UNIVERSITY OF OSLO
Department of Informatics

User Survey of Practices Using Cfengine

Master thesis

Raheel A. Chaudhry

6th June 2005

Abstract

This Master thesis is the final project at Oslo University College which is a part of the international Master degree in Network and system administration, in collaboration with the University of Oslo. This thesis includes a literature review covering the important topics and methodologies. The starting point of this thesis is a user survey of practices using cfengine, performed within cfengine users. A questionnaire was developed and published to receive participants' feedback. This made it possible to present and analyse the received data in an analytical manner. The result of the analysis may help develop a new version of cfengine.

Acknowledgements

This thesis is written for the Master of Science Degree at Oslo University College in collaboration with University of Oslo.

First of all, I want to thank God for giving me patience and strength to complete this Master thesis.

I would like to thank my advisor, Kirsten Ribu for having faith in me and supporting me through the period of project. I would also like to thank Mark Burgess for his co-operation through the Masters course, for sharing his experience and ideas with me and letting me use some of his figures. Both of them really know how to turn stone into gold.

I want to thank, Frode Eika Sandnes, Kyrre Begnum, Siri Fagernes, Hårek Haugerud, Simen Hagen and Tore Møller Jonassen for all the support through the Masters degree.

I want to thank my classmates for the feedback and co-operation during these last two years. Especially, Ole Arild Rindalsholt for being one of my best friend, and guide through the five last years.

I also want to thank all of my friends for all support and for understanding me when I was sour and grumpy last months.

I also want to thank the users of cfengine who took some time to participate in the questionnaire.

Most of all, I want to thank my mom and dad for supporting me, cooking for me and praying for me through all my life, I can never thank them enough.

Oslo June 2005

Raheel A. Chaudhry

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is written as the last project in the Master of Science Degree at Oslo University College in collaboration with University of Oslo. The aim of this thesis is to do a user survey of practices using cfengine.

Cfengine was developed by Mark Burgess, a professor at Oslo University College. He is planning to develop a new version of cfengine. He wants some feedback from users of cfengine to have some information before he can start his work. The survey is a mixture of qualitative and quantitative results. It is important to ask smart questions, so we can get users interest and simultaneously get good feedback. We want to find out: how easy cfengine is to use or to learn, how good it is at solving the problems of configuration and maintenance. Cfengine is special, in that it implements a set of principles. We want to find out if users really do understand these principles and why they are important.

First of all we must try to define what software quality is. Quality is more like a concept than a single idea. By this term we mean what a developer of software should think of when developing software:

- Capability
  - Ability to do what it is developed for
- Usability
  - Possible to use it in the way it is made for
- Performance
  - Ability to achieve
- Reliability
  - Ability to be trusted
- Installability
  - Easy to install and implement
- Maintainability
  - Possible to maintain
- Documentation
  - Good documentation – easy to read and understand
- Availability
  - Easily accessible

But if all of the points above are completely considered, it will be software with high quality. There is no perfect and faultless software. Users have important influence, when developing software, because they are the one who will use it. Price, performance, reliability and satisfaction are the point to be noted here. So, users' requirement must be analyzed before developing a product.

Cfengine is a tool for maintain a heterogeneous network. Cfengine embraces a stochastic model of system evolution. That is one of the main reasons why cfengine is different from other tools in configuration management. The most configuration agents either want a human being to make a change or rewrite the same constant configuration many times. Cfengine's configuration approach is to always move the system closer to an ideal state. An ideal state is defined as, when a system compiles with policy, it is "healthy", when it deviates, it is "sick".

Cfengine has a certain number of components.

- Cfagent
  - An autonomous configuration agent
- Cfservd
  - A file server and remote activation service
- Cfexecd
  - A scheduling and report service
- Cfenvd
  - An anomaly detection service
- Cfenvgraph
  - A help tool for cfenvd
- Cfkey
  - Key generation tool
- Cfrun
  - A tool for executing one or more remote agents
- Cfshow
  - A tool for showing the contents of the internal databases used by cfengine in its operation

A cfengine agent is run on every host on the network, so a class structure is possible. Every host which runs a cfengine agent makes a list of its attributes/classes which the host belongs to.

There are three common types to obtain feedback from users about their satisfaction with the product; face-to-face interview, telephone interview and questionnaires. In this user survey, we will use last one. The reasons are, it costs less than other methods and we can use a web-page to administer it. The aim is to get answers form the most users of cfengine as possible. Their response will be analyzed. Most of the questions are yes/no questions, where we will get numbers to deal with. Some of the questions require users to write down some sentences. In the end we will try to draw a conclusion of what users are satisfied whit and what thy are dissatisfied with.

Chapter 2

Background

2.1 Cfengine

Cfengine is an open source system administration tool. It is an abstract programming language for system administrators of huge heterogeneous environment. With cfengine, system administrators have an easy way to maintain complicated networks.

The founder of cfengine is Mark Burgess, professor at Oslo University College. He started with this project in 1993, and worked with it since than. We already have two versions of cfengine, cfengine 1.0 and cfengine 2.0. Burgess is planning to develop a new version of it. It is essential for him to do a user survey of practices using cfengine. This way he can get important feedback from user. From the feedback he for instance pick out he weaknesses and improve them in a new version

2.1 Motivations

This project was presented by Mark Burgess early in October. I made an early decision to take this project when I first read about it on the College's web-pages. I thought this might be the right project for me since I was interested in Cfengine and would more than willing to learn more about this tool and its features and components. From earlier courses at Oslo University College, we had an introduction in cfengine, but that was all. I was already impressed in Mark Burgess' work, who had used so many years in researching and developing.

A questionnaire seemed very interesting, this way we could come in touch with people who really are using cfengine. Their experience with the tool would be very important for this thesis and of course for Mark Burgess who is the founder of this tool. This would also be a good experience for me.

In the start I had to do research, to find relevant papers written on the topic. I got a hint from Mark Burgess to visit *http://www.cfengine.org*. On this web page I found a lot of relevant information regarding to cfengine and some published papers. Most of the papers were written in English. It was very easy to get the overview after reading the papers. It was very important to pick out the most important and relevant papers from the web and read them more carefully to get the big picture.
The web page had its own forum, where users could discuss their experiences with the tool. Some of the threads are interesting, because some

of the users had same problems. This forum could help a lot to formulate some of the questions in the questionnaire.

In this project on must also think of software quality. It seemed exciting to find out how users would evaluate cfengine and to find out if the really understand all the features of cfengine and its purpose and philosophy.

Such a survey of cfengine were never done before, so that was a motivation of it self. Mark Burgess tried once, but it was never completed. We knew that we had to ask smart questions in the questionnaire, to get users interested and get good information – information we can analyze.

This user survey would help Mark Burgess to improve or maybe develop a new version of cfengine. It is important for Burgess to get some feedback; it would help him a lot in improvement.

2.2 Aims

Cfengine is a powerful tool for a system administrator, if all of the features, purpose, and philosophy of it are completely understood. It is a configuration management tool that is widely used. This project considers the design of a questionnaire about user practices with cfengine and collecting data from users. Results will be classified and then presented geographically and for large/small organizations etc in order to correlate practices. The aim of this project is to find the user satisfaction.

There are some questions Burgess wants answers to. The biggest aim of this project is to get answers to these questions. The "big" questions are:

- How easy is it to use/learn?
- How good is it at solving the problem of configuration and maintenance?
- Do users really understand the principles of cfengine?
- Do they feel safe using cfengine?
- How easy is to express own policy ideas?
- What improvements are needed in cfengine?

Answer to these questions will show us what kind of users we are dealing with. It will be an easy way to find out if the online documentation and support for the tool is useful or not. It was important to develop smart questions that gave us some result we could analyze and work with. A checklist was made for the questions:

- Does the question fit into a model for testing the software?
- Does the question fit a model for managing the network?
- Can I use the result of this question to draw a conclusion?
- Can I get meaningful statistics, or only qualitative comments?

The most important thing was to get meaningful statistics, so the result could be classified. It is a lot easier to work with numbers than comments. This survey can help Burgess to improve the tool and the documentation.

2.3 What Do I Expect

I assume that the most of the users use cfengine in an ad hoc way, to solve a random selection of issues. I expect some to also use it to lay out plan for the whole system systematically. Cfengine has a feature that deletes for instance junk files, lot users probably use the feature for garbage collection of files and processen. Cfengine has a lot of components, like *cfagent, cfservd, cfrun, cfenv, cfenvgrapg,* etc. Expected use of each component is:

- Cfagent – almost everyone uses it
- Cfservd – almost everyone uses it
- Cfexecd – a lot uses it
- Cfrun – not common
- Cfshow – not common
- Cfenvd – not common
- Cfenvgraph – not common

Cfenvgraph can give you the trends, but not many users prefer to use it. It should be a lot better if it was possible to se cfenvgrap data via web. You can't use cfengine properly without Cfagent and cfservd.

- Users should use import to break up the *cfagent.conf* configuration and split into several files
- The configuration instructions should be grouped by the role of the host in the network
- The smartest way to solve cfengine problems is by using shell commands embedded in cfengine.
- The amount of configuration files depends on the size of a organization/company
- Terms like adoptions or adaptive lock care important to understand.

All users of cfengine should understand its purpose and philosophy. It's no point in using such a tool without understanding the main purpose. It's also important to understand expressions like convergence, the cfengine language interface and classes. Basic functionality of the components should be understood before using the tool. Policy regarding to security is also important – the policy cfengine implements.

It should be easy to learn cfengine, it's not that complicated. Reading the documentation and help manuals can be an advantages. The documentation has a good quality and it is easy to read. It covers the most important topics. However, I don't think many users have taken any course for understanding the cfengine according to the mailinglists. But I rather expect almost every

user have read the textbooks about system administration and cfengine. The web and discussing groups are a primary source for information. I assume that the discussing groups are used frequently.

Users should trust the engine that it does what you asks it to do. Features like pull rather than push should not be a problem to solve a file distribution problem. Every user should know:

- Complexity is a potential security risk.
- It’s not necessary for a human to monitor hosts all the time.
- The organization of a policy doesn’t have to be sequential.
- Networks should be organized hierarchically.
- Simplest is usually best.
- It is not necessary to specify every detail of a machine's configuration.
- Machines in a network should be as similar as possible
- Users must sacrifice some freedoms for the good of the networks

It will be interesting to know if the users think cfengine could be made easier to adopt. It will also be interesting to find out what features users think are missing in the current version of cfengine. It will be important to Mark Burgess to find out which changes users will prefer in a new version of cfengine.

Chapter 3

Literature Review

3.1 Software Quality and Analysis of Data

3.1.1 Introduction

"Quality must be defined and measured if improvement is to be achieved" [10]. It is very easy to misunderstand the term quality. The reasons can be that people think differently when they try to define the word. Quality is more like a concept then a single idea. When we talk about quality we can refer it in its broadest sense or just refer to its specific meaning. Quality is a word we use almost daily without really thinking over what we really are trying to say, because the popular and professional use of it can be different.

The word quality can be discussed, felt and judged, but can't be weighed or measured [10]. In this chapter we will discuss software quality. By this term we mean what a developer of software should think of when developing software.

- Capability
  - Ability to do what it is developed for
- Usability
  - Possible to use it in the way it is made for
- Performance
  - Ability to achieve
- Reliability
  - Ability to be trusted
- Installability
  - Easy to install and implement
- Maintainability
  - Possible to maintain
- Documentation
  - Good documentation – easy to read and understand
- Availability
  - Easily accessible

3.1.2 Software quality

By experience we know that a lot of software often contains "bugs". It's a great challenge to develop software without any kind of functional defects. We have two ways to express the definition of software quality:

- Defect rate
- Reliability

By defect rate we mean the number of defects per million lines of source code, per function point, or other unit. And by reliability we mean number of failures per *n* hours of operation, mean time to failure, or the probability of failure-free operation in a specifies time [10]. When we want to find out about the user satisfaction, we usually use user satisfaction surveys. IBM for instance monitors satisfaction with its products in level of CUPRIMDSO (capability, usability, performance, reliability, installability, maintainability, documentation, service and overall) [11]. These points are emphasized differently by different users. Some users are more interested in good documentation, or the software should be easy to install etc. But users with sophisticated networks, performance and reliability are the most important factor. Figure 1 shows a relationship between the different quality attributes. Some relationships are supportive, some are negative, and some are not clear because of the types of users and applications [10]. A lot of software has different type of users and therefore there is difficult to set goals for the quality attributes. Am mentioned in the article [12], 15% or more of all software defects are requirements errors. Software will always be poor-quality software if the development process does not address requirements quality.

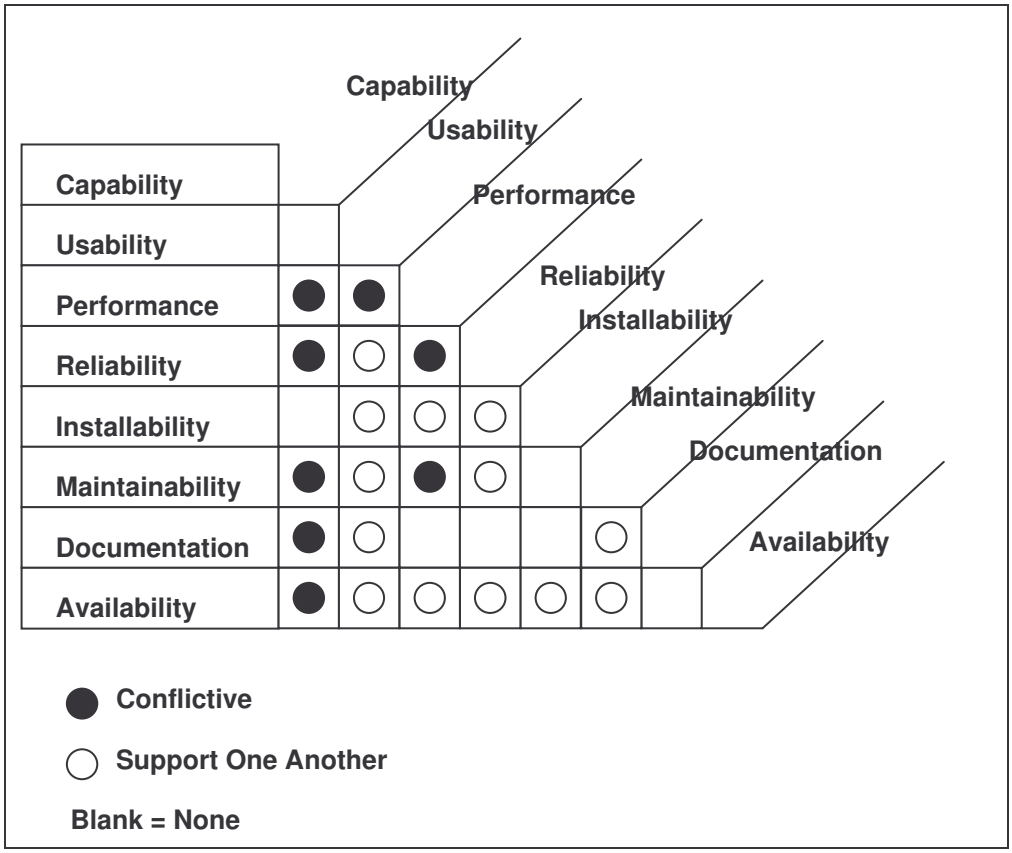

Figure 3.1.1: Interrelationship of Software Attributes – A CUPRIMDA Example [10]

Another important thing about software quality is the process quality compared with end-product quality. The development process includes some states. Each state is dependent on the preceding state. Each set has quality attributes that can affect the end product.

3.1.3 User's role

It is difficult to decide the role of the user, when relating to quality. The points a user emphasizes are:

- Price
  - This is an important factor when buying software. Do we really need this software? Do we really afford this software?
- Performance
  - Are we pleased with the performance of the software? Did we get what we want?
- Reliability
  - Is this software reliable? Reliability is also an important factor.
- Satisfaction
  - Are we satisfied with the software? Does this software fulfil our needs?

