Softwareprojekt 2005-2007

SW-Architektur

Version: 1.3
Letzte Änderung: 03.02.2006 11:16

Projektleiter	Peter Hofer	
Projektleitung 4DHD	Reinhard Pointner	
Verantwortlich	Peter Hofer	
Verantwortlich 4DHD	Reinhard Pointner	
Erstellt	05.10.2005 13:34	
Bearbeitungszustand		In Bearbeitung
		Vorgelegt
	*	Fertiggestellt
V-Modell-Version	XT 1.1	

Änderungsverzeichnis

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor	Zustand
	Datum	Version				
1	30.1.2006	1.1	Alle	Initiale Produkterstellung	jp	
2	1.2.2006	1.2	Alle	Verzeichnis, Beschreibungen	ph	
3	3.2.2006	1.3	Dekomposition	Erläuterung, Klassendiagramme, GUI-Prototypen	ph, ce	

Prüfverzeichnis

Datum	Geprüfte Version	Anmerkungen	Prüfer	Neuer Produktzustand
3.2.06	1.3	-	ce	Fertiggestellt

Inhaltsverzeichnis

1. Einleitung

Für jede in der Systemarchitektur identifizierte SW-Einheit wird eine SW-Architektur erstellt. Ausgehend von den funktionalen und nicht-funktionalen Anforderungen an die SW-Einheit ist es Aufgabe des SW-Architekten, eine geeignete SW-Architektur zu entwerfen. Das Produkt SW-Architektur dient dabei sowohl als Leitfaden zum Entwurf als auch zur Dokumentation der Entwurfsentscheidungen.

Wie in der Systemarchitektur werden richtungweisende Architekturprinzipien festgelegt und mögliche Entwurfsalternativen untersucht. Entsprechend der gewählten Entwurfsalternative wird die Zerlegung (Dekomposition) der SW-Einheit in SW-Komponenten und SW-Module beschrieben. Beziehungen und Schnittstellen zwischen den Elementen und zur Umgebung werden identifiziert und im Überblick dargestellt. Ein Datenkatalog der an den Schnittstellen ausgetauschten Datenstrukturen wird erstellt.

Die gewählte Architektur wird hinsichtlich ihrer Eignung für das geforderte System bewertet. Offene Fragen können beispielsweise im Rahmen einer prototypischen Entwicklung geklärt werden.

Der Entwurf der SW-Architektur kann Änderungen der Systemarchitektur nach sich ziehen. Abhängig von den Vorgaben im Projekthandbuch wird die Änderung vom Systemarchitekten geprüft und gegebenenfalls direkt eingearbeitet. Im Einzelfall kann ein expliziter Änderungsantrag notwendig sein.

Hauptverantwortlicher für den Entwurf der SW-Architektur ist der SW-Architekt. Unterstützt wird er dabei vom SW-Entwickler sowie von verschiedenen Experten zu Einzelthemen wie Logistik, Systemsicherheit oder Ergonomie.

Die SW-Architektur stellt das zentrale Dokument für die Erstellung weiterer Produkte dar. Sie legt alle SW-Komponenten und SW-Module der SW-Einheit fest. Entsprechend ihren Vorgaben werden die jeweiligen Elemente mit ihren Spezifikationen erstellt.

2. Architekturprinzipien und Entwurfsalternativen

2.1 Architekturprinzipien

- Bibliotheken und ausführbare Programme werden in Komponenten getrennt. Bei Bibliotheken wird weiterhin zwischen Backend und Frontend unterschieden („Kernklassen“, GUI-Klassen).
- Zur Interprozesskommunikation wird das DCOP-Protokoll verwendet, da es im KDE, auf den dieses Projekt aufbaut, schon integriert ist und im großen Maße genutzt wird.

2.2 Entwurfsalternativen

2.2.1 Unterteilung nach Komponenten

Die oben beschriebene Trennung in *ausführbare Programme, Frontend- und Backend-Library* bietet folgende Vorteile:

- ✔ Kernklassen sind von Benutzerschnittstelle getrennt
- ✔ Programme, die nicht auf GUI-Komponenten angewiesen sind, brauchen diese nicht zu laden
- ✔ Erhöhte Wiederverwendbarkeit
- ✔ Leichte Wartung

2.2.2 Unterteilung nach Funktionalität

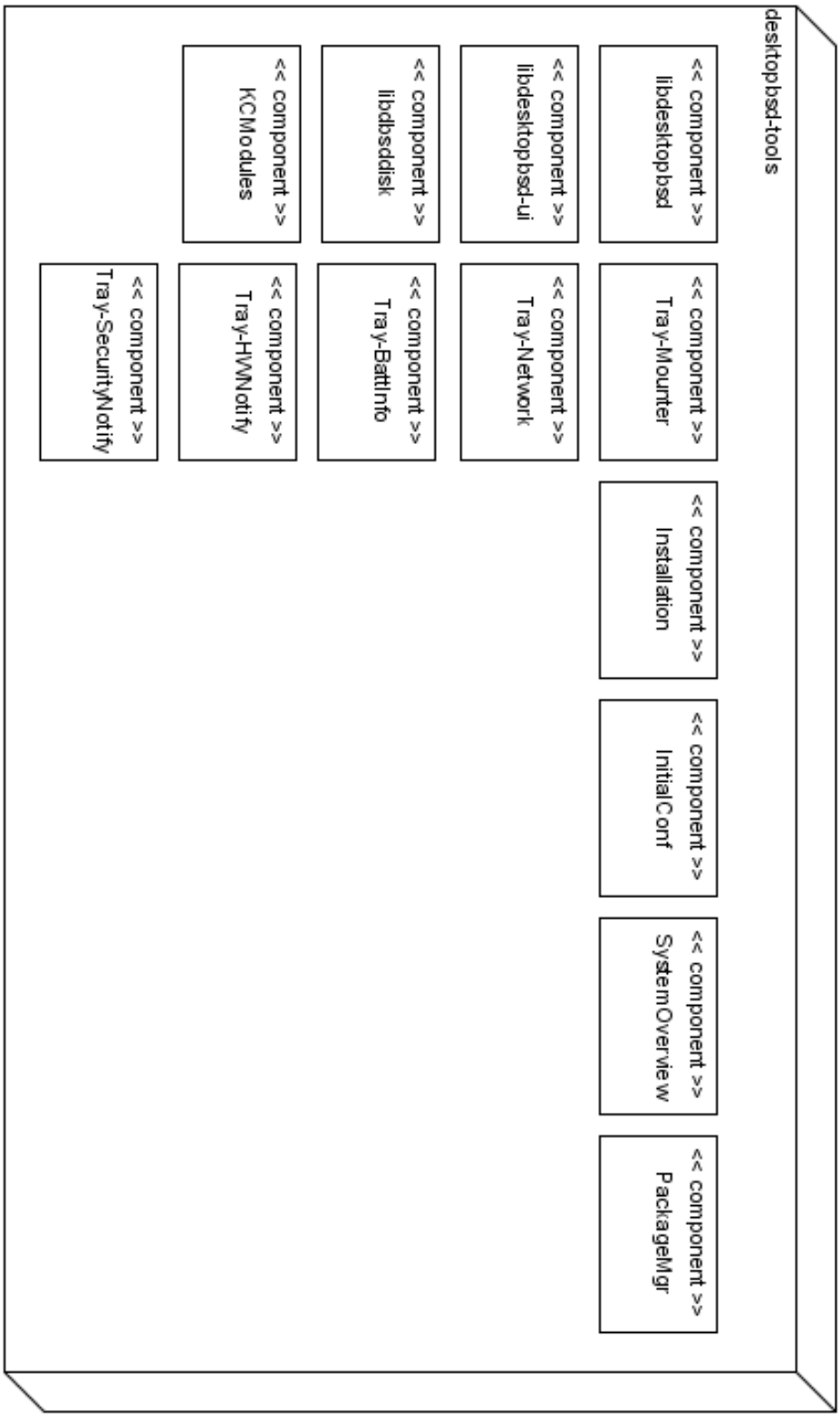
Eine Komponente umfasst alle Kernklassen und Benutzeroberfläche, die für dessen Funktion nötig sein (beispielsweise Netzwerkkonfiguration).

