Designing a scalable architecture from the start

Tom Hendrickx, Yuri Schaeffer

March 29, 2009

Abstract

This paper describes our research in setting up a scalable IT infrastructure in small or starting organisations. We will explain what it means and why one would want to do it. After highlighting which principles are relevant to achieve scalability, we will present a guide to help system administrators set up their infrastructurein the form of a checklist to identify the required steps.

Contents

1 Introduction

For the vast majority of people in this world a computer is nothing more than a tool, they need to do their work and use a computer to do this efficiently. It is not of any interest how it works as long as it *just works*. This phenomenon causes a tendency for ad-hoc solutions for ICT problems: if it works now, it is good enough.

This approach could be the cause for problems in the future, especially for rapid growing or changing organisations. This is nothing new for any large, mature organisation. These will hopefully have dedicated ICT management, documented infrastructure and processes. However, many learned this the hard way. A way paved with problems, service downtime and excessive spending of effort and money.

1.1 Research question

When setting up or running a small company, it might be tempting to solve problems the quick and easy way. "Quick and easy" can be commonly read as "cheap" since in most small companies the financial resources are quite limited (see equation 1 for this line of thinking). It might, however, be a good idea to set up the ICT infrastructure with scalability in mind. In this paper we will look into how relatively small companies can set up and plan their infrastructure in such a way that over a long period of time they reduce their ICT cost. As a result we aim to produce a checklist which could be helpful for a starting company. We expect this to require more initial effort and knowledge thus we will justify this investment.

$$\left.\begin{array}{rcl} IT & = & Time \times Money \\ Time & = & Money \end{array}\right\} \Rightarrow IT = Money^2 \tag{1}$$

Goal: Create a checklist for setting up an ICT infrastructure, which is useful for starting and small companies and find answers on questions such as:

- Can we justify the initial costs?
- How to set up a scalable infrastructure?
- Where does it go wrong, when SME's construct their ICT infrastructure?

1.2 Scope

In our research we will limit ourselves to small and medium sized enterprises (SME). The services we will be looking at are employee centred, meaning services which support an employee in her daily work and help continuing the work flow (think e-mail or backup management).

1.3 Scalability

As an introduction, we provide the following summary which we feel to represent the general requirements and benefits of a scalable architecture:

- The architecture must be able to start out small, and scale to one's needs.
- Physical limitations, such as processor power, must not limit its ability to scale.
- Scaling the architecture must not increase its costs exponentially, but it should increase linear or preferably even better.

2 Motivation to scale

Why would one want to set up a scalable, but more "expensive" ICT infrastructure? Whether scalability is useful depends on one thing entirely: money. A geek may find a particular approach pretty or elegant, but all that matters for a business is the financial aspect. An approach that is eventually much cheaper than another, but not necessarily better in the long run, may still be preferred. Whether this will happen at all depends on multiple variables. The next section tries to unveil these dependencies.

2.1 Cost per employee

It is to be expected that setting up a scalable infrastructure is initially much more expensive than an ad-hoc solution. This does not have to be a bad thing. If in return the costs per employee decrease when the amount of employees increase, there will be a point where having the scalable solution gets cheaper. To be able to justify this initial expense one should know at which amount of employees this point will be. As an example we will define two approaches C that give the cost per employee. C_s gives a reducing cost per employee as a function of the amount of employees whilst C_n has a constant cost.

Independently of the approach, the cost of hardware for each user will be a constant H. We define the following functions:

$$C_n(e) = H + S_1 \tag{2}$$
$$C_s(e) = H + S_2 + S_3 \times 2^{-De} \tag{3}$$

with S_1 the service costs for 1 employee, S_2 the minumum service costs for 1 employee, S_3 the initial service costs for a scalable solution, D some damping factor and e the amount of employees. Of course these functions are assumptions and not completely accurate, they give however a good idea of our hypothesis. We need to find the intersection of both functions:

$$C_n(e) = C_s(e) \tag{4}$$
$$H + S_1 = H + S_2 + S_3 \times 2^{-De} \tag{5}$$

This gives us

$$e = \frac{\log_2 S_2 - \log_2 (S_1 - S_2)}{D} \tag{6}$$

When we know the values for S_1, S_2, S_3 and D we can calculate at how many employees e the scalable method will be cheaper. Unfortunately this does not tell us *when* it will be cheaper, which depends on the growing rate of the company. When a company grows very fast, e will be reached soon and the time span in which the scalable approach is more expensive is shorter. Therefor it will sooner be profitable to have a scalable solution.

