Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Jan Henrik Ohlhoff

Entwurf und Realisierung einer Software zur Auswertung, Abrechnung, Nachweis und Überwachung der Lizenznutzung von standortübergreifend-nutzbaren Applikationen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan Henrik Ohlhoff

Entwurf und Realisierung einer Software zur Auswertung, Abrechnung, Nachweis und Überwachung der Lizenznutzung von standortübergreifend-nutzbaren Applikationen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 28. Mai 2013

Jan Henrik Ohlhoff

Thema der Arbeit
Entwurf und Realisierung einer Software zur Auswertung, Abrechnung, Nachweis und Überwachung der Lizenznutzung von standortübergreifend-nutzbaren Applikationen

Stichworte
Lizenzmanagement, Java, FlexLM, Drei-Schichten-Architektur

Kurzzusammenfassung
Software zum Lizenzmanagement dient dazu, Lizenzen effizient zu verwalten und Einsparungspotentiale zu erkennen. In dieser Arbeit wird eine Lizenzmanagement-Applikation entworfen und realisiert. Die Anwendung wird eine Drei-Schichten-Architektur verwenden.

Jan Henrik Ohlhoff

Title of the paper
Design and implementation of a software for the evaluation, billing, verification and monitoring of license usage by applications usable across sites

Keywords
license managment, Java, FlexLM, three-tier-architecture

Abstract
Software license management is used to manage licenses efficiently and to identify potential savings. In this thesis, a license management application is designed and implemented. The application will use a three-tier architecture.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel folgt eine kurze Einführung, welche die Motivation, das Ziel und den Aufbau dieser Bachelorarbeit erläutert.

1.1 Motivation

Software ist heute für Unternehmen eine wichtige Ressource, um am Markt wettbewerbsfähig und produktiv zu bleiben. Die Software wird meist in Form von Lizenzen erworben. Eine effiziente Verwaltung der Lizenzen ist für viele Unternehmen wichtig, um die Kosten im Rahmen zu halten. Der Entwurf und die Realisierung einer Applikation zur Lizenzverwaltung ist Ziel dieser Bachelorarbeit.

Durch Überlizenzierung kann dem Unternehmen ein wirtschaftlicher Schaden entstehen. Es müssen Kosten für nicht benötigte Lizenzen vom Unternehmen zusätzlich getragen werden. Die Unterlizenzierung stellt dagegen einen Lizenzverstoß dar. Dieser Verstoß kann je nach Lizenzvertrag schwerwiegende Folgen für das Unternehmen haben.

Bei der Firma Rheinmetall stellt besonders die Unterlizenzierung einen kritischen Punkt dar. In diesem Fall jedoch nicht im oben genannten Sinne eines Lizenzverstoßes, sondern in dem Sinne, dass die Anwender der CAD/PDM-Systeme gegebenenfalls ihre Anwendungen nicht mehr starten können und somit arbeitsunfähig wären. Weiterhin beinhalten beispielsweise sogenannte Freigabe-Workflows Arbeitsschritte, welche im Hintergrund auf die gleichen Systeme und somit Lizenzen zurückgreifen – diese Workflows würden in diesem Falle stehen bleiben.

Das Problem der Über- bzw. Unterlizenzierung kann durch den Einsatz einer geeigneten Lizenzmanagement-Software beseitigt bzw. zumindest verbessert werden. Durch die hier zu entwickelnde Applikation kann das Nutzungsverhalten aufgezeichnet und analysiert werden. Wenn festgelegte Schwellenwerte erreicht werden, soll eine Warnung per E-Mail an den Administrator versendet werden.

Durch Ablage der gesamten Lizenz-Nutzungsdaten in einer Datenbank kann bei einem Compliancecheck auch ein Nachweis geführt werden, dass in der Vergangenheit keine Vertragsverstöße aufgetreten sind.

Eine kostenstellen-, abteilungs- oder standortbezogene Abrechnung der anfallenden Lizenzkosten bietet bei Verwendung des Concurrent-User Lizenzmodells (siehe Abschnitt 2.1) den Vorteil, dass die Kosten dem Verursacher in Rechnung gestellt werden können.

1.2 Zielsetzung

Die Arbeit hat das Ziel, eine Applikation zu entwerfen und zu realisieren, die es möglich macht Lizenzen zu verwalten. Es sollen Nutzungsdaten grafisch ausgewertet und analysiert werden können, um eventuelle Einsparungspotentiale frühzeitig zu erkennen. Eine genaue Abrechnungsmöglichkeit der genutzten Lizenzen soll integriert werden. Des Weiteren werden alle erhobenen Daten in einer Datenbank archiviert, um einen Nachweis für eine spätere Kontrolle zu führen. Eine Überwachung der Lizenzserver soll ebenfalls möglich sein.

Die Applikation soll für andere Lizenzen und Lizenzserver erweiterbar sein. Die Applikation wird in drei Komponenten unterteilt:

- Eine Datenbank, in der die Nutzungsdaten gespeichert und archiviert werden
- Ein Dienst, der die Daten in die Datenbank einträgt
- Ein Frontend, welches die Daten auswertet und benutzerfreundlich darstellt

1.3 Gliederung

Die Bachelorarbeit teilt sich in acht Kapitel auf. Nach der Einleitung folgen in Kapitel 2 die Grundlagen. Hier werden unterschiedliche Lizenzierungsarten beschrieben. Des Weiteren wird auf schon vorhandene Lizenzmanager und ihre Möglichkeiten eingegangen. In den Abschnitten 2.3, 2.4 und 2.6 wird die Lizenzumgebung, die Applikationen und der Lizenzmanager der Firma Rheinmetall erläutert. In Kapitel 3 folgt die Analyse, in der die Anforderungen unter Abschnitt 3.1 beschrieben werden. In Kapitel 4 werden die Anforderungen spezifiziert. In Kapitel 5 wird die Konzeption und der Entwurf der Anwendung näher beschrieben. In Kapitel 6 wird die Realisierung der Anwendung an einigen Beispielen beschrieben. In Kapitel 7 folgt die Evaluierung und Bewertung der eigenen Arbeit. Zum Abschluss wird in Kapitel 8 die Arbeit zusammengefasst und ein kurzer Ausblick gegeben.

2 Grundlagen

In diesem Kapitel werden die Grundlagen beschrieben. Es wird unter anderem eine Einführung in die verschiedenen Lizenzmodelle, Lizenzserver, Applikationen zur Lizenzauswertung und die Lizenzumgebung der Firma Rheinmetall gegeben.

2.1 Arten der Lizenzierung

Die Hersteller von Software bieten den Unternehmen vielfältige Lizenzierungsarten an. So können die Lizenzen je nach Lizenzierungsart an einen Nutzer, an ein Gerät oder an die Anzahl der CPUs gebunden sein. Die wichtigsten Lizenzierungsarten werden im Folgenden beschrieben. Die Lizenzierungsformen wurden hauptsächlich von Wikipedia (http://www.wikipedia.de/), [Bürkner, 2003] und der Flexera Homepage (http://www.flexerasoftware.de/) entnommen.