- ✔ Bei Änderungen an einzelnen Funktionen muss nur eine Komponente/ein Modul neu übersetzt werden
- ✔ Leichte Wartung

- Auch, wenn ausschließlich Kernklassen benötigt werden, müssen die entsprechenden GUI-Teile inklusive abhängiger Programmbibliotheken geladen werden
  - ✗ Die Trennung von GUI und Kernklassen wird im Projektverlauf schwieriger
  - ✗ Schlechtere Wiederverwendbarkeit

3. Dekomposition der SW-Einheit

3.1 Übersicht über die SW-Einheit



3.2 Dekomposition

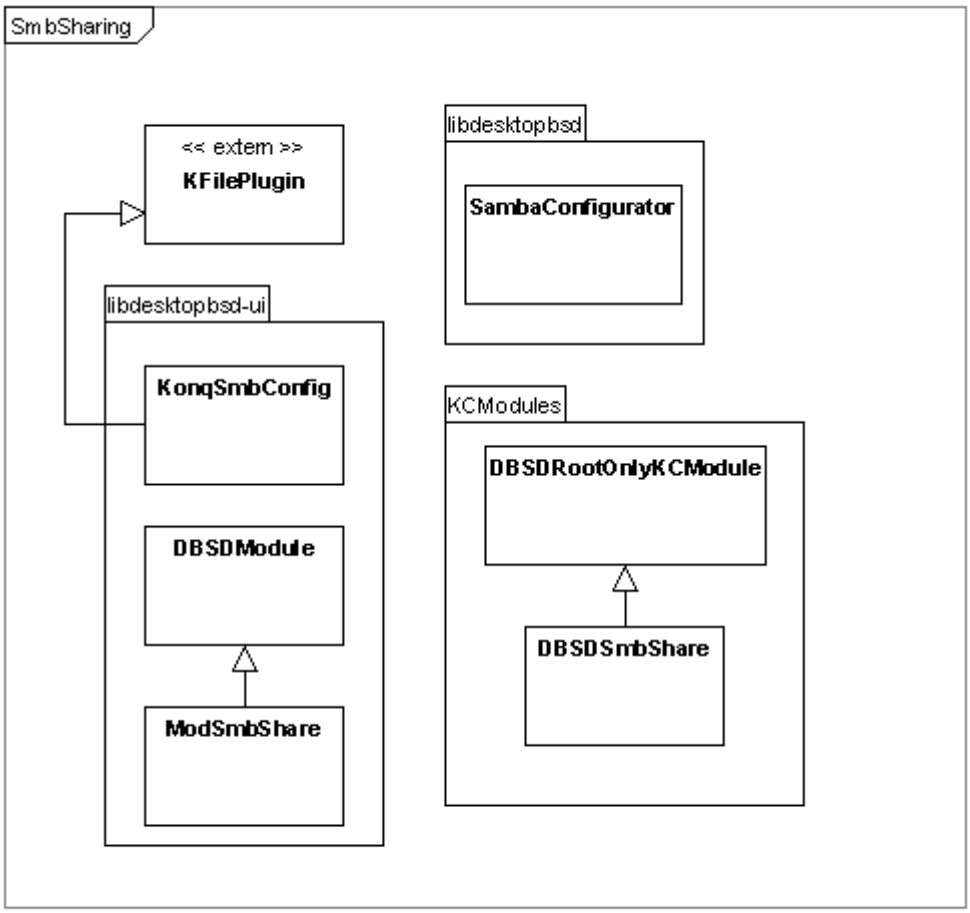
Wegen den oben beschriebenen Architekturprinzipien, die nicht nach Funktionalität trennen, sondern nach Frontend und Backend sowie Ausführbarkeit, ist es hier zielführender, für jede zu implementierende Funktionalität die SW-Module in den SW-Komponenten aufzuführen.

3.2.1 Allgemeines Layout

Wie an den nachstehenden Klassendiagrammen ersichtlich, gibt es:

- Mindestens eine Klasse in *libdesktopbsd*, die die eigentliche Funktionalität implementiert (Back-End)
- Mindestens eine Klasse in *libdesktopbsd-ui*, die von *DBSDModule* abgeleitet ist und die grafische Oberfläche implementiert (Front-End)
- Für jede von *DBSDModule* abgeleitete Klasse in *KCModules* eine Wrapper-Klasse, die die grafische Oberfläche in das KDE-Kontrollzentrum einbindet

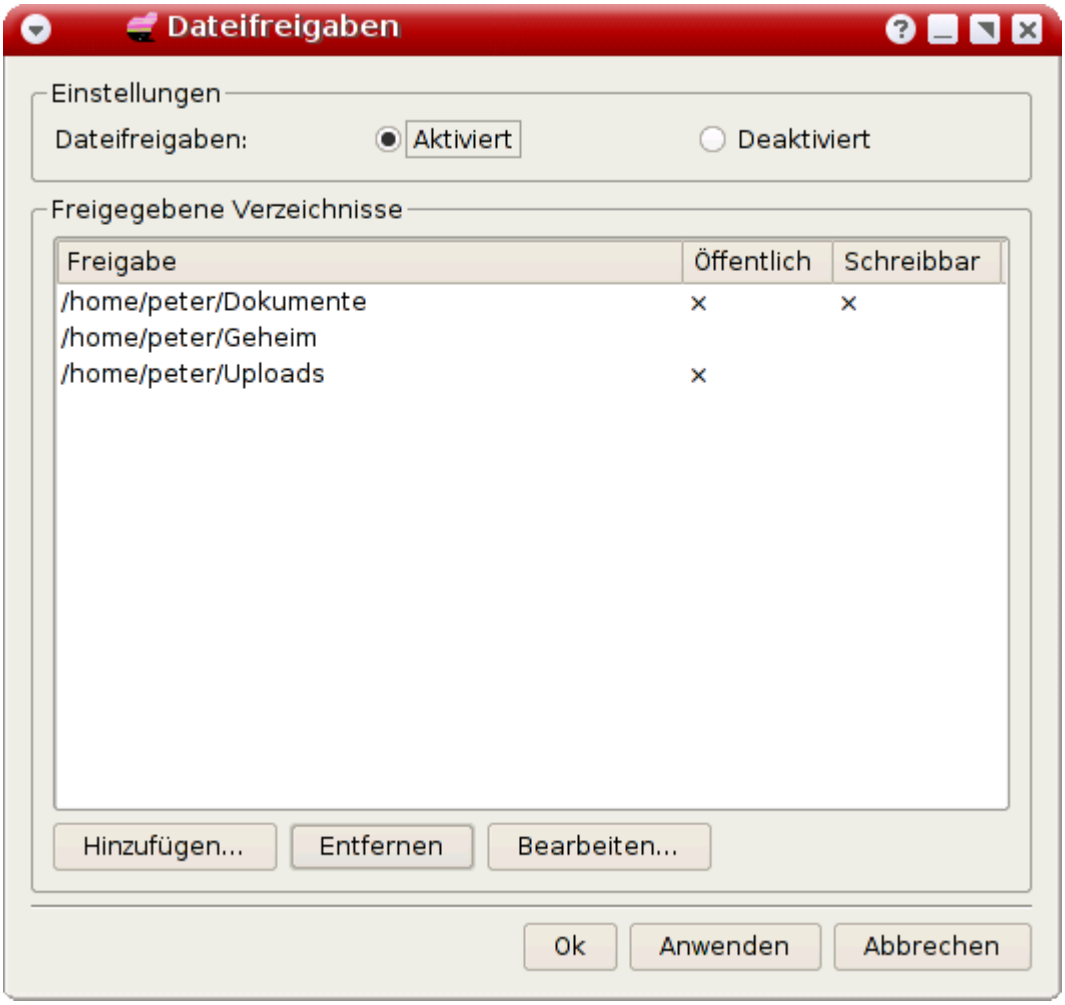
3.2.1 Grafische Konfiguration von Dateifreigaben

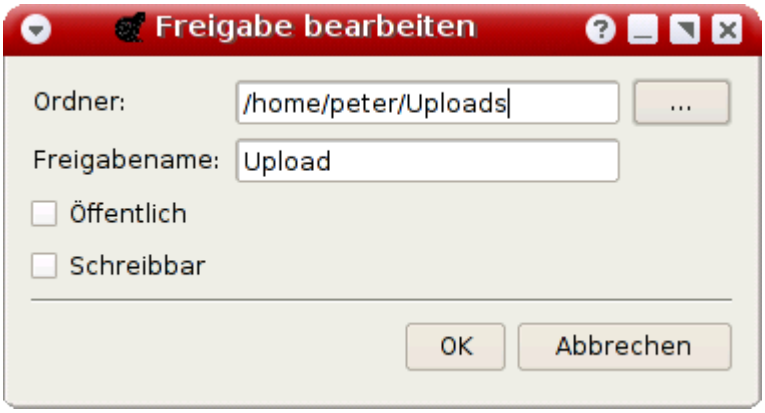


SambaConfigurator enthält Methoden zum Hinzufügen, Entfernen und Bearbeiten von Dateifreigaben. KonqSmbConfig ist eine Erweiterung für den