In Guasparis book *I Know It When I See It* [13], he discusses quality in the user's context as follows:

> *"Your customers are in a perfect position to tell you about quality, because that's all they're really buying. They're not buying a product. They'rebuying your assurance that their expectations for that product will be met.And you haven't really got anything else to sell them but those assurances.You haven't really got anything else to sell but quality."*

When making software, users' requirements must be first gathered and analyzed; specifications to meet those requirements must be produced. The product must be developed accordingly [10]. The whole process of developing software will contain different phases. Errors can occur in each phase that can affect the quality of the finished product. Reasons for errors can be:

- Bad planning
  - Proper planning is very important when developing a software
- Erroneous requirements
  - Determine requirements before developing software
- Human error
  - Human errors are almost unavoidable

Users are most interested in a product that conforms to requirements and is fit to use. From the producer's perspective, once requirements are specified, developing the product in accordance with the specifications is the path to achieving quality [14]. Important points for good quality are the lack of defect and good reliability. The definition of quality consists of two levels [10]:

- Level 1: small *q* (*q* for quality)
    - Refers to product's defect rate and reliability
- Level 2: big *Q*
    - Includes:
        - Product quality
        - Process quality
        - User satisfaction

This approach was not in practice before late 80's. User satisfactions were not the most important factor and product requirements were decided without user input. That was the reason why the products were not what users wanted. Users' role is important when developing a good product.

3.1.4 Data collection

There are different ways to obtain feedback from users about their satisfaction with the product. We have three common methods to gather survey collection [10, 17]:

- Face-to-face interviews
    - The answers must be recorded
    - The data has high degree of validation
    - Less misunderstandings about the questions being asked
    - Interviewer can affect the result
    - Cost a lot more than the two other methods
- Telephone interviews
    - Less expensive than face-to-face interviews
    - Supervisors can monitor the interviews and ensure that the right interview procedure is followed.
    - A computer-system can reduce costs and increase efficiency
    - Should be short and impersonal
- Self-administered questionnaires
    - No interviewers are needed
    - Cost less than the two other methods
    - Chances of getting response are lower than the two other methods
    - Questions must be carefully constructed, validated, and pretested
    - Development of questions requires professional knowledge and experience

Type of Survey	Cost	Sampling	Response Rate	Speed	Flexibility	Observ-vations	Length of Interview	Validity
In Person	- -	+ -	+ -	+ -	+ +	+ +	+	+ +
Phone	+	+	+ +	+	+	-	-	+
Mail	+ +	-	- -	-	-	-	+	-

- = Disadvantage + + = Best
- = Worst + - = Could be an Advantage or a Disadvantage
+ = Advantage

Figure 3.1.2: Advantages and Disadvantages of Three Survey Methods [16]

Figure 3.1.4 shows advantages and disadvantages of the three survey methods with regard to number of attributes.

3.1.5 Analyzing the data

A common satisfaction scale is five-point satisfaction scale:

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

This satisfied scale is often used in user satisfaction survey. The data we get from the user is usually summarized in percent. The result is often represented by charts and bar charts to show the trend of percent satisfaction. Percent satisfied is the most used metric, companies such as IBM choose to monitor the inverse, the percent nonsatisfied [11]. One of the advantages of monitoring user satisfaction is that data we collect can provide information for improvement. The results we get from users often indicate strength and weakness of the software product.
When analyzing and presenting user satisfaction survey data, the confidence interval and margin of error should be included. Usually, the 95% confidence level is used for forming confidence intervals and the 5% probability is used for significance testing.

Expected Satisfaction	80% Confidence		85% Confidence		90% Confidence		95% Confidence	
	+/- 5%	+/- 3%	+/- 5%	+/- 3%	+/- 5%	+/- 3%	+/- 5%	+/- 3%
80%	104	283	133	360	171	462	240	639
85%	83	227	106	289	137	371	192	516
90%	59	161	75	206	97	265	136	370
95%	31	86	40	110	51	142	72	199

Figure 3.1.3: Examples of Size (for 10,000 users) in Relation to Confidence Level and Error Margin [10]

Figure 5 illustrates the sample size for 10.000 customers for various levels of confidence with both 5% and 3% margins of error. A point to be noted is that the required sample size decreases as the user satisfaction level increases [17].
Good analysis is most important in transforming data into useful information and knowledge. In satisfaction surveys, satisfactions with specific quality attributes of a product are often queried, in addition to overall satisfaction. Attributes with lowest levels of satisfaction should not get the highest priority for improvement.

3.1.6 How Good Is Good Enough?

How much user satisfaction is good enough? The goal for each software developer should be 100%, total user satisfaction. Here we have some questions that should be answered:

- Should my company invest $5,000,000 to improve satisfaction from 85% to 90%?
- Given that my company's customer satisfaction is at 95%, should I invest another million dollars to improve it or should I do it later?

Answers to the questions lies in relationship between user satisfaction and market share. A basic assumption is that the satisfied users will continue buying products from the same company and dissatisfied users will most probably buy from another companies. So, as long as market competition exist, user satisfaction is very important to keep customers. Even in monopoly markets, user dissatisfaction encourages the development of competition. Studies of business have lent strong support to this assumption [16, 17].

From Babich's simple model and examples, the answer to the "How good is good enough? [16]" is simple: A company who wants to survive must be better than competitors. The important thing here is to not only measure one's own customer satisfaction but also the satisfaction of one's competitors. A good customer satisfaction management process must

include measurement, analysis and actions. Such a process should cover the following elements[10, 16, 17]:

- Measure the overall customer satisfaction over time, one's own as well as competitors.
- Perform analyses on products strengths, weaknesses, prioritization, and other relevant issues.
- Set satisfaction targets by taking competitors' satisfaction levels into consideration.
- Formulate and implement action plans based on the above.

3.2 Cfengine

3.2.1 Introduction

Cfengine is a tool for setting up and maintaining computer systems. GNU cfengine is an abstract programming language for system administrators of huge heterogeneous networks. With cfengine, system administrators have an easy and elegant way to maintain complicated networks [2].

Cfengine is a distributed agent framework for performing policy-based network and system administration, used on hundreds of thousands of Unix-like and Windows systems[1]. It is a UNIX administration tool that aims to make the easy administrative tasks automatic, and the hard tasks easier. Its goal is system convergence from any state towards an ideal state. According to its author, Mark Burgess, cfengine always brings your system closer to the configuration you have defined; it never makes it worse [1, 2, 4].

3.2.2 Purpose of Cfengine

In general, Cfengine is a way of doing Network Information System (NIS) types of things like keeping `/etc/passwd` and `/etc/shadow` files, distributing a master file system, making sure all systems have a standard set of GNU utilities and a working GNU Compiler Collection (GCC) environment, fixing those `/etc/init.d` links which always seem to get deleted, and so fort [1, 2]. It will do all this across far-flung machines which may have different operating systems, and will do it without the dependencies of a Network File System (NFS) server or NIS master. The reason is that it's all done in files, in guaranteed transaction, and the client machines retain local copies of the files between updates [1].

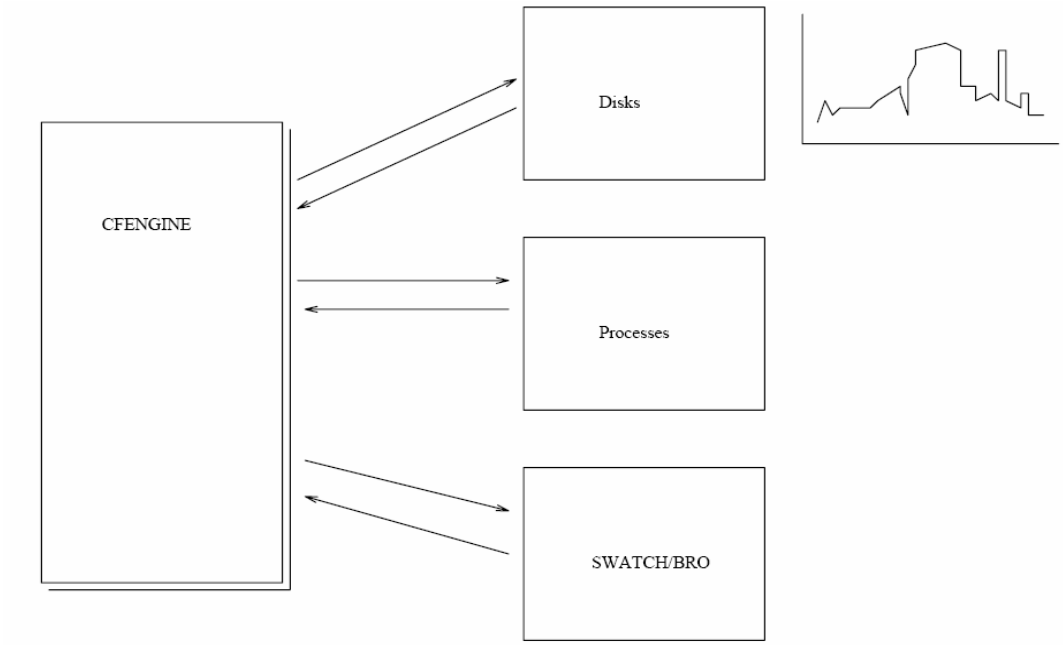

Figure 3.2.1: Cfengine communicates with its environment in order to stabilize the system. This communication is essential [3]

Stability:

- Defined as resistance to unintended changes
- Convergence is implemented through sets of rules
- Rules will try to move the system to an ideal state, for instance
    - recreate symbolic links every time cfengine runs
    - *init.d* startup scripts can be copied from a trusted repository, whenever the local copies are modified
- Processes can be restarted if they are not running according to cfengine

Reliability:

- Ability of a machine to survive when problems occur
- A disk failure is for instance a reliability test
- A system can be near the ideal state when conferencing to an ideal state
- Convergence simplifies the reliability, when reliability is not executed by cfengine alone
- Stability is important in relation to reliability, because a stable system will not easily be affected by problems
- Cfengine's convergence makes it possible to move the system to a desired state

Desired and ideal states:

- Does it only exist one desired state?
- There is no ideal state for all machines in reality

- A machine is classified by:
    - Location
    - Task
    - Connectivity
    - Users
    - Operating systems
- These points does not have equally importance for all machines
- Each machine will have their own ideal state
- Cfengine's job is to find this ideal state and converge to it.

Predictability

- Ability of a machine to act as expected
- Convergence takes care of predictability by making a system stable and reliable
- We expect that a new machine behaves just like the old one it replaced – after it has converged to an ideal state
- Software we are using for system can expect them to be in a ideal state
- System resource should be in a predictable state – should not treat every system as hostile unknown territory [1, 3, 4].

3.2.3 Cfengine philosophy

The following phrases are used in discussing cfengine [1, 5].

Ad hoc: following no predefined pattern. Ad hoc is a self-organizing network of peers.
Alphabet: a set of independent symbols that belong together.
Arrival process: The arrival of different kinds of events over time is called an arrival process. For instance, the arrival of network traffic, or the arrival of instructions to a computer. It is generally assumed that the arrival of events is random.
Autonomous: A host or peer system is autonomous if the policy does not originate from another host peer.
Coarse-graining: this is the process of taking a detailed viewpoint and eliminating detail to form coarser, less specific classes. Coarser grains are like black box version of parts of a system.
Digitization, classification: This is a form coarse-graining in which an observed information stream is reduced into classes of known extent.
Information, complexity: this is a precise theoretical meaning in terms of the average entropy of a digital observation.
Graph: a number of nodes connected together by links or edges [1].

"*Cfengine embraces a stochastic model of system evolution*" [1]. That is one of the main reasons why cfengine is different from other programs in configuration management. The focus of cfengine, and supporting work, is the willingness of accept the idea of increasing random entropy of

configuration through interaction [1]. Cfengine has some policy and a set of principles that refers to an immunity model. The aim is to get a correct/optimal configuration. One of the risks is users. Their behaviour can disorder the system configuration over time [3, 6]. These embody the following features:

- "Centralized policy-based specification, using an operating system independent language, which conceals implementation details.
- Distributed agent-based action in which every host node is responsible for its own maintenance.
- Convergent semantics encourage every transaction to bring the system closer to an "ideal" average-state.
- Once the system has converged, action by the agent desists, or more usually, does not even start at all, when convergence was assured on a previous run of the agent" [1]

The last two points are the most important. The most configuration agents either want a human being to make a change or rewrite the same constant configuration many times. Cfengine's configuration approach is to always move the system closer to a ideal state [1, 2]. An ideal state or a "healthy" state is defined by Mark Burgess [1] as, when a system compiles with policy, it is healthy; when it deviates, it is sick. Cfengine makes this process of "maintenance" into an error-correction is meant in the sense of Shannon [7].

3.2.4 Components of the cfengine

Here are the components of cfengine [1, 2]:

cfagent: An autonomous configuration agent
cfservd: A file server and remote activation service
cfexecd: A scheduling and report service
cfenvd: An anomaly detection service
cfenvgraph: A help tool for cfenvd
cfkey: Key generation tool
cfrun: A tool for executing one or more remote agents
cfshow: A tool for showing the contents of the internal databases used by cfengine in its operation

Cfagent:
Cfagent runs on every host and parses a file-set so the configuration of the host is checked against this file. If desired, any problems will be fixed. *Cfagent* also performs security checks as installing and repairing the configuration.
Cfservd:
Cfservd is a file server which starts the cfengine remotely. It also has the control over access that is based on RSA authentication and IP address. The server daemon is controlled by a file called `cfservd.conf`. The syntax of this configuration file is deliberately modelled on cfengine's own configuration file, but despite the similarities, they are separate.

Cfexecd:
Cfexecd is a wrapper for execution of cfagent. It sends the output of cfagent as mail to for instance the system administrator.
Cfenvd:
Cfenvd has the responsibility for the anomaly detection. It is a part of *cfagent*. It provides the agent with information. Cfenvd is optional so it don't need any configuration.
Cfenvgraph:
Cfenvgraph is a tool for *cfenvd*. It makes graphs of the system performance both before and after a possible anomaly.
Cfkey:
This module generates public-private key for authentication.
Cfrun:
Cfrun executes one or more remote agents. It contacts *Cfservd* which starts an authorized agent. So *cfrun* can not send instructions direct to an agent.
Cfshow:
Cfsow shows the contents of the database which is used by cfengine. Usually you can dump this to a text file.

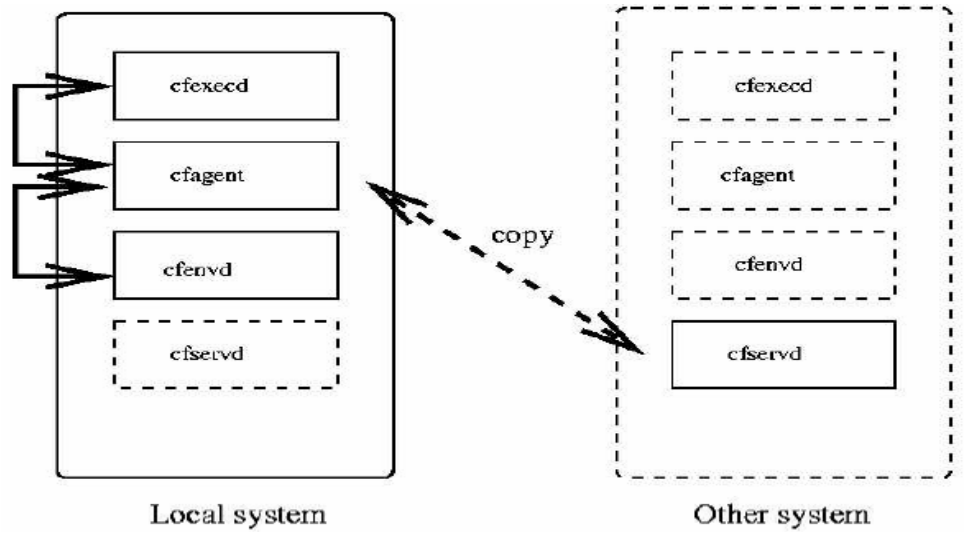

Figure 3.2.2: Cfengine components [1]

Functionality of the components is now described, but I will now go in more details of the functionality of *cfagent*.

3.2.5 Functionality

The notes above give us a idea of what cfengine can be used for [8].

- Configuring network interface.
  - It checks and configures the network interface
- Edit textfiles
  - Edits textfiles for all users and for the system
- Make symbolic links

    - It makes symbolic links and multiple links from a single command
- Permissions
    - Checks and sets permission and ownership of files
- Deletes junk
    - It gets rid of junk files which can disorder the system
- Mounting
    - Systematic automated mounting of NFS filesystems
- Presence of files
    - It checks the presence of important files and filesystems.
- Execution of scripts
    - Controlled execution of user scripts and shell commands
- Structure
    - Cfengine follows a class-based decision structure.
- Process management.
    - Kill irrelevant processes

Cfagent can be run as a *cron* job or manually. One can run cfagent as many times as you like. The engine determines if there is something to do, every time when cfagent is ran. That means if every thing is ok, nothing will be done. If you use cfagent to configure a whole network, then you should run cfengine often with help from cron or cfexecd.

