



EINRICHTUNG EINES NETZWERK-MONITORING-TOOL

Abschlussprojekt 2015
Edin Ahmetovic
Fachinformatiker
FR Systemintegration
Prüflingsnummer: 50070

Inhaltsverzeichnis

1. Einleitung	3
1.1 Ausbildungsbetrieb	3
1.2 Projektumfeld/Problembeschreibung	3
1.3 Prozessschnittstellen	3
2. Ausgangssituation und Projektbeschreibung	3
2.1 IST-Analyse	3
2.2 SOLL-Konzept	4
3. Evaluierung der Software	4
3.1 Informationsbeschaffung bezüglich Software	4
3.2 Vorstellung der verschiedenen Softwareprodukte	4
3.2.1 Nagios	4
3.2.2 Paessler	5
3.2.3 Big Brother BTF	5
3.3 Auswahl der Software	6
4 Ressourcen und Zeitplanung	7
4.1 Zeitplanung	7
4.2 Personalplanung	7
4.3 Sachmittelplanung	7
4.3.1 Auswahl der Linux Distribution	7
4.3.2 Hardwarespezifikationen	7
4.3.3 Hardwarekosten	8
4.4 Kostenplanung	8
5. Durchführung	8
5.1 Zusammenfassung der Clients und Dienste	8
5.2 Erstellung der Virtuellen Maschine	8
5.3 Vorbereitende Installationen und Konfigurationen	9
5.3.1 Hinzufügen der benötigten Libraries	9
5.3.2 Anbindung in die bestehende Windows Domäne	9
5.4 Installation und Konfiguration von Nagios	10
5.4.1 Benutzer und Gruppen	10
5.4.2 Installation von Nagios 4.0.7	10

5.4.3 Installation des Apache Webserver	11
5.4.4 Einrichtung der Sicherheitsaspekte	11
5.4.5 Konfiguration von Nagios	11
5.4.6 Erklärung der Template Datei	12
5.4.7 Erstellung von Hostgruppen	12
5.4.8 Erstellung von Hosts	12
5.4.9 Erstellung von Kontaktgruppen	13
5.4.10 Erstellung von Services	15
5.4.11 Konfiguration des NSClient++	15
6 Projektabschluss	16
6.1 Abschlusstests	16
6.2 Übergabe	16
7 Fazit	16
8. Literaturverzeichnis	17
8.1 Allgemeine Informationen	17
8.2 Produktinformationen	17
9 Anhang	17
9.1 Vorlagen der Template datei	17
9.2 Gantt – Diagramm	21

1. Einleitung

1.1 Ausbildungsbetrieb

Die Radiologie Vechta ist eine Ärztepartnerschaft mit ca. 100 Mitarbeitern. In dem Unternehmen werden verschiedene diagnostische Abteilungen wie z.B. die Radiologie, Mammographie, Strahlentherapie und Nuklearmedizin vereint. Die Radiologie Vechta besitzt neben dem Hauptstandort im Marienhospital Vechta noch zwei weitere Standorte in Wildeshausen und Cloppenburg. Die interne EDV-Abteilung der Radiologie Vechta, welche aus zwei Personen besteht, pflegt und wartet das interne Radiologie Informationssystem (RIS), das Archivsystem (PACS), die virtualisierte Serverlandschaft sowie das Netzwerk.

1.2 Projektumfeld/Problembeschreibung

In der Radiologie Vechta werden alle erstellten Bilder des Patienten digital gespeichert. Alle Ärzte arbeiten an einer Workstation, die extra für die digitale Befunderstellung ausgelegt sind. Die Bilder, die zuvor an einem medizinischen Gerät erstellt wurden, werden über das Netzwerk an das zentrale Archiv, auch PACS genannt, geschickt. Das PACS verteilt danach alle Bilder automatisiert an die Workstations der Ärzte.

Doch hierbei kommt es vermehrt vor, dass diverse Dienste des internen PACS Programms ausfallen oder die internen Festplatten der Workstations volllaufen und die Ärzte somit keine Röntgenbilder mehr erhalten. Dies fällt jedoch erst nach geraumer Zeit auf, die Ärzte können in dieser Zeit nicht arbeiten und es kommt zu unnötigen Wartezeiten für die Patienten.

Des Weiteren wird der Ausfall von Netzwerkkomponenten (z.B. Switch) erst spät bemerkt und die Behebung nimmt viel Zeit in Anspruch. Diesbezüglich wurde beauftragt, ein Monitoring-Tool für 50-100 Windows Clients, die auf drei Standorte aufgeteilt sind, zu planen und einzuführen, damit schon im Vorfeld erkannt wird, dass Dienste ausgefallen sind oder Festplatten volllaufen.

Um nicht nur bei einem Ausfall von Netzwerkkomponenten schneller reagieren zu können, sollen ebenfalls die Erreichbarkeit wichtiger Server, deren Dienste sowie die Erreichbarkeit wichtiger Clients überwacht werden.

1.3 Prozessschnittstellen

Meine Auftraggeber und Ansprechpartner in der Radiologie Vechta waren sowohl Herr Meyer in der Funktion als Administrator sowie Herr Müller in der Funktion als Geschäftsführer und Verwaltungsleiter.