Floating in einem Netzwerk (Concurrent-User Lizenzmodell)

Das Concurrent-User-Lizenzmodell beschreibt eine Lizenzierungsform, bei der die maximale Anzahl an Nutzern, die gleichzeitig auf eine Ressource zugreifen dürfen, in einer Datei auf dem Lizenzserver festgelegt wird. Diese Lizenzdatei wird durch den Lizenzgeber bereitgestellt. Die auch Floating-Lizenzen genannten Software-Güter können auf jedem Rechner installiert werden. Begrenzt und überwacht durch den zentralen Lizenzserver wird nur die gleichzeitige Nutzung der Lizenzen. Sind alle Lizenzen an die Benutzer vergeben, muss ein konkurrierender Benutzer warten bis eine Lizenz wieder freigegeben ist.

Dieses Lizenzmodell wird unter anderem auch bei der Firma Rheinmetall im CAD/CAM/CAE- und PDM/PLM-Umfeld eingesetzt.

Nutzerbezogen (Named-User Lizenzmodell)

Bei diesem Lizenzmodell ist die Lizenz an den Login-Namen des Benutzers im Netzwerk gebunden. Da eine Anmeldung an verschiedenen Rechnern im Netzwerk möglich ist, ist auch die Nutzung mit demselben Benutzer auf unterschiedlichen Rechnern möglich, solange der

Login-Name in der Liste der erlaubten Nutzer steht. Die gleichzeitige Nutzung der Lizenz auf mehreren Rechnern mit demselben Login-Namen ist meist nicht gestattet.

Demo (Evaluierungs-Lizenz)

Demo-Lizenzen dienen zur Evaluierung einer Software. Die Nutzung kann entweder zeitlich begrenzt sein oder eine bestimmte Anzahl an Ausführungen ist gestattet. Des Weiteren kann die Funktionalität der Demo-Lizenz von dem normalen Produkt abweichen und gegenüber der Vollversion eingeschränkt sein. Eine Demo-Lizenz wird auch als TBYB (Try-Before-You-Buy) bezeichnet.

Enable / Disable Produkt-Features

Ein Software-Produkt kann aus mehreren Leistungsmerkmalen (Features) bestehen. Diese Features können entweder selbst zusammengestellt werden oder sind schon vorab definiert. Zum Beispiel kann eine Software als Light-Version mit eingeschränkter Funktionalität, als Standard-Version oder als Professional-Version mit noch mehr Features lizenziert werden.

Hardwaregebunden (node locked)

Die Software wird für einen unbegrenzten Zeitraum an einen bestimmten Rechner gebunden. Dies geschieht über die Bindung der MAC-Adresse, Dongle-ID oder die Seriennummer der Festplatte an die Lizenz.

Pakete (packages)

Pakete oder Bundles definieren Funktionen von Produkten, die als zusammengehöriges Produkt lizenziert werden. Die Zusammenstellung wird im Lizenzzertifikat festgelegt. Der Lizenzgeber hat die Möglichkeit, das Produkt an die verschiedenen Branchen anzupassen und vorzudefinieren. Der Lizenznehmer kann dann ein Paket / Bundle mit den gewünschten Funktionen auswählen.

Dieses Lizenzmodell wird unter anderem auch bei der Firma Rheinmetall im CAD/CAM/CAE- und PDM/PLM-Umfeld eingesetzt.

Domänen Lizenz (Company Lizenz)

Bei diesem Lizenzmodell werden die Lizenzen einer Domäne (Intranet oder Internet) zugewiesen und können unternehmensweit genutzt werden.

Netzwerk-Segmente (site license)

Bei dieser Lizenzierungsform können die Lizenzen von allen Rechnern in einem bestimmten Netzwerksegment genutzt werden. Unterschieden wird hier anhand der im Unternehmensnetz eindeutigen IP-Adresse. Mit dieser Form der Lizenzierung können Lizenzen auf Regionen, Abteilungen oder Unternehmensbereiche eingeschränkt werden. Mit sogenannten Wild-Cards können ganze Subnetze für eine Lizenz freigeschaltet werden.

2.2 Gängige Lizenzmanager

Die nachstehend aufgelisteten Lizenzmanager werden im CAD/CAM/CAE- und PDM/PLM-Umfeld eingesetzt.

2.2.1 FlexLM

FlexLM, jetzt bekannt als FLEXnet von der Firma Flexera Software ist ein Lizenzmanager, der es ermöglicht, das Concurrent-User-Lizenzmodell umzusetzen. Es wird ein Pool an Lizenzen erworben, der dann von unterschiedlichen Benutzern zur gleichen Zeit benutzt werden kann, bis die Obergrenze an Lizenzen erreicht ist. Des Weiteren kann mit FlexLM die Lizenz an einen Rechner gebunden werden (node locked). FlexLM wird sehr häufig für Applikationen aus dem CAD/CAM/CAE- und PDM/PLM-Umfeld als Lizenzmanager eingesetzt. Eine detaillierte Beschreibung über die Arbeitsweise von FlexLM ist im Abschnitt 2.6 zu finden. FlexLM besitzt weiterhin die Möglichkeit für einen „Failover“-Modus, der in Abschnitt 2.3 näher beschrieben wird.

2.2.2 License Use Management

License Use Management (LUM) ist ein Lizenzmanager der Firma IBM. Ein LUM Lizenzserver kann Lizenzen an verschiedene Clients im Netzwerk verteilen oder direkt auf einem lokalen Rechner laufen (node locked) und dort die Lizenz zuteilen.

Eine weitere Variante der Serverkonfiguration für spezielle Lizenztypen ist der *Central Registry License Server*. Dieser Server ist ein Verzeichnisdienst auf den alle anderen Lizenzserver zugreifen können, um z. B. reservierbare Lizenzen zu steuern.

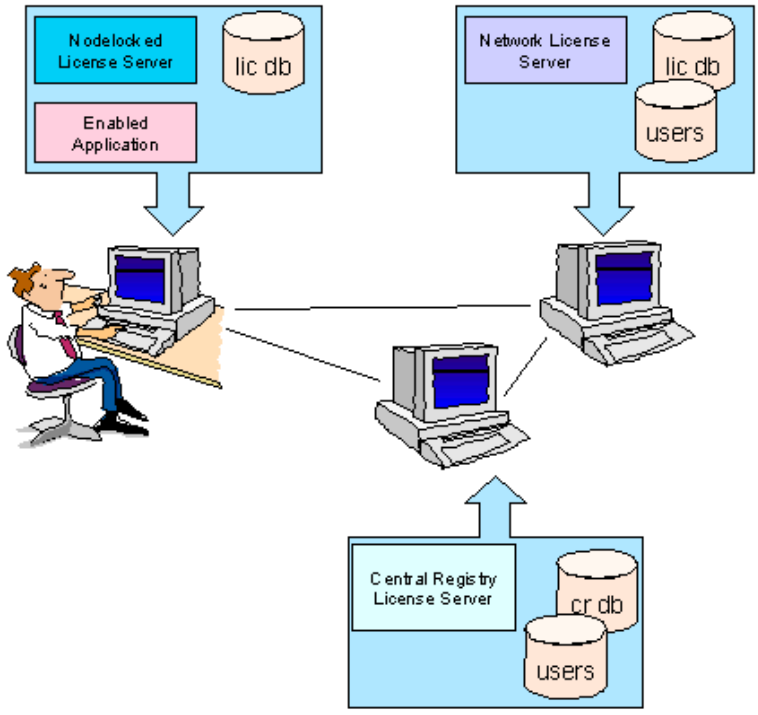

Abbildung 2.1: License Use Management [Grafik entnommen aus IBM, 2013]

2.2.3 Dassault Systèmes License Server (DSLS)

Der Lizenzserver der Firma Dassault Systèmes bietet das Concurrent-User-Lizenzmodell an, sowie einen „Failover"-Modus. Bei diesem Modus gibt es drei Lizenzserver, die den Ausfall eines Lizenzservers tolerieren und trotzdem wie gewohnt Lizenzen ausgeben. „Node Locked"-Keys werden in der neuesten Version nicht mehr unterstützt. Der Dassault Systèmes License Server dient hauptsächlich für die im eigenen Haus vertriebenen Produkte wie CATIA oder SolidWorks.

