are running, this is essential. Example result for a report: Visual Studio 2400 seconds, Explorer 1000 seconds, Word 200 seconds. This report shows that the local user had the visual studio window as the active window 40 min of that hour etc.
Dependencies: None.
Priority: High

C-1.1. Send report
Requirement: The agent must send the log report created by requirement C-1 to a server who stores usage data for all users. When the report is sent successfully to the server, it should be deleted on the local computer.
Rationale: In order to be able to make any analysis, the data needs to be stored in a central place. This is essential.
Dependencies: C-1.
Priority: High

C-1.2. Delete unreadable report
Requirement: If a local report is unable to be read, it shall be deleted from the computer.
Rationale: The reports which can not be read needs to be deleted from the computer to not take up any unnecessary space.
Dependencies: C-1.
Priority: Medium

C-1.3. Log activity in stages
Requirement: The agent should log different stages of passitivity that depends on how long ago the mouse or keyboard was used.
Rationale: In order to filter away times when the user is away from the computer and the computer is idle but also to help analyse different ways a program is used.
Dependencies: C-1.
Priority: Medium

C-1.4. Log computer disturbances
Requirement: The active window logger must check for hibernation and sleep and must save this in the reports.
Rationale: To be able to find disturbances when logging and to not skew the data that is collected.
Dependencies: C-1.
Priority: Low

D-1. Server creation
Requirement: A server which receives all the log files and store the information in a database shall to be created.
Rationale: Having a server which receives the data from the active window reports before storing it in a database is to make sure no false data is stored. It will also make sure only one service is directly connected to the database which makes it more secure.
Dependencies: None.
Priority: High

D-1.1. Data validation on server
Requirement: The server shall check if the data is valid for the database before it is stored.
Rationale: This requirement will work as an extra filter to shield the database from strange data as well as checking that the data is valid.
Dependencies: D-1.
Priority: High

D-1.2. Invalid data error report
Requirement: If the data in a report is invalid the server shall discards the data and store an error report.
Rationale: Storing when something goes wrong is essential when troubleshooting as well as to see if the data can be trusted. If there are many error reports, the data that is stored might not be the whole truth about software usage.
Dependencies: D-1.
Priority: Medium

D-1.2.1. Viewing error reports
Requirement: There should be an easy way to view error reports through the Compliance Manager interface.
Rationale: Enabling license managers to see which users give valid data and which users don't.
Dependencies: D-1.2.
Priority: Medium

E-1. Concurrent users graph - Used
Requirement: The concurrent user graph shall have a new graph-area added upon the already existing one for running processes. This shall show the amount of concurrent users who have have at least once had a window from the program as foreground window during that hour.
Rationale: This will help showing actual usage of the programs and not only everyone who has the program running in the background. This may help identify if there are people who never shuts programs down even if they are not using it. For clarification, see green area in Figure 1.
Dependencies: B-1.
Priority: High

E-2. Concurrent users graph - Used actively

Requirement: The concurrent user graph should have a new graph-area added upon the already existing one for running processes. This will show the amount of concurrent users who have have had a window from the program as foreground window at least X minutes during that hour.

Rationale: Adding another depth to the usage analysis, to identify how many users are actually using the program and not just briefly checks it every once in a while. The X minutes is just an example of where to put the limit, it should be user specified. For clarification, see yellow area in Figure 1.

Dependencies: B-1.