2. Ausgangssituation und Projektbeschreibung

2.1 IST-Analyse

Da in der Radiologie Vechta noch kein Monitoring-Tool im Einsatz ist, ist die Fehlersuche bei Ausfällen sehr zeitintensiv. Die Administratoren werden oftmals erst durch die Benutzer auf die Probleme aufmerksam gemacht. Anschließend müssen diese unter starken Zeitdruck gelöst werden, da bei einem Netzerkausfall oder Ausfall einer Workstation keine Röntgenbilder mehr befundet werden können. Dies führt zu unnötigen Wartezeiten und Unzufriedenheit bei den Patienten.

Es befinden sich mehrere Switches in verschiedenen Gebäuden und Stationen des Krankenhauses. Wenn es zu einem Netzwerkproblem kommt, müssen unter Umständen alle EDV-Räume aufgesucht werden um einen fehlerhaften Switch zu finden.

Alle aktuellen Server sowie einige Clients der Radiologie Vechta werden auf einem redundanten ESXI 5.5 Server betrieben.

2.2 SOLL-Konzept

Das Monitoring-Tool soll alle wichtigen Dienste, Netzwerk- und Hardwarekomponenten sowie deren Erreichbarkeit an allen drei Standorten überwachen. Wenn ein Dienst an einer Workstation oder einem Server ausfällt, soll der Administrator umgehend eine Benachrichtigung darüber erhalten, damit das Problem schnellstmöglich gelöst werden kann. Ebenfalls soll der Administrator im Vorfeld benachrichtigt werden, wenn die Festplattenkapazität einen bestimmten Stand erreicht hat. Dadurch sollen Arbeitsausfälle vermieden werden.

Insgesamt sollen Zeit und Aufwand gespart werden um somit die Ausfallzeiten zu senken. Der Server für das Monitoring-Tool soll, wie alle anderen Server auch, virtualisiert werden, da hierfür bereits eine physikalische Redundanz gegeben ist.

Das Monitoring-Tool soll nur geringe Kosten aufweisen. Ebenfalls sollen neue IT Mitarbeiter in der Lage sein, das Tool nach kurzer Einarbeitung bedienen zu können. Zur Umsetzung und Einhaltung der Ressourcenanforderungen sollen verschiedene Softwarelösungen miteinander verglichen werden.

3. Evaluierung der Software

3.1 Informationsbeschaffung bezüglich Software

Die Informationen wurden eigenhändig durch die Recherche im Internet bezogen. Mehrere Lösungen kamen in die engere Auswahl.

3.2 Vorstellung der verschiedenen Softwareprodukte

3.2.1 Nagios

Nagios ist ein Host- und Servicemonitor auf OpenSource Basis. Es wurde zwar primär für den Einsatz unter Linux entwickelt, lässt sich aber auch unter anderen unix basierten Betriebssystemen betreiben.

Das Ziel von Nagios ist es, IT Mitarbeiter frühzeitig über auftretende Probleme zu informieren.

Je nach Konfiguration testet Nagios Server, Clients, Router, Switches etc. und Dienste wie beispielsweise SMTP, POP3, IMAP in definierten Abständen. Auch die Hardware sowie die Umgebungsparameter wie Temperatur und Luftfeuchte lassen sich mit geeigneten Sensoren überwachen.

Treten Probleme auf, werden Administratoren per E-Mail oder alternativ via SMS benachrichtigt, ebenso kann eine Benachrichtigung erfolgen, wenn das Problem gelöst ist. Diese Benachrichtigungen lassen sich über unterschiedliche Kontaktgruppen detailliert steuern. Im Webbrowser lässt sich der Status der überwachten Systeme überblicken, ebenso kann darüber auf Logfiles und Auswertungen zugegriffen werden.

Vorteile	Nachteile
Individuell konfigurierbar.	Kenntnisse in UNIX sind erforderlich.
Kompatibilität mit Windows durch ein Zusatzmodul (NSClient+) ist gegeben.	Offizielle Supportverträge sind sehr kostenaufwändig.
Geringe Kosten.	/
Durch hohe Verbreitung sind viele Dokumentationen und Softwarebeschreibungen zu finden.	/
Es können beliebig viele Clients hinzugefügt und überwacht werden.	

Tabelle 1: Produktvorstellung Nagios

3.2.2 Paessler

Paessler läuft auf einem Windows-System im Netzwerk und sammelt verschiedene Nutzungsdaten von den ausgewählten Rechnern, Anwendungen und Geräten. Die automatische Erkennung von Geräten ist mit dieser Software ebenfalls möglich. Die Daten werden in einer Datenbank abgelegt, so dass langfristige Statistiken zur Verfügung stehen. Über ein Webinterface kann das System administriert, Sensoren eingerichtet, Berichte konfiguriert und die Ergebnisse ausgewertet werden. Mitarbeitern kann der Zugang zu Grafiken und Tabellen mit Echtzeitdaten, sowie Berichte zur Auslastung gewährt werden. Paessler verfügt über mehr als 200 Sensortypen für alle üblichen Netzwerkdienste (z.B. Ping, HTTP, SMTP, POP3, FTP usw.). Ein Sensor wird als Messpunkt definiert, der einen bestimmten Aspekt auf einem Gerät überwachen kann. Benachrichtigungen über Ausfälle können per E-Mail, SMS oder Pager verschickt werden.