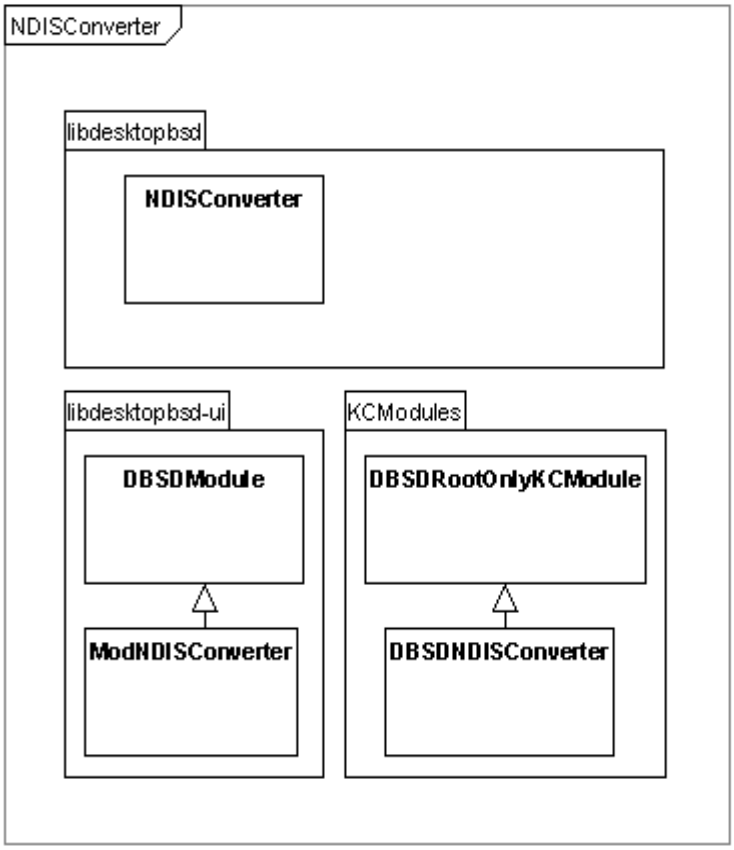
KDE-Dateimanager *Konqueror.* Öffnet der Benutzer in Konqueror den Eigenschaftendialog eines Ordners, werden ihm zusätzliche Optionen zur Freigabe des Ordners angeboten.

3.2.1.1 GUI-Prototypen



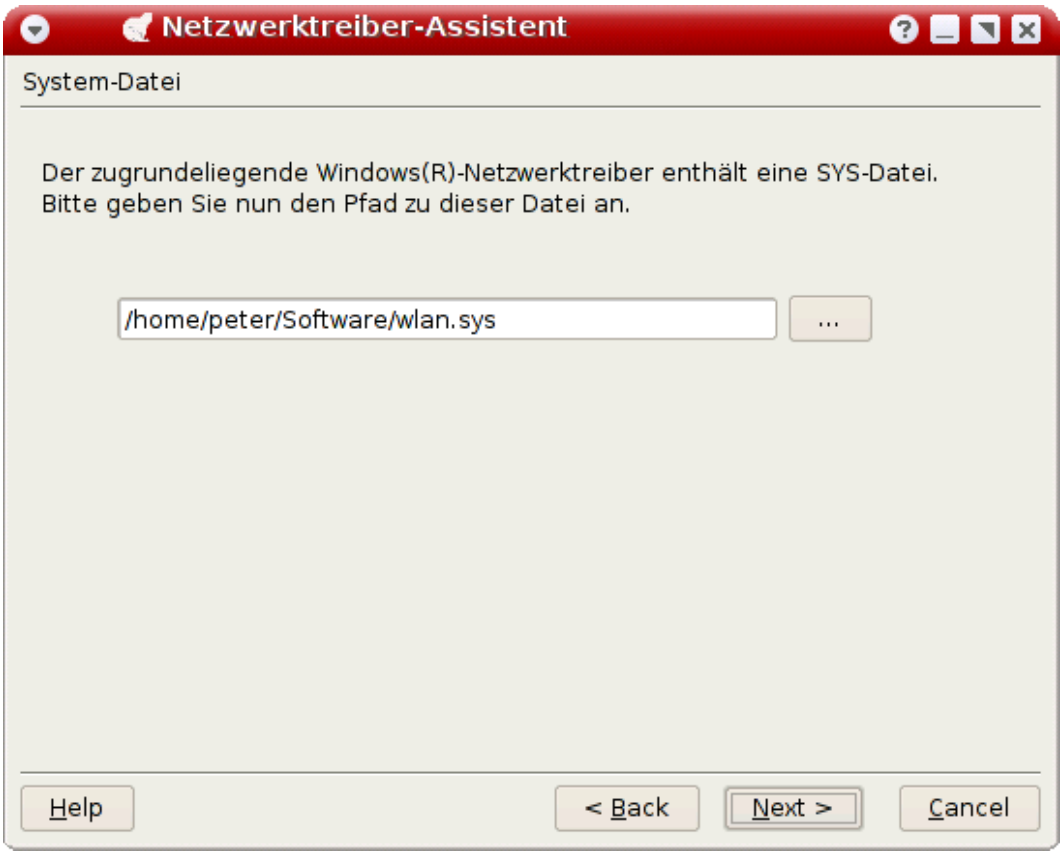


3.2.2 Konvertieren von Windows-Netzwerktreiber zu DesktopBSD-Treibern



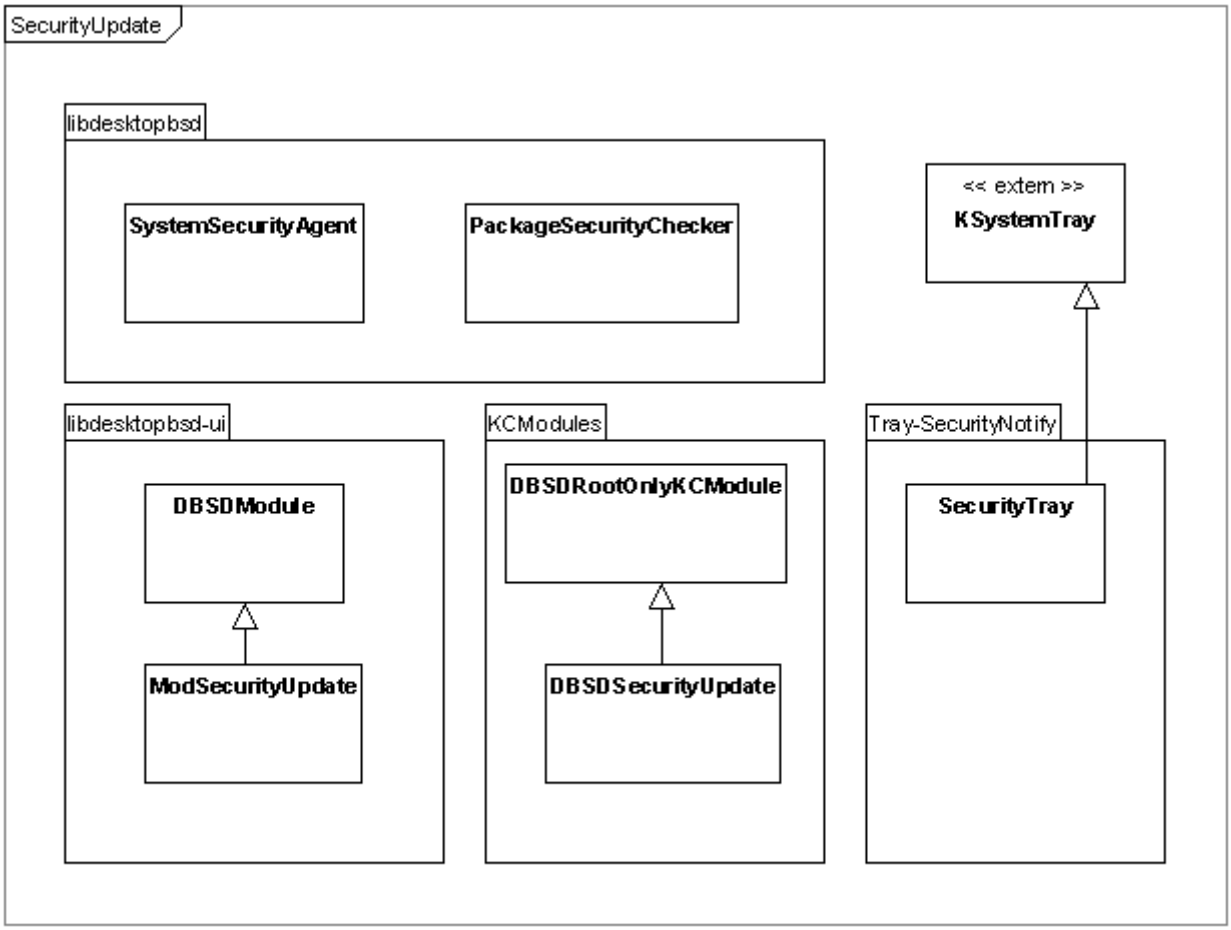
3.2.2.1 GUI-Prototypen


Netzwerktreiber-Assistent
Willkommen
Willkommen beim Netzwerktreiber-Assistent!
Dieser Assistent hilft ihnen, aus ihrem Windows(R)-Netzwerktreiber einen DesktopBSD-Treiber zu generieren.
Help
< Back
Next >
Cancel