2.2 Total costs

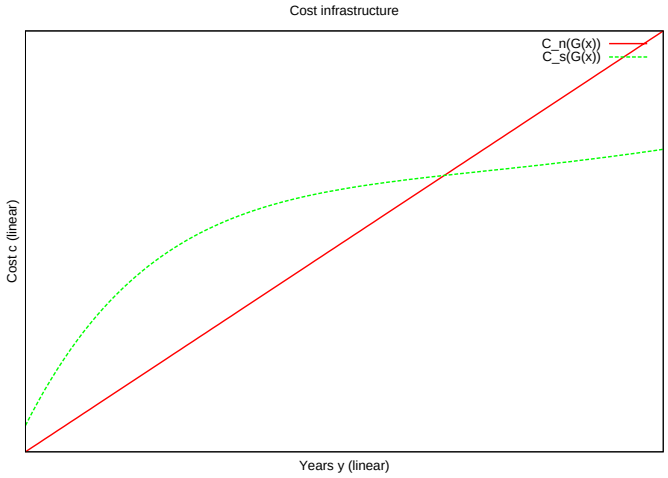

Figure 1: Cost of infrastructure for each year with linear growth.

To make an estimation in what year a scalable solution will start paying off we have to know how fast the company will grow. Naturally this is beyond our control, but a company is expected to have some idea about its growth. We define a function for the amount employees expected in each year $G(y)$. This can be any function, but for sake of an example we assume the following:

$$G(y) = 10 \times y \tag{7}$$

Each following year y the company grows by ten employees. Using this information we can calculate, for each year, what the total costs will be for all employees. For this we introduce the functions $Y(y)$, which are shown in Figure 1, together with initial setup costs for each approach I. This graph is of course greatly simplified, in reality the expenses are less smooth over time. Our aim is a scalable solution which requires a minimum of large expenses over a longer period of time.

$$Y_n(y) = I_n C_n(G(y)) = I_n 10y(H + S_1) \tag{8}$$
$$Y_s(y) = I_s C_s(G(y)) = I_s 10y(H + S_2 + S_3 \times 2^{-10Dy}) \tag{9}$$

If we take the integral of these functions we know how much the solutions have cost so far *in total* and for each year. We can use these integrals to find when the scalability starts paying off. We need to solve the following equation:

$$\int_0^y Y_n(y)dy = \int_0^y Y_s(y)dy$$

3 Pillars of scalability

We now know what a scalable architecture actually is, but how can it be achieved? This section will present a more overall guideline and will substantiate each component's importance. The next points are what we find to be essential to achieve a scalable architecture:

1. The architecture should be set up in modules as much as possible.
2. Open standards should be used wherever possible.
3. Repetitive tasks should be automated, for which policies and processes must be set in place.
4. Every module, process and policy should be documented from the very beginning.
5. All services must be monitored to predict and anticipate growth.

These points may sound theoretical, boring and like a lot of extra work, but they are the fundamental principle to limit the operational costs of an infrastructure. In the next few subsections, we elaborate on each of them to clarify their usefulness and purpose.

3.1 Modularisation

Modularisation reflects on the ability to view every part of the architecture as a separate component. Implementing this design enables:

- Taking away one service/module, without harming the system as a whole.
- Separating the software layer from the hardware layer.
- Building new modules on top of existing ones.