Vorteile	Nachteile
Leichte und schnelle Installation.	Sehr hohe Kosten.
Schnelle Einarbeitung.	Individuelle Gestaltung ist nicht geben.
Offizieller Support.	Nur eine begrenzte Anzahl an Sensoren, wenn diese überschritten werden, müssen nachträglich neue gekauft werden.
Kompatibel mit Windows.	/

Tabelle 2: Produktvorstellung Paessler

3.2.3 Big Brother BTF

Big Brother überwacht den Zustand der Server und der ausgewählten Hardware auf ihre Verfügbarkeit und zeigt auf einem Webinterface für jede einzelne Komponente deren Status an. Big Brother BTF benutzt kein SNMP um den Status abzufragen, sondern fungiert als Client-Server Model und benutzt sein eigenes Netzwerk Kommunikationsprotokoll und empfängt dabei TCP Pakete auf dem Port 1984. Es können ebenfalls Daten und Statistiken für andere Mitarbeiter zur Verfügung gestellt werden. Zusätzlich ermöglicht die integrierte Alarmierung den IT Mitarbeiter über Fehler, per E-Mail oder SMS zu kontaktieren.

Vorteile	Nachteile
Offizieller Support.	Hohe Wartungs- und Installationskosten.
Einfache Installation.	Individuelle Gestaltung ist nicht gegeben.
Kompatibel mit Windows.	Nur eine begrenzte Anzahl an Clients, wenn diese überschritten werden, müssen nachträglich neue Lizenzen gekauft werden.
	/

Tabelle 3: Produktvorstellung Big Brother BTF

3.3 Auswahl der Software

Kriterium	G	Nagios		Paessler		Big Brother BTF	
Punkte (max. 10) u. Gewichtung		P	P*G	P	P*G	P	P*G
Kosten	50	9	450	4	200	6	300
Integrierbarkeit	30	8	240	9	270	6	180
Aufwand der Implementierung	20	7	140	8	160	7	140
Erweiterbarkeit	40	9	360	9	360	8	320
Einarbeitung anderer Mitarbeiter	40	7	280	9	360	8	320
Summe			1470		1350		1260

Tabelle 4: Produktvergleich

Die Wahl der Software fällt auf Nagios, aufgrund mehrerer Aspekte. Es wurde von der Radiologie Vechta eine Softwarelösung verlangt, welche nur geringe Kosten aufweist. Somit ist dies der am höchsten gewichtete Punkt. Hierbei schneidet Nagios am besten ab.

Die anderen zur Auswahl stehenden Produkte waren für den Kunden zu teuer, da je nach Umfang der Clients eine Lizenzgebühr anfällt. Dies ist bei Nagios jedoch nicht der Fall. Ebenfalls ist durch den NSClient++ eine gute Integrierung in die bestehende Windowsumgebung gegeben. Hierzu muss lediglich die Software auf allen zu überwachenden Clients verteilt oder eigenhändig installiert werden.

Die Einarbeitung anderer Mitarbeiter erfolgt nach der fertigen Installation und ist je nach Kenntnisstand in Linux schnell zu erlernen. Ein weiterer Aspekt für Nagios ist, dass der derzeitige Administrator vor Ort bereits sehr gute Kenntnisse in Linux besitzt. Sollte jedoch zukünftig der Support von Nagios ausgelagert werden, könnte der derzeitige Dienstleister dies ebenfalls in den bestehenden Wartungsvertrag integrieren.

4 Ressourcen und Zeitplanung

4.1 Zeitplanung



Tabelle 5: Gantt diagramm zur Zeitplanung

* In voller Größe im Anhang unter Punkt 9.2 zu finden.

4.2 Personalplanung

Folgende Personalressourcen werden für das Projekt eingesetzt:

Mitarbeiter	Std. -Satz	Aufgaben	Zeitbedarf	Kosten
Herr Ahmetovic	6,35€	Hauptverantwortlicher	35 Std.	222,25€
Herr Meyer	23,50€	Mitwirkung an Konzeptionen	2 Std.	47,00€
Herr Müller	50,00€	Mitwirkung an Konzeptionen	1 Std.	50,00€
			38 Std.	319,25€

Tabelle 6: Personalressourcenplanung

4.3 Sachmittelplanung

4.3.1 Auswahl der Linux Distribution

Da der Fokus auf einer kostengünstigen Lösung liegt, stehen folgende OpenSource Linux-Distributionen zur Auswahl:

- CentOS
- OpenSUSE
- Debian
- Ubuntu

Alle oben genannten Distributionen sind mit der Softwarelösung Nagios im vollsten Umfang kompatibel.

In Absprache mit Herrn Meyer wurde sich für die Distribution OpenSUSE entschieden, da Herr Meyer sich mit dieser Distribution bereits auskennt und somit diese Distribution einsetzen möchte.

4.3.2 Hardwarespezifikationen

Es werden folgende Hardwarekomponenten für die Linux Distribution benötigt, damit das Monitoring-Tool Nagios einwandfrei betrieben werden kann.