Netzwerktreiber-Assistent
System-Datei
Der zugrundeliegende Windows(R)-Netzwerktreiber enthält eine SYS-Datei.
Bitte geben Sie nun den Pfad zu dieser Datei an.
/home/peter/Software/wlan.sys
...
Help
< Back
Next >
Cancel

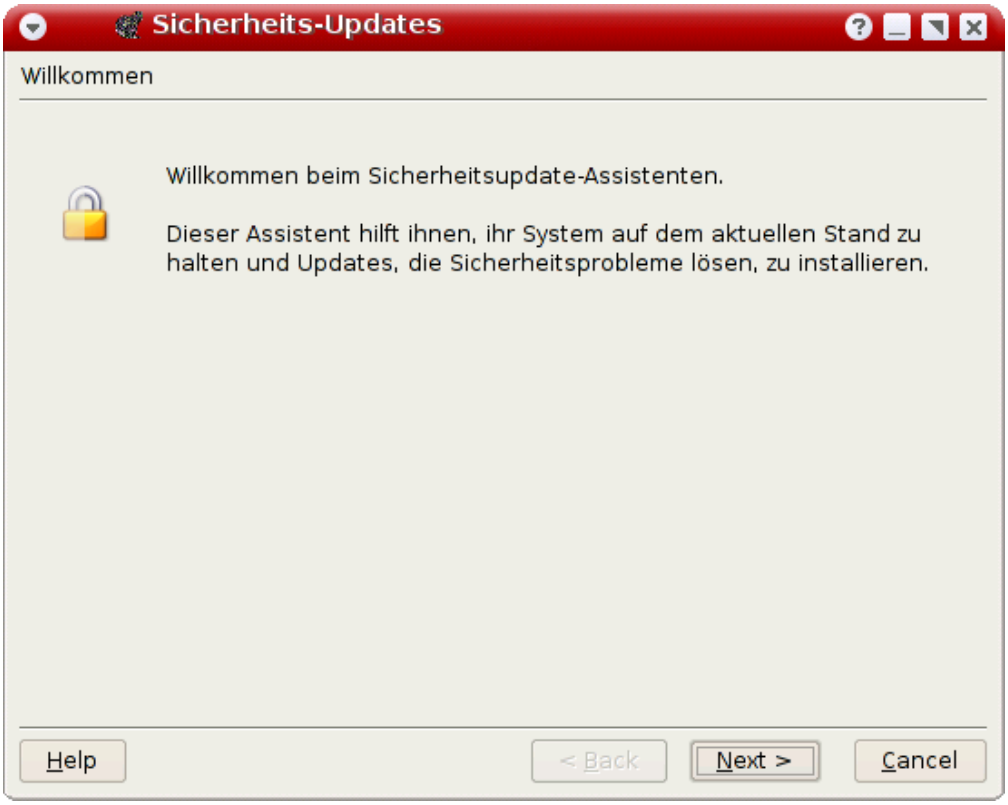
(In den weiteren Schritten werden INF-Dateien und etwaige Firmware-Dateien angegeben, danach wird der Treiber generiert)

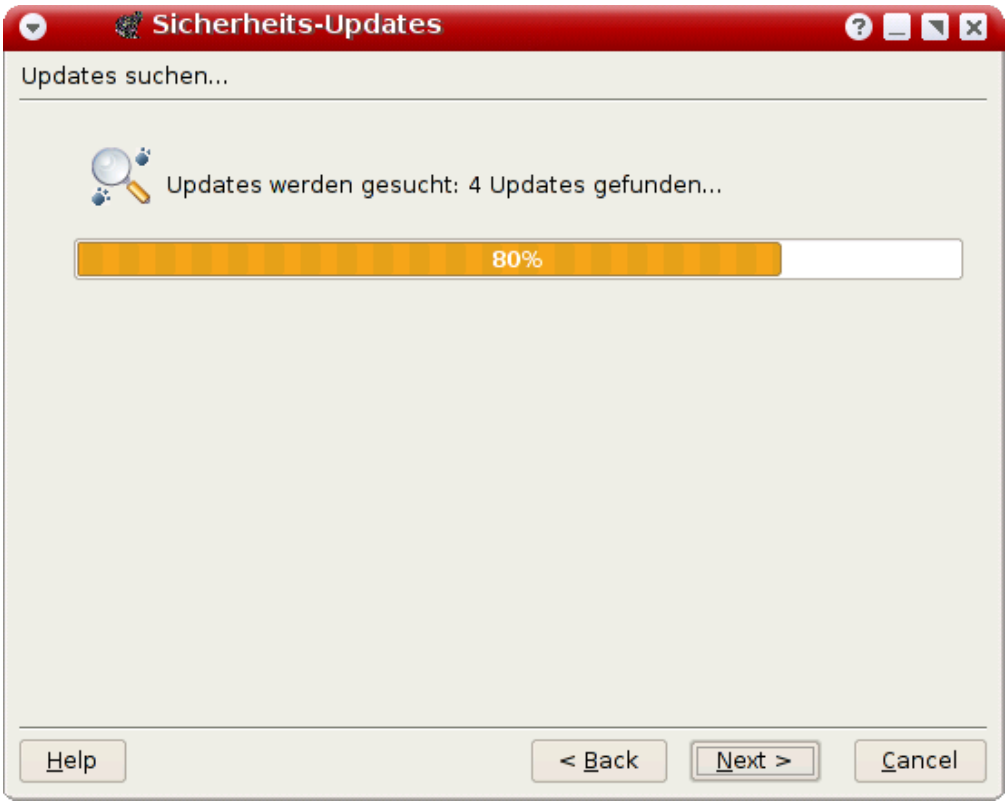
3.2.3 Automatisiertes Einspielen von Aktualisierungen am Betriebssystem