2.3 Lizenzumgebung der Firma Rheinmetall

2.3.1 Lizenzmodell

Das Concurrent-User-Lizenzmodell wird für NX und Teamcenter Engineering bei der Firma Rheinmetall eingesetzt. Für NX gibt es mehrere Bundles (z. B. DESIGNER, P1, NX12420, NX13500 und ADV5XMACH), die wiederum mehrere Komponenten beinhalten (siehe Abbildung 2.2). Es sollen sowohl die einzelnen Bundles, als auch die Komponenten ausgewertet

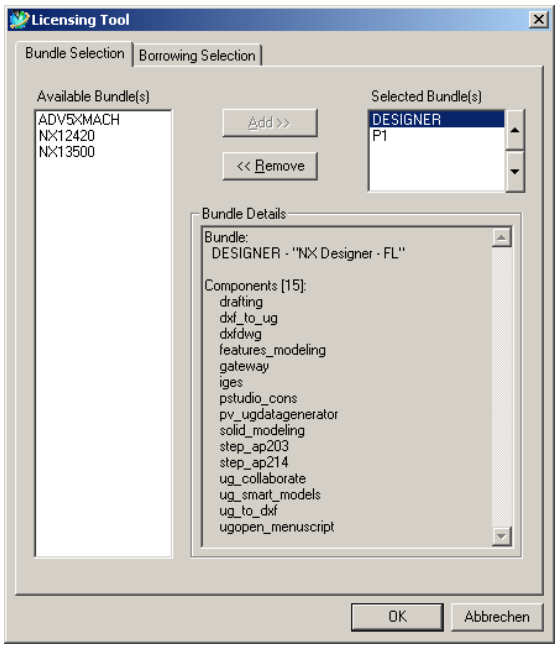

Abbildung 2.2: Bundles und deren Komponenten für NX

werden können. Es wird also eine Mischung aus dem in Abschnitt 2.1 aufgelisteten Concurrent-User-Lizenzmodell und dem Paket-Lizenzmodell eingesetzt. Der Zugriff erfolgt von verschiedenen Standorten aus.

Des Weiteren werden noch die Bundles ICIDO zur 3D-Visualisierung, Origin Pro, BCT Raster und Abaqus auf einem anderen Port des Lizenzserver-Triples bereitgestellt. Diese Bundles sollen jedoch nicht mit der Applikation ausgewertet werden.

2.3.2 Lizenzserver-Triple

Es besteht ein Lizenzserver-Triple, das von mehreren Standorten aus genutzt wird. Das Triple ist über zwei Standorte hinweg verteilt. Von dem Triple müssen je zwei Server erreichbar sein, um eine Lizenz an einen Client zu vergeben. Es besteht also eine Ausfallsicherheit falls ein Server des Triples nicht mehr erreichbar ist. Die Lizenzen müssen nur auf einem Server gepflegt werden. Die Server senden sich periodisch gegenseitig eine Nachricht um sicherzustellen, dass mindestens zwei Server erreichbar sind. Die Lizenzserver werden als primär, sekundär oder tertiär identifiziert. Ein Server wird als Master zugeordnet und ist für folgende Aufgaben zuständig:

- Bereitstellung von Lizenzen für „FLEXenabled“ Aplikationen

- Aufnahme von Informationen in das Debug-Logfile
- Aufnahme von Informationen in das Report-Logfile

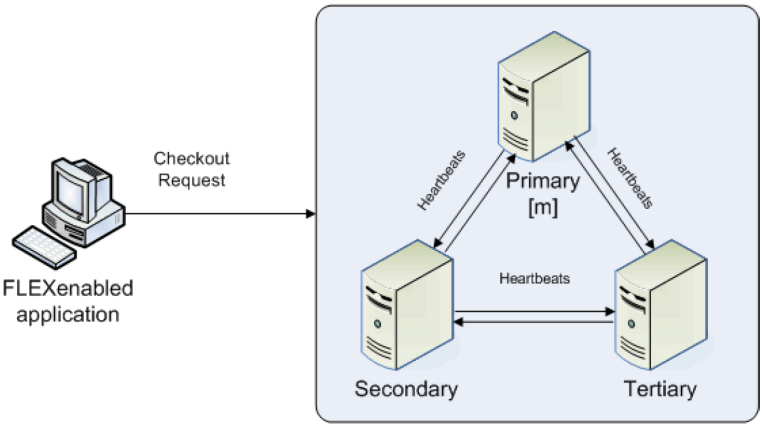

Abbildung 2.3: Server Triple [Grafik entnommen aus Acresso, 2008, S. 66]

Wenn der Master-Server nicht mehr erreichbar ist, wird ein anderer Server der Master. In der Grafik (siehe Abbildung 2.3) ist der primäre Server auch der Master [m]. Wenn eine „FLEXenabled“ Applikation eine Lizenzanfrage sendet, antwortet der Master-Server und stellt die Lizenz dem anfragenden Client bereit.

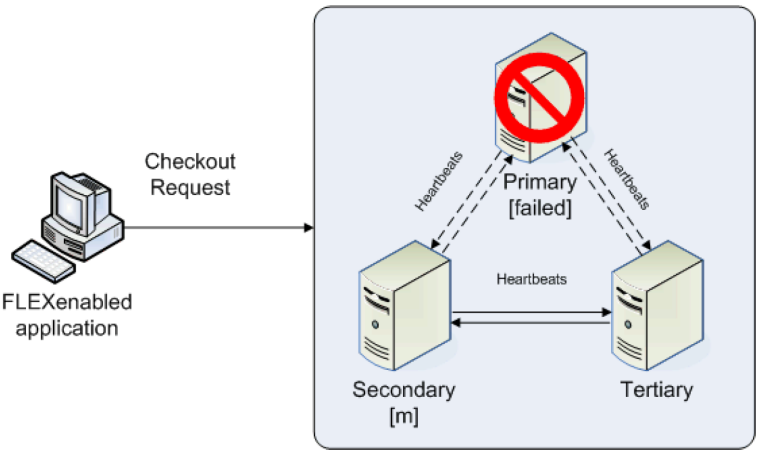

Abbildung 2.4: Server Triple mit Ausfall [Grafik entnommen aus Acresso, 2008, S. 67]

Wenn der Master-Server nicht mehr verfügbar ist, wird der sekundäre Lizenzserver der Master [m] und stellt Lizenzen an die Clients bereit. Der tertiäre Lizenzserver kann niemals der Master werden. Wenn der primäre und der sekundäre Lizenzserver ausfallen, können keine Lizenzen

mehr an die Clients bereitgestellt werden. Der Master-Server stellt keine Lizenzen bereit, solange es keine zwei Lizenzserver in der Triade gibt, die auch untereinander kommunizieren.