- CPU Intel i3
- 4 GB RAM
- 150 GB HDD Kapazität

4.3.3 Hardwarekosten

Die Maschine soll auf der vorhandenen Virtuellen Umgebung bereitgestellt werden. Folgende Kosten fallen anteilig jährlich für die virtuelle Maschine an, die rund um die Uhr laufen soll:

Beschreibung	Kosten
Anteilige CPU Benutzung	48,33€
4 GB RAM (anteilig)	51,00€
150GB HDD Kapazität (anteilig)	169,00€
Stromkosten (bei 8760 Std. Jährlich)	69,92€

Tabelle 7: Hardwarekostenplanung

4.4 Kostenplanung

Für das Projekt ist geplant, dass möglichst geringe Kosten entstehen, deshalb werden auf preisintensive Anschaffungen verzichtet. Für das Projekt sind folgende Kosten geplant:

Beschreibung	Kosten	Jährlich
Personalkosten für die Einrichtung und Durchführung	222,25€	/
Mitwirkende Personalkosten	97,00€	/
Anteilige Hardwarekosten der Virtuellen Maschine	268,33€	/
Jährliche Stromkosten (bei 8760 Std.)	/	69,92€
Gesamt:	587,58€	69,92€

Tabelle 8: Gesamtkostenplanung

5. Durchführung

5.1 Zusammenfassung der Clients und Dienste

Vor der eigentlichen Installation wurden in Zusammenarbeit mit Herrn Meyer alle zu überwachenden Clients dokumentiert. Nach der Dokumentation aller Clients wurde geplant, welche Dienste überwacht und in welchem Zeitraum Benachrichtigungen versendet werden sollen.

Es wurde sich für die E-Mail Benachrichtigung entschieden.

Alle normalen Arbeitsplatzrechner sollen lediglich während der Arbeitszeit (08:00 Uhr – 17:00Uhr) Benachrichtigungen an den Administrator senden. Alle Netzwerkkomponenten und Server sollen permanent Benachrichtigungen über Fehler und Ausfälle per E-Mail an den Administrator senden.

5.2 Erstellung der Virtuellen Maschine

Es wird die Linux OpenSUSE Distribution 13.2 installiert. Die Distribution ist virtuell und liegt auf einer Hypervisor-Struktur. Als Software wird ein ESXi 5.5 Server im Zusammenhang mit VMware genutzt. Über die Software kann auf dem ESX-Server eine neue virtuelle Maschine installiert werden. Es wird eine Virtuelle Maschine mit den benötigten Hardwarespezifikationen erstellt und mit dem Namen „VM-Nagios“ versehen, anschließend wird die Linux Distribution OpenSUSE 13.2 über ein kostenloses Image, was im Vorfeld heruntergeladen wurde, installiert. Die Daten werden redundant auf einem SAN, mit SAS-

Festplatten in einem RAID5-Verbund gesichert. Die Installation lief ohne Fehler standardmäßig durch und war somit erfolgreich.

5.3 Vorbereitende Installationen und Konfigurationen

Der OpenSUSE Client bekam die IP Adresse 192.168.10.104/24 und wurde somit in das bestehende Netzwerk 192.168.10.0/24 eingebunden. Um die Softwarepakete aktuell zu halten, wird eine Internetverbindung empfohlen.

Des Weiteren müssen im Vorfeld einige Installationen und Konfigurationen auf dem OpenSUSE Client durchgeführt werden, damit Nagios installiert und betrieben werden kann. Hierzu gehören folgende Produkte:

- Apache2
- GD Graphic Library
- PHP5 Libraries
- C/C++ Development Libraries

Außerdem muss sichergestellt werden, dass Nagios eine korrekte Namensauflösung innerhalb der Windowsdomäne aufweisen kann. Hierzu wird der Nagios-Client in den DNS auf dem Domänencontroller eingetragen. Anschließend wird die Namensauflösung mittels eines PING Befehls auf einen Namen getestet. Der Test wurde erfolgreich abgeschlossen.

5.3.1 Hinzufügen der benötigten Libraries

Es werden mehrere Libraries für Nagios benötigt um sicherzustellen, dass alle Checks die via Nagios ausgeführt werden, einwandfrei funktionieren und die Installation korrekt übersetzt wird.

Die in Punkt 5.3 genannten Libraries wurden alle mittels YaST installiert. Da YaST alle bereits installierten und fehlenden Softwarepakete anzeigt und die abhängigen Pakete automatisch mit installiert, ist YaST in diesem Fall die beste Lösung.

5.3.2 Anbindung in die bestehende Windows Domäne

Der OpenSUSE Client wurde mit folgenden Netzwerkeinstellungen versehen, damit eine allgemeine Verbindung in das Netzwerk 192.168.10.0/24 gegeben ist.

Beschreibung	Wert
IP Adresse	192.168.10.104
Subnetzmaske	255.255.255.0
Gateway	192.168.10.11
DNS 1	192.168.10.217
DNS 2	192.168.10.218

Tabelle 9: Netzwerkkonfiguration des Servers

Nachdem alle erforderlichen IP Adressen erreichbar waren, musste der OpenSUSE Client noch in die Windowsdomäne eingebunden werden. Hierfür wurde Samba verwendet. Samba dient zur Einbindung eines UNIX Clients in eine Windowsdomäne.

Sobald auf der Linux Maschine DNS-Abfragen oder Benutzerkonto-Abfragen benötigt werden, leitet Samba diese an den zuständigen Windows Domänencontroller weiter.