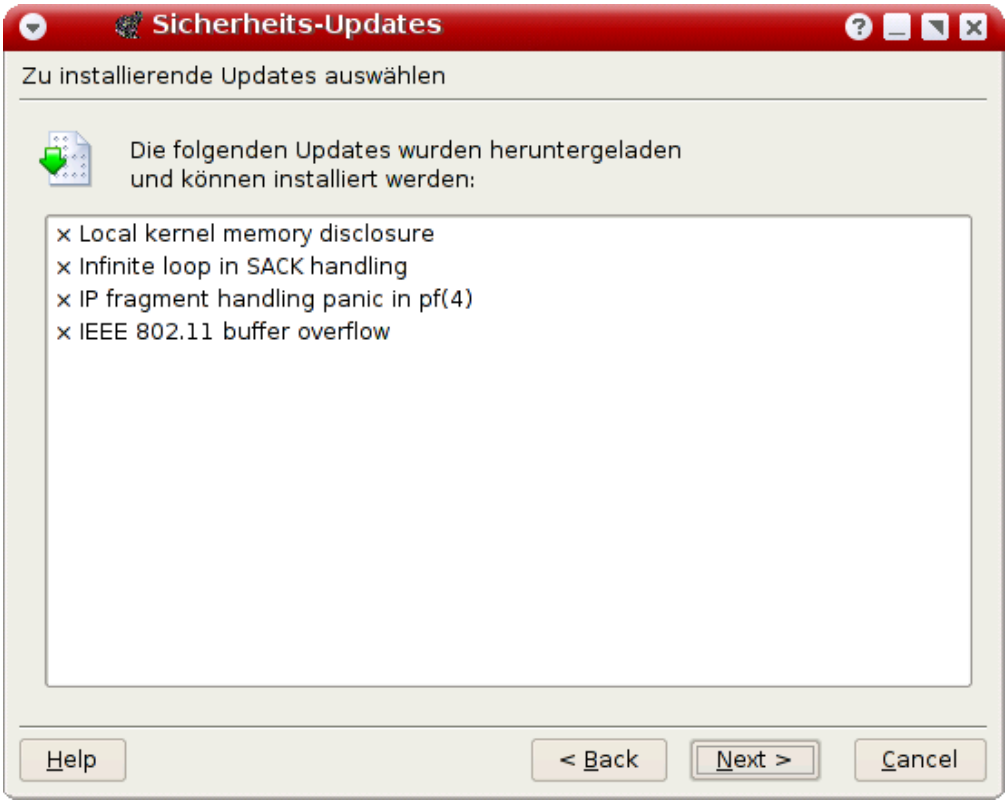


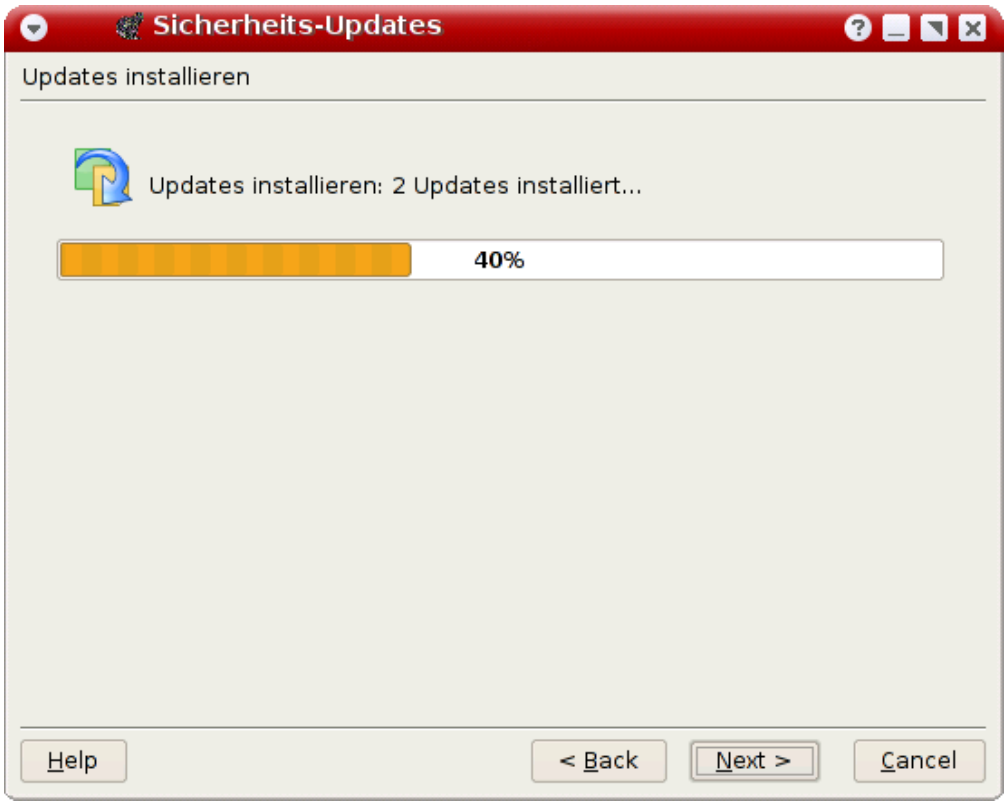
SystemSecurityAgent verfügt über Methoden, mit denen auf Sicherheitsprobleme des Betriebssystems und verfügbare Updates geprüft werden kann, weiters können diese Updates installiert werden. PackageSecurityChecker enthält Methoden zum Überprüfen installierter Software (die nicht zum Betriebssystem gehört) auf Sicherheitslücken. SecurityTray ist ein Symbol im Systembereich der Kontrollleiste (Tray-Icon), das je nach Verfügbarkeit von Updates ein anderes Bild zeigt und weiters den Benutzer benachrichtigt, sobald Probleme gefunden wurden. ModSecurityUpdate bietet die Möglichkeit, Einstellungen zur Regelmäßigkeit der Prüfungen zu treffen.

3.2.3.1 GUI-Prototypen





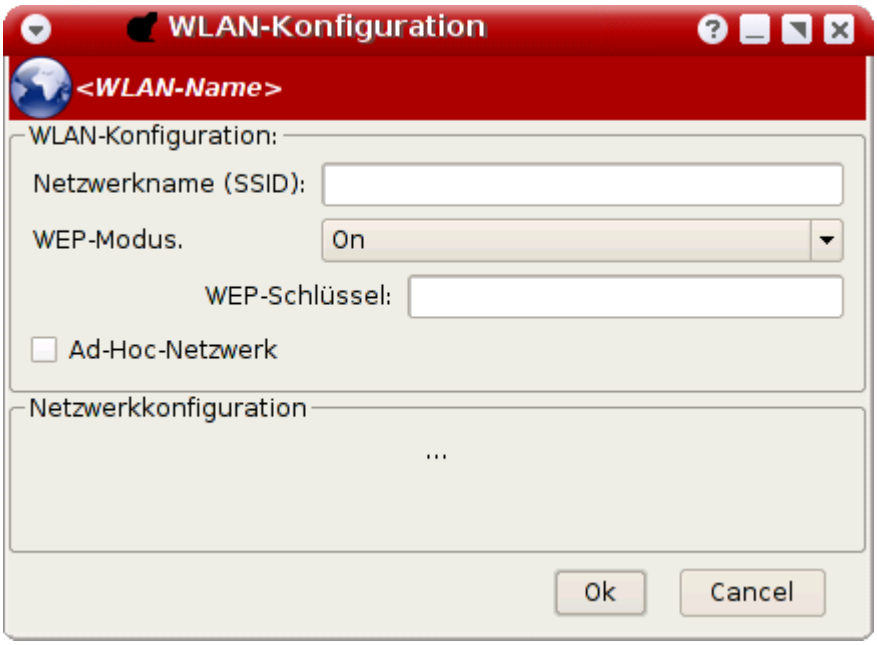

Sicherheits-Updates
Zu installierende Updates auswählen
Die folgenden Updates wurden heruntergeladen
und können installiert werden:
× Local kernel memory disclosure
× Infinite loop in SACK handling
× IP fragment handling panic in pf(4)
× IEEE 802.11 buffer overflow
Help
< Back
Next >
Cancel


Sicherheits-Updates
Updates installieren
Updates installieren: 2 Updates installiert...
40%
Help
< Back
Next >
Cancel

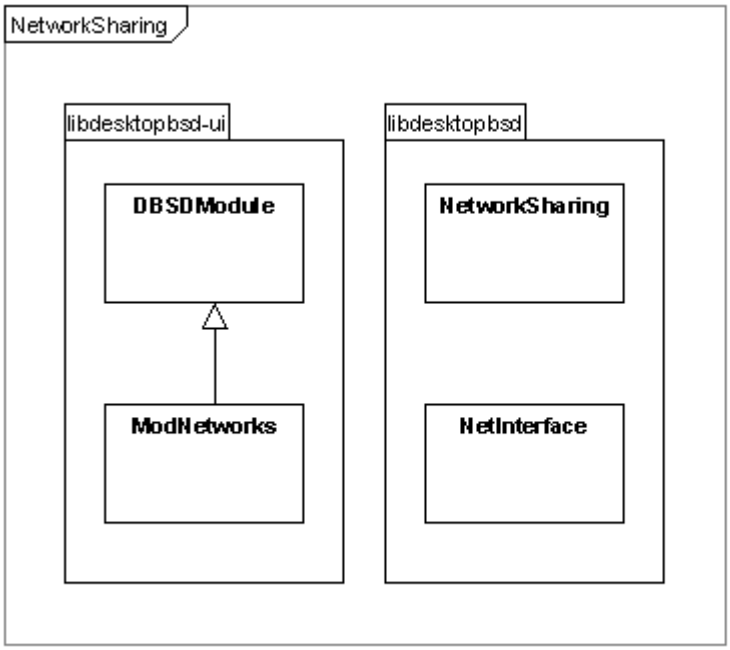
3.2.4 Konfiguration von Ad-Hoc-WLAN

Bei der Implementierung dieser Funktionalität werden nur bestehende Klassen geändert, deswegen wird die Planung erst in SE5 – Feinplanung vorgenommen.