Zu Beginn liest jeder Lizenzserver die Lizenzdatei ein und prüft, ob er mit den anderen Lizenzservern kommunizieren kann. Bis zu Herstellung der ersten Verbindung mit den anderen Lizenzservern sendet der Lizenzserver periodisch Nachrichten. Wenn die erste Kommunikation erfolgreich war, sendet jeder Lizenzserver periodisch einen „Heartbeat“ an die anderen Server. „Heartbeats“ sind Nachrichten die über TCP/IP versendet werden. Der Lizenzserver wartet auf die Rückmeldung der anderen Lizenzserver. Wenn ein Lizenzserver keine Rückmeldung bekommt, beendet er die Lizenzbereitstellung.

2.3.3 Lizenzserver-Triple bei der Firma Rheinmetall

In der nachstehenden Abbildung 2.5 wird die Verteilung des Lizenzserver-Triples über die Standorte hinweg gezeigt. Ausgewählt für die Server wurden die Standorte mit den performantesten Internetanbindungen. Es fällt auf, dass der Standort Unterlüß zwei Server beinhaltet. Die Server befinden sich jedoch in getrennten Netzwerk-Segmenten und so können bei einem Ausfall eines Lizenzservers in Unterlüß immer noch Lizenzen vergeben werden. In grün sieht man die Standorte der Tochtergesellschaft Rheinmetall Waffe Munition GmbH. In blau dargestellt die der Tochtergesellschaft Rheinmetall Landsysteme GmbH und in gelb die Rheinmetall Defence Electronics GmbH.

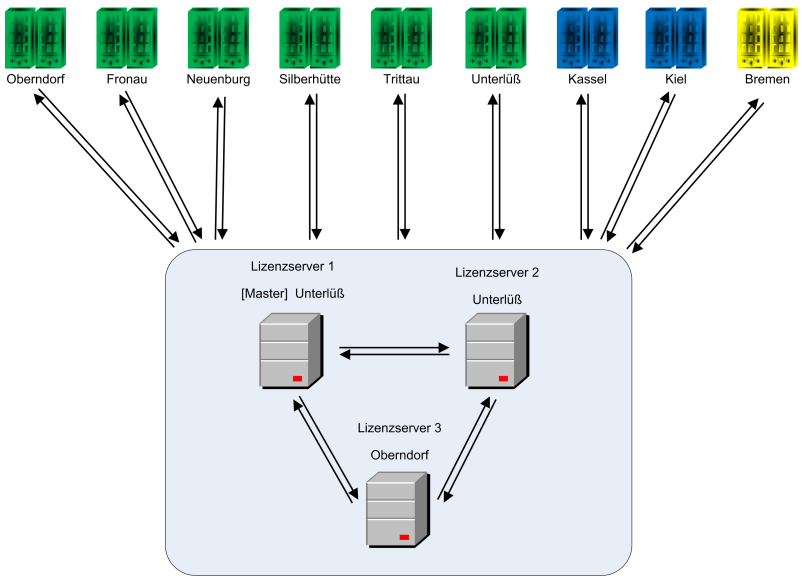

Abbildung 2.5: Lizenzserver-Triple und Standorte bei Rheinmetall

2.4 Applikationen zur Lizenzauswertung

Für den Lizenzmanager FlexLM und meist auch andere Lizenzmanager gibt es verschiedene kommerzielle und freie Applikationen zur Lizenzauswertung wie *JTB FlexReport*, *X-Formation*, *Open iT*, *phpLicenseWatcher*, *.inCharge* und *OpenLM*.

Zwei dieser Applikationen werden nachfolgend vorgestellt. Die Applikationen *phpLicenseWatcher* und *.inCharge* wurden bei der Firma Rheinmetall installiert und getestet.

Bei den anderen Applikationen wären weitere Aufwände in Form von Schulungen durch den Anbieter bzw. Lizenzgebühren beim Kauf entstanden, die so nicht im Budget vorgesehen sind.

2.4.1 .inCharge – Management und Analyse von Softwarelizenzen

.inCharge ist eine professionelle Softwarelösung, die es ermöglicht, zentral die Nutzung des Lizenzbestandes zu überwachen, zu managen und diesen entsprechend auszuwerten und zu optimieren.

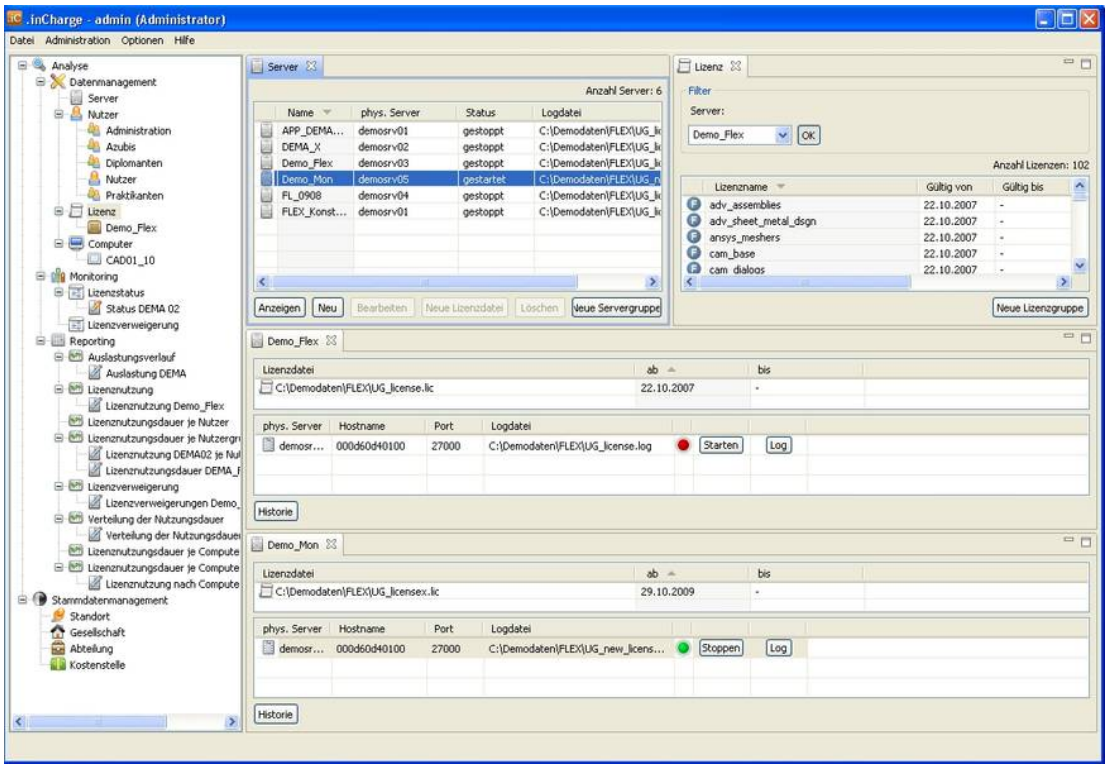

Abbildung 2.6: Die Lizenzauswertungs-Software .inCharge [Grafik entnommen aus NovaTec, 2012]

.inCharge ist die Lösung, die die Flexibilität ermöglicht, eine große Anzahl von hochwertigen Software-Investitionsgütern, die mit FlexNet, LUM oder Dassault Systèmes License Server (DSLS) gemanagt werden, zu monitoren, zu managen und deren Nutzung auszuwerten und zu verrechnen.