Die Samba Konfiguration wird ebenfalls mittels YaST durchgeführt. Nach der

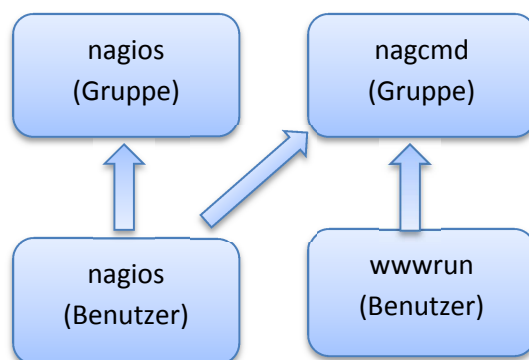
abgeschlossenen Konfiguration wird mit dem Befehl „*Host -f*“ geprüft, ob die lokale Maschine ein Mitglied in der Windowsdomäne ist und ein Fully qualified domain name (FQDN) ausgegeben wird. Dies lief ohne Probleme ab und in anschließenden Tests konnte sichergestellt werden, dass alle Namen und relevanten Clients in der Windowsdomäne erreichbar sind.

5.4 Installation und Konfiguration von Nagios

5.4.1 Benutzer und Gruppen

Für Nagios wird auf dem Server eine eigene Gruppe sowie der Benutzer „nagios“ erstellt. Mitglieder der erstellten Gruppe werden der Webserver sowie der neu erstellte Benutzer „nagios“ sein.

Anhand folgender Grafik werden die neu erstellten Benutzer und Gruppen sowie deren Zusammenhang in den Gruppen veranschaulicht:



5.4.2 Installation von Nagios 4.0.7

Um Nagios installieren zu können, werden die aktuelle Version sowie die aktuellen Plugins von www.sourceforge.net heruntergeladen. Anschließend werden die Nagios Dateien mit folgendem Befehl entpackt:

```
tar xzf nagios-3.0.tar.gz tar xzf nagios-plugins-1.4.11.tag.gz
```

Im Verzeichnis „nagios-3.0“ werden die Nagios Dateien kompiliert, die Nagios-Plugin Dateien im Verzeichnis „nagios-plugins-1.4.11“.

```
./configure --with-command-group=nagiosgroup make all  
make install  
make install-init  
make install-config  
make install-commandmode
```

```
./configure --with-nagios-user=nagios --with-nagios-group=nagiosgroup make  
make install
```

Nachdem die Installation abgeschlossen wurde, kann mit der Konfiguration begonnen werden.

5.4.3 Installation des Apache Webservers

Damit das Nagios Webinterface später aufgerufen werden kann, muss der Apache Webserver installiert werden. Es wurde die Version Apache2 genommen, da dies die aktuellste Version und diese bereits im Paketmanager von OpenSUSE enthalten ist. Nach der Installation müssen noch Verweise auf die HTML und CGI Dateien von Nagios eingerichtet werden. Dies geschieht in den Konfigurationsfiles des Apache Servers. Nach der Konfiguration müssen noch diverse Sicherheitsaspekte durchgeführt werden, damit ausschließlich der Nagios Benutzer Zugriff auf das Webinterface besitzt.

5.4.4 Einrichtung der Sicherheitsaspekte

Nun wird das Webinterface von Nagios mit einer Benutzer- und Passwortauthentifizierung gesichert. Mit dem Programm „htpasswd“ wird eine verschlüsselte Datei erzeugt, die den Benutzernamen und das Passwort beinhaltet. Das Passwort wird nicht im Klartext, sondern verschlüsselt in der Datei hinterlegt. Jeglicher Zugriff auf das Webinterface soll über den Benutzer „nagios“ erfolgen. Dies wird mit folgendem Befehl gewährleistet.

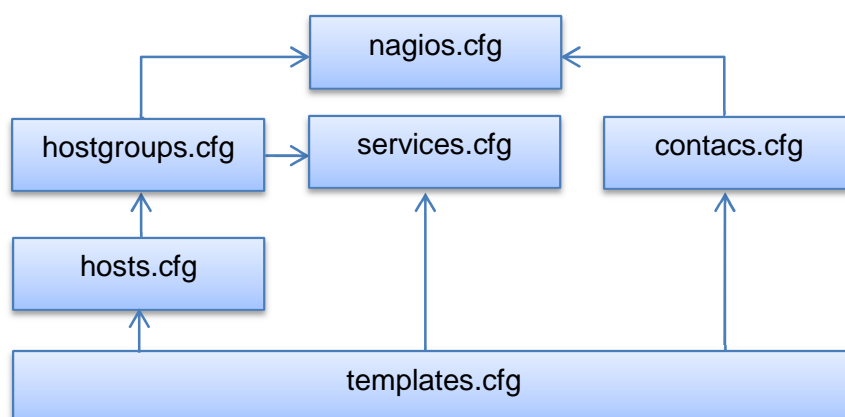
```
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagios
```

Nach dem Bestätigen des Befehls erscheint eine Abfrage, in der nach dem Passwort des Benutzers gefragt wird. Wenn zukünftig ein Zugriff auf das Nagios Webinterface stattfindet, müssen der Benutzername sowie das zugehörige Passwort eingegeben werden. Diese werden nach der Eingabe in der Datei „htpasswd.users“ überprüft. Bei einer Übereinstimmung wird der Zugriff erlaubt.