Maybe an obvious point, but not yet always implemented. For cost-saving aspects, some choose to install every service on a single server. When services must be replaced, removed or moved, this non-modular approach makes it hard to isolate that specific service. The service might become insufficiently isolated or damage other services in the process. Another problem is that it might be too dependant on the hardware which it currently runs on, which makes it more difficult to convert to newer hardware equipment.

3.2 Open standards

Using open standards have been labelled with both pros and cons. Not surprisingly, the cons mostly come from the vendor's side, whom might discredit open standards as limiting to their innovative capabilities. We on the other hand like to stimulate and promote the use of open standards in a commercial organisation. Using open standards prevents becoming the victim of *vendor lock-in*. Using proprietary protocols and formats can create a dependency for future projects on a specific vendor. This limits the variety of choice, but also makes one dependant of a vendor's price policy.

Secondly, open standards can be seen as an enabler of modularisation. As an example, imagine a mail server configured with the open protocol IMAP. The server is used in an organisation to access its mail, by use of a standard mail client. When customers, however, decide that their mail client is insufficient or out of date, it can be easily replaced by any mail client speaking the IMAP protocol. Their is no obligation at all to stay with a certain product, because (mostly) all speak the same protocol.

Open standards are however, not available for all purposes but we suggest to use products implementing them wherever possible.

3.3 Automation

Automation is mostly self-explaining, but might still be an issue. Automating certain tasks takes time, and because time means money, this aspect might be postponed for the long run. Once an organisation is growing, these unautomated tasks will require even more time to complete and may eventually become a heavy load to deal with for the system administrators. Although, automating tasks might be regarded unnecessarily, the sooner it is done, the sooner an infrastructure can scale more efficiently. The most simple example on this matter

is the automatic instalment and updating of computers. A few may be done without any problem by hand, but once the need is there to reinstall a lot of computers, one will really wish to have an automatic installation method in place for the operating system and needed software packages.

3.4 Documentation

Documentation may take some time and is probably not the most favourite occupation of a system administrator, but it does save a lot of time and problems in the long run. Having a good documentation enables to more easily introduce new members to the system administration team and delegate certain components to others. It also serves for one's own purpose as it helps remembering important design decisions and syntax.

Especially the former reasons are important with respect to scalability. With sufficient documentation and modularisation it is trivial to delegate a task completely to someone else. When you do not need to be queried about a service, it can be taken completely of your workload and mind and you will be available for other important projects.

3.5 Monitoring

As a final pillar we present monitoring, which implies to the monitoring of (but not exclusively):

- Network:
  - Load
  - Peaks
  - etc.
- Server:
  - CPU, memory usage
  - Storage capacity
  - etc.

This must provide the ability to timely predict the need to scale. For example, when monitoring the storage disks, one might see on time when only a few gigabytes of storage are left. This might then be enough to buy the needed disks of storage before the former disks are out of capacity. A timely prediction of the need to scale is a huge benefit because it avoids the need to make quick fixes, e.g when suddenly a e-mail server is out of capacity. Instead, when one notices that the mail server needs an upgrade in the coming months, it can be anticipated and plans can be made on time.

To be able to monitor the whole infrastructure, it is practical to aggregate as much as possible to one monitoring tool. Like this, one is able to have a complete overview over the whole infrastructure on a single screen.

4 Scaling issues for SME's

Because lots of SME's construct their ICT infrastructure without the future in mind, they will most likely face problems when their organisation grows. In this section, we will provide some examples of common mistakes, which can easily be prevented with some planning in advance. Some of these examples come from our own experiences, our course book[1] and others from a questionary which we presented to a company which handles ICT for multiple SME's.

4.1 User management

There are a few problems with the implementation of user management. Without the future in mind, this is mostly done in a very uncoordinated and scattered manner. This could be because different services are set up each on its own, without any links in between or because new services are added later on, without taking the existing services into account.

If this happens, it is possible that their exists a database holding user data from the email-server, another one holding user data with log-in permissions and so on. Such a set-up, however, means that multiple databases have to be kept up-to-date and synchronised. With just a few employees and databases it is still feasible, although rejective. Once the organisation reaches a certain size, this method will have lost its manageability. Databases will contain entropy, because data is not removed from each database, some databases may not be up-to-date and so on.