Es ist möglich, eine Vielzahl an Auswertungen über die Lizenznutzung zu erhalten, um Informationen zu beschaffen und einen Gesamtüberblick zu bekommen, wer wann von welchem Arbeitsplatz welche Lizenz wie lange genutzt hat. Informationen von ganzen Unternehmenseinheiten (Projekt, Abteilung, Kostenstelle, Standort) können ausgewertet werden.

.inCharge sammelt diese Informationen in einer eigenen zentralen Datenbank, so dass auf diese Informationen jederzeit über die Reporting-Schnittstelle von *.inCharge* zugegriffen werden kann. [vgl. NovaTec, 2012]

Beim Test dieser Software stellte sich heraus, dass sie nur schwer auf die Bedürfnisse der Firma Rheinmetall anpassbar ist und dieses wiederum mit hohen Kosten verbunden ist.

2.4.2 phpLicenseWatcher

phpLicenseWatcher ist eine Open-Source Applikation, mit welcher der Zustand eines FlexLM oder LUM Lizenzservers angezeigt werden kann. Des Weiteren kann die Auslastung der Lizenzen bzw. das Ablaufdatum einzelner Lizenzen aktuell angezeigt werden.

License port@server	Description	Status	Current Usage	Available features/license	Master	LM Hostname	lmgrd version
☐	28000@C71117SR0703	NX TestServer	UP	Details	Listing/Expiration dates	C71L17SR0703	v11.6

Abbildung 2.7: Die Software phpLicenseWatcher: Übersicht

Der *phpLicenseWatcher* schreibt seine Daten in eine MySQL-Datenbank. Die Daten dienen zum Darstellen der Lizenznutzung über einen gewissen Zeitraum. Es ist möglich, eine Warnmeldung per E-Mail zu versenden, bevor eine Lizenz abläuft. Die Applikation kann anzeigen, wann eine Lizenzanforderung abgewiesen wurde oder wie oft ein Nutzer eine bestimmte Lizenz gezogen hat. Der *phpLicenseWatcher* bietet keine weiteren Reports, die z. B. zu einer Abrechnung herangezogen werden können. Deswegen schied die Verwendung dieser Applikation für die Firma Rheinmetall aus.

These licenses will expire within 10 days. Licenses will expire at 23:59 on the day of expiration.

Server	Server description	Feature expiring	Expiration date	Days to expiration	Number of license(s) expiring
28000@C71117SR0703	NX TestServer	wave	31-jul-2012	**Already expired**	1
28000@C71117SR0703	NX TestServer	cattoug	30-jun-2012	**Already expired**	1
28000@C71117SR0703	NX TestServer	catv5_nx_sca	30-jun-2012	**Already expired**	1

Abbildung 2.8: Die Software phpLicenseWatcher: abgelaufene Lizenzen

2.5 Teamcenter und NX

Teamcenter und NX von der Firma Siemens PLM Software werden nachfolgend vorgestellt. Diese beiden Systeme und die von diesen verwendeten Bundles sollen hauptsächlich in Bezug auf die Lizenznutzung analysiert werden.

2.5.1 NX

NX ist ein interaktives CAD/CAM/CAE-System. Die Funktionen der Applikation sind in Pakete bzw. Komponenten aufgeteilt. Diese Funktionen müssen entsprechend lizenziert sein. Der Einstiegspunkt in das Programm ist *NX Gateway*, wofür jeder Benutzer eine Lizenz besitzen muss. Andere Anwendungen sind optional und können frei konfiguriert werden, wobei es schon vorkonfigurierte Bundles für die jeweilige Branche gibt.

NX ist ein 3D-System für Entwicklung und Konstruktion, Zeichnungserstellung, Simulation und Fertigung. Es ermöglicht die Volumen- und Flächenkonstruktion in voll-, teil- und nicht-parametrisierter Form zu erstellen. NX basiert auf dem Parasolid-Kern (Basis der Geometriedarstellung).

Nachdem die Konstruktion abgeschlossen ist, ermöglicht die Anwendung „Manufacturing“ (CAM) das Eingeben von Fertigungsdaten, wie dem Werkzeugdurchmesser, sowie das automatische Erzeugen einer Quelldatei für die Werkzeugpositionierung (Cutter Location Source File, CLSF), die zum Steuern der meisten NC-Maschinen verwendet werden kann.

Die CAE-Funktionen bieten viele Möglichkeiten für die Simulation von Produkten und Baugruppen im gesamten Bereich der Konstruktion. [vgl. Wikipedia, 2012] [vgl. Klette u. a., 2011]

2.5.2 Teamcenter

Die Teamcenter Produktlinie von Siemens PLM Software ist ein Produkt-Daten-Management-System (PDM) und dient der Definition und Verwaltung digitaler Produktmodelle über den ganzen Produktlebenszyklus.

Mit Teamcenter sind Unternehmen in der Lage unterschiedlichste Produktdaten zu definieren, zu verwalten, diese abzurufen, zu integrieren und über verteilte Standorte sowie mit Partnern und Lieferanten auszutauschen.

Dies ermöglicht eine effiziente Gestaltung der Unternehmensprozesse. Die Informationen, die abgebildet werden, umfassen Produktanforderungen, Projektdaten, Engineering-Daten, Komponenten, Dokumente, Produktkonfigurationen und mehr. Teamcenter ist eine modular aufgebaute Software, Komponenten für die Visualisierung und Zusammenarbeit führen Abteilungen und Unternehmen mit Geschäftspartnern und Kunden in einer einheitlichen Umgebung zusammen.

Teamcenter ist ein offenes System und der Einsatz von Industrie-Standards ermöglichen Interoperabilität zwischen unterschiedlichsten IT-Anwendungen. [vgl. Graf u. a., 2012]

2.6 Lizenzierung der Software mit FlexLM

FlexLM ist der Lizenzmanager, der bei der Firma Rheinmetall eingesetzt wird. Nachfolgend wird die Funktionsweise des Servers erläutert. Es werden die Hauptkomponenten von FlexLM und der Lizenzanforderungs-Prozess vorgestellt.