5.4.5 Konfiguration von Nagios

Nagios wird ausschließlich in Konfigurationsdateien konfiguriert. Die Hauptkonfiguration erfolgt in der Datei „nagios.cfg“. Um nicht alle Konfigurationsoptionen unterzubringen, werden zur Übersichtlichkeit diverse Konfigurationsdateien erstellt, die miteinander verbunden werden.

Anhand der folgenden Grafik werden die Beziehung und Dateinamen der Konfigurationsdateien veranschaulicht:



5.4.6 Erklärung der Template Datei

Mit Hilfe der Template Datei werden alle Konfigurationen zur Erstellung von Services oder Hosts um einiges vereinfacht. In der Template Datei werden Vorlagen erstellt die bereits einige Standardparameter enthalten. Somit müssen nicht bei jedem Host oder Service alle Parameter angegeben werden. Die Vorlagen können mittels des Befehls „use“ benutzt werden. Genauere Beschreibungen werden in den jeweiligen Konfigurationsbeispielen in Punkt 5.4.8 und 5.4.10 gezeigt.

Alle Vorlagen, die sich in der Datei „templates.cfg“ befinden, sowie die Werte der Parameter, werden im Anhang unter Punkt 9.1 eingefügt.

5.4.7 Erstellung von Hostgruppen

Die Hostgruppen werden benutzt, um mehrere Hosts in einer Gruppe zusammenzufassen. Mit Hilfe von Hostgruppen lassen sich somit diverse Services leichter auf mehreren Clients im Netzwerk anwenden. Dies sorgt nicht nur für eine erleichterte Konfiguration und Fehlersuche, sondern beschleunigt die Implementierung der Dienste im späteren Verlauf. Alle Hostgruppen werden in der Datei „hostgroups.cfg“ erstellt und bearbeitet.

Die Definition einer Hostgruppe sieht wie folgt aus:

```
define hostgroup {  
    hostgroup_name    Radiologie  
    alias             Alle Clients des Radiologie Netzes.  
    member            Client_1, Client_2, Client_3  
}
```

Option	Erklärung
hostgroup_name	Name der Hostgruppe.
alias	Beschreibung der Hostgruppe.
member	Mitglieder der Hostgruppe.

Tabelle 10: Begriffserläuterung der Hostgruppenparameter

5.4.8 Erstellung von Hosts

In der Datei „hosts.cfg“ werden Nagios alle Hosts bekannt gemacht. Das Hostobjekt ist der zentrale Punkt, auf dem alle Host- und Service – Checks basieren. Das Hostobjekt beinhaltet den Namen, der im Webinterface und den Benachrichtigungen angezeigt wird, sowie die IP-Adresse des Hosts.

Alle weiteren Parameter, die ein Hostobjekt definieren, werden in der Datei „templates.cfg“ in Vorlagen zusammengefasst und vordefiniert.

Ein Host wird wie folgt erstellt:

```
define host {  
    use                HOST  
    host_name          Client_1  
    alias              Beispiel Client.  
    address            192.168.10.110  
}
```

Option	Erklärung
use	Benutzung des erstellten Templates.
host_name	Name des Hosts.
alias	Beschreibung des Hosts.
address	IP-Adresse des Clients.

Tabelle 11: Begriffserläuterungen der Hostparameter

Folgende Parameter befinden sich in dem HOST Template, welches vorher erstellt wurde. Diese Parameter können zusätzlich noch in der Datei „hosts.cfg“ unter einer Hostdefinition angegeben werden, falls der erstellte Host abweichende Eigenschaften aufweisen soll. Somit wird der Wert in der Vorlage außer Kraft gesetzt.

Option	Erklärung
check_period	
Check_command	Name des Prüfverfahrens des Hosts.
Max_check_attempts	Anzahl der maximalen Service Überprüfungen.
Check_interval	Interval für die Hostprüfung (Minuten)
Notification_period	Benachrichtigungszeitraum für Fehler.
Notification_interval	Wiederholtes Senden der Nachrichten. (Minuten)
Notification_options	Wann Benachrichtigungen geschickt werden.
contact_groups	An wen Benachrichtigungen geschickt werden.

Tabelle 12: Begriffserläuterungen der Hostparameter in der Templatedatei

5.4.9 Erstellung von Kontaktgruppen

In der Datei „contacs.cfg“ lässt sich festlegen, wer im Falle eines Fehlers benachrichtigt werden soll. Die einzelnen Kontakte werden wie folgt in der Datei „contacs.cfg“ angelegt:

```
define contact {  
  
    contact_name      Herr Meyer  
    alias             EDV-Leiter der Radiologie Vechta.  
    email             kontakt@radiologie-vechta.de  
    contactgroups     Administratoren  
  
    host_notifications_enabled 1  
    host_notification_options  d,u,r,f  
    host_notification_commands  notify-host-by-email  
    host_notification_period   24x7
```

```

        service_notifications_enabled      1
        service_notification_options      w,u,r,f,s
        service_notification_commands      notify-service-by-email
        service_notification_period        24x7
    }

```

Diese Kontakte können nach der Erstellung in einer Kontaktgruppe zusammengefasst werden, falls mehrere Personen Benachrichtigungen über einen Fehlerzustand bekommen sollen.

```

define contactgroup {
    contactgroup_name      Administratoren
    alias                  Gruppe der Administratoren die benachrichtigt werden.
}

```

Nachfolgend sind alle Parameter und Eigenschaften genauer erläutert.