The general thought is to implement one central database which then can be used by systems such as: ActiveDirectory, (Open)LDAP, printer configurations, e-mail account informations, home directory's, permissions, etc.

4.2 User support / helpdesk

This possible issue is well demonstrated by an example which we received on our questionary. We received some numbers of how a workplace really costs on support, and these prices were something in the line of our expectations. As such, an organisation with 5 to 10 employees had a service cost of approximately €50 for each workspot, per month. When a larger organisation is taken into consideration, between 10 to 25 employees, the costs have already risen to €75 for each workspot, per month. This increase in costs, however, had nothing to do with the size of the technical environment, but with the needed support per user.

Whilst the smaller firm delegated, ICT related, questions to the other employees, this was not the case in the slightly larger organisation. Here was noticed that the same kind of questions were asked by different small "islands" inside the organisation. We assume the same counts even more for larger organisations, with more than 25 employees.

We believe that these issues can be solved by a well centralised support point. A possibility could be a wiki or a FAQ (Frequently asked questions)

page, where employees can find the answers they seek. This could be combined with a request tracker system such that a solved question can be posted on the wiki or FAQ on a website. Both idea's are not expensive to implement, and we are convinced that it results in a serious saving.

4.3 Server acquisition

Another typical example is how servers are set up in general. This means one piece of server hardware is bought to save expenses and all services are then provided from this single server. Again, when few employees need to use these services, the one server will be satisfactory. However, it does not comply with one of our previous key pillars: modularisation. Such as explained in section 3.1, this approach will not scale well when the demand grows.

What we want most is one server for each service, this is infeasible because of budget limitations. New technologies have changed this. Virtualization brings consolidation and modularisation together. It might be a buzz word, but virtualization is the way to go to both save expenses and separate one's services over different machines.

We have heard of two approaches in this field to achieve scalability.

- Acquire one big and more expensive server.
- Acquire one or more smaller servers.

Both approaches have their pros and cons, which we will set forth.

Acquiring the bigger server will be more expensive in the beginning and might be considered overkill. However, it does provide more "scalability" than two smaller servers having the same capacity and performance together. This is because the two smaller servers lose some "slicing capacity". For instance, when each server has one GB RAM left, but the system administrator needs a new server with two GB RAM, a new server will need to be bought. A virtual image will not be able to be partially on both servers, this problem will not occur with the bigger server.

Buying the smaller servers, however, has the advantage of being able to first buy one server and when the need arises buy the second one and so on. This makes possible to:

- Spread costs over a longer period.
- Buy a better second server, for the same price as the first.

These reasons make us prefer the smaller servers, over one big server. We feel that limiting expenses, whilst still preserving server consolidation, has a higher priority than losing that small slice of capacity by using multiple servers.

Whilst we advise smaller less expensive servers, this does not mean that we recommend to use desktop equipment instead of specific server hardware. It is important to buy decent equipment, which is manufactured for server purposes.

First of all, a server needs to be much more reliable than a normal desktop, and specific server hardware has been manufactured to serve this purpose. The costs might be higher in the beginning, but an organisation's infrastructure can not be run from an unreliable server. Using unreliable hardware would mean spending more time and money into keeping the infrastructure running and ultimately having more service down-time. It also has the problem of having the same appearance of an ordinary desktop (which it actually is). This has as a direct consequence that those servers, which are mostly put in a corner somewhere can be accidentally tripped over, have water poored over it or can be found and used by an employee who is in need of a new desktop. This server, sitting in the corner, might then be assumed to being used by no one and to be nothing more then an ordinary desktop. The employee may then claim this server as a personal desktop, make a clean install of an operating system on it and by doing so, remove all the server material. One might argue that this will never happen in its organisation, but the company from our questionary experiences those accidents at least once a year on one of its customers' sites.