2.6.1 Hauptkomponenten Siemens PLM Lizenzierung / FlexLM

Die Siemens PLM Lizenzierung besteht aus fünf Hauptkomponenten:

1. License Manager Daemon (lmgrd)
2. Vendor Daemon (ugslmd)
3. License File (splm.lic)
4. Application Program
5. Server Setting

License Manager Daemon (lmgrd)

Der License Manager Daemon (lmgrd) stellt eine erste Verbindung mit dem Anwendungsprogramm her und übergibt die Verbindung an den Vendor Daemon. Der License Manager Daemon startet den Vendor Daemon oder startet ihn bei Bedarf neu.

Vendor Daemon (ugslmd)

Der Vendor Daemon (ugslmd) verfolgt, wie viele Lizenzen ausgecheckt sind und wer sie in Benutzung hat. Wenn ugslmd aus irgendeinem Grund beendet wird, verlieren alle Benutzer ihre Lizenzen. Benutzer bekommen normalerweise automatisch wieder ihre Lizenzen, wenn der License Manager Daemon ugslmd neu startet.

License File (splm.lic)

Die Lizenz-Datei ist eine Textdatei, die die Lizenzierungs-Daten beinhaltet. Die Lizenz-Datei muss für jede Maschine, die als Lizenzserver definiert ist, zugänglich sein. Die Lizenz-Datei enthält alle spezifischen Informationen zur Lizenz:

1. Server Name(s)
2. Host Identifier(s)
3. Vendor Daemon Name
4. PACKAGE Information (Optional)
5. INCREMENT / FEATURE Information

Application Program

Ein Softwarepaket, das Siemens PLM Licensing für seine Lizenz-Überwachung nutzt, wird üblicherweise vom Client ausgeführt. Das Anwendungsprogramm muss in der Lage sein, sich mit dem ugslmd Daemon zu verbinden, um Lizenzen zugewiesen zu bekommen.

Server Setting

Die Server-Einstellung muss auf jeder Siemens PLM Licensing basierten Anwendung gesetzt werden. Die Server Einstellung identifiziert den Lizenz-Server-Port (Port 28000 für NX und 27000 für Teamcenter Engineering) und den Hostnamen und wird zunächst durch das Installationsprogramm voreingestellt. [vgl. Siemens, 2012, S. 5-6]

Für die bei Rheinmetall eingesetzte Triple-Server Lösung müssen alle drei Servernamen inklusive Port eingetragen werden.

2.6.2 Lizenzanforderungs-Prozess

Die folgenden Schritte und die Abbildung 2.9 beschreiben, wie Applikationen mit dem ugslmd Daemon interagieren: [vgl. Siemens, 2012, S. 7]

- Das Anwendungsprogramm findet den Lizenzserver nach der Interpretation der Informationen in der Server-Einstellung
- Das Anwendungsprogramm stellt eine Verbindung mit lmgrd her, um den Port zu finden, auf dem sich der Vendor Daemon befindet
- Der lmgrd Daemon bestimmt, welchen Port ugslmd anspricht und sendet die Informationen zurück an den Client
- Der Client stellt eine Verbindung mit ugslmd her und stellt eine Lizenzanfrage
- Der ugslmd Daemon prüft in seinem Speicher, ob Lizenzen verfügbar sind und sendet entweder eine Gewährung oder eine Verweigerung zurück an den Client
- Der ugslmd Daemon zeichnet die Gewährung oder Verweigerung der Lizenz Anfrage in der Debug-Protokolldatei *ugslicensing.log* auf
- Das Lizenz-Modul in der Client-Anwendung gewährt oder verweigert die Verwendung der Funktion

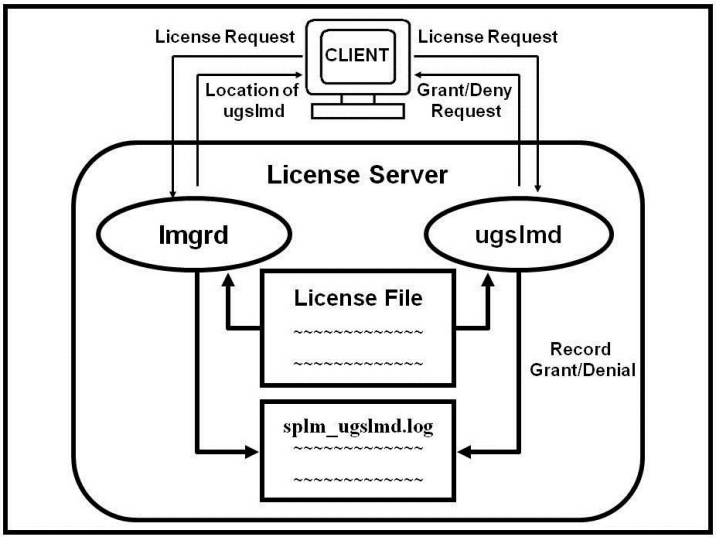

Abbildung 2.9: Der Lizenzanforderungs-Prozess [Grafik entnommen aus Siemens, 2012, S. 7]

3 Analyse

In diesem Kapitel folgt der Analyseteil, der sich in die beiden Unterkapitel Anforderungen und Anwendungsfälle aufteilt.

3.1 Anforderungen

Die Anforderungen wurden mit den Verantwortlichen der Technischen Datenverarbeitung der Firma Rheinmetall in einem Workshop erarbeitet und verabschiedet. Orientiert haben sich die Projektverantwortlichen an den Möglichkeiten und Funktionen am Markt verfügbarer Lizenzmanager bzw. Auswertungs-Tools sowie den eigenen Anforderungen, die in den kommerziellen Produkten nicht integriert sind, aber benötigt werden.

Eine Erhebung bei einer Konkurrenz-Firma wäre denkbar gewesen. Jedoch ist es schwierig, in der Branche, in welcher die Firma Rheinmetall tätig ist (Verteidigungs- und Sicherheitsindustrie), an Informationen von Konkurrenz-Firmen zu gelangen. Firmen in dieser Branche legen sehr großen Wert auf die Sicherheit ihrer Daten – gerade im Bereich der Konstruktion und der dafür benötigten Lizenzierung der CAD/CAM/CAE-Systeme. Damit scheidet diese Erhebungsmethode aus und es werden nur die Anforderungen dargestellt, die im Workshop verabschiedet wurden.

3.1.1 Funktionale Anforderungen

Nachstehend die funktionalen Anforderungen – gegliedert nach Komponenten:

Auswertungskomponente

A-1 Die Applikation soll die Lizenzen von Teamcenter und NX überwachen und auswerten, um Transparenz in der Lizenznutzung zu erlangen.

A-2 Die Applikation soll eine grafische Benutzeroberfläche (GUI) haben. Die GUI soll von den Mitarbeitern der Technischen Datenverarbeitung ohne weitere Einweisung benutzt werden können.

A-3 Es gibt monatlich einen Nachweis /Abrechnung über die genutzten Lizenzen, damit die Wartungskosten auf die entsprechenden Entwicklungsbereiche umgelegt werden können.

A-4 Es soll der Vergleich der Lizenznutzung zwischen Tag, Woche, Monat und Jahr möglich sein.

A-5 Analyse und Auswertung der Spitzenwerte bzw. maximal genutzter Lizenzen je Bundle.