Table 3.2.1 shows a list over all functionality of cfengine and which component does what. As we can se from the table *cfagent* has a lot more jobs to do than other components. That makes it the most important component and other components are dependent of it.

Functionality	**Component**
Configuring network interface	Cfagent
Edit textfiles	Cfagent
Make symbolic links	Cfagent
Permissions	Cfagent
Delete junk	Cfagent
Mounting	Cfagent
Presence of files	Cfagent
Execution of scripts	Cfagent
Structure	Cfagent
Process management	Cfagent
Security checks	Cfagent
Authentication	Cfservd
File copying	Cfservd
Execution of cfengine	Cfexecd
Send mail to administrator	Cfexecd
Anomaly detection	Cfenvd

Graphs (system oerformance)	Cfenvgraph
Generate keys for authentication	Cfkey
Execution of remote agents	Cfrun
Showing contents of the database	Cfshow
Checksum verification	Cfagent

Table 3.2.1: Functionality-table of the components

3.2.6 State diagram

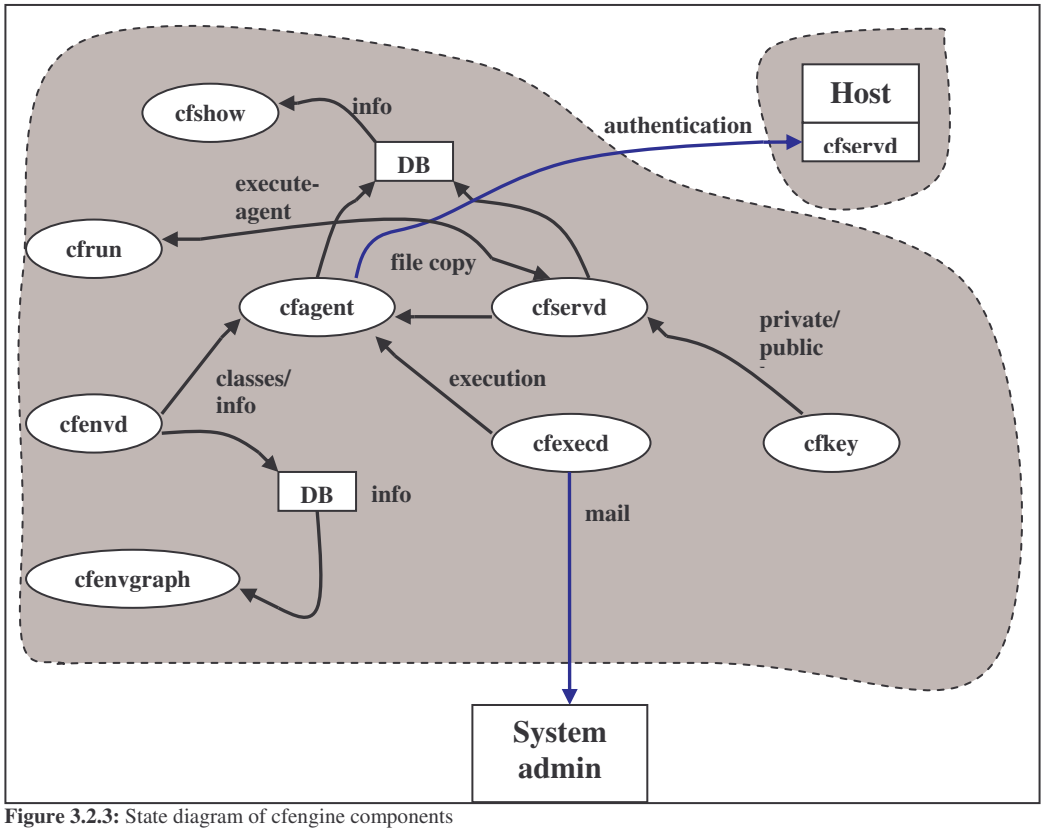

Figure 3.2.3: State diagram of cfengine components

This state diagram shows the activity between the different components of cfengine. Here we can see how the components communicate among each other. A host will be authenticated by *cfservd*, when a user logs into the system. There is communication between *cfservd* and *cfagent*, files are sent from *cfservd* to *cfagent*. *Cfkey* generates public/private keys for users and sends information to *cfservd*. *Cfagent* and *cfservd* sends information to a database. *Cfshow* shows this information or contents of the database. This information can be dumped to a text file. There are also communication between *cfservd* and *cfrun*. If we want a immediate change or want something to run faster, there is a opportunity to execute a remote agent. *Cfrun* contacts *cfenvd* when a new agent is to be started. *Cfenvd* provides the agent with information and classes. It sends the information to a database to which is collected by *cfenvgraph*. *Cfexecd* executes the agent and sends the output of *cfagent* as mail to system administrator. As figure 3.2.3 shows, there is a cfengine installed on the host to. So there are two

machines on the picture. Everything I have explained here is done in the "host" too.

3.2.7 Classes

As mentioned in Mark Burgess' paper "A brief overview of the implementation of principles of system administration in cfengine", "*cfengine uses the idea of host classification to dissect a distributed environment into overlapping sets*" Figure 3.2.4 shows this overlapping:

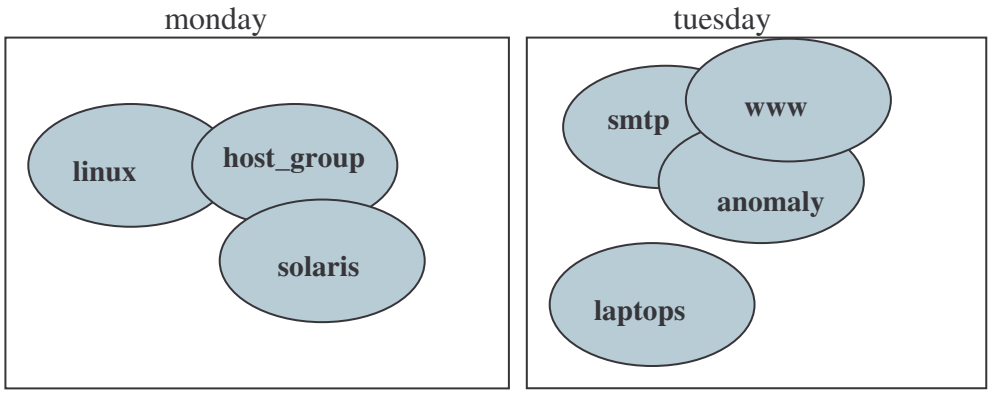

Figure 3.2.4: Overlapping classes[1]

As mentioned, a cfengine agent is run on every host on the network, so a class structure is possible. Every host can find out whether it belongs to a given group or not, because it knows its own name and the operating system it is running. Every host which runs a cfengine agent makes a list of its attributes/classes which the host belongs to [1]. Important classes are [1, 8]:

- Machines identity
  - Hostname
  - IP-address
  - Network it belongs to
- The operating system it is running
- The group it belongs to
- Time and date
- Strategy
- The logical combination of:
  - AND (.)
  - OR (|)
  - NOT (!)

Since the environment is very large, so it is very difficult to describe it precisely. That's why cfengine classifies it so it will be suitable. If an agent is running on a host, it can pick everything it needs from the global policy, since policy is also marked with the classes.

3.3 Summary

In this chapter we have discussed the definition of software quality and customers role. Quality must be measured and defined for improvement and is best defined as “conformance to customers’ requirements”. In software, operational definition of quality consists of two levels: the product quality (small *q*) and customer satisfaction (big *Q*).

There are various methods to measure customer/user satisfaction. The three common methods of survey data collection are, face-to-face interview, telephone interview, and mailed questionnaire. Each method has its advantages and disadvantages. The results are often presented in percent.

Cfengine is an open source tool for setting up and maintaining a network of a computer system. Cfengine consist of a number of components. Each component has its own functionality. *Cfagent* is the most important component, because it runs on every host and has the most important functionality. Cfengine’s functionality makes it easy for a system administrator to maintain a network.

Components communicate between each others too. A component called *cfenvgraph*, shows a graphical presentation of system behaviour. This makes it easy to se the changes both before and after a possible anomaly. This can be a very useful component for a system administrator. Cfengine embraces a stochastic model of system evolution. That is one of the main reasons why cfengine is different from other programs in configuration management. Cfengine has a set of policy that refers to an immunity model. The aim is to get an optimal configuration.

Chapter 4

Methodology

4.1 Survey Method

There are three most common types of survey methods [10, 16].

- Face-to-face interviews
- Telephone Interviews
- Self-administered questionnaires

The last one, self-administered questionnaire was chosen without any kind of doubt. First of all, this method of doing survey was the cheapest. Another advantage is that questionnaires are easy to analyze. Data entry for almost all surveys can be easily done by software packages. Most of the people have completed some kind of questionnaire before, so they have experience with it. Questionnaire can both give us qualitative and quantitative feedback.

Some disadvantages can be that it can be lower response rate than other methods, because it can end up in people's junk-mail. Low response can lead to poor statistical analysis. A questionnaire must include some area for comments. So, participants can qualify their answers. Another disadvantage can be error margins like, misunderstanding of the questions or just guessing [16].

A questionnaire should be designed clearly and easy to follow [18]. Some participant may just don't want to answer if they don't see the meaning of it. So the aim of the questionnaire should be clarified with some few sentences in the beginning. A start text should also include some instructions of how to complete the questionnaire [10, 18]. These instructions should be easy to understand. Spelling mistakes must be avoided. The language should be simple and direct. It is very important to hold participants interest. First impression is very important. A questionnaire should only include relevant questions, so participants don't feel that they are answering some junk. The questions must be carefully constructed, validated and pretested [16].

Since the users of cfengine came from a wide geographic are, this would be the best method. By using this method, no interviewers are needed. Users of cfengine received a link to the web-page where the questionnaire was placed. The plan was to wait some weeks and then send them a reminder if we don't get enough participants. As starting point, we would be pleased with 20 participants. But when I was told by Mark Burgess that there are about 300 companies/organizations using cfengine, we were expecting a lot more participants.

The questions should be of good quality. It is important that the questions are objective. Non-objective questions can lead to force participants to answer the way you want. Another way is to add some alternatives, where participants can pick the alternative that fits them best. We must have in mind that we want participants answer, what they think. Some questions can include sensitive data about for instance a company/organization. In this case we have to clear the policy on confidentiality [18]. We don't want participants to feel threatened in any way. Questionnaires with multiple choice questions are most common. These types of questions are easy to answer for participants and easy to analyze for a researcher. Rating questions on a scale form for instance 1 – 5 is also an alternative. But some things are harder to rate than other [16]. And that ends up with rating without thinking over it. It is important to ask questions we know, participants can answer. It would be wrong to ask a question like: *How much money does your organization spend on various software?* This question can lead to guessing. In these type of questions, one should include a alternative, *don't know*. It is also important to use simple words and language – we don't want any misunderstandings [10, 18].

A questionnaire should not be too long, or contain too many questions. It is very easy for a participant to lose interest if a questionnaire includes too many questions. We can be ending up with few participants, if a questionnaire is too long. Our questionnaire is placed on a company's web-page, *http://www.kompetanseweb.no*. Users of cfengine have received a mail with a link to this web-page. They don't have to log in; the link will directly lead them to the first page of the questionnaire. The questionnaire contains seven parts, with several questions related to each part. It includes multiple choice questions, yes/no-questions, rating-questions and some questions where participants can write down some text.

4.2 Questionnaire Statement and Briefing

The questionnaire was developed in co-operation with Mark Burgess. The questionnaire is divided into seven parts. Each part includes a number of questions and has its own meaning and aims:

- General Items
  - The purpose of these questions is to get a qualitative impression of participants needs and the challenges you face.
- Awareness
  - From this part we can get an idea of how well acquainted users are with the fundamental concepts of cfengine.
- Language Interface
  - The purpose of this part is to evaluate users' impressions of the cfengine language interface.
- Modus Operandi and Usage
  - This part will tell us users understanding and practical use of cfengine.

- Security
    - The purpose of this section is to measure users' ideas and impressions about security in the context of cfengine.
- Adoption
    - This part is about whether cfengine could be made easier to adopt.
- Training and documentation
    - The purpose of this part is to evaluate users' impressions of the documentation and help groups and training possibility.

4.2.1 General Items

This section contains some background information. Here we want to know who the participants are, where they come from. We will try to find out where the most of the participants are form. We also want to find out how big, participants companies are, how many users/host they deal with. Furthermore, we want to find out whether they use cfengine or not, and if they are planning to use it in the future. If they use cfengine, we would want to know how many hosts their configuration cover. It is interesting to find out what kind of operating systems participants use.

4.2.2 Awareness

In this part we are wondering how familiar, participants are with the basic concept and philosophy of cfengine. These questions are basic and we expect users have the basic knowledge about cfengine before using the tool. Understanding the convergence concept is important. An action or operation in cfengine shall always lead to the same final result, regardless of how many times you run cfengine. In cfengine, configuration is individual to each host, with other words are independent. Furthermore, we want to know if users know what adaptive locks are and what they are good for.

4.2.3 Language

The purpose of this section is to get some feedback form users when it comes to cfenigne's language interface. We want to know hove users handle their configuration files. If they use the import command to break their configuration files into several, and how they group their configuration instructions. We want to know which scripting language users prefer to use when expressing their ideas for system configuration. We also want to know whether users think they are able to "think" in cfengine terms. Users impression of the consistency of the language is important to know, whether they get syntax error when they try to express their own ideas. Another interesting thing to find out is how many configuration files users deal with, and their total byte count of configuration files.

4.2.4 Modus Operandi and Usage

In this section we want to find out users practical use of cfengine. Here we will try to find out which of cfengine's functionalities most of the participants use. We are talking about functionalities like, garbage collection of files/processes, tripwire functionality for checsumming and version control on their configuration files. It is also interesting to find out in which way use cfengine. Do they use it in an ad hoc way, proscriptively or as a file-copying/editing utility? Furthermore, we want to know what of the components of cfengine most of the users use. The last questions in this section are some control questions to test users' knowledge.

4.2.5 Security

The purpose of this part is to measure users' impressions about security in the context of cfengine. The first questions are about comparing cfengine and it components with similar tools. We want to find out whether the pull-only architecture of cfengine is difficult for users to understand or not. We also want to know if users' company/organization has firewall and whether they operate cfengine through a firewall or not. In this section we also have some security statements which we want users to agree or disagree with.

4.2.6 Adoption

In this part we want to find out whether cfengine is easy to adopt or not. First of all we want to know how users summarize the development of understanding of cfengine over time. We want to know how easy it is to debug in cfengine and what tools they would prefer to help debug problems. It is also interesting to find out how their configuration has grown over time. We also want top know if it would be easier for users to adopt cfengine if their company/organization could pay for it. The last question here is which software users consider as alternative to cfengine.

4.2.7 Training and Documentation

The purpose of the last section is to measure users' impression of the documentation and help groups. We want to know if users ever have taken some course in cfengine or ever read textbooks about system administration. Basically, we want to know if users prefer documentation or discussion as their primary source of information. Users are suppose to rate their experience with the documentation and web-support – its quality, consistency, coverage of important topics, including important topics and relevant examples. We also want o know how often users upgrade and if they ever experience problems with upgrading. In the last questions, users are supposed to come with suggestions to improvement in a possible new version of cfengine.

4.3 Model of Analysis

To evaluate or analyze the data, no special tools or method are been used. Most of the questions give quantitative result. But there are some questions that give us qualitative result too. So, both qualitative and quantitative data has been analyzed on different ways. Tables have been preferred rather than bar charts. It was easier to see the data this way. The figures of bar charts became really large, and when the figures were decreased, the text was difficult to read under each bar. Almost every question has its own table, both for quantitative data and qualitative data.

4.3.1 Quantitative Analysis

By quantitative analysis we mean handling numbers. Most of the questions give us quantitative data, with other words we have to handle a lot of numbers. These numbers are been directly put into a table. By these kinds of questions we can find out how many of the participants for instance agree or disagree with our questions. An example can be:

Q: Do you know what an adaptive lock is?

Table with participants answer:

	In number	**In percent**
Yes	16	26%
No	44	73%

Table 4.3.1.1: Example 1

The table includes both number of participants, who have answered yes/no and answers in percent. When analysing such data, we must try to find out why participants have answered yes or no. In this case, we can wonder why 26% have answered no. Haven't they read the documentation? Do they care what an adaptive lock is? Are they newcomers, and have not come to this point? The aim of analysis is to try to find answers to these kinds of questions. So when we operate only number, we call it quantitative analysis. Another example can be:

Q: Rate on a scale from 1 – 5, (where 5 is best) how easy it is to debug problems in cfengine.