Priority: High

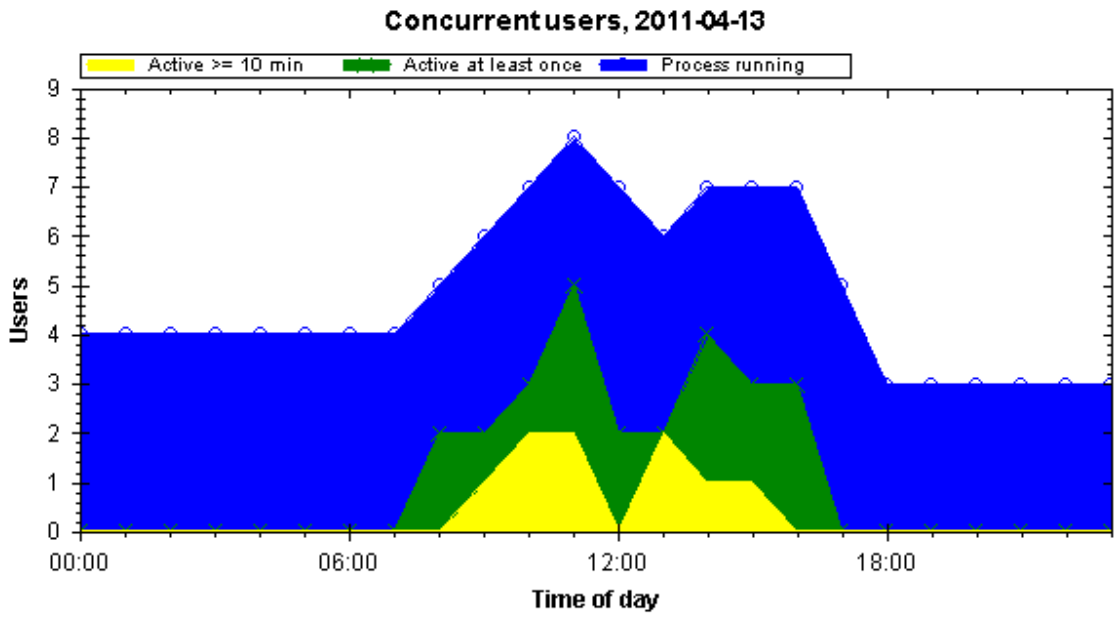

Figure 1: An example of how the concurrent users graph might look. The blue area shows users having the process running, the green area shows users having the program as the foreground window at least once during the hour and the yellow area shows users having the program as the foreground window for at least 10 minutes during the hour.

E-3. Concurrent users - Clickable graph

Requirement: The user shall be able to click the concurrent user graph in order to get specific information about who the users are that are using the program at a given time. When clicking the graph, the results shall show the three different types of concurrent users: Having the process running, actively using the program and actively using the program for more than X minutes.

Rationale: An addition to the requirement B-1.2 in Project 1, which includes more information about finding which users are actually using the program at any point in time. This is to make the concurrent users graph more manageable and useful for license managers.

Dependencies: B-1.2, E-2, E-3.

Priority: High

E-4. Process activity - Active window information addition
Requirement: Builds upon the process activity information from Project 1. Active window time for each user to shall be added to the information that is queried from the database.
Rationale: This will show the relation between how long the process is running and how much the program is actually used actively and will help with identifying users who are running certain programs but not using them.
Dependencies: B-2.
Priority: High

E-4.1 Filter on active window time
Requirement: There must be functionality to also filter the query on true usage activity, namely active window time.
Rationale: To refine the filtering options and make it possible to narrow the search for specific types of users.
Dependencies: E-4.
Priority: Medium

E-4.2. Process activity - Usage activity
Requirement: The process activity information table should show how actively a user is using the program when having the program as foreground window. How much time, in relation to total active usage time, is spent in the different stages of passivity logged by the agent, example time spent in interval 0->5 seconds since last keyboard/mouse action, 5->10, etc.
Rationale: Example: 5->30 sec / 0->60 sec would give a higher number if the program is used less actively. If the the program is used more for reading and not editing, the time between mouse/keyboard-input will be higher and the value for the mentioned formula will be higher. This way, users who only uses a program for reading and not editing may only need a simpler version of that program; Example: Adobe Reader vs Adobe Professional.
Dependencies: E-4.
Priority: Low

E-4.2.1. Process activity - Usage activity type
Requirement: There should exist functionality to change the formula for the data presented in requirement E-4.2.
Rationale: To be able to customise the query results to the customer needs rather than to specify a formula beforehand which might be meaningless.
Dependencies: E-4.2.
Priority: Low

E-5. Process comparison
Requirement: Should be able to compare how different groups of programs are used by using the information of passivity from requirement C-1.3.
Rationale: This can help in the analysis of how users use groups of programs. An example can be comparing different programs to see if the cheaper alternative is used in the same way as the other and then possibly drawing the conclusion that it can be used instead of the more expensive one.
Dependencies: C-1.3.
Priority: Low

E-5.1. Process comparison - Grouping programs
Requirement: Should be able to specify groups of programs manually directly in Compliance Manager.
Rationale: Customising which programs should be grouped together is essential to make requirement E-5 usable. An example of this can be grouping web browsers together to see how actively the common users use them.
Dependencies: E-5.
Priority: Low

E-5.2. Process comparison - Grouping users
Requirement: Should be able to group users from Active Directory groups directly in Compliance Manager.
Rationale: If a certain work group with certain types of assignment uses the programs differently than another group of users, there might a cheaper alternative one of the groups.
Dependencies: E-5.
Priority: Low

E-6. Most used programs
Requirement: Show should used programs based on total time that the program has been the foreground window for a given group of users.
Rationale: Information about most used programs is needed to see if groups of users actually use their licensed software and how much they use it actively. It can be shown as a pie chart.
Dependencies: C-1.
Priority: Medium

E-7. Process activity - Pie chart
Requirement: There should be a pie chart displaying a user's or a group of users' activity over all programs.
Rationale: The active time of the users shows how actively they use their computers, which can be good information to have.
Dependencies: C-1.
Priority: Medium