A-6 Schwellenwerteinstellung mit Warnnachricht über das E-Mail-System. Beim Erreichen des Schwellenwerts wird einmalig eine Warnnachricht an den E-Mail-Verteiler der Technischen Datenvearbeitung gesendet.

A-7 Überwachung temporär vergebener bzw. ausgeliehener Lizenzen.

A-8 Überwachung / Monitoring der Lizenzserver (Dienste, Laufzeit, CPU, Speicher).

Eintragungskomponente

A-9 Die erzeugten Logdateien sollen in einer Datenbank archiviert werden, um bei späteren Nachfragen einen Nachweis zu haben.

Datenbankkomponente

A-10 Die zu archivierenden Daten sollen in Form von Binary Large Objects (BLOBs) abgelegt werden. Im SQL-Server 2008 wird die Spalte zur Datenspeicherung als *varbinary(max)* ausgewiesen. In einem BLOB können große Binärdateien, also auch die hier vorhanden Logdateien gespeichert werden und später wieder in dem Archiv durch eine Suchfunktion abgerufen werden. Das *max* bei *varbinary(max)* gibt an, dass die maximale Speichergröße $(2^{31}) - 1$ Byte beträgt.

3.1.2 Nicht-funktionale Anforderungen

Nachstehend die nicht-funktionalen Anforderungen:

A-11 Konfigurationsmöglichkeiten für weitere Lizenzen und Lizenzserver aus dem Teamcenter und NX Umfeld.

A-12 Portierbarkeit: Die Anwendung soll auch auf Unix lauffähig sein. Durch die Umsetzung der Applikation in Java SE (Version 7) ist die Anwendung plattformunabhängig und kann mit geringen Änderungen nach Unix portiert werden.

A-13 Installations- und Konfigurationsanleitung: Für die Migration der Datenbank auf einen anderen Server und für die Konfiguration weiterer Lizenzen und Lizenzserver soll es eine Anleitung geben.

3.2 Anwendungsfälle

In diesem Abschnitt wird ein Use-Case Diagramm vorgestellt, das fünf verschiedene Anwendungsfälle zeigt (siehe nachstehende Abbildung 3.1).

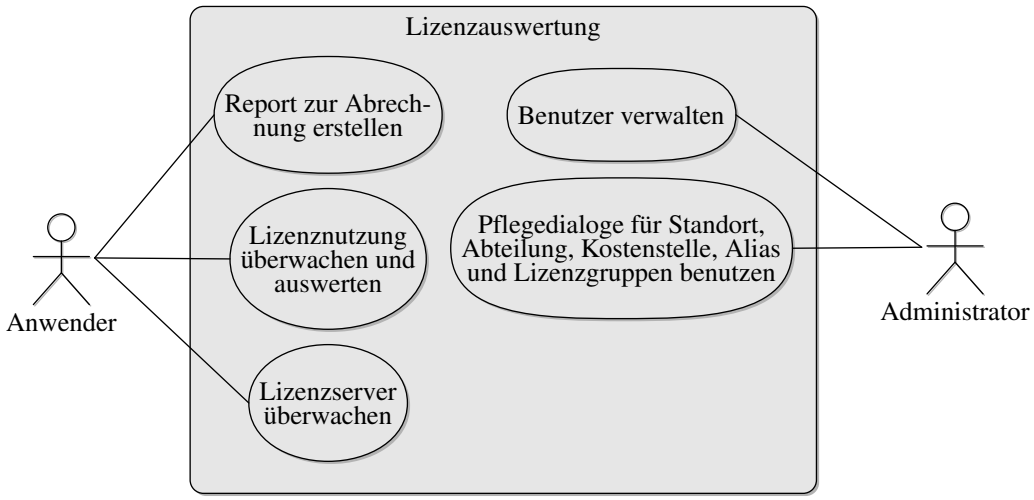

Abbildung 3.1: Anwendungsfalldiagramm für fünf Anwendungsfälle

Auf den nächsten Seiten werden die für die obigen fünf Anwendungsfälle umzusetzenden Inhalte in tabellarischer Form weiter definiert.

3.2.1 Report zur Abrechnung erstellen

Titel	Report zur Abrechnung erstellen
Akteur	Anwender
Ziel	Report ist abgespeichert
Auslöser	Anwender möchte Report generieren
Vorbedingung	Daten stehen zur Verfügung
Nachbedingungen	Report ist erstellt
Erfolgsszenario	
1. Der Anwender startet das Programm zur Lizenzauswertung. 2. Der Anwender wählt im linken Baum den Punkt Reporting und dann Report1, Report2, Report3 aus. 3. Der Anwender füllt das Formular mit den gewünschten Daten aus. 4. Der Anwender startet die Generierung des Reports. 5. Das System überprüft die Eingaben des Anwenders. 6. Das System generiert den Report und zeigt ihn im rechten Frame der Anwendung an. 7. Der Anwender speichert den Report ab.	
Erweiterungen	
Fehlerfälle	
5a. Datumsgrenze liegt in der Zukunft: Das System meldet einen Fehler und fordert zur Korrektur auf. 5b. Lizenzen oder Abteilungen wurden im Report nicht angegeben: Das System meldet einen Fehler und fordert zur Eingabe der Daten auf.	
Anforderungen	A-3

Tabelle 3.1: Anwendungsfall „Report zur Abrechnung erstellen“

3.2.2 Lizenznutzung überwachen und auswerten

Titel	Lizenznutzung überwachen und auswerten
Akteur	Anwender
Ziel	Lizenznutzung ist analysiert
Auslöser	Anwender möchte Lizenznutzung analysieren
Vorbedingung	Daten sind von der Eintragungskomponente in die Datenbank eingetragen
Nachbedingungen	Lizenznutzung ist analysiert
Erfolgsszenario	
1. Der Anwender startet das Programm zur Lizenzauswertung. 2. Der Anwender wählt im linken Baum den Punkt Analyse und dann Lizenznutzung aus. 3. Der Anwender füllt das Formular mit den gewünschten Daten aus. 4. Der Anwender startet die Generierung des Diagramms. 5. Das System überprüft die Eingaben des Anwenders. 6. Das System generiert das Diagramm und stellt es im rechten Frame der Anwendung dar.	
Erweiterungen	
Fehlerfälle	
5a. Datumsgrenze liegt in der Zukunft: Das System meldet einen Fehler und fordert zur Korrektur auf. 5b. Lizenzen wurden im Formular nicht angegeben: Das System meldet einen Fehler und fordert zur Eingabe der Daten auf.	
Anforderungen	A-4 und A-5

Tabelle 3.2: Anwendungsfall „Lizenznutzung überwachen und auswerten“

3.2.3 Lizenzserver überwachen

Titel	Lizenzserver überwachen
Akteur	Anwender
Ziel	Lizenzserver Status analysiert
Auslöser	Anwender möchte Lizenzserver überwachen
Vorbedingung	keine
Nachbedingungen	keine
Erfolgsszenario	
1. Der Anwender startet das Programm zur Lizenzauswertung. 2. Der Anwender wählt im linken Baum den Punkt Monitoring aus. 3. Der Anwender füllt das Formular mit den gewünschten Daten aus. 4. Der Anwender startet die Überwachung. 5. Das System überprüft die Eingaben des Anwenders. 6. Das System zeigt den Status des Lizenzservers an und stellt ihn im rechten Frame der Anwendung dar.	
Erweiterungen	
Fehlerfälle	
Anforderungen	A-8