Table with participants rating:

	In number	**In percent**
1	2	3%
2	11	20%
3	23	43%
4	16	30%
5	1	1%

Table 4.3.1.2: Example 2

Once again, the table includes both number of participants and rating in percent. The first thing we have to see here is what the most of the participants have rated, to find a possible average. And afterwards we must try to find out why participants have rated the way try have. As we can see from this table, the average is almost three. So the next step is to find out why one of the participants has rated five and why two participants have rated one.

So from these type of questions where participants are suppose to rate or answer yes/no, we only get to deal with numbers. It is easy to deal with this kind of data, because you only have to plot them in a table and than analyze them.

4.3.2 Qualitative Analysis

In qualitative analysis we talk about handling text and not numbers. These types of questions don't contain any alternatives. Participants are just supposed to write down their ideas, impressions or their meaning. Participants can be asked to comment something or give us their ideas or suggestions. The first thing we have to do is to classify the data. That is the hardest and most time-consuming part. What we have to do is to find similarities in the text form each participant and categorize them. Then we can try to find out which category includes most of the participants. An example can be:

Q: What improvement would you prefer in a new version of cfengine?

Table contains participants' suggestions to improvements in cfengine.

	In number	**In percent**
Simpler setup	24	30,4%
Better Policies	45	57,0%
Integration of scripting languages	3	3,8%
Other	7	8,8%

Table 4.3.2.1: Example 3

This table includes the qualitative data. As we can see, this table is similar to the table that included quantitative data. But work that is done before making the table is the important part. First of all, all the text must be carefully read through. The next step is to find the similarity in the text. The comments that resemble to each other can be one category. The same must be done with the other comments. In the end, group them by a word or a short sentence. Then, we can make a table by counting the similar comments. The most common comments are also been added in the analysis. After the discussion of the table, priorities are been given to each suggestion or idea. All the tables of this type include an "other" group. This group contain the comments that don't resemble to any other, or just resemble to a few.

- Highest Priority
    - These comments must be taken seriously
- Medium Priority
    - These comments can be discussed
- Lowest Priority
    - These comments are not fully important

4.4 Margin of Error

A questionnaire will always include a margin of error. Different type of points that can lead to margin of error can be:

- Misunderstanding of a question
    - Interviewers' fault
- Guessing
    - Participants' fault
- Not answering all of the questions
    - Participants' fault and interviewers' fault

All of these points can lead to a wrong picture of the result of a questionnaire.

It is important to use simple words and language to avoid misunderstanding. Difficult language can easily lead to misunderstandings. A participant can end up in answering something totally different from what is asked in the question. An interviewer should always be objective when asking questions. Participant should not feel forced to answer for instance yes or to a question. It will anyway be wrong to force a participant to answer in the way you want. Alternative answers are very important to add to a question. Another important point to note is to have a clear design which is easy to follow. It will almost always be interviewer's fault, if participants misunderstand a question.

To guess an answer to a question will always be a problem. Participants who guess answers will ruin the result. We will get a wrong picture of the result of a questionnaire. These kinds of participants are almost impossible to point out. An interviewer should add a sentence in the beginning of a questionnaire, where he/she could clearly say that this questionnaire is a serious kind of a questionnaire. These kinds of messages can exclude some of them. In this case, we can't blame the interviewer for the margin of error in the results.

One problem can be that some participants do not complete the whole questionnaire before submitting. Reasons for this can be located both at the interviewer and participants. A questionnaire should not be too long, or contain too many questions. It is very easy for a participant to lose interest if a questionnaire includes too many questions. The more participants, the

better results. So, the result will change if some participant jump over some questions or don't complete the questionnaire. A participant should go roughly trough the questionnaire, before starting to answer the questions. Then he/she will get an idea of how long it is and how much time it can take. It would be better to not participate in the questionnaire than answering just the half of the questions.

Chapter 5

Results and Discussions

5.1 The Result of the Survey

5.1.1 Part 1: General Items

From the first question we ca see that the major part of the users who have participate the questionnaire are from the USA. A lot of them are universities. We also have a lot from the Europe, countries like France, Germany, and United Kingdom etc. The distribution is like this:

- USA 70%
- Europe 25%
- Asia 5%

These numbers are very rough. This was expected because most of the users on the list on *http://www.cfengine.org* are Americans.

On the question *How many users do you deal with,* we get a big variety of number of users. The size of an organization will determine the number of users. If it is a very large company, we may deal with thousands of users, is there a small company, we may deal with one or to users. Since the variations are so huge, I had to divide them into groups:

	1 – 99	**100–999**	**1.000 – 9.999**	**10.000 – 99.999**
Number of users	17	18	14	3
In Percent	32,69%	34,62%	26,92%	5,77%

Table 5.1.1.1: Number of users participants deal with.

From this table we can se that the gap between the three first groups; 1-100, 100-1.000 and 1.000-10.000 is not very big. So we are here dealing with small and medium companies. As mentioned there are several universities in the survey. A university can have from 3.000 – 8.000 students, and all of them have their own users account. Medium companies can have from hundreds to thousands users. In the first group, we have users down to 5, 3 or only one. Here we are talking about really small companies. There are many one-person companies, and this should not prevent them from using for instance a tool like cfengine. The fourth group is the smallest one, but here we are talking about really big companies. The biggest amount of the users we are dealing with here is 60.000 users. According to Mark Burgess, who is the founder of cfengine, there is possible to use cfengine on 100.000 hosts. But cfengine has only been tried on 20.000 hosts. So there would not

be a problem for them to use such a tool. So the number of users varies from one single user to 60.000 users.

85% of the users are currently using cfengine. That means 14% of the users who participated have experience with cfengine. The 14% are not using the tool currently. The reasons why they are not using it currently can be:

- Not satisfied with the tool
- Have moved from a company that used Cfengine.
- Have not started yet.

96% of those who uses cfengine today are planning to use in the future to. Only 3% think they will not use in the future. Users have write down some comments on this, let us go through them:

	Number of users	**In percent**
Will surely use cfengine in the future	19	47,5%
It depends on the situation	7	17,5%
100% sure of using cfengine in the future	12	30%
Will not use cfengine in the future	2	5%

Table 5.1.1.2: Using cfengine in the future.

47,5% of the users says that they will most probably use cfengine in the future. The most common answers of this group are:

- I enjoy using it, will most probably use it in the future.
- I like it and I'm going to keep using it.
- It has been very useful to date.
- Would like to extend cfengine to more machines and more uses.
- Would like to continue to work on total configuration management using cfengine.
- Initial roll-out and usage has been positive. Would like to explore File Editing and Package Management more in the future.

Seven users or 17,5 % of the total amount of the users says that it depends on the situation. There can be some different reason for this kind of answer. Some of the answers of this group are:

- If I could, I would use only the OS distribution and rely on its package management systems. That, however, is rarely possible. Any addition to the OS introduces the possibility of breaking applications
- At the moment we're only using it on our machines. We would like to extend this to customer machines in the future, if it's possible.
- I will use it in the future if more dynamic toolset is implemented.
- I will continue using cfengine if I don't move to another job that is not using cfengine.

There are not many users who really are dissatisfied with cfengine. As we can see from the answers, most of them will or most probably use cfengine in the future. 30% of the users who have participated, are 100% sure of using cfengine in the future. Typical answers from them are:

- We can't live without it anymore
- I am now using and I will guaranteed use it in the future
- We've found nothing which can match cfengine's capabilities.
- Of course we will use it in the future
- After having used cfengine for the last year, I cannot imagine trying to manage a network without it. So yes, I will definitely use it in the future.

A lot of the users do enjoy using it and have benefit form it. It is very important to completely understand the philosophy and the purpose of cfengine to get most out of the tool. Only two or 5% of the participants have answered that they will not use the cfengine in the future. One of them is moving to another job where they don't use cfengine. The other one answered that their company did not have enough manpower to keep up using cfengine.

One of the background questions was what kind of OS the participant use, and the result showed this:

	Number of users	**In percent**
Solaris	4	7,41%
Linux	13	24,07%
Windows/Solaris/Linux	7	12,96%
Solaris/Linux	20	37,04%
Windows/Linux	10	18,52%

Table 5.1.1.3: Operating systems participants use.

The table shows that almost every user uses Linux. The reason is that basically cfengine is a UNIX administration tool that aims to make the easy administrative tasks automatic, and the hard tasks easier. As we can se, there are only 4 users that don't use Linux. Windows is an easy-to-use operating system, so that is the reason why companies use such an operating systems. But usually the serves always runs Solaris or Linux. As we can see no one of the participants only use Windows. A reason can be that you can't run cfengine directly on a Windows OS. You have to use a tool called *cygwin*, emulating a UNIX system under Windows.

	0 – 999	**100 – 499**	**500 – 999**	**1.000 – 2.999**
Number of hosts	39	12	3	4
In Percent	67,24%	20,69%	5,17%	6,90%

Table 5.1.1.4: Number of hosts cfengine configuration cover.

Table 5.1.1.4 shows the how many host cfengine configuration covers in each company. Here we can clearly see that most of the companies use cfengine on 0 to 100 hosts. Some of the participants have answered 0 hosts

too. The reason can be that they are in a test period and have not implemented cfengine in their network yet. In group three (500 – 1.000) and group four (1.000 – 3.000) we have participants who deal with hosts between 500 and up to 2000. These companies must have had good experience with cfengine since their cfengine configuration covers so many hosts. The founder of cfengine, Mark Burgess has asserted that it is possible for cfengine to cover over 100.000 hosts. So covering 2000 host should not be a problem at all. So the group one (0 – 100) is in a development stage. Group three can be consisting of medium companies compared to group three and four which can be large companies.

Another interesting ting was to find out whether the participants use or used other configuration management tools.

	In number	In percent
None	11	27,5%
Home grown	8	20,0%
Perl	3	7,5%
CVS	4	10,0%
Scripting	3	7,5%
Rsync	4	10,0%
Other	7	17,5%

Table 5.1.1.5: Different configuration management tools, participant have used or use.

Table 5.1.1.5 shows us the different kind of configuration management tools user have used or use together with cfengine. The most of the participants use no configuration tool besides cfengine. That means that they are satisfied with the tool and that it covers their needs or wants. 20% of the participants use home grown tools; the advantage here is that they can develop tools after their own wishes and needs. Scripting and perl which are 7,5% respectively can also be considered as home grown. Scripting in UNIX environment is a excellent way to express one's needs and wants. CVS and *rsync* are two configuration management tools that are most popular among the participants beside cfengine. But are these tools better than cfengine when it comes to reliability, predictability and stability? But people usually use the product they are most satisfied with, the tool that covers their needs. It is also common to use the tool we are most familiar with, even thou we know there exist better tools. The group, other, contain different tools that are a lot less popular among the participant. We are talking about tools like, *up2date*, *kickstart*, *nagios*, and *isoconf*

5.1.2 Part 2: Awareness

This part contains question about the principles and concept of cfengine. The purpose of these questions is if the users have understood the principles and purpose of the tool.

The first question is a basic general question that will tell us whether the participants have understood the principles on which cfengine operates. The

participants are supposed to rate their knowledge be rating it on a scale from 1 – 5.

	In number	In percent
1	1	1%
2	4	6%
3	20	32%
4	23	37%
5	13	21%

Table 5.1.2.1: Participant rating of understanding the principles of cfengine

As we can see from the table, most of the participants have placed themselves as 3, 4 and 5. 21% of the participants understand the principles completely. The biggest group has rated themselves with four. The average is also four, so we can easily see that almost everyone claims that they have understood the principles to a certain extent. 7% have rated themselves on 2 and 1. The reason can be that they have recently started to use cfengine or they simply just don't understand the principles. Maybe they have not adjusted themselves to the principles yet or because they don't think it is necessary. Maybe they think that as long as tool does what they want, they don't have to understand all the principles. It can also happen that they are pleased with the basic principles or functionality; that they don't use the entire tool, just some parts of it.

The next questions are based on understanding of the tool. The answers to these questions will show us the participants' basic knowledge. The first question is about the convergence. What we are trying to ask is, which property cfengine refers to. The alternatives are:

a) An action or operation should lead to the same final result, regardless of how many times you run cfengine.
b) The same actions are repeated again and again, regardless of when you run cfengine
c) The actions of the system are randomized to spread the load

	In number	In percent
a)	59	96%
b)	2	3%
c)	0	0%

Table 5.1.2.2: Participants understanding of convergence

The correct answer is a), *an action or operation should lead to the same final result, regardless of how many times you run cfengine.* This is a good start, 96% or 59 of the participants have answered correct on the first question. This is the basic functionality of cfengine that we have an ideal state to reach to. You are always supposed to end up in the same state, independent of how many times you run cfengine. Every one who runs cfengine should now this. Only two participants have answered wrong on this question. But it is still worrying that they don't understand the basic functionality. Maybe the first two alternatives look alike each other. The last

alternative is not even close, so it's good no one answered that. The next question is also a basic one. We are wondering what of the three alternatives best summarize cfengine. The alternatives are:

a) Configuration is individual to each host. Security means that no external system can alter the configuration of a host, nor send it instructions. Hosts are fully autonomous and may choose to download new instructions from somewhere if they wish.
b) A central manager can decide to change any host in the network and must be obeyed. Each host does the work of the manager. Security is based on encryption of the transmitted instructions.
c) A central configuration file determines the rules by which every host will be patched. Security is based on having an authorized MD5 checksum on every file to patch, like Tripwire.

	In number	In percent
a)	35	59%
b)	16	27%
c)	8	13%

Table 5.1.2.3: This table show how participants summarize cfengine.

Once again, the right answer is a). As we can se only 59% of the participants have answered correctly. Cfengine attaches importance to that no external system or host can change configuration of a host, that's where security comes in the picture. But the host on the other hand are free to download instructions from anywhere. Since the host are autonomous, alternative b) and c) are wrong. The host don't have to obey a central manager or be dependent of a central configuration file. Participants who have answered b) have most probably mixed cfengine with another tool, or maybe a tool they have used earlier. Few have answered c), maybe they have mixed the tool with Tripwire? Anyway, the most of the participants have answered correct on this question.

Further, we are wondering if the order of which operations are declared does matter or not.

	In number	In percent
Does not matter	23	39%
Is important	33	56%
Don't know	2	3%

Table 5.1.2.4: The table shows us what participants think of the order of operations.

In cfengine it does not matter in which order operations are declared. Documents of cfengine say clearly that the order doesn't matter. The most of the participants have answered the opposite. This functionality is also an advantage, because then you don't have to think of the order when operations are declared. 33 of the participants have answered wrong. It is possible to enforce ordering through temporarily defines classes as in example:

```
control: actionsequence = ( copy.general copy.host_specific )

copy.general::
           /var/local/archive/general dest=/ recurse=inf

copy.host_specific.foo::
               /var/local/arcive/hosts/foo dest=/ recurse=inf
```

In the next question we are wondering if users know what an adaptive lock is.

	In number	In percent
Yes	16	26%
No	44	73%

Table 5.1.2.5: The table shows if the participants know what an adaptive lock is.

Cfengine treats all of its operations as transactions which are locked. Locking these operations prevents contention from competing processes and it also places limits on the execution of the program. By locking these operations, many cfengine programs can coexist without problems. There are two locking parameters which controls the locking:

- IfElapsed
  - Tells operations that they only can perform when a certain period of time has elapsed since the last time the action was performed.
  - Takes care of spamming protection
- ExpireAfter
  - Tells cfengine that there are given length of time for an action
  - Protects against hanging sub-processes [8].

Only 26% of the participants know what cfengine's adaptive lock is. This is an important feature that users should be aware of. As mentioned above, it takes care of spamming problem and sub-process hanging. This feature is also described clearly in the documentation of cfengine.

	In number	In percent
Yes	38	63%
No	22	36%

Table 5.1.2.6: The table tells if participants have experienced that cfengine tells it's "too soon" to do something.

The next question was if participants had experienced that cfengine tells them it is "too soon" to do something. Table 5.1.2.6 shows that 63% of the participants have experienced that. The adaptive locks are controlled by "too soon" parameter and control the execution time. There are fixed bounds for the execution time even when scheduling requests occur randomly in addition to the periodic scheduling time[9]. 22% of the participants have not experienced that; the reason can be that these users have not experienced any requests that have exceeded the fixed bounds of execution time.

	In number	In percent
True	9	15%
False	49	84%

Table 5.1.2.7: Is there possible to implement a cfengine script by a Perl script

Table 5.1.2.7 shows us the participants answer to the question; could a cfengine script be implemented by a Perl script. The correct answer is yes, it is possible. Cfengine is a very high level language, much higher-level than Perl or shell. As we can se from the table 5.1.2.7, most of the participants have answered correct. Nine participants have answered wrong, the reason can be:

- Just guessing
- Never made or edit a cfengine script
- Just use the default scripts

5.1.3 Part 3: Language interface

This part will evaluate participants' impressions of the cfengine language interface.