4.4 Namespace management

The most occurring problem with namespaces is the possibility of collisions, through a lack of planning. A simple example for this is naming home directories with the first name of an employee. This might again be no problem with just a few employees, but when a company scales to the dozens, it is doubtfully that a few names will not collide. With a proper policy in place, this could easily be solved. For example, it could state to use "given name.surname" which has a higher chance of being a unique ID in the organisation. This could also be used for the email address, computer name (ID_PC), etc.

Another problem occurs when naming the servers and services in a confusing fashion. For example, imagine a server which provides the mailing service, but after a while also the printing service. If the server was firstly named as a mailing server, this might get really confusing when addressing the printing server. Instead, if the server itself is generically named (e.g. server01), and services can be addressed with cnames (e.g. mail.org.nl, print.org.nl), a lot of confusion can be prevented.

4.5 Desktop standardisation

Desktop standardisation comes from two angles. First, we will consider the software side. A way this can go wrong is if a company leaves its employees rather free in their choice of software packages, operating system and by giving them the necessary rights to install software by themselves.

This brings of course the necessary security risks with them, but it also makes managing every computer a total nightmare for the system administrator. This

last problem can be neglected if every employee would manage its own computer. Although in most cases this is not the fact, because most employees do not care about managing their own computer or do not have the necessary skills to do it proficiently. Then, if random packages are still installed by employees, a system administrator will have a really hard time when he needs to troubleshoot one of their computers. His job will be even harder when it includes educating and helping employees to work with a certain software package. Of course, this can not be expected if lots of different packages are installed. This is why we suggest to install standardised software packages on each desktop. Not only does this make it easier to troubleshoot any problems, it is also a huge relief when new installations or updates have to be done. If every computer can be handled in the same way, these installations and updates can be automated and as mentioned, this is one of the pillars for scalability.

So how does this save money? Not only management and training costs will decrease this way, but also the costs for acquiring the necessary software licences. Ultimately it comes down to this: the fewer in-house knowledge there must be about different products, the more money can be saved.

If there is no standardisation on software packages, the expenses will increase because one of the following two reasons: either an external firm, which handles the ICT department, will have many more questions of different nature to handle or more multiple in-house administrators will be needed who must be familiar with all the different used software packages and operating systems. This is a serious drain on a SME's budget, and as such we strongly recommend to standardise the desktop environment as much as possible. Combined with our suggestions of centralising all frequently asked questions at a local website or wiki, it will be even more clear. Because most employees are using the same software packages and desktops, they will probably encounter much of the same problems. Having these problems handled by a centralised FAQ, saves load on the system administration.

The second point to consider is the acquisition of the hardware itself. This point really depends of the current company's size, but from a certain point, it must be possible to buy a large amount of desktops from the same batch. This mainly has the following advantages:

- A homogene environment which is more easily to manage.
- Possible price reductions.
- Spare desktops are available when one is broken.

We acknowledge that this is not feasible for all SME's, but if a certain growth is expected, it should be possible to come to an arrangement with one's vendor to make a deal to be able to buy more from the exact same hardware at a later point.

4.6 Desktop/server management

If the previous point is taken into consideration and most computers have a standardised package of software, it is already much easier to manage them. The managment on itself, however, may not be forgotten. This means for instance that all software packages must be timely updated and that the computers must most preferably be scanned for viruses on a timely basis. Again, this might sound very obvious, but as well in our questionary as in personal encounters have we seen this to go wrong.

One example from our experience is a virus scan, which was always scheduled on a specific hour in an organisation which was specialised in selling heavy construction equipment. This was around lunch-time, but most employees had a different lunch-schedule. Some people were still working when the virus scanner activated. They turned it immediately off as it interfered with their work. (Partially because their computer hardware was not up to date anymore.)

From our questionary we learned the following. When an organisation forgets to update its software frequently and scan everything for viruses, there will be a certain point when the computers will have to be rebuild to a clean state, because too much entropy interferes with their proper working. This has as a consequence that much overtime hours will have to be made to get all computers back to their clean state. Overtime hours are equal to very expensive hours (think €125/h). Additionally, the company will be unable to resume its work for the good part of three days, whilst the equipment is being fixed.