3.2.4.1 GUI-Prototypen

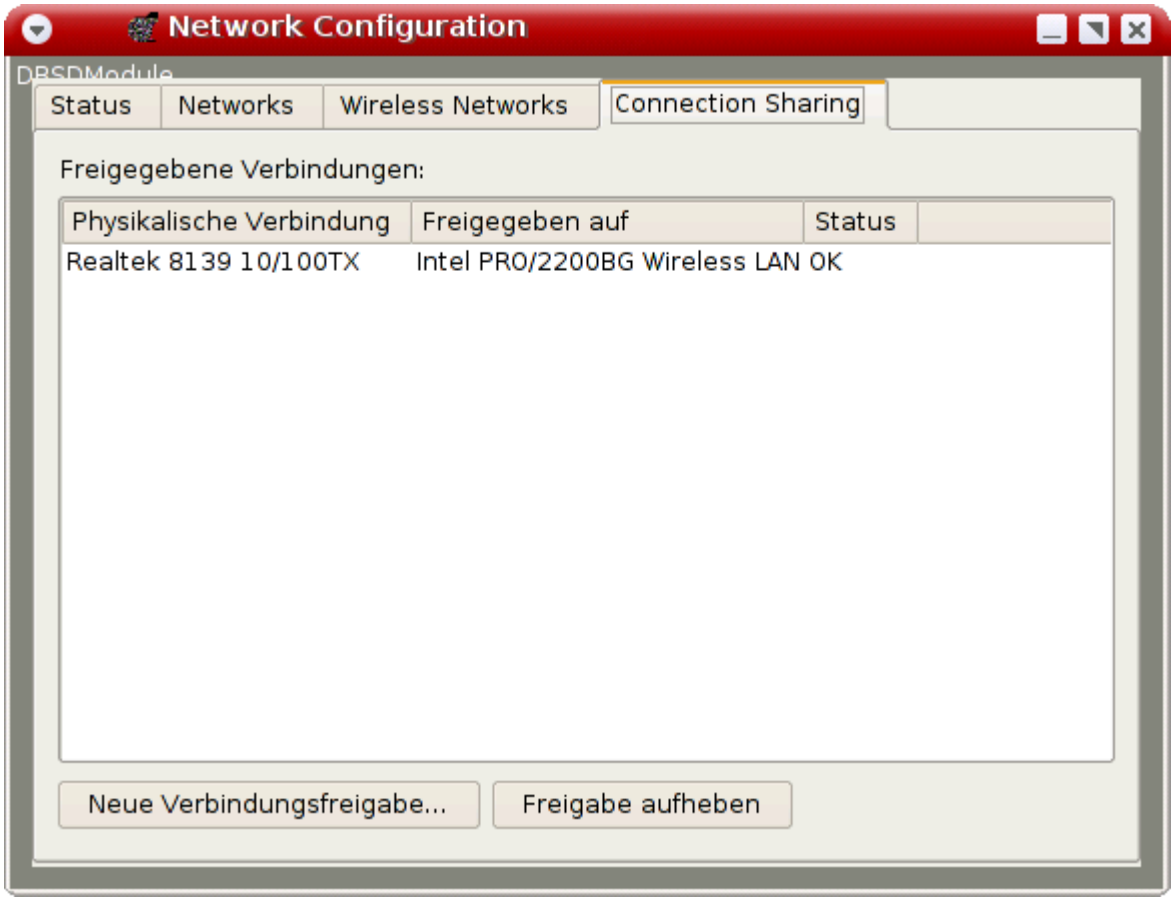


3.2.5 Internetverbindungsfreigabe



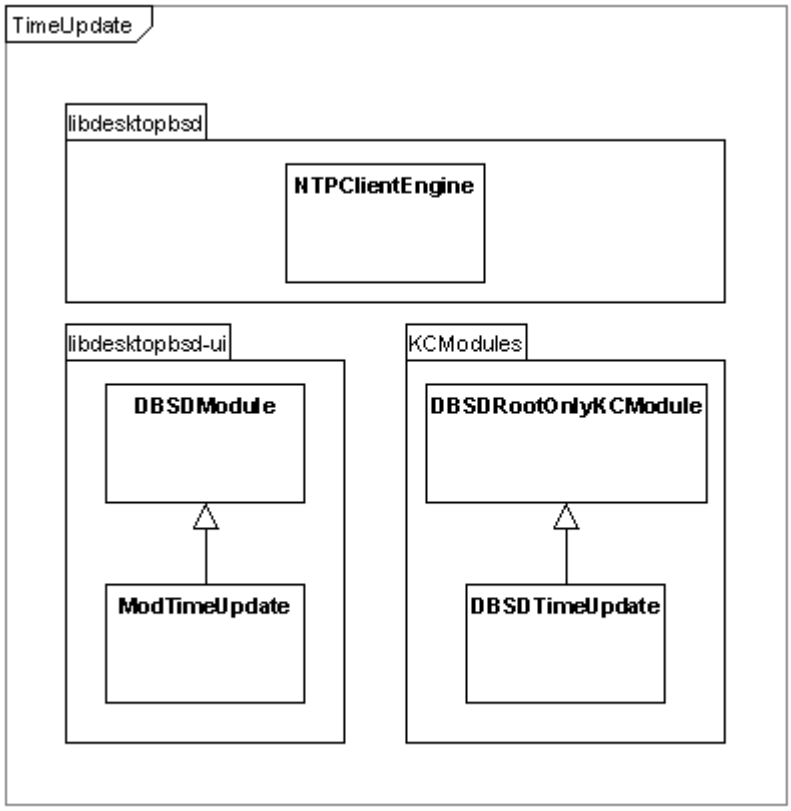
3.2.5.1 GUI-Prototypen

Die Klasse NetworkSharing bietet Möglichkeiten, über Bridging oder Routing zwei Netzwerkinterfaces zu „verbinden“, um von einem Netz auf das andere zugreifen zu können. NetInterface ist eine Netzwerkkarte, die entsprechende



Methoden zum Setzen einer IP-Adresse etc. enthält.

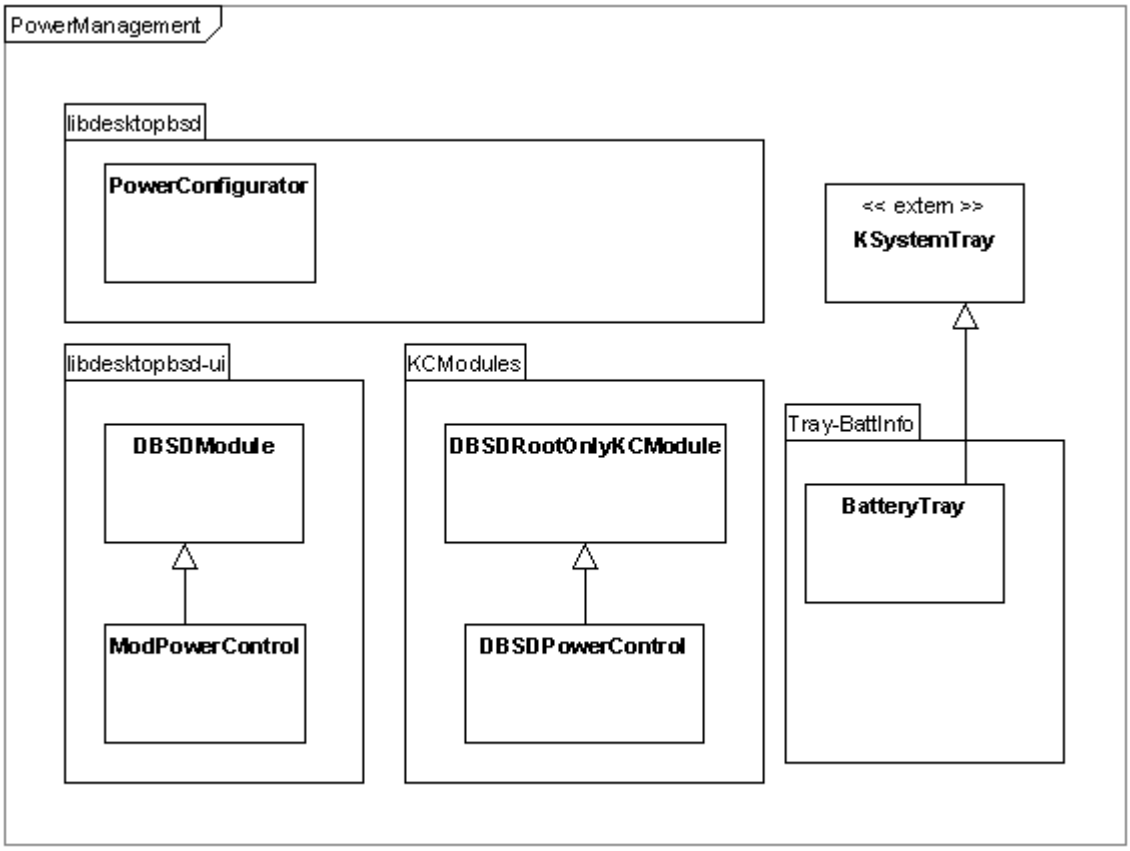
3.2.6 Zeitabgleich mit Timeserver



NTPClientEngine ist eine Klasse, die den Befehl ntpdate verwendet, um von einem entfernten NTP-Zeitserver die aktuelle Zeit abzurufen und als Systemzeit zu setzen. ModTimeUpdate erlaubt das Einstellen des Zeitservers.