	In number	In percent
Yes	47	85%
No	8	14%

Table 5.1.3.1: Does participants break cfagent.conf configuration into several files

Table 5.1.3.1 shows us if participants use import to break up their cfagent.conf configuration split into several files. It is possible to break up files into convenient modules and to import common resources, such as list of groups. The syntax for breaking a configuration file is:

```
Import:

     any::

        cf.global_classes


     linux::

        cf.linux_classes
```

An important point to be noted is that, if you define variables in an imported file they will not be defined in their parent files.

It can be seen as advantage to break up large configuration files into smaller files. An advantage can be that you will get a better overview of your configuration files. 47 or 85% of the participants have answered that they break their cfagent.conf configuration into several files. Big companies, who deal with 1000's of machine, can get really big configuration files. It is more relevant for them to break their configuration files into smaller than small companies who only deal with a few machines. So the remaining

participants don't break up configuration files most probably because of they have smaller files.

The next question we were wondering how users mainly groups their configuration instruction. The alternatives are:

a) By operating system type
b) By organizational group (department)
c) By the role of the host in the network

	In number	In percent
a)	12	32%
b)	2	5%
c)	23	62%

Table 5.1.3.2: The table shows how participants group their configuration instructions.

As we can see from the table 5.1.3.2, most of the users group their configuration instructions by the role of the host in the network, while only two of the participants group their configuration instructions by department. We don't have a right/wrong answer to this question; this is up to each company to decide. The reason why the most of the participants have answered c) can be that it seems obvious to group like this, because the most important host will also contain the most important, relevant configuration instructions. The biggest/ most important instructions should be given to the hosts that can manage to handle them. As long as you run *cygwin* on for instance a Windows operating system, it is no point of grouping the instructions by operating systems.

	In number	In percent
Yes	35	63%
No	20	36%

Table 5.1.3.3: This table shows us if participants use shell commands embedded in cfengine.

Table 5.1.3.3 shows us if participants use shell command embedded in cfengine to solve problems. Cfengine gives access to the shell environment variables and allows defining variables of your own too. It is a good opportunity to use these shell commands to solve problems. As we can see from the table, 63% of the participants often use the shell command. But there are quite lot participants that don't use these shell commands. These participants either use it infrequent or they use another method to solve their problems.

	In number	In percent
Perl	10	20%
Cfengine	33	67%
Shell	6	12%
Tcl	0	0%

Table 5.1.3.4: The table shows us how participants express their ideas in cfengine.

Table 5.1.3.4 show us the answers of the next question. We were wondering which scripting language users express their ideas for system configuration in. as we can see from the table; most of the users use the cfengine script to

express their ideas. 67% of the participants use the cfengine script. The basic scripts in cfengine are in cfengine, once you are used to it, it will be much practical to use cfengine script. Other most common scripts are perl and shell. Some users don't like changes and will keep on using the scripting language they are used to. As mentioned earlier, cfengine is a much higher-level language than perl and shell. This is interesting for Mark Burgess, that most of the participants will rather use cfengine scripts than other scripting languages.

The next question is about rating how well users feel that they are able to "think" in cfengine terms. The participants are supposed to rate on a scale from 1 – 5.

	In number	In percent
1	1	1%
2	6	10%
3	21	38%
4	19	34%
5	8	14%

Table 5.1.3.5: Participants rating of how well they can "think" in cfengine terms.

As we can see from the table the most of the users have placed them selves on 3, 4 and 5. 14% of the participants think they completely understand the cfengine terms. The biggest group, 38% have placed them selves on 3. They gradual understand and can think in cfengine terms. In the start it will be hard to think in cfengine terms, but the understanding will be improved gradually. Only 11% have placed them selves on 2 and 1. The reason can be that they are fresh users. It is important to read through the documentation of cfengine and go carefully through the examples. When you have done that, you will be more able to "think" in cfengine terms. The document of cfengine covers the most important topics.

In the next questions we are wondering if users ever have experienced syntax error when trying to express their ideas.

	In number	In percent
Often	10	20%
From time to time	33	67%
Seldom	6	12%
Never	0	0%

Table 5.1.3.6: This table shows us how often users get syntax error when expressing their ideas.

From the table 6.1.3.6, we can see that every one of the participants have experienced problems when they try to express their ideas in cfengine. 20% of the participants have experienced that often, that is a very high number. Whole 67% have answered that they have experienced syntax error from time to time when expressing their ideas. There can be many reasons for that. You can get syntax error by:

- Using wrong command

- Scripting in a language that cfengine don't support
- Mixture of different scripting languages
- Implementing a cfengine script by a perl script
- How intuitive the language is
- If particiants find it difficult to learn/recall syntax

It is easy to make those faults when making own scripts in cfengine. Only 6% have answered that they have seldom experienced syntax error when expressing their ideas. This group can consist of two types of users. Users who never express their ideas in cfengine or are happy with the default scripts. The other type is users who understand cfengine completely and have long experience with the tool. A little fault can lead to a syntax error. No one is perfect.

The next question is about rating the consistency of the cfengine language on a scale from 1 – 5 (where 5 is best).

	In number	In percent
1	1	1%
2	6	10%
3	21	38%
4	19	34%
5	8	14%

Table 5.1.3.7: The table shows participants rating of consistency of cfengine language.

As we can see from the table 5.1.3.7, most of the users have rated the consistency of the cfengine language on 3 and 4. There are only eight participants who are completely pleased with the language. Cfengine uses a declarative language for describing policy. This can often be confusing to newcomers who are more familiar with imperative scripting languages as perl [1]. This would be the main reason for why so many have rated the consistency of the language on 3 and 4. Participants, who have rated the consistency on 4 and 5, are most probably experienced users. They are now used to the declarative language and have no problems wit it. The one participant who has rated on 1 is most probably a newcomer.

The next question is also rating question. This time we are wondering if the participants think the language is well-suited to its task. We are using the same scale, 1 – 5.

	In number	In percent
1	0	0%
2	5	9%
3	16	29%
4	26	48%
5	7	12%

Table 5.1.3.8: This table shows us if participants think the language is well-suited.

As we can see from the table 5.1.3.8, most of the participants have rated 3 and 4. Most of the participants think that the cfengine language is well-

suited to its task. No one has rated 1 and only 5 participants have rated 2. Participants who have rated lower than 3 are most probably not satisfied with the language at all, the reason is simply that they are unaccustomed to it. Participants who have rated who have rated on 4 or 5 are more used to it now and they are taking the language as a matter of course. As you get more used to the cfengine's declarative language, you will automatically think that the language is well-suited to its task.

In the next question, we were wondering how many configuration files the participants have.

	In Number	In Percent
1 – 9	14	26,9%
10 – 19	14	26,9%
20 – 29	12	23,1%
30 – 39	5	9,6%
40 – 49	3	5,8%
50 – >	4	7,7%

Table 5.1.3.9: The participants' configuration files.

Table 5.1.3.9 shows us the number of configuration files participants have. As we can se from the table most of the participants have number of configuration files between 1 and 29. The group 1 – 9 and 10 – 19 are equal, both on 26,9%. Large companies will always have a lot more configuration files than small companies. So the size of a company makes the biggest different. In the group 50 – >, we are dealing with really big companies. The 4 participants who are placed here deal with 70, 97, 130 and 140 configuration files. Especially, the two last numbers are very big. There can be two reasons for why the total amounts of configuration files are so large. One reason is that we are dealing with a very big company with a lot of hosts. In part one, two of the participant answered about 2.200 hosts on the question, how many host does you cfengine cover? It is possible to think that these are the same participants who have 130 and 140 configuration files. The other reason can be that they have used the *import* command to break their configuration files into several. A strange thing is that two of the participants have answered 0 configuration files. Cfengine can do nothing for you, if no configuration files exist. So, either the participants who have answered 0 didn't understand the question or they are in the installing period and not come so far.

	In Number	In Percent
0 – 999	7	14,3%
1.000 – 9.999	4	8,2%
10.000 – 99.999	21	42,9%
100.000 – 999.999	15	30,5%
1.000.000 – 9.999.999	2	4,1%

Table 5.1.3.10: The table shows total byte count of participants' configuration files.

Table 5.1.3.10 shows us total byte count of participants' configuration files. As we can see from the table the biggest group is 10.000 – 99.999 with

42,9%. Most of the users have their files between 1.000-and 999.999 bytes. As mentioned above, the biggest companies will have larger configuration files than smaller companies. Total byte counts of configuration files are independent from number of configuration files. It can happen that some large companies have not broken their configuration files into several, and rather have a few large files. Two of the participants have total byte count of configuration files over 1.000.000. There is a possibility that we again are dealing with the same two participants who had 130 and 140 configuration files. But there can be some other who have larger and consequently fewer configuration files. Two of the seven participants who are in group 0 – 999, have answered 0 on total byte count of configuration files. Those must be the two who answered that they had no configuration files at all. Another thing we can do is to divide total byte amount with number of configuration files. Then we can find out how large one configuration file is on average. Some configuration files will always be larger than others. In this survey, it will be hard to find out number of bytes for each configuration files, because there are more participants who have answered last question than this one.

Next question is about the discussion of changes in cfengine configuration. We were wondering if participants discuss the changes as team or everyone in the team makes changes to the syntax on an ad hoc basis.

	In number	In percent
Discuss	26	50%
ad hoc	26	50%

Table 5.1.3.11: This table shows how participants discuss changes in cfengine configuration.

As we can see form the table 5.1.3.11, half of the participants discuss changes in cfengine configuration as a team and half of them make changes on an ad hoc basis. By discussing the changes, everyone in the team can come with good suggestions. By making changes to the syntax on an ad hoc basis we mean, making changes to the configuration files only in this special case. There is no correct/wrong answer to this question. In some cases it is better to discuss changes to the configuration file as team and in other situations it's better to make changes on an ad hoc way.
In the next question we were wondering what changes the participants would like to see in syntax.

	In number	In percent
Consistency	9	30,0%
Better editfiles	10	33,3%
Integration of other scripting languages	2	6,7%
No actionsequence	2	6,7%
Better iteration	2	6,7%
Simplified	2	6,7%
None	3	10,0%

Table 5.1.3.12: Participants suggestion to changes in syntax.

As we can see from the table 5.1.3.12, most of the participants want more consistency and better editfiles. Participants, who have asked for more consistency, allude to:

- Significantly more consistency in general
- Use of attributes
- Built-in variables
- Order of operations
- Case consistency
- More consistence in what is required and accept
- Consistency in syntax between functions and rest of cfengine

The participants, who have asked for better edit files, allude to:

- Tidy editfiles
- Editfiles should be made into a separate language
- Editfiles should have a one-function method of setting variables in configuration files
- Editfiles commands should be a bit more complete
- Editfiles are cumbersome
    - Lacking some features
    - Hard to accomplish
    - Import/include is hard to understand
- Better syntax coherence between editfiles

Two of the participants have wants better iteration. By better iteration they mean, better handling of list iteration. The same variable should not expand recursively. Integration of other scripting languages are also been mentioned. Both of the participants who have mentioned that, wants an integration of perl. Two of the participants want less confusion about actionsequence order, imports, and *define/elsedefine* scopes. Another two participants just want a simpler syntax. Three of the participants are happy with the existing syntax, they don't want any changes.

In the next part we just wanted the participants to give us some other comments. Some of the participants had some other comments. Here are some of the comments:

- The cfengine package management support needs to be rethought to mesh better with how admins manage packages.
- More and better documentation
- I would like to see embedded version control which will allow me to use several different branches of cfengine configuration
- Hooks to tie into cvs would be nice - checkout/update a directory from cvs
- Why is cfengine written in C? It seems to us that writing cfengine itself in a scripting language would make it easier to debug when there are problems.

- In general cfengine needs to be rethought from the top-down. It's been developed bottom-up, but it's past time to reassess cfengine as a whole.
- I really do like cfengine. It's the best thing going.
- Not consistent, not good.
- Don't get rid of editfiles, don't give into the temptation to declare them evil.
- The documents should specify what is to be done on a server, what is to be done on a client, and how do you push the */etc/motd* to the client.
- Small examples are good way to suggest how to go about learning cfengine.

Highest priority: As we can see from the table, better editfiles and consistency in a future version of cfengine

Lowest priority: Integration of different scripting languages, better iteration, simplification of the language

5.1.4 Part 4: Modus Operandi and Usage

The purpose of these questions is to map out participants understanding and practical use of cfengine.

	In number	In percent
Yes	29	53%
No	25	46%

Table 5.1.4.1: The table shows if participants perform garbage collection of files.

In the first question, participants are supposed to tell us if they perform garbage collection of files. As we can see from the table 5.1.4.1, most of the participants perform garbage collection of junk files. One of cfengien's functionality is to get rid of junk files which can disorder the system. This is a useful functionality, that user should be aware of. 46% of the participants have answered "no" on the question. Most probably does this group of participants perform garbage collection manually or use another tool to handle them. No one likes unnecessary junk files. Users should take advantage of functionalities that are built-in the tool, because these are built-in for a reason. Another reason can be that users are afraid of deleting something valuable.

In the next question, we are wondering if participants use tripwire functionality for checksumming.

	In number	In percent
Yes	22	40%
No	32	59%

Table 5.1.4.2: The table shows if participants use tripwire functionality for checksumming.

Cfengine contains "tripwire functionality" for MD5 checksums. It means if you define a checksum database and activate verification, *cfagent* will build a database of file checksum and warn you when files' checksum changes. This functionality makes *cfagent* act like Tripwire. Example:

```
control:

      ChecksumDatabase = ( /etc/cfengine/cache.db )

files:

      /filename checksum=md5 ....
```

As we can see from table 5.1.4.2, 40% of the participants use this tripwire functionality. This is also a useful functionality users should be aware of. This tripwire functionality is also a useful functionality, where user can get control over checksum changes. From table 5.1.4.2, we can see that most of the participants have answered "no" on this question. This group of participant may already use Tripwire from previous, and is used to it. Or they just don't need it.

In the next question we ask why those 32 participants don't use tripwire functionality.

	In number	**In percent**
Don't need it	5	19,2%
Use other tools	9	34,6%
Dissatisfied	6	23,1%
Will use it in future	6	23,1%

Table 5.1.4.3: The reasons why participants don't use the tripwire functionality.

The typically answers from those participants were:

- Don't need it in the environment I am working on
- I haven't had the need yet since the security functions of cfengine are not my utmost importance.
- Because I use package management, and that changes files all the time anyway
- /etc/daily is already doing it
- We tried it, but it takes so long to run.
- It is not useful to me. My machines are not internet-exposed and security breaches are not a primary concern.
- We use osiris for tracking system changes.
- Seems like the load is high and the risk is high.
- Switched to using radmind as a tool to do this.
- Get the impression from the mailing list that checksums is buggy/prone to cause issues.
- I use Tripwire and a home brewed summary report generator for intrusion detection.
- It is planned for the future.

As we assumed, most of the participants use other tools for checsumming. The tools for checksumming participants have mentioned are:

- /etc/daily
- Tripwire
- osiris
- radmin

All of these tools are almost similar. It will always be easy for a user to use a tool he/she is familiar with.
23% of the participants have answered that they are planning to use the checksumming functionality in future. The reasons can be that they haven't implemented it yet or haven't had time to learn it.
Some of the participants have answered that they don't need it because they don't care or they just don't need it in their environment. Six participants have answered that they are dissatisfied with this functionality. The reasons for that are:

- It takes to long time to run
- Updating the database after files change is too much of as hassle
- Warnings in the huge amount of output created by cfengine are easily overlooked

In the next question, participants are supposed to tell if they perform garbage collection of processes.

	In number	**In percent**
Yes	19	40,4%
No	26	55,3%
Sometimes	2	4,3%

Table 5.1.4.4: The table show whether participants perform garbage collection of processes.

Table 5.1.4.4 shows whether participants perform garbage collection of processes. Garbage collection of processes is very important. Too many unnecessary processes can be a security risk for instance denial of service attacks on the system. Reasons why process tables fill up with unterminated processes are:

- Faulty X-terminal software which does not kill its children at logout.
- Programs like netscape tend to go into loops from which they never return
- If the host concerned has important duties then this lack of responsiveness can compromise key services.