Hopefully when reading this, one will think again when previously having assumed to be saving money, by being spare on computer management.

4.7 Efficient storage/backup

One of the heard mistakes made with tape robots is that most robots are bought without thinking ahead in time. A starter might only need small tapes, and will be perfectly happy with a maximum tape capacity of 300 GB and the lower price payed for the device. If the company, however, grows and the need arises for bigger tapes, they will be stuck with their old robot and need to buy a new one.

If a serious growth is predicted, it would be more economically to buy a robot which can handle also 1 TB tapes, for instance. This would allow to store backups on the appropriate size of tape and when a larger storage is needed, the switch can be made without the acquisition of a new tape robot.

Another problem which serves as an example to effectively manage the storage facilities comes from our questionary. A certain organisation has bought a brand new storage array, which purpose was to run as a backup. After a while, it appeared that the system was powered on, but running without even having a simple operating system installed on it. This indicates the importance of not only “taking backups”, but that it is also needed to test them. Even if it

certainly is running, it might be a good idea to check the backups once in while to see if it actually contains the data you expect.

In lots of cases when backups are being made, this only includes backing up the real "workable data". Additionally we advise to backup your server configurations. Whilst it might not seem the most "important" data to backup, having backups from these configuration files may save a lot of money when a server needs to be restored. If it is a complex server which needs to be restored, it can become a hazardous and lengthy task to get it in its original configuration, with lots of overtime hours to get the job done. (We assume of course that the configurations are also well documented.)

5 Checklist

The previous chapters consist of a lot of theory. What if you do not care too much about academics, but just want to set up IT infrastructure and do it right now. Of course you want to do it correct from the start but time and money are limited resources for a small starting company. What would you do?

This section aims at condensing that theory in a more concrete form which is supposed to be actually usable by real administrators.

5.1 Checklist for new services

When setting up a service in an existing organisation, either introducing a new service or upgrading an existing one, one should always keep the future in mind.

The system administrator is the person for designing the infrastructure and implementing it. No situation is the same and the sheer number of different configurations make a generic solution impossible. Therefor we can not hand over specific implementations but rather talk about concepts. In this section, for the sake of simplicity we will assume a service is implemented by a collection of software packages.

5.1.1 Step one: Define your needs

A good project starts with thinking, lots of it, even more so than doing. Instead of hasty installing the software you *assume* to meet your needs, it is wise to start with defining just what those needs are.

With these requirements you can find out what software fits best to your needs. It is a good idea to sort the requirements by importance. In practice, chances are that no existing package or even no self written software will completely meet your needs. Sorting requirements eases selecting candidate packages and adjusting the initial requirements.

5.1.2 Step two: Research dependencies

Once you know what you want and need for your new service you should investigate its dependencies. Likely your service consists of a set of software packages and some hardware. First of all these modules should not conflict with the existing soft and hardware. It is a good idea to inventory what set of services must be up and running in order for this new service to remain available. You should have a clear idea about the importance of the service and its required availability. All modules which your service depend on *should get at least* the same importance. Doing so helps in identifying existing services which importance has increased over time. For example a DNS service was first only required to browse the web, but over time many other services started to rely on it. It is important to notice this on time so appropriate action can be taken to ensure high availability of the DNS service. Equally important is to know what services depend on this one to better establish its own importance.

Another thing to keep in mind besides availability is the modularisation, to ensure that different packages can function correctly when one or more parts are being replaced by other, which may possibly be from different vendors. We strongly suggest to pay good attention to open source and open standards in general to prevent vendor lock-in.

5.1.3 Step three: Automation

After implementing the service, automate as much as possible. Even for a new service a list of most frequent uses and changes is easily compiled. Make sure you start automating the most repetitive tasks. This step can be somewhat delayed till one is familiarised with the software and its usage. However we recommend to do it while you are still actively busy with setting up the service and its usage is still fresh in mind. Be sure to look at the configuration of depending services, e.g. when creating a new user automatically, set up an e-mail box etc.