Option	Erklärung
contact_name	Name des Kontakts.
alias	Beschreibung des Kontakts.
email	E-Mailadresse des Kontakts.
contactgroup	Zugehörige Kontaktgruppe.
host_notifications_enabled	Entscheidung ob bei Hostfehlern benachrichtigt wird.
host_notification_options	Benachrichtigungsoptionen.
host_notification_commands	Art der Benachrichtigung.
host_notification_period	Zeitraum in dem benachrichtigt werden soll.
service_notifications_enabled	Entscheidung ob bei Servicefehlern benachrichtigt wird.
service_notification_options	Benachrichtigungsoptionen.
service_notification_commands	Art der Benachrichtigung.
service_notification_period	Zeitraum in dem benachrichtigt werden soll.

Tabelle 13: Begriffserläuterungen der Kontaktparameter

Service Benachrichtigungsoptionen	Erklärung
w	Benachrichtigung bei WARNING Zustand.
u	Benachrichtigung bei UNKNOWN Zustand.
f	Benachrichtigung beim Flattern des Services.
s	Benachrichtigung bei geplantem Serviceausfall.
n	Keine Benachrichtigung versenden.

Tabelle 14: Begriffserläuterungen der Service Benachrichtigungsoptionen

Host Benachrichtigungsoptionen	Erklärung
d	Benachrichtigung bei DOWN Zustand.
u	Benachrichtigung bei UNREACHABLE Zustand.
r	Benachrichtigung bei UP Zustand.
f	Benachrichtigung beim Flattern des Hosts.
s	Benachrichtigung bei geplanten Hostausfall.
n	Keine Benachrichtigung versenden.

Tabelle 15: Begriffserläuterungen der Host Benachrichtigungsoptionen

5.4.10 Erstellung von Services

Alle Checks, die Nagios ausführen soll, werden in der Datei „services.cfg“ definiert und konfiguriert. Diese werden im Vorfeld in der Datei „templates.cfg“ definiert und können dann mittels des Befehls „use“ verwendet werden. Somit ist die Erstellung von weiteren Checks einfach gehalten. Es erfolgt lediglich eine Zuweisung der Hosts in der Service Definition. Wenn beispielsweise eine Überwachung der Antwortzeiten eines Pings überprüft werden sollen, sieht dies wie folgt aus:

```
define service{
    use                Ping
    hostgroup_name     Clients_Vechta
    host_name          Test_Server
}
```

Alle Parameter die in dem Template für den Check „PING“ definiert wurden, sind in der Datei „templates.cfg“ zu finden. Gleich wie bei den Hosts, können diese Parameter zusätzlich eingefügt werden, um die Werte in dem Template zu umgehen. Alle weiteren Parameter werden hier veranschaulicht:

Befehl	Erklärung
Hostgroup_name	Name der Hostgruppe die überwacht werden soll.
Host_name	Name des Hosts der überwacht werden soll.
Service_description	Beschreibung des Services.
Check_command	Überprüfungsbefehl.
Max_check_attempts	Anzahl der Service Überprüfungen.
Check_interval	Überprüfungsintervall.
Retry_interval	Überprüfungsintervall bei Fehlern.
Check_period	Zeitraum in dem überprüft werden soll.
Notification_period	Zeitraum in dem Benachrichtigungen verschickt werden.

Tabelle 16: Begriffserläuterungen der Serviceparameter

5.4.11 Konfiguration des NSClient++

Nach der erfolgreichen Installation und Konfiguration von Nagios, wurden alle Hosts und Services erstellt. Damit alle erstellten Checks ausgeführt werden können, muss der NSClient++ an allen Clients installiert sein. Der NSClient++ muss ausschließlich auf

Windowssystemen installiert werden um eine Kommunikation zwischen dem UNIX-System zu gewährleisten.

Die Installation erfolgt über eine grafische Oberfläche, die sich nach dem Starten der Datei NSClient.exe öffnet. Hier muss lediglich die IP-Adresse des Nagios Servers eingetragen werden. Weitere Einstellungen sind nicht vorzunehmen.

Nach der Installation können allen Clients überwacht werden.

6 Projektabschluss

6.1 Abschlusstests

Um Nagios zu testen, wurde der Server einem Neustart unterzogen. Hiernach wurden zu überwachende Dienste beendet sowie die Kapazitäten diverser Hardwarekomponenten ausgelastet um zu testen, ob Nagios diese als „Critical“ oder „Warning“ im Webinterface anzeigt. Ebenfalls wurde somit die E-Mail Benachrichtigung getestet.

Folgende Situationen wurden Simuliert:

Test	Anzeige im Webinterface	E-Mail Benachrichtigung
Host heruntergefahren	Bestanden	Bestanden
CPU Auslastung >80%	Bestanden	Bestanden
RAM Auslastung >80%	Bestanden	Bestanden
Festplattenspeicher <20%	Bestanden	Bestanden
Dienst beendet	Bestanden	Bestanden
Ping Latenzzeit > 70ms	Bestanden	Bestanden

Tabelle 17: Abschlusstests

Im Rahmen der Testphase gab es Probleme mit dem Starten des Nagios Dienstes. Sobald Nagios eine fehlerhafte Konfiguration in den „.cfg“ Dateien findet, startet dieser Dienst nicht. Um den Fehler zu lokalisieren, wurde der Befehl „`/usr/local/nagios/bin/nagios -v /etc/nagios/nagios.cfg`“ ausgeführt.