If users always log out at the end of the day and log in again the day after then this is easy to address with cfengine. How to kill commonly hanging processes:

```
processes:

          linux|freebsd|sun4

                 SetOptionString "aux"

           any::

           "Jan|Feb|Mar|Apr"

                 signal=kill

                 include=ftpd
              include=xterm
              include=netscape
              include=ftp
              include=perl
              include=java
              include=passwd
```

As we can see from the table 5.1.4.4, most of the participants don't perform garbage collection of processes, as mentioned above; it can lead to a denial of service attacks on the system. Two of the participants perform garbage collection of processes form time to time. And 40,4% of the participants perform garbage collection regularly.

Next question is: Do you use cfengine

a) in an ad hoc way – to solve a random selection of issues
b) proscriptively – to lay out plan for your whole system systematically
c) as a file-copying utility
d) as a file-editing utility

	In number	**In percent**
a)	13	31,7%
b)	25	61,0%
c)	2	4,9%
d)	1	2,4%

Table 5.1.4.5: This table shows in which way participants use cfengine.

As we can see from the table 5.1.4.5, most of the participants use cfengine proscriptively. That means to lay out plan for the whole system systematically. 31,7% of the participants use cfengine to solve a random selection of issues. 3 of the participants use cfengine as a file-copying or file-editing utility. Cfengine will be used according to users needs. Not everyone use all of the functionalities of cfengine. When we are talking about the two first alternatives, users usually pick one of the possibilities. Either solving problems in an ad hoc way or proscriptively. It is no point in installing or using cfengine, if you are just planning to use it for file-copying or file-editing utility.

In the next question we were wondering which of the cfengine programs/components users' use.

	In number	In percent
Cfagent	52	98%
Cfservd	48	90%
Cfexecd	45	84%
Cfrun	21	39%
Cfshow	6	11%
Cfenvd	23	43%
Cfenvgraph	7	13%

Table 5.1.4.6: Which components of cfengine, participants use.

As we can se for table 5.1.4.5, most of the participants use *cfagent, cfserv and cfexecd*. This table shows that users follow recommended usage quite well. These components have the most important functionality. These three are as expected the most popular amongst the participants. *Cfagent* is a configuration agent which checks host against file-sets it parses to every host. It performs also security checks. The engine determines if there is something to do, every time when *cfagent* is ran. Table 3.2.1 shows that *cfagent* have a lot more jobs than any other component. *Cfservd* is a file server which starts the engine remotely. *Cfservd* does also control the authentication. *Cfenvd* is a wrapper for execution of *cfagent*. *Cfrun* and *cfenvd* is next on the list on most used components. Both of these components have their advantages. *Cfrun* executes one or more remote agents, but it can't do it directly, it has to contact *cfservd*. *Cfenvd* has the responsibility of anomaly detection. Both of these components are recommended. As we can see from the table, *cfshow* and *cfenvgraph* are used by just a few participants. These are components that are not so highly recommended. *Cfshow* shows the contents of the database which is used by cfengine. This information can be interesting for users. *Cfenvgraph* makes graphs of the system performance both before and after a possible anomaly. It can be interesting to se how system acts after an attack.

	In number	In percent
Yes	6	11%
No	47	88%

Table 5.1.4.7: Participants who use cfenvgraph component.

Table 5.2.4.6 show how many participant who use the component, cfenvgraph. Only 6 of them do. In the next question we were wondering what participant are looking for in the graph. Participant answers are:

- Look for pattern, trends and for interest
- Used once to find a random lock-up problem
- Used it once on a mailserver in DMZ to watch *http, smtp, pop and imap* traffic
- Client status, connection, success and failures

Participants, who have used *cfenvgraph*, have used it for either just check the trends, for interest or just used a few time when some problems have occurred. It is interesting to check the system performance form time to time and find trends and later try to find the reasons for changes.

	In number	In percent
Yes	35	74%
No	12	25%

Table 5.1.4.8: This table shows if participants would prefer to see the data from cfenvgraph via a web-page.

Table 5.1.4.7 show if participants would find it useful to be able to see *cfenvgraph* data via a web-page. 74% have answered yes. That is a lot more than those participants who use *cfenvgraph* at this moment. It can happen that more users will use the component, *cfenvgraph,* if it is possible see *cfenvgraph* data via web in the future.

Further, we want to know if participants test their configuration changes before rolling them out.

	In number	In percent
Yes	44	83%
No	9	16%

Figure 5.1.4.9: This table shows if participants check their configuration files before rolling them out.

It is an advantage to test the configuration before use it. The possibility of discovering bugs/fault is bigger then. As we can see from table 5.1.4.8, 83% of the participants do check their configuration files before implementing them.

In the next question we were wondering if users of cfengine use any kind of version control on their configuration files.

	In number	In percent
CVS	17	36,2%
RCS	11	23,4%
Subversion	10	21,3%
None	9	19,1%

Table 5.1.4.10: Version control tool participants' use.

Version control tools help users to make changes to files without ruining each others work or making confusion in version. Functionality of a version control is:

- Changes that are made are not lost
- A log of modifications
- Allow you to revert to older version

The most popular tools for version control are:

- CVS – Concurrent Version Control
- RCS – Revision Control System
- Subverion

As we can see from table 5.1.4.8, most of the users do use a version control tool. Only 19,1% does not. The remaining 80,9% use one of the three tools

we have mentioned above. It is better to use on of the version control tools than just change files and either try to remember earlier version or take manually notes. You never know when you can need the old version. When it's about which tool you should use, it's up to each user. All of the tools have in principle the same functionality.

The next six questions are some control questions. By these questions we are trying to find out how well participant are know with cfengine and its components. The questions are:

Q1. *Cfagent* can copy files from *cfservd*
Q2. *Cfrun* sends policy instructions to *cfagent*
Q3. A misconfigured *cfenvd* can be a security risk
Q4. *Cfenvd* is used to monitor remote hosts
Q5. A misconfigured *cfagent* can be a security risk
Q6. A misconfigured *cfservd* can be a security risk

The participants are supposed to answer true or false on each question.

	In Number		**In Percent**	
	True	**False**	**True**	**False**
Q1	48	4	92%	7%
Q2	15	37	28%	71%
Q3	20	32	38%	61%
Q4	9	42	17%	82%
Q5	41	12	77%	22%
Q6	50	3	94%	5%

Table 5.1.4.11: Participants answer to the control questions.

Table 5.1.4.9 shows participants answer to the control question. Correct answer to each question is:

Q1. true
Q2. false
Q3. false
Q4. false
Q5. false
Q6. true
Q7. true

As we can se from the table, most of the participants have answered correct on each question. But on some questions have quite a lot answered wrong. Some reasons can be that some of the participant have misunderstood the question or mixed the components. These are some basic question that users should be aware of.

5.1.5 Part 5: Security

The purpose of this part is to make participants to give us their ideas and impressions about security in the context of cfengine.

The first question is a general question about cfengine. We are wondering if participants trust cfengine to behave as they expect.

	In number	In percent
Yes	42	79%
No	11	20%

Table 5.1.5.1: Participants trust cfengine.

As we can see from table 5.1.5.1, most of the participants trust cfengine to act as they expect. You will know how cfengine by reading the documentation and if it does as documentation says, you will automatically trust it. The trust comes with experience. A user who has long experience with cfengine will also be reliable if he/she tells newcomers that cfengine is trustable. 20% of the participants don't trust cfengine to act as they expect. Reasons can be that those of the participants are newcomers or not have enough experience. It is normal to do some faults in the beginning that one don't take note of and rather blame cfengine.

In the next questions, participants are supposed to compare cfengine components with other similar tools. The questions are:

Q1. Do you trust ssh more than you trust *cfservd*?
Q2. Do you trust Tripwire more than you trust *cfagent*?
Q3. Do you trust *http* more than you trust *cfservd*?

	In Number		In Percent	
	Yes	**No**	**Yes**	**No**
Q1	25	27	48%	51%
Q2	12	39	23%	76%
Q3	22	30	42%	57%

Table 5.1.5.2: Cfengine components compared to similar tools

As we can se from table 5.1.5.2, most of the participants trust *cfserved* more than *ssh*. But the gap between them is not so big. *Ssh* is a tool to log onto another host over a network and for instance move files from one host to another. *Cfserved* dose the same and it has the control over access that is based on RSA (A public-key encryption) authentication and IP addresses. With other words, it is more secure to use *cfservd* than *ssh*. Further in the table we can see that most of the users trust *cfagent* more than tripwire. Tripwire is a tool which makes a database with information about all the files on the disk and their checksum. By running a check against the database we can find out which files are been changed. One can tell tripwire via a configuration file which files to monitor. Advantages of tripwire are:

- Invisible for an attacker
    - Noting prevent a attacker to do his things – hard to discover
- Uses strong encryption
    - Impossible for an attacker to make changes to the databasefile
- Uses a file-based databasefile
    - Can only be copied away – burned on a CD

Cfengine and tripwire have almost the same functionality. The only difference between these tools is that tripwire makes a checksum of file properties too like the file permissions. Cfengine make the checksum only out of the contents of a file.
The reason why most of the participants do trust *cfagent* more than tripwire can be that they are cfengine users or that they have good experience with it and its functionality. Some few have answered that they trust tripwire more than *cfagent*. These participants most probably use tripwire and have good experience with it. On the last question, we can see that most of the participants have answered that they trust *http* more than *cfservd*. From earlier experience, we know that most of the users of cfengine will rather use tools like *rsynch* and *ftp* than *cfservd*. The reason can be that they have always used *http* and just don't see the point in implementing *cfservd* instead for it. There is actually no rational reason for why some trust one tool over another. The question was asked to discover whether information change perceptions.

In the next question we are wondering if users often use *cfagent* with –K flag.

	In number	**In percent**
Yes	15	29%
No	36	70%

Table 5.1.5.3: Participants use of the –K flag.

As we can see from table 5.1.5.3, most of the participant doesn't use *cfagent* with –K flag. Cfengine has a built-in functionality that makes sure that configuration actions don't take place too frequently. If you run *cfagent* with the –K flag, it will ignore the locks. Participants, who have answered "no" on this question, most probably don't have the necessity of it. Use of the –K flag will take place in special case where you have to take many configuration actions.

	In number	**In percent**
Yes	16	30%
No	36	69%

Table 5.1.5.4: Participants experience with the pull-only architecture.

Table 5.1.5.4 show us if participants ever have found it difficult to solve a file distribution problem in cfengine due to it's pull-only architecture. This pull-only architecture is unfamiliar for most of the users, because they are used to push architecture. That can be the reason why the most of the

participants have answered “no” this question. Another reason can be that some participants have never used it, so do they have problems with it?

	In number	In percent
Yes	47	88%
No	6	11%

Table 5.1.5.5: The table shows if participants use firewall.

	In number	In percent
Yes	20	37%
No	33	62%

Table 5.1.5.6: This table shows if participant operate cfengine through a firewall.

Table 5.1.5.5 show if participants’ organization/company has a firewall. As we can see from the table most of the users have firewall at their organization. A firewall prevents unauthorized access to or from a network. With firewall you have a better control of the incoming and outgoing traffic. Some operating systems nowadays have an integrated firewall.
Table 5.1.5.6 show if participants operate cfengine through a firewall. As we can se from the table, most of the participants don’t do that. As mentioned earlier cfengine pull files than pushing them. With other words cfengine’s security model is about protecting each individual host. Firewall thinks different; it says that some host are more important than other. A problem can occur when you try to copy files from a host inside firewall to a host outside the firewall. There are two ways to get files through firewall:

- Cfengine – pull from inside to outside
- A manual push – push from the inside to outside

So there can be an advantage to operate cfengine through a firewall.

	In number	In percent
Yes	37	71%
No	15	28%

Table 5.1.5.7: Participants trust on their own ability to use cfengine safely.

Table 5.1.5.7 show participants own ability to trust that they use cfengine safely. As we can see from the table, 72% of the participants do trust their own ability to use cfengine safely. Users, who have used cfengine for long time, will trust their actions more than newcomers. The more you use cfengine, the better will the ability be to trust yourself, if you don’t make to many mistakes.

In the next questions, participants are supposed to answer yes or no to some statements. The statements are:

S1. Complexity is a potential security risk
S2. It is necessary for a human to monitor hosts at all times
S3. The organization of a policy should be sequential
S4. Networks should be organized hierarchically
S5. Simplest is usually best

S6. It is necessary to specify every detail of a machine's configuration.
S7. Machines should be as similar as possible
S8. It is desirable to have a diverse collection of configurations
S9. The freedom of the end user is a high priority in a system
S10 Users must sacrifice some freedoms for the good of the networks

	In Number		In Percent	
	True	**False**	**True**	**False**
S1	52	1	98%	1%
S2	8	44	15%	84%
S3	11	39	22%	78%
S4	29	22	56%	43%
S5	46	7	86%	13%
S6	7	44	13%	86%
S7	48	5	90%	9%
S8	13	38	25%	74%
S9	25	26	49%	50%
S10	43	10	81%	18%

Table 5.1.5.8: Participants answer to some statements.

Table 5.1.5.7 shows if participants agree or disagree with the statements we have mentioned. Most of the statements are advices and some of them are unwritten rules. These statements can't be forced on some one; it is up to each system administrator. Let us go through the statements:

- S1:
  - Complexity is a security risk – if a system is Complex, it will be easier for an attacker
- S2:
  - It is not necessary – why do we have cfengine?
- S3:
  - Order of policy doesn't matter
- S4:
  - It is easy to follow a hierarchically organized network
- S5:
  - Unwritten rule – simplest is usually best
- S6:
  - That is not necessary – just specify the relevant information
- S7:
  - If the machines are as similar as possible, it will be a lot easier for a system administrator to administer them.
- S8:
  - Not necessary, up to each system administrator
- S9:
  - Not highly prioritised – can cause a lot of junk, system overload, etc
- S10:
  - Users must expect that they have to sacrifice some freedoms for the good of the network.

The two last statements have been placed there by purpose.

- The freedom of the end user is a high priority in a system
- Users must sacrifice some freedoms for the good of the networks

The thing we must keep in mind is that this questionnaire is directed to system administrators. As we can see from table 5.1.5.7, the gap between participants who have answered "yes" and those who have answered "no" on S9 is minimal. With other words, about half of them have answered that the freedom of the end user is a high priority in a system. But on the next statement have 81% of the participants answered "yes" on the statement that users must sacrifice some freedoms for the good of the network. These two statements are almost the same. Have some of the participants contradicted themselves?

5.1.6 Part 6: Adoption

This part is about whether cfengine could be made easier to adopt.

The first question is about how participants summarize the development of understanding of cfengine over time. The alternatives are:

a) A straight line growing with time
b) Slow start and rapid progress later
c) Rapid start and slow progress later

	In number	In percent
a)	12	22%
b)	29	54%
c)	12	22%

Table 5.1.6.1: Participants summarization of their development of understanding of cfengine.

As we can se from table 5.1.6.1, most of the participants have answered that they had a slow start and rapid progress later. This type of development is quite normal and was expected. Cfengine is not that difficult to understand. It can be hard to understand in the beginning, but when you understand the basic principles; your progress will increase rapidly. 54% of the participants have answered b). As we can see from the table, 22% of the participants have answered that they have had a straight line growing with time. These participants must have read the documentation properly. These users of cfengine had most probably no problems with understanding of cfengine. Another 22% have answered that they had rapid start and slow progress later. These users had most probably no problems with the basic functionality, but the problems occurred gradually when they maybe started to use the more advanced features.

In the next question, participants are supposed to rate how easy it is to debug problems they have with cfengine, on a scale from 1 – 5.

	In number	In percent
1	2	3%
2	11	20%
3	23	43%
4	16	30%
5	1	1%

Table 5.1.6.2: Participants experience with debugging problems with cfengine.

As we can see from the table 5.1.6.2, most of the participants have rated three and four. 13 of the participants have experienced problems with debugging in cfengine. Newcomers will always have problems with various aspects of cfengine. Participants who have rated one or two are most probably newcomers. The one participant, who has rated five, has to be a pretty experienced user of cfengine. So, an average Jo will experience problems with debugging in cfengine from time to time.

	In number	In percent
Better documentation	3	13,0%
Improved error messages	5	21,7%
Better reporting from the components	5	21,7%
A debugger component	6	26,1%
Don't know	4	17,5%

Table 5.1.6.3: Improvements participants want in debugging.

Table 5.1.6.3 shows what kind of improvements, participants want in debugging in cfengine. As we can see from the table, most of the users want:

- Improved error messages
- Better reporting
- A separate debugging component

The typically answers form the participants were:

- A debugger component
- Better error messages & documentation.
- Clearer log messages
- Ability to debug just certain sections, like editfiles or files, etc.
- Improved *cfservd* debug output.
- Better log messages
- Some sort of logging mechanism for error messages

As we can see from the table, 26% of the participants want a separate debugging component.