Tabelle 3.3: Anwendungsfall „Lizenzserver überwachen“

3.2.4 Benutzer verwalten

Titel	Benutzer verwalten
Akteur	Administrator
Ziel	Benutzer angelegt oder bearbeitet
Auslöser	Änderungen an den Benutzerdaten
Vorbedingung	Erfolgreiches Login des Administrators
Nachbedingungen	keine
Erfolgsszenario	
1. Der Administrator startet das Programm zur Lizenzauswertung. 2. Der Administrator loggt sich in das Programm mit Administratorrechten ein. 3. Der Administrator wählt im linken Baum den Punkt Benutzerverwaltung aus. 4. Der Administrator ändert das Formular mit den gewünschten Daten. 5. Der Administrator speichert die Angaben. 6. Das System überprüft die Eingaben des Administrators. 7. Das System übernimmt die geänderten Daten in die Datenbank.	
Erweiterungen	
Fehlerfälle	
Anforderungen	

Tabelle 3.4: Anwendungsfall „Benutzer verwalten“

3.2.5 Pflegedialoge benutzen

Titel	Pflegedialoge benutzen
Akteur	Administrator
Ziel	Standort, Abteilung, Kostenstelle, Alias oder Lizenzgruppe bearbeitet
Auslöser	Änderungen bei Standort, Abteilung, Kostenstelle, Alias oder Lizenzgruppe
Vorbedingung	Erfolgreiches Login des Administrators
Nachbedingungen	keine
Erfolgsszenario	
1. Der Administrator startet das Programm zur Lizenzauswertung. 2. Der Administrator logt sich in das Programm mit Administratorrechten ein. 3. Der Administrator wählt im linken Baum den Punkt Pflegedialoge aus. 4. Der Administrator wählt den Unterpunkt Lizenznutzer oder Lizenz aus. 5. Der Administrator ändert den Eintrag im Formular mit den gewünschten Daten. 6. Der Administrator speichert die Angaben. 7. Das System überprüft die Eingaben des Administrators. 8. Das System übernimmt die geänderten Daten in die Datenbank.	
Erweiterungen	
Fehlerfälle	
Anforderungen	

Tabelle 3.5: Anwendungsfall „Pflegedialoge für Standort, Abteilung, Kostenstelle, Alias und Lizenzgruppen benutzen“

4 Spezifikation

In diesem Kapitel wird die Anwendung nach den unter Abschnitt 3.1 festgehaltenen Anforderungen spezifiziert.

4.1 Auswertungsdatei

Die Auswertungsdatei wird von der Eintragungskomponente alle 5 - 15 Minuten (nach gewünschter Genauigkeit) erzeugt. Die nachfolgend erklärten Werte werden in die Datenbank geschrieben. In dieser Datei stehen alle relevanten Angaben der momentanen Lizenznutzung. Die Auswertungsdatei wird in Java mittels regulären Ausdrücken geparst. Die Muster wurden so eindeutig gewählt, dass auch bei Fehlern bei der Erzeugung der Auswertungsdatei keine falschen Daten in die Datenbank geschrieben werden.

Die erstellten Auswertungsdateien sollen für eine spätere Überprüfung laut den in 3.1 definierten Anforderungen A-9 und A-10 archiviert werden.

In dem nachfolgenden Listing 4.1 ist eine Auswertungsdatei dargestellt. Diese Datei wird per Batchskript erstellt. Pro Lizenz wird der Befehl:

```
1 lmutil lmstat -f DESIGNER -c 28000@C71117SR0703
```

ausgeführt und erstellt einen Block in der Auswertungsdatei. Der Parameter `-f` mit dem Wert DESIGNER gibt an, dass nur zu diesem Feature Statistiken erhoben werden sollen. Der Parameter `-c` gibt an, wo die Lizenzdatei liegt; hier wird der Server mit vorangestelltem Port angegeben.

Ausgewertet und in die Datenbank eingetragen werden die im Listing 4.1 blau hervorgehobenen Werte. Aus den Zeilen 2 und 6 wird das Datum und die Uhrzeit der Abfrage des Lizenzstatus zusammengesetzt. In Zeile 9 stehen die abgefragte Lizenz, die maximal verfügbaren Lizenzen und die zum Abfragezeitpunkt benutzen Lizenzen. Die hier markierten Werte dienen dazu, die unter 3.1 definierte Anforderung A-4 abfragen zu können. In Zeile 14 steht der Benutzer, der

die Lizenz gerade in Benutzung hat, sowie sein Computername und der Startzeitpunkt des Programms, das die Lizenz nutzt.

```
1
2  Benutzte Lizenzen um  10:10:31  ,71
3  ------------------------------------------------------------------------
4  lmutil - Copyright (c) 1989-2005 Macrovision Europe Ltd. and/or Macrovision
5  Corporation. All Rights Reserved.
6  Flexible License Manager status on Mon  3/11/2013  10:10
7
8  [Detecting lmgrd processes...]
9  Users of DESIGNER:  (Total of 43 licenses issued;  Total of 36 licenses in use)
10
11   "DESIGNER" v27.0, vendor: ugslmd
12   floating license
13
14     ohlhof_j C71L17WS0109 C71L17WS01090.0 (v27.0) (C71L30SR0702/28000 17550), start Mon 3/11  10:04
15      .
16      . weitere 35 Lizenzeinträge
17      .
18 ------------------------------------------------------------------------
19 lmutil - Copyright (c) 1989-2005 Macrovision Europe Ltd. and/or Macrovision
20 Corporation. All Rights Reserved.
21 Flexible License Manager status on Mon  3/11/2013  10:10
22
23 [Detecting lmgrd processes...]
24 Users of P1:  (Total of  21  licenses issued;  Total of  16  licenses in use)
25
26   "P1" v27.0, vendor: ugslmd
27   floating license
28
29     ohlhof_j C71L17WS0109 C71L17WS01090.0 (v27.0) (C71L30SR0702/28000 17550), start Mon 3/11  10:04
30      .
31      . weitere 15 Lizenzeinträge
32      .
33 ------------------------------------------------------------------------
34      .
35      . weitere Lizenzpakete
36      .
```

Listing 4.1: Auszug aus der Auswertungsdatei

4.2 Fachliches Entity-Relationship Model

In dem nachfolgend abgebildeten Entity-Relationship Model (Abbildung 4.1) sind die Schlüssel jeweils unterstrichen dargestellt.

Die Entitätstypen *Abteilung* und *Standort* haben jeweils einen künstlichen Schlüssel und ein Attribut (*Abteilungsname* und *Standortname*).

Der Entitätstyp *Benutzer* hat einen künstlichen Schlüssel. Ein Benutzer ist genau einer Abteilung bzw. einem Standort zugeordnet, jedoch kann eine Abteilung bzw. ein Standort mehrere Benutzer beinhalten. Des Weiteren hat der Entitätstyp *Benutzer* drei Attribute (*Username*, *Kostenstelle* und *Computername*).