Dieser Befehl prüft alle bekannten Konfigurationsdateien von Nagios. Wenn hierbei ein Fehler gefunden wird, wird dieser unter Angabe des Dateinamens sowie der dazugehörigen Zeile angezeigt.

In diesem Fall wurde eine falsche Hostgruppenbezeichnung angegeben.

Nachdem der Fehler korrigiert wurde, konnten alle anderen Tests erfolgreich ausgeführt werden.

6.2 Übergabe

Nach der Fertigstellung des Projektes wurde der Nagios-Server durch Herrn Meyer abgenommen und für die Inbetriebnahme freigegeben. Auf eine Kundendokumentation wird verzichtet und stattdessen die Projektdokumentation verwendet.

7 Fazit

Das Projekt wurde erfolgreich in der geplanten Zeit durchgeführt. Schwerwiegende Fehler oder Abweichungen zu der Planung gab es nicht.

Alle vom Kunden geforderten Aspekte wurden in vollem Umfang erfüllt und die SOLL-Analyse wurde zufriedenstellend erfüllt. Nach der erfolgreichen Implementierung in den Betrieb waren alle Beteiligten über die Funktionalität und Erleichterung von Nagios erfreut.

8. Literaturverzeichnis

8.1 Allgemeine Informationen

- Enzyklopädie Wikipedia (<http://de.wikipedia.org/wiki/>)

8.2 Produktinformationen

- <http://www.nagios.org>
- <http://www.monitoring-portal.org/>

9 Anhang

9.1 Vorlagen der Templatedatei

```
define service{
    name                PING
    use                  generic-service
    service_description  PING
    check_command        check_ping!100.0,20%!500.0,60%
    max_check_attempts   3
    normal_check_interval 20
    retry_check_interval  1
    check_period         24x7
    notification_interval 120
    notification_period   workhours
    notification_options  n
    contact_groups        admins
    register              0
}

define service{
    name                DRIVE_SPACE_C
    use                  generic-service
    service_description  C:\ Drive Space
    check_command        check_nt!USEDISKSPACE!-l c -w 80 -c 90
    max_check_attempts   3
    normal_check_interval 2
    retry_check_interval  3
    check_period         24x7
    notification_interval 0
}
```

```

        notification_period      24x7
        notification_options     w,u,c,r,f,s
        contact_groups           admins
        register                 0
    }
    define service{
        name                     CPU_LOAD
        use                      generic-service
        service_description      CPU load
        check_command            check_nt!CPULOAD!-l 5,80,90
        max_check_attempts      3
        normal_check_interval    2
        retry_check_interval     3
        check_period            24x7
        notification_interval    120
        notification_period      24x7
        notification_options     w,u,c,r,f,s
        contact_groups           admins
        register                 0
    }
    define service{
        name                     MEMORY_USAGE
        use                      generic-service
        service_description      Memory Usage
        check_command            check_nt!MEMUSE!-w 80 -c 90
        max_check_attempts      3
        normal_check_interval    20
        retry_check_interval     3
        check_period            24x7
        notification_interval    120
        notification_period      24x7
        notification_options     w,u,c,r,f,s
        contact_groups           admins
        register                 0
    }
    define service{
        name                     AMI_CONNECT
        use                      generic-service
        service_description      AMI connect
        check_command            check_nt!PROCSTATE!-l connect.exe -d SHOWALL
        max_check_attempts      3
        normal_check_interval    1
    }

```

```

        retry_check_interval      3
        check_period              24x7
        notification_interval     120
        notification_period       24x7
        notification_options      w,u,c,r,f,s
        contact_groups            admins
        register                  0
    }
    define service{
        name                      AMI_DATABASE
        use                       generic-service
        service_description       AMI Database
        check_command              check_nt!PROCSTATE!-I database.exe -d SHOWALL
        max_check_attempts        3
        normal_check_interval     1
        retry_check_interval      3
        check_period              24x7
        notification_interval     120
        notification_period       24x7
        notification_options      w,u,c,r,f,s
        contact_groups            admins
        register                  0
    }
    define service{
        name                      AMI_PRINT
        use                       generic-service
        service_description       AMI Print
        check_command              check_nt!PROCSTATE!-I print.exe -d SHOWALL
        max_check_attempts        3
        normal_check_interval     1
        retry_check_interval      3
        check_period              24x7
        notification_interval     120
        notification_period       24x7
        notification_options      w,u,c,r,f,s
        contact_groups            admins
        register                  0
    }

    define service{
        name                      ID_DICOM_RECEIVE
        use                       generic-service

```

```

service_description      ID DICOM RECEIVE
check_command            check_nt!PROCSTATE!-l receivescp.exe -d SHOWALL
max_check_attempts       3
normal_check_interval    1
retry_check_interval     3
check_period             24x7
notification_interval    120
notification_period      24x7
notification_options      w,u,c,r,f,s
contact_groups           admins
register                 0
}

```

9.2 Gantt - Diagramm