Highest priority: Improved error message, better reporting and a possible separate debugging component

Lowest priority: Better documentation

In the next question, we were wondering how users configuration has grown over time. The alternatives are:

a) A straight line growing with time
b) Slow start and rapid progress later (concave)Rapid start and slow progress later (convex)

	In number	In percent
a)	22	46%
b)	25	53%

Table 5.1.6.4: This table shows how participants' configuration has grown over time.

As we can see from table 5.1.6.4, 53% of the participants have answered that their configuration has grown concave or convex. This type of progress is usual as mentioned earlier. Participants who have answered a), most probably had full control over their configuration.

	In Number	In Percent
1 – 10	19	36%
10 – 100	29	55%
100 – 1.000	4	7%

Table 5.1.6.5: How many hosts, participants would need to have before they consider cfengine as cost-effective.

As we can see from table 5.1.6.5, most of the participants think they would need to have between 10 and 100 host before they consider cfengine as cost-effective. 7% have answered 100 – 1.000 hosts. As mentioned earlier, it depends on the size of the organization and how many users and host they deal with. Small companies with a few hosts will never consider cfengine as cost-effective. However, we know that cfengine can cover up to 100.000 hosts. So covering host should not be a problem

In the next question we were wondering if it would be easier for users' organization/company to adopt cfengine if they could pay for it.

	In number	In percent
Yes	3	5%
No	50	94%

Table 5.1.6.6: The table shows if it would be easier to adopt cfengine if they could pay for cfengine.

As we can se from table 5.1.6.6, almost no one thinks that it will help them if they could pay for cfengine. Some companies/organizations don't want to use tools that don't have any licence. 5% of the participants have answered that it would be easier for them to adopt cfengine if they could pay for it. These participants are most probably newcomers. Maybe they think they will get more/better support if they could pay for the tool.

Next question is an open one. Participants are supposed to mention which software they consider to be an alternative to cfengine.

	In number	In percent
Scripts	10	32,3%
Radmind	6	19,3%
Opsware	3	9,6%
Others	6	19,4%
None	6	19,4%

Table 5.1.6.7: Participants suggestion to alternative tools to cfengine.

As we can from table 5.1.6.7, 32,3% of the participants have answered that home grown script can be alternative to cfengine. This is the biggest group. Participants have mentioned scripting languages like, shell, perl and python. Some of the participants have mentioned Radmind and Opsware. Those participants may have some experience with these two tools, since they mentioned that theses tools can measure up to cfengine. 19,4% of the participants have mentioned some other tools like Microsoft SMS, rsync, PIKT, Apple Remote Desktop and ISconf. Six of the participants have answered that none other software is comparable to cfengine. They are most probably very satisfied with cfengine.

5.1.7 Part 7: Training and Documentation

The purpose of this part is to get some feedback from users about documentation and the online help they can get.

	In number	In percent
Yes	9	17%
No	43	82%

Table 5.1.7.1: This table shows if participants have taken course to learn about cfengine.

Table 5.1.7.1 show if participants have ever taken a course to learn cfengine. As we can see from the table, only nine of the participants have taken course to learn about Cfengine. Most of the participants have not taken any courses; they have managed to understand cfengine by reading the documentation. The founder of cfengine, Mark Burgess, has held talks about cfengine several times among other things, at LISA conferences. He has presented it several times at Oslo University College too.
Next question is about if users are willing to take a course to improve their understanding of cfengine.

	In number	In percent
Yes	30	60%
No	20	40%

Table 5.1.7.2: This table shows if participants are willing to take a course to improve their understanding of cfengine.

As we can see form table 5.1.7.2, most of the participant are willing to take a course to improve their understanding of cfengine. 40% of the participants have answered that they are not willing to take a course. This group of participants can include the participants who answered no yes on the last

question. Since there are so many of them who want a course, maybe Mark should hold one for them.

	In number	In percent
Yes	41	78%
No	11	21%

Table 5.1.7.3: Participants answer to whether they read textbooks about system administration or not.

Table 5.1.7.3 show if participants read textbooks about system administration. As we can se from the table, most of the users do read text books. 21% have answered that they don't. It is important to read som textbooks to improve the knowledge. One textbook to recommend is: *Principles of Network and System Administration*, written by Mark Burgess. This textbook includes good guidance to administer a network. It contains a lot of important principles and unwritten rules due to administer a network.

In the next question we were wondering if the web and discussion groups are their primary source of information.

	In number	In percent
Yes	48	94%
No	3	5%

Table 5.1.7.4: This table shows if discussion groups are participants' primary source of information.

As we can see from table 5.1.7.4, 94% of the participants have answered that web and discussion groups are their primary source of information. There is wiki on the web for cfengine, *http://cfwiki.org*. Here, users can share their experience with each other and try to help those who have any type of problems with the tool. This is excellent way to get help for those who need that. Three of the participants have answered "no" on the question. There primary source is most probably the documentation.

In the next questions, participants are supposed to rate the documentation and support on a scale from 1 – 3, where 1 is bad, 2 is ok and 3 is good. The different questions are:

R1. The quality of documentation
R2. How easy the documentation is to understand
R3. Whether it covers important topics
R4. Whether it has relevant examples
R5. The quality of support you get from mail groups
R6. The speed with which you get help
R7. The speed at which bugs are fixed

	In number			In percent		
	1	**2**	**3**	**1**	**2**	**3**
R1	3	33	16	5%	63%	30%
R2	10	29	13	19%	55%	25%
R3	3	20	28	5%	39%	54%
R4	8	31	11	16%	62%	22%
R5	3	17	31	5%	33%	60%
R6	3	20	28	5%	39%	54%
R7	1	29	19	2%	59%	38%

Table 5.1.7.5: Participants rating of different questions about documentation and support.

Table 5.1.7.5 shows how participants have rated the different aspects of support and documentation.

- Most of the participants think that document is "ok"
  - Maybe some improvements are needed?
- Most of the participants think that the documentation is easy to read – "ok"
  - Is it possible to make the document a bit easier?
- All the important topics are covered – "good"
  - Most of the participants have rated with a three.
- The document includes relevant examples – "ok"
  - Some more relevant examples should be added to the documentation.
- Quality of support form mail groups are "good"
  - Participants are happy with the quality of mail groups
- Participants get help fast – "good"
  - Most of the participants are happy with the speed of help
- Bugs are fixed pretty fast – "ok"
  - Is it possible to fix bugs faster?

As we can see from the table, participants are more satisfied with the support from mail groups than the documentation in general. I t should be possible for Mark to write a improved documentation, maybe a bit more detailed and document that are easier to read. Document could also include some more relevant examples for newcomers. The thing participants are less satisfied with is the speed at which bugs are fixed. Some bugs are easy to fix and some can take long time. Maybe it should be established a team whit cfengine experts who could help fixing bugs. So, in general we can say that the support and documentation is better than applicable.

The next question is about what users mainly use cfengine for. The alternatives are:

a) Warn about problems
b) Fix problems
c) Something else

	In number	In percent
a)	2	4%
b)	37	74%
c)	11	22%

Table 5.1.7.6: Participants answer to what they use cfengine mainly for.

As we can see from the table, 74% of the participants use cfengine for fix problems and only 2 of the participants use cfengine for warn about the problems. As mentioned earlier, cfengine has a lot of functionalities; it is up to each user what of the functionality they want to utilize. 11% of the participants have answered that they use cfengine for something else. Other functionalities users can utilize are; a file-copying/editing utility, garbage collection of both files and processes, controlled execution of scripts, check and set permissions etc.

In the next two questions, participants are supposed to rate in a scale of 1 – 3, where 1 is infrequent, 2 is "once a while" and 3 is often. The questions are:

R1. How often you upgrade
R2. Ever had problems with upgrading

	In number			In percent		
	1	**2**	**3**	**1**	**2**	**3**
R1	9	26	15	18%	52%	30%
R2	23	16	10	46%	32%	20%

Table 7.1.7.7: Participants rating of upgrading in cfengine.

As we can see from table 7.1.7.7, most of the participants upgrade "once in a while". One should upgrade frequently, or mail should be sent to user when new upgradings are available. Some upgradings are more important than others, so if you want to be on the safe side, upgrade frequently. The table shows that most of the participants, 46% have experienced problems with upgrading. The most common problems can be; slowness, server overload etc.

In the two next questions, participants are supposed to feel free to come with suggestions about:

- Do cfengine miss any features
- What changes users would prefer in a new version of cfengine

Since these two questions are about the same, the suggestions are been melting together.

	In number	In percent
Simpler setup	8	19,0%
Consistency	7	16,7%
Integration of scripting languages	2	4,8%
Simpler (in general)	5	11,9%
Better language	2	4,8%
Better editfiles	3	7,1%
Better management	5	11,9%
Update files directly from cvs	3	7,1%
Other things	5	11,9%
Everything is ok	2	4,8%

Table 7.1.7.8: This table show what improvements participants want in cfengine.

As we can see from table 7.1.7.8, most of the participants want simpler cfengine in general, simpler setup etc. Consistency is still a hot theme. Most of these suggestions have already been mentioned earlier in the questionnaire. The group "other things" contains No actionsequence, background processes, update checksum silently and secure way to push up information. A priority will be given to these suggestions.

Highest priority: Simpler cfengine, consistency and better management (file and user)

Medium priority: Integration of scripting languages, better language, better editfiles and update files directly from cvs.

Lowest priority: No actionsequence, background processes, update checksum silently and secure way to push up information.

In the last question, we were wondering if participants would be willing to pay for cfengine consulting help.

	In number	In percent
Yes	12	37,5%
No	17	53,1%
Maybe	3	9,4%

Table 7.1.7.9: This table shows whether participants are willing to pay for consulting help.

As we can see from table 7.1.7.9, most of the participants are not willing to pay for the consulting help. 37,5% pf the participants would way for a consulting help. These participants can be newcomers or users who have experienced problems with cfengine for a long time. The alternatives are to establish consulting help or help free through mailing groups.

5.2 Summary

Part 1 has given us some background information of the companies where participants work. As expected, most of the users are form the USA, but cfengine is spreading around the world. A lot of participants have answered

that they in future will expand their use of cfengine. As mentioned, the most of the participants will most probably use cfengine in the future. It seems like no one are directly dissatisfied with the tool, but rather want some changes. A combination of the operating systems Solaris and Linux is the most popular amongst the participants. No one runs cfengine directly on windows operating systems. Most of the users are so satisfied with the tool that they don't consider other configuration management tools. Home grown tools and scripting are the most popular tools amongst participants who use other tools beside cfengine.

Part 2 has given us some information about the participants' most basic understanding of the principles of cfengine. The basic understanding is very important for users who are planning to extend their use of cfengine for instance from one machine to several. Most of the participants have the basic philosophy under control. Those of the participants who have implemented cfengine completely in their network, most probably understand the cfengine completely. They usually make their own scripts and make it more personalize. Understanding of convergence is important. We have an ideal state, and we are always supposed to end up in this state, independent of how many times you run cfengine. It is important to know that each host has individual configuration. That means no external system can make changes to a host or send it instructions. The host are autonomous. Adaptive locks and the "too soon" functionality gives the users control over their operations. This is also a useful functionality that users should be aware of. It was a bit disappointing that the most of the participants didn't know what adaptive locks were. Adaptive locks take care of spamming problem and sub-process hanging. This is important, whit a view to security. Most of the users have answered that they have experienced that cfengine tells that it is "too soon" to do something. Didn't those participants wonder why cfengine is saying so? Didn't they bother of this or did they think that this is normal? If they had read the documentation, they had discovered that the "too soon" functionality have connection with adaptive locks. Otherwise, most of the participants have answered right on the rest of the questions.

Part 3 has given us some information about the cfengine's language interface, and what participants think of it. We have now found out that most of the participants break up their configuration files split into several files. An advantage can be that we will get a better overview of our configuration. Most of the participants are able to "think" cfengine terms and understand the language. Since declarative language of cfengine is unfamiliar for a few users, they often get syntax error when they try to express their own ideas. Gradually, when user get used to the cfengine, most of them will rather express their ideas in cfengine scripts than perl scripts. Most of the participant thinks that the language is well-suited to its task. Many of the participants deal with a very large amount of configuration files. Large companies total byte count will always be much bigger than smaller companies total byte count. Most of the participants want more consistency in the language and better editfiles.

From part 4, we have now got some information about the usage of cfengine amongst the participants. We have got a view over what components participants use and which they don't and why. A lot more participants will use the component, *cfenvgraph* in the future if the data from this component can be seen on a web-page. As expected the most popular components are *cfagent*, *cfservd* and *cfexecd*. These are also the most important ones to. We have for instance found out that most of the participants use garbage collection of files but the gap between them and those who don't use it is not so big. Most of the participants don't use cfengine's tripwire functionality for checksumming. Participants have also mentioned some reasons for why they don't use the tripwire functionality. Most of the participants, who don't use it, use other tools, like tripwire, *osiris* etc. Most of the user use cfengine in an ad hoc way or proscriptively. Participants, who use a version control tool, prefer to use CVS, RCS or Subversion. Most of the participants have right on the control questions about the components. Simultaneous have a quite a lot of them answered wrong too. Are the components good enough explained in the documentation?

By part 5, we were trying to find out participants impression about security in the context of cfengine. Most of the users do trust in cfengine, that it behaves as they expect. Users that have long enough experience with cfengine trust the different component like *cfagent* and *cfservd* more than other similar tools like tripwire and *ssh*. One thing we must keep in mind is that a person will always trust a tool he/she is used to, as long as it satisfies him/her. It will always be difficult to make that person to use a new tool. Tripwire is a useful tool for a system administrator. It makes a database of files on the disk and their checksum. This way we can find out if the files are been changed lately. Cfengine and tripwire have almost the same functionality. The only different is that tripwire also includes information about file permissions in the database. Cfengine have a pull-only architecture. That means cfengine pull files than pushing them. This architecture is unaccustomed for most of the users, since they may are used to a push-architecture from earlier tools. Almost every participant use firewall, but only a minority of them operate cfengine through their firewall. It is an advantage to operate cfengine through a firewall; you can for instance avoid the problem with file copying. Cfengine can pull from inside to outside. There are a lot of unwritten rules in networking, or we can call them good advices. Most of the participants agree to the right statements.
When it comes to security, most of the participants have pretty good experience and understand the most important principles. A network is secure in these participants hand.

From part 6, have we got some information form participants about whether cfengine could be easier to adopt. We have received some informative feedback. Most of the participants have summarized theire development understanding of cfengine as slow start and rapid progress later. It depends on how quickly and properly you read the documentation. Most of the participants experience problems when debugging they have with cfengine. We have found out that most of the participants want some improvements in debugging. They want improved error messages, better reporting and a

separate debugging component. Since most of the participants have mentioned these tools, Mark Burgess should have this in mind when developing a new version of cfengine. For most of the participants, the configuration has grown as concave/convex over time. But the gap between them and the other, who answered that their configuration has grown as a straight line growing with time, is minimal. Most of the participants have mentioned that if they would need 100 – 1.000 host before they can consider cfengine as cost-effective. Almost no one of the participants are would adopt cfengine, if they could pay for it. A lot of the participants have suggested Radmind, Opswate as comparable to cfengine.

Part 7 have given us some important information about the documentation and the support mailing groups. Only a few numbers of participants have taken any course to learn about cfengine, but a lot more want to take course to improve their understanding of cfengine. Most of the participants have read textbooks about system administration. A very big group of participants use discussion groups on the web as their primary source of information. Most of the participants are medium satisfied with the documentation. Most of the participants are satisfied with he mailing/discussion groups. They feel they get quick help and good information. Most of the users upgrade “once in a while”. Users of cfengine should upgrade frequently, because of high importance of some of them. Most of the participants have experienced problems with upgrading, something to think for Mark. Things Mark should prioritize highly are; Simpler cfengine, consistency and better management (file and user). These suggestions have been mentioned several times in the questionnaire. Some other things Mark should be aware of are; Integration of scripting languages, better language, better editfiles and update files directly from cvs. This is the judgement from the participants.

Chapter 6

Conclusions and Summary

6.1 Concluding Words

This project concerns a user survey about practices using cfengine. The aim was to find out if users of cfengine have understood the purpose and philosophy of cfengine. The big questions we wanted answers to were:

- How easy is it to use/learn?
- How good is it at solving the problem of configuration and maintenance?
- Do users really understand the principles of cfengine?
- Do you feel safe using cfengine?
- How easy is to express your policy ideas?
- What improvements are needed in cfengine?