5.1.4 Step four: Monitoring

The next step is guaranteeing a continuous service. Make sure you are able to see in an instance, in what state a service is and preferably aggregate it to existing monitoring software. Also check the correct functioning and level of verbosity of the log files from the start, because doing so at the point when you need them is always too late. Not only are you interested in the current availability, but also in future availability. You want to know as soon as possible if and when it will run out of resources. Doing so enables you to be in control of the work to be done as it enables you to plan maintenance from up-front and whilst giving enough time to test and apply those changes.

5.1.5 Step five: Documenting

Although last in the enumeration, one should really document throughout the entire process of implementing and testing the service. Even the documentation of step 1 and 2 might prove useful in the future as changes are made or new services added. In the same line of thinking it is good to document important design decisions.

Make sure you write down how to perform common tasks even if you find them trivial at the moment. The documentation should be able to guide a new system administrator when first encountering the software. A good practice would be upon finishing the documentation to ask a peer to retrieve the document from where he would expect it, read it and preferably use it as a guide. By this, possible inaccuracies can be found and adapted in the documentation.

6 Conclusion

In our paper we tried to define what a scalable architecture is and why we find it so important. Then we discussed the fundamentals of such an architecture and the common issues found in practice. Finally we presented a checklist which guides system administrators of SME's in setting up an infrastructure in a scalable fashion.

In our introduction we asked ourselves a couple of questions which we will review in the following sections.

6.1 How to set up a scalable infrastructure?

We have answered this question by means of presenting five key pillars: modularisation, open standards, automation, documentation and monitoring. These five pillars cover the fundamentals of a scalable infrastructure and are explained in section 3.

Later on, in section 5, we condense the theory in the form of a checklist. This checklist has the purpose of guiding system administrators in setting up services in a new or existing architecture with respect to scalability. Hereby we keep the focus on small and beginning companies with a limited IT budged.

6.2 Can we justify the initial costs?

In section 2 we reasoned about the cost of IT management with respect to the number of workspaces. For an ad-hoc solution as well as a more structured scalable approach in setting up the IT architecture. Our hypothesis states that the latter approach is initially more expensive but starts to pay of over a relative short period of time.

Although we lack enough relevant financial information to back this up we believe from experiences of our own and other people, this will generally hold true. Most likely the truth is even less bright as bigger companies, when not set up correctly, tend to have their cost per workplace increased as the company grows.

6.3 Where does it go wrong, when SME's construct their ICT infrastructure?

This question was really important because we believe that SME's can learn a lot from the general mistakes which were made in the past by others. By summarising them in section 4, we want to give an overview of the most important issues which can easily be overcome by SME's when constructing their own ICT infrastructure.

In general it all boils down to an underestimation of the importance of a good architectural setup. This is related to a general lack of financial resources in small companies. Therefor the IT infrastructure is set up without the future

sufficiently in mind. We feel it is necessary to create awareness and we hope our work to be inspiring for SME's.

7 Future research

We feel it would be a good idea to create an additional *road map to a scalable ICT* to be used alongside our checklist. This road map should help new system administrators to prioritise the implementation of new services and help them in deciding when the optimal time would be to scale up services. It might not be necessary to set up all services in a scalable fashion right from the start, in case switching at a later time is relatively cheap. If one could identify these services and define the turning point, the system as a whole could still be scalable but financially more attractive.

To create awareness in SME's we believe it would help to have some kind of list containing common services and their price to implement. This list should make a very specific comparison of both approaches for each service in the long term. This will help educate the importance of a proper set up to the management.

8 Words of thanks

Special thanks goes to Jeffrey Barendse, who was so kind to share his experiences with us and has given us a better understanding of the impact of a more well-considered ICT infrastructure in a SME. His company WireITup, is an organization which helps mostly SME's in setting up and managing their ICT infrastructure.

References

[1] Thomas A. Limoncelli, Christina J. Hogan and Strata R. Chalup, *The Practice of System and Network Administration - Second Edition*