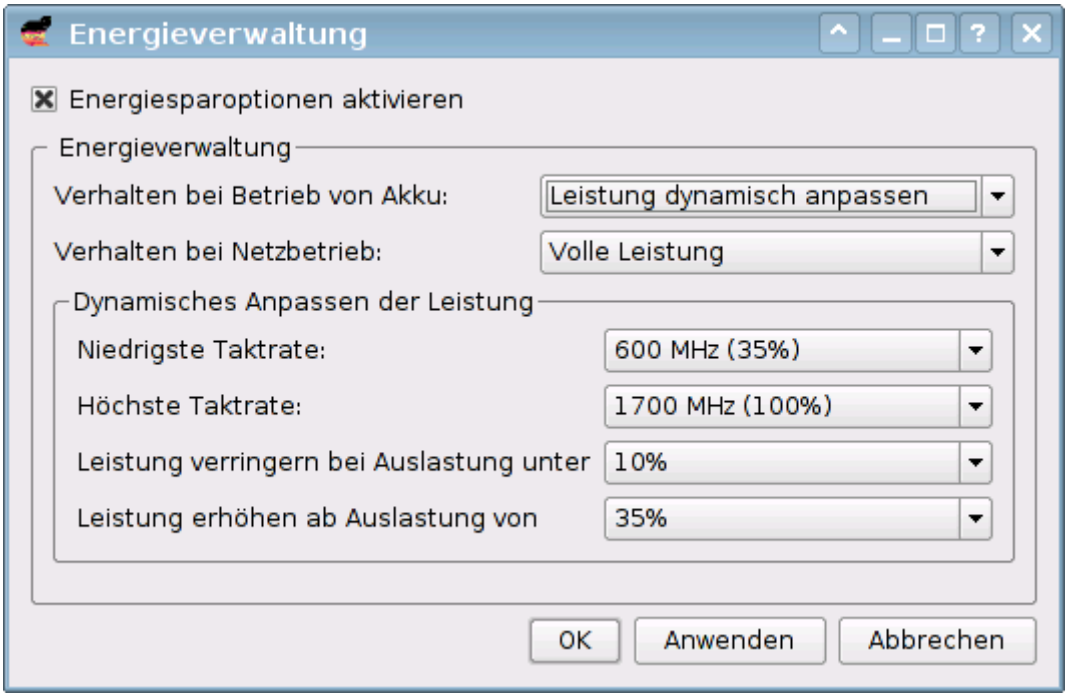
3.2.6.1 GUI-Prototypen

3.2.7 Konfiguration von Energiesparmechanismen



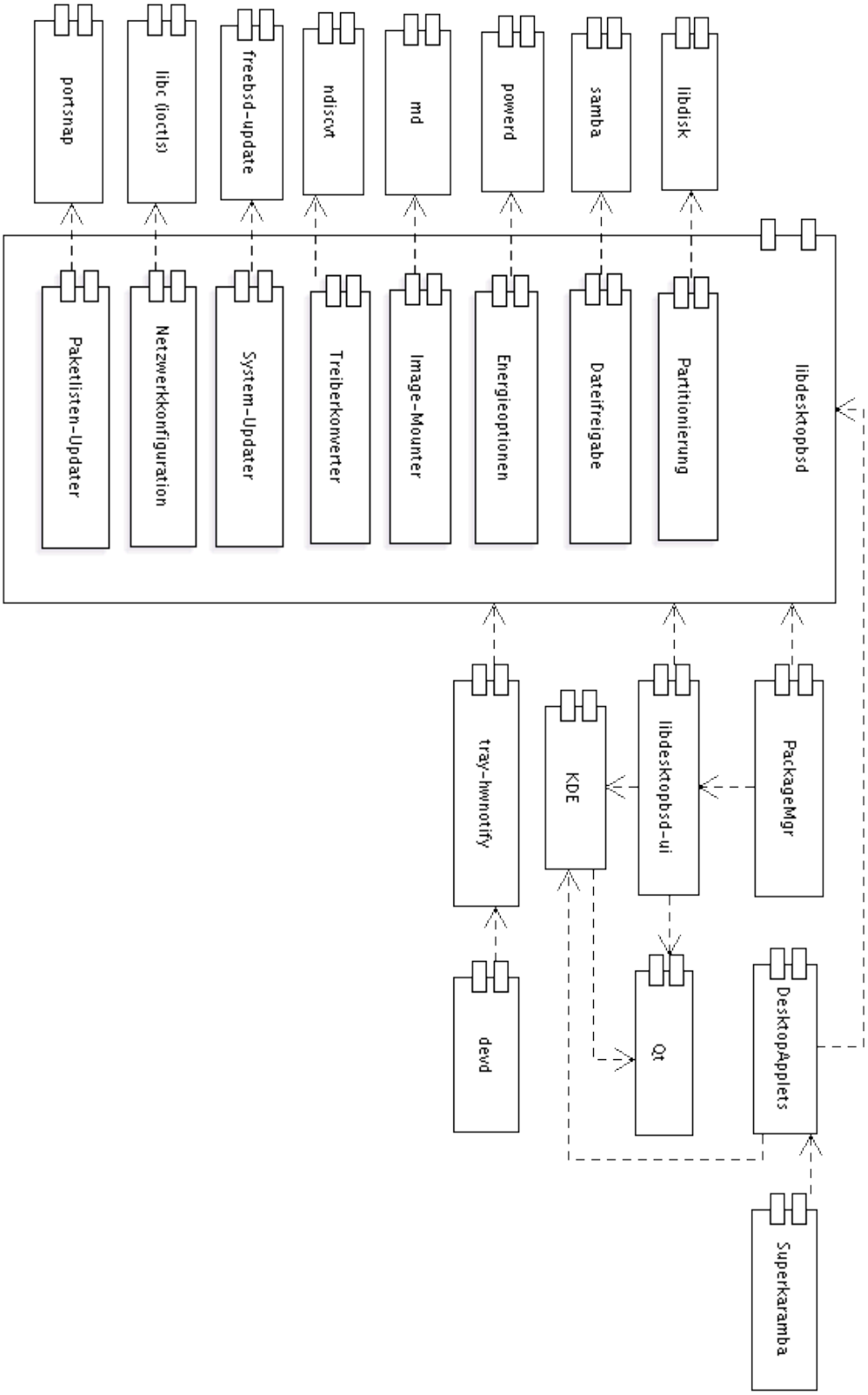
PowerConfigurator setzt die entsprechenden Einstellungen für powerd, um die Regelung der Leistung vorzunehmen. ModPowerControl ist eine grafische Oberfläche, mit der die Parameter der Leistungsanpassung festgelegt werden können (siehe GUI-Prototyp). BatteryTray ist ein Symbol im Systembereich der Kontrollleiste (Tray-Icon), das den Typ der Stromversorgung (Akku oder Netzbetrieb) und den Akkustand anzeigt. Weiters kann mit Doppelklick auf dieses Symbol oder das Rechtsklickmenü die Energieverwaltung (ModPowerControl) aufgerufen werden.

3.2.7.1 GUI-Prototypen


Energieverwaltung
Energiesparoptionen aktivieren
Energieverwaltung
Verhalten bei Betrieb von Akku:
Leistung dynamisch anpassen
Verhalten bei Netzbetrieb:
Volle Leistung
Dynamisches Anpassen der Leistung
Niedrigste Taktrate:
600 MHz (35%)
Höchste Taktrate:
1700 MHz (100%)
Leistung verringern bei Auslastung unter
10%
Leistung erhöhen ab Auslastung von
35%
OK
Anwenden
Abbrechen

4. Schnittstellenübersicht

4.1 Grafische Darstellung



Jedes ausführbare Programm wird als Komponente entworfen. Weiters erfolgt die Einteilung in Bibliotheken für *Kernklassen* (Backend), *Benutzerschnittstelle* (Frontend) und *Unterstützungssysteme*.

\+ Kernklassen sind von Benutzerschnittstelle getrennt.

- ✔ Programme, die nicht auf GUI-Komponenten angewiesen sind, brauchen diese nicht zu laden
- ✔ Erhöhte Wiederverwendbarkeit
- ✔ Leichte Wartung

5. Datenkatalog

Im Datenkatalog der SW-Architektur werden die an den Schnittstellen der SW-Einheit ausgetauschten Datenstrukturen mit Attributen, Datentypen und Wertebereichen beschrieben. Jede Programmiersprache und Plattform bietet hier eigene Lösungen, die bei der Definition zu berücksichtigen sind.

5.1 libdisk

Funktion: Partitionierung

- Interne Gerätebezeichnung (Zeichenkette)
- Struktur „disk“:

```
struct disk {
        char            *name;
        u_long          flags;
        u_long          bios_cyl;
        u_long          bios_hd;
        u_long          bios_sect;
        u_char          *bootmgr;
        u_char          *boot1;
        u_char          *boot2;
        struct chunk    *chunks;
        u_long          sector_size;
};
```

- Struktur „chunk“:

```
struct chunk {
        struct chunk    *next;
        struct chunk    *part;
        struct disk     *disk;
        daddr_t         offset;
        daddr_t         size;
        daddr_t         end;
        char            *name;
        char            *oname;
        chunk_e         type;
        int             subtype;
        u_long          flags;
        void            (*private_free)(void*);
        void            *(*private_clone)(void*);
        void            *private_data;
};
```

5.2 Samba

Funktion: Dateifreigabe über SMB-Protokoll

- Änderung der Konfigurationsdatei *smb.conf*
  - Einzelne Freigabe:

```
[Freigabename]
path = /verzeichnispfad
```

- Servername

```
[global]
...
server string = Servername
```

- Arbeitsgruppe

```
[global]
...
workgroup = Arbeitsgruppe
```

5.3 powerd

Funktion: Energieoptionen konfigurieren

- Änderung der Konfigurationsdatei *rc.conf*
  - Variable *powerd_enable*
    - *powerd_enable="YES"* aktiviert Energiesparmechanismen
    - *powerd_enable="NO"* (Standard) deaktiviert Energiesparmech.
  - Variable *powerd_flags*
    - *-a, -b, -n:* Verwendetes Profil bei Netzteilbetrieb, Akkubetrieb, nicht erkennbarem Zustand
      - *max*: Maximale Leistung
      - *min*: Kleinstmögliche Leistung
      - *adaptive*: Dynamisches Anpassen je nach Leistungsbedarf
    - *-i, -r:* Auslastung, bei der dynamisch hinunter-/hochgetaktet werden soll
    - *-p:* Intervall in ms, in dem der Wert abgefragt werden soll

Beispiel:

```
powerd_enable="YES" # Energiesparmechanismus aktiviert

# Bei Netzteilbetrieb maximale Leistung, bei Akkubetrieb dynamisch
# angepasst, bei 33% Leerlauf hochtakten, bei 66% Leerlauf hinuntertakten,
# Wert alle Sekunden abfragen
powerd_flags="-a max -b adaptive -r 33 -i 66 -p 1000"
```

5.4 md [mdconfig]

Funktion: Images als virtuelles Gerät registrieren (Images mounten)

- Kommandozeilenparameter

- *-a*: Neues Image registrieren
- *-d*: Image entfernen
- *-f <datei>*: Image-Datei
- *-t <typ>*: Typ des Images [standardmäßig vnode]

Beispiel:

```
mdconfig -a -t vnode -f CD.iso
```

Erstellt ein virtuelles Gerät aus der Datei CD.iso und gibt deren Systembezeichnung aus (zB `md0`).

5.5 ndiscvt

Funktion: Windows-Treiber zu DesktopBSD-Treiber konvertieren

- Kommandozeilenparameter:
  - *-i <inf-file>*: Treiber .INF-Datei
  - *-s <sys-file>*: Treiber .SYS-Datei
  - *-n <dev-name>*: Name, unter dem sich der Treiber später registriert
  - *-o <out-file>*: Pfad der resultierenden Headerdatei für einen Treiber
  - *-f <firmware-file>*: Angabe einer zusätzlichen Firmware

Beispiel:

```
ndiscvt -i netX500.inf -s pcx500.sys -o ndis_driver_data.h
```

Erzeugt aus der INF-Datei netX500.inf und der SYS-Datei pcx500.sys die Headerdatei ndis_driver_data.h, mit deren Hilfe ein Treiber für DesktopBSD erstellt werden kann.

5.5 freebsd-update

Funktion: System-Updates durchführen

- Kommandozeilenparameter:
  - *fetch*: Auf Updates prüfen und diese herunterladen
  - *install*: Heruntergeladene Updates verifizieren und installieren

5.6 libc (ioctls)

Funktion: Netzwerkinterfaces kontrollieren und konfigurieren

```
#include <sys/ioctl.h>

int
```

```
ioctl(int d, unsigned long request, ...);
```

d: Filedeskriptor eines Sockets zu dem Device

request: Auszuführende Operation

Dritter, optionaler Parameter: Datenstrukturen des Typs *ieee80211req*, *ifreq* und *ifmediareq*.

Auszug: Betriebssystem-Include net80211/ieee80211_ioctl.h

```
struct ieee80211req {
        char            i_name[IFNAMSIZ];       /* if_name, e.g. "wi0" */
        u_int16_t       i_type;                 /* req type */
        int16_t         i_val;                  /* Index or simple value */
        int16_t         i_len;                  /* Index or simple value */
        void            *i_data;                /* Extra data */
};
```

Auszug: Betriebssystem-Include net/if.h

```
/*
 * Interface request structure used for socket
 * ioctl's.  All interface ioctl's must have parameter
 * definitions which begin with ifr_name.  The
 * remainder may be interface specific.
 */
struct  ifreq {
        char    ifr_name[IFNAMSIZ];             /* if name, e.g. "en0" */
        union {
                struct  sockaddr ifru_addr;
                struct  sockaddr ifru_dstaddr;
                struct  sockaddr ifru_broadaddr;
                short   ifru_flags[2];
                short   ifru_index;
                int     ifru_metric;
                int     ifru_mtu;
                int     ifru_phys;
                int     ifru_media;
                caddr_t ifru_data;
                int     ifru_cap[2];
        } ifr_ifru;
#define ifr_addr        ifr_ifru.ifru_addr      /* address */
#define ifr_dstaddr     ifr_ifru.ifru_dstaddr   /* other end of p-to-p link */
#define ifr_broadaddr   ifr_ifru.ifru_broadaddr /* broadcast address */
#define ifr_flags       ifr_ifru.ifru_flags[0]  /* flags (low 16 bits) */
#define ifr_flagshigh   ifr_ifru.ifru_flags[1]  /* flags (high 16 bits) */
#define ifr_metric      ifr_ifru.ifru_metric    /* metric */
#define ifr_mtu         ifr_ifru.ifru_mtu       /* mtu */
#define ifr_phys        ifr_ifru.ifru_phys      /* physical wire */
#define ifr_media       ifr_ifru.ifru_media     /* physical media */
#define ifr_data        ifr_ifru.ifru_data      /* for use by interface */
#define ifr_reqcap      ifr_ifru.ifru_cap[0]    /* requested capabilities */
#define ifr_curcap      ifr_ifru.ifru_cap[1]    /* current capabilities */
#define ifr_index       ifr_ifru.ifru_index     /* interface index */
```

```
};
```

Auszug: Betriebssystem-Include net/if.h

```
struct ifmediareq {
        char    ifm_name[IFNAMSIZ];     /* if name, e.g. "en0" */
        int     ifm_current;            /* current media options */
        int     ifm_mask;               /* don't care mask */
        int     ifm_status;             /* media status */
        int     ifm_active;             /* active options */
        int     ifm_count;              /* # entries in ifm_ulist array */
        int     *ifm_ulist;             /* media words */
};
```

5.7 portsnap

Funktion: Paketliste mit Server synchronisieren

- Kommandozeilenparameter
  - *fetch*
    - Ist keine Paketliste vorhanden, wird eine heruntergeladen.
    - Ist eine Paketliste vorhanden, werden neue Updates heruntergeladen
  - *update*: Heruntergeladene Updates verifizieren und installieren
  - *extract*: Extrahiert eine Paketliste, wenn zuvor eine neue mit *fetch* heruntergeladen wurde

5.8 devd

Funktion: Benachrichtigung bei neu angesteckten/entfernten Geräten

devd ist ein Dienst, der beim Systemstart gestartet wird und beim Hinzufügen und Entfernen von Geräten in der Konfigurationsdatei *devd.conf* vorgegebene Aktionen ausführt.

Beispiel für die Syntax zur Konfiguration zum Aufrufen eines bestimmten Kommandos bei An-/Abstecken eines Gerätes:

```
# Anstecken
attach 0 {
        device-name "geraete-id";
        action "kommando";
};

# Abstecken
detach 0 {
        device-name "geraete-id";
        action "kommando";
```

```
};
```

Im konkreten Fall wird ein Remote-Prozeduraufruf (RPC) mittels DCOP vorgenommen, der die entsprechenden Komponente des DesktopBSD-Hardwareüberwachungs-Programms über das neue/entfernte Gerät informiert:

```
# Anstecken
attach 0 {
        device-name "geraete-id";
        action "dcop dbsd-hwnotify Notify attachDevice ${device-name}";
};
```

5.9 Superkaramba

Funktion: Anzeige von Informationen am Desktop.

Superkaramba ist ein Werkzeug, um einfach interaktive Anzeigen zum Desktop hinzuzufügen. Beispiele für Anwendungsgebiete sind:

- *Anzeige von Systeminformation wie CPU-Auslastung*
- *Information vom Internet anzeigen, wie Wetter und Schlagzeilen*

6. Designabsicherung

Der verwendete Entwurf ist Unterteilung in Komponenten:

Jedes ausführbare Programm wird als Komponente entworfen. Weiters erfolgt die Einteilung in Bibliotheken für *Kernklassen*, *Benutzerschnittstelle* und *Unterstützungssysteme*.

\+ Kernklassen sind von Benutzerschnittstelle getrennt.

- ✔ Programme, die nicht auf GUI-Komponenten angewiesen sind, brauchen diese nicht zu laden
- ✔ Erhöhte Wiederverwendbarkeit
- ✔ Leichte Wartung
- ✔ Da das Design bereits im vorhandenen System verwendet wird und sich bislang in den genannten Punkten bewährt hat, wird angenommen, dass das Design die Anforderungen in gewünschter Weise umsetzt.

7. Zu spezifizierende SW-Elemente

Keine Komponenten oder Module besitzen ausreichende Kritikalität oder Anforderungskomplexität, um eine weitergehende Spezifikation zu benötigen.

8. Abkürzungsverzeichnis

Abkürzung	Erklärung
NTP	*Network Time Protocol zur Abfrage der Zeit von einem entfernten Server*
SMB	*Dateifreigabe-Protokoll; u.a. implementiert von Microsoft Windows und Mac OS X*

9. Literaturverzeichnis

–

10. Abbildungsverzeichnis

–