The first thing we have to be aware of is that there are about 300 organizations/companies that use cfengine. We got only 63 answers. So, the result of the analysis will not give us the right picture of the users. All the participants have not answered all the questions. On some questions we only received answers form about 30 participants. Another point to note is that we are only dealing with participants who use cfengine. When it comes to comparing cfengine or its components with other similar tools, most of them will agree that cfengine is better than other similar tools. The reason is simple, they use cfengine on daily basis and they are used to it. We must also assume a margin of error. Some of the questions could have been misunderstood by some participants. The result of misunderstandings can be that those participants jump over the question or just guess an answer. The margin of error is almost impossible to avoid.

How easy is cfengine to use?
Most of the participants have answered that cfengine is easy to use to a certain degree. Most of them have summarized their development of understanding of cfengine as slow start and than rapid progress. Reading the documentation and maybe taking some course can lead to a better start. Some users have problems with the declarative language of cfengine. But most of them think that the language is well-suited. Most of the participants think that they are able to "think" in cfengine terms and understand the language. Most of the participants are familiar with the most of the components of cfengine and their functionality. The most used components are *cfagent, cfservd* and *cfexecd*. Documentation has an important part when it comes to the understanding the tool. The analysis shows that participants think the discussion groups are a better source to get information than the

documentation. When a new tool is adopted, most of the people have starting problems. These problems can be avoided by reading carefully through the documentation.

How good is it at solving problem of configuration and maintenance?
The first thing to be mentioned here is that most of the participants have experienced problems with debugging in cfengine. Most of the participants break their configuration files into several files, to most probably get better overview. Big companies/organizations will always have larger configuration files than small companies/organizations. Some of the participants have experienced problems when adding their own ideas in configuration files. Scripting language can be a reason. Maybe the pull-only architecture of cfengine is a problem for some users. Most of the participants use the functionality, garbage collection of files/processes. A few participants have also got syntax error when they try to express their own ideas. Reasons can be that they are using the wrong command, the syntax is difficult to learn/recall, mixture of different languages, etc.

Do users really understand the principles of cfengine?
The basic understanding of cfengine's principles is important, if you are planning to implement the tool in your network. Most of the participants have the basic philosophy and understanding of principles under control. Some of the participants have understood it completely and manage to make their own scripts to personalize it. Almost every one of the participants understands the term convergence. By convergence we mean that we have an ideal state, and we are always supposed to end up in this state, independent of how many times you run cfengine. Most of the participants have understood that the configuration is individual to each host. All the host are fully autonomous and may choose to download new instructions from somewhere if they wish. Security means that no external system can alter the configuration of a host. But, most of the participants were not familiar with the "too soon" functionality and adaptive locks. Adaptive locks and "too soon" functionality give users full control over their operations.

Do they feel safe using cfengine?
As we can see from the analysis most of the participants trust cfengine, and that it acts as expected. Users that have long enough experience with cfengine trust components like *cfagent* and *cfservd* more than similar tools like tripwire and *ssh*. Users of a tool will always trust the tool they are most satisfied with. Almost every participant use firewall at their company/organization, but only a few of them operate cfengine through a firewall. By operating cfengine through a firewall we can for instance avoid the problems with file copying. There are a lot of unwritten rules in networking, we can call them good advises. We added some statements in the questionnaire, and wanted participants to agree or disagree with them. Most of the participants agree to the right statements. When it comes to

security, most of the participants have pretty good experience and understand the most important principles.

How easy is to express own policy ideas?
As mentioned earlier in this chapter, most of the participants have experienced problems when they try to express their ideas. The reasons can be that they are using the wrong command, mixtures of different scripting languages etc. Maybe a improvement is needed. The documentation should include some more examples of how to add policy or express own ideas.

What improvements are needed in cfengine?
In the last section of the questionnaire, participants were supposed to mention what changes they would prefer in a new version of cfengine. Participants have mentioned several suggestions here, and some improvements have been repeated several times. A list over what improvements must be prioritised is mentioned in the next section of this chapter.

In the starting point of the project, we wanted answers to some "big" questions. The whole questionnaire was built around these questions. We have now got answers, and hopefully, Mark Burgess is satisfied with the result. He has also got some other information that can inspire him when developing a new version of cfengine.

6.2 Future Work

After reading through the analysis of the questionnaire, Mark Burgess should have a pointer to what should be improved or added to a new version of cfengine. All the information has been informative. From the questionnaire we both get the functionality users are satisfied with and dissatisfied with. The improvements and desired features are been classified.

Highest priority:

- Better editfiles
- Consistency
    - Mentioned several times in the questionnaire
- Improved error messages
    - Self-explained error messages
- Better reporting system
    - A reporting component can solve the problem
- Better debugging system
    - A separate debugging component can solve the problem
- Simpler cfengine
    - Improved documentation can help

- Better management
    - Improve user and file management

Medium priority:

- Integration of different scripting languages
    - Integration of perl, shell, etc
- Better language
- Update files directly from CVS
    - Add a version control functionality in cfengine

Lowest priority:

- Better documentation
- Better iteration
- No actionsequence
- Background processes
- Update checksum silently
- A secure way to push up information

We must have one thing in mind that only a minority of cfengine users have participated in the questionnaire, and those who have participated have their suggestions of improvements. We are not sure of the rest of the users will agree with these improvements. The point under highest priority has been mentioned by many participants. These points should be included in a possible new version of cfengine. The category, medium priority is mentioned by some participants, but not as many as in highest priority. To include these points or not in a new version is up to the developer. The last category, lowest priority includes improvements mentioned by a few participants. There are too few participants to take any action. It would be advantage to find out what the rest of the users have to say before thinking of improvements that are mentioned in medium priority and lowest priority. The improvements under highest priority are mentioned by so many participants that these should be included.

References

[1] M. Burgess, “A brief overview of the implementation of principles of system administration in cfengine”, Available at: http://www.iu.hio.no/~mark/papers/Cfengine_Principles.pdf, 2004

[2] M. Burgess, “Cfengine concepts”, http://www.cfengine.org/docs/cfengine-Tutorial.pdf, 2001

[3] M. Burgess, “Computer Immunology”, LISA, 1998 - usenix.org, 1998

[4] M. Burgess, “Cfengine: A Site Configuration Engine”, Available at: http://www.iu.hio.no/~mark/papers/paper1.pdf, 1995

[5] T. Zlatanov , “Intro to cfengine for system administration”, http://www-106.ibm.com/developerworks/linux/library/l-cfe.html, 2002

[6] M. Burgess, H. Haugerud, T. Reitan and S. Straumsnes “Measuring host normality”, ACM Transactions on Computing Systems, 2001.

[7] C. E. Shannon and W. Weaver, “The Mathematical Theory of Communication”, Bell System Technical Journal - portal.acm.org, 1948

[8] M. Burgess, “Cfengine-Tutorial”, Available at: http://www.cfengine.org/docs/cfengine-Tutorial.html

[9] M. Burgess and D. Skipitaris, “Adaptive Locks for Frequently Scheduled Tasks with Unpredictable Runtimes”, LISA, 1997 - usenix.org, 1997

[10] St. H. Kan, “Metrics and models in software quality engineering”, ACM SIGSOFT - portal.acm.org, 1996

[11] P. E. Munis, R. W. Phillips, J. T. Harding and R. A. Radice, “A Programming Process Study”, IBM Systems Journal - portal.acm.org, 1985

[12] C. Jones, “Critical Problems in Software Measurement”, Burlington, Mass.: Software Productivity Research, 1992

[13] J. Guaspari, “I Know It When I See It: A Modern Fable about Quality”, AMACOM American Management Association, 1985

[14] P. B. Crosby, “Quality is Free: The Art of Making Quality Certain”, New York: New American Library - dx.doi.org, 1979

[15] W. E. Deming, “Out of the Crises”, Cambridge, MA, MIT/Centre for Advanced Engineering, 1986

[16] P. Babich, “Customer Satisfaction: How Good Is Good Enough?”, Available at: http://www.geocities.com/Athens/Crete/3108/goodenuf.pdf, 1992

[17] T. Toarmina, “From Quality to Business Success”, Quality Progress Journal, 2002

[18] D. S. Walonick, “Survival Statistics”, StatPac Inc , 1997 - 2004

Appendix

The questionnaire:

GENERAL ITEMS
The purpose of these questions is to get a qualitative impression of your needs and the challenges you face. Please tell us:

1.1) **The name of your organization**

1.2) **How many users do you deal with?**

1.3) **Are you currently using cfengine in your company?**

Yes
No

1.4) **Would you like to use cfengine in the future?**

Yes
No

1.5) **Please comment on the previous answer.**

1.6) **What OSes do you use and how many machines of each do you have?**

1.7) **How many hosts does your cfengine configuration cover?**

1.8) **Mention any other configuration management tools that you use or have used.**

AWARENESS
The purpose of this section is to get an idea of how well acquainted you are with the fundamental concepts of cfengine. Please choose the best answer in each case:

2.1) **Please rate, on a scale of 1-5 (were 5 is good) how you preceive your grasp of the principles on which cfengine operates?**

1
2
3
4
5

2.2) **Convergence refers to the property that**

a) an action or operation should lead to the same final result, regardless of how many times you run cfengine.
b) the same actions are repeated again and again, regardless of when you run cfengine
c) the actions of the system are randomized to spread the load

2.3) **Which of the following best summarizes cfengine?**

a) Configuration is individual to each host. Security means that no external system can alter the configuration of a host, nor send it instructions. Hosts are fully autonomous and may choose to download new instructions from somewhere if they wish.
b) A central manager can decide to change any host in the network and must be obeyed. Each host does the work of the manager. Security is based on encryption of the transmitted instructions.
c) A central configuration file determines the rules by which every host will be patched. Security is based on having an authorized MD5 checksum on every file to patch, like Tripwire.

2.4) **The order in which operations are declared in a cfengine program**

Does not matter
Is Important
Don't know

2.5) **Do you know what a cfengine adaptive lock is?**

Yes
No

2.6) **Have you ever experienced that cfengine tells you it is "too soon" to do something**

yes
No

2.7) **A cfengine script could not be implemented by a Perl script**

True
False

LANGUAGE INTERFACE
The purpose of these questions is to evaluate your impressions of the cfengine language interface.

3.1) **Do you use import to break up your cfagent.conf configuration split into several files?**

Yes
No

3.2) **Do you mainly group your configuration instructions**

By operating system type
By organizational group (department)
By the role of the host in the network

3.3) **Do you often use shell commands embedded in cfengine to solve problems?**

Yes
No

3.4) **Do you feel more comfortable expressing your ideas for system configuration in**

perl
cfengine
shell
tcl

3.5) **How well do you feel that you are able to "think" in cfengine terms?**
Scale 1-5 (were 5 is good)

1
2

3
4
5

3.6) **How often do you experience syntax errors when trying to express your ideas?**

Often
from time to time
seldom
never

3.7) **Rate your impression of the consistency of the cfengine language**
Scale 1-5 where 5 is good

1
2
3
4
5

3.8) **Do you find the language to be well-suited to its task? (1-5)**

1
2
3
4
5

3.9) **How many configuration files do you have?**

3.10) **What is the total byte count of your configuration files?**

3.11) **Do you discuss changes in cfengine configuration as a team, or does everyone in your team make changes to the syntax on an ad hoc basis?**

Discuss
ad hoc

3.12) **What changes would you like to see in syntax?**

3.13) **Other comments:**

MODUS OPERANDI AND USAGE
The purpose of these questions is to map out your understanding and pratical use of cfengine.

4.1) **Do you perform garbage collection of files?**

Yes
No

4.2) **Do you use the Tripwire functionality for checksumming?**

Yes
No

4.3) **If the answer to the previous question was no, please say why not.**

4.4) **Do you perform garbage collection of processes?**

4.5) **Do you use cfengine**

in an ad hoc way – to solve a random selection of issues
proscriptively – to lay out plan for your whole system systematically
as a file-copying utility
as a file-editing utility

4.6) **Which of the cfengine programs do you use?**

cfagent
cfservd
cfexecd
cfrun
cfshow
cfenvd
cfenvgraph

4.7) **Do you look at the data from cfenvgraph?**

Yes
No

4.8) **If the answer to the previous question was yes, what are you looking for in the graphs?**

4.9) **Would you find it useful to be able to see cfenvgraph data via a web-page?**

Yes
No

4.10) **Do you test configuration changes before rolling them out?**

Yes
No

4.11) **Do you use any kind of version control on your configuration files? (Specify)**

4.12) **Please answer true or false to the following questions: Cfagent can copy files from cfservd**

True
False

4.13) **Cfrun sends policy instructions to cfagent**

True
False

4.14) **A misconfigured cfenvd can be a security risk**

True
False

4.15) **Cfenvd is used to monitor remote hosts**

True
False

4.16) **A misconfigured cfagent can be a security risk**

True
false

4.17) **A misconfigured cfservd can be a security risk**

True
False

SECURITY
The purpose of this section is to measure your ideas and impressions about security in the context of cfengine.

5.1) **Do you trust cfengine to behave as you expect?**

Yes
No

5.2) **Do you trust ssh more than you trust cfservd?**

Yes
No

5.3) **Do you trust Tripwire more than you trust cfagent?**

Yes
No

5.4) **Do you trust cfservd more than you trust http?**

Yes
No

5.5) **Do you often use cfagent with the -K flag?**

Yes
No

5.6) **Have you ever found it difficult to solve a file distribution problem in cfengine due to its pull-only architecture?**

Yes
No

5.7) **Does you organization/company have a firewall?**

Yes
No

5.8) **Do you have to operate cfengine through a firewall?**

Yes
No

5.9) **Do you trust your own ability to use cfengine safely (i.e. to not make serious mistakes)?**

Yes
No

5.10) **Which of the following do you agree with?**

5.11) **Complexity is a potential security risk**

Yes
No

5.12) **It is necessary for a human to monitor hosts at all times**

Yes
No

5.13) **The organization of a policy should be sequential**

Yes
No

5.14) **Networks should be organized hierarchically**

Yes
No

5.15) **Simplest is usually best**

Yes
No

5.16) **It is necessary to specify every detail of a machine's configuration.**

Yes
No

5.17) **Machines should be as similar as possible**

Yes
No

5.18) **It is desirable to have a diverse collection of configurations**

Yes
No

5.19) **The freedom of the end user is a high priority in a system**

Yes
No

5.20) **Users must sacrifice some freedoms for the good of the networks**

Yes
No

ADOPTION

These questions are about whether cfengine could be made easier to adopt.

6.1) **Which of the following curves most closely summarizes the development of understanding of cfengine over time**

A straight line growing with time
Slow start and rapid progress later
Rapid start and slow progress later

6.2) **How easy is it for you to debug problems you have with cfengine? (1-5)**

1
2
3
4
5

6.3) **What tools would you like to have to help debug problems?**

6.4) **How has your configuration grown over time?**

A straight line growing with time
Slow start and rapid progress later (concave)
Rapid start and slow progress later (convex)

6.5) **How many hosts would you need to have before you considered it cost-effective to implement cfengine?**

1-10
10-100
100-1000

6.6) **Would it be easier for your organization/company to adopt cfengine if you could pay for it?**

Yes
No

6.7) **Which software would you consider to be an alternative to cfengine?**

TRAINING AND DOCUMENTATION

7.1) **Have you ever taken a course to learn about cfengine?**

Yes
No

7.2) **Would you like to take courses to improve your understanding of cfengine?**

Yes
No

7.3) **Do you read textbooks about system administration?**

Yes
No

7.4) **The web and discussion groups are you primary source of information**

yes
No

7.5) **How would you rate the documentation on a scale of 1 to 3 (where 1 is bad, 2 is ok and 3 is good)**

7.6) **The quality of documentation**

1
2
3

7.7) **How easy the documentation is to understand**

1
2

3

7.8) **Whether it covers important topics**

1
2
3

7.9) **Whether it has relevant examples**

1
2
3

7.10) **How do you rate**

7.11) **The quality of support you get from mail groups**

1
2
3

7.12) **The speed with which you get help**

1
2
3

7.13) **The speed at which bugs are fixed**

1
2
3

7.14) **What do you mainly use cfengine for?**

Warn about problems
Fix problems
Something else

7.15) **3) How would you rate in a scale of 1 to 3 (where 1 infrequent, 2 is “once a while" and 3 is often)**

7.16) **How often you upgrade**

1
2
3

7.17) **Ever had problems with upgrading**

1
2
3

7.18) **4) Do you feel cfengine is missing any features? If so, please mention these**

7.19) **What changes would you prefer in a new version of cfengine?**

7.20) **Would you be willing to pay for cfengine consulting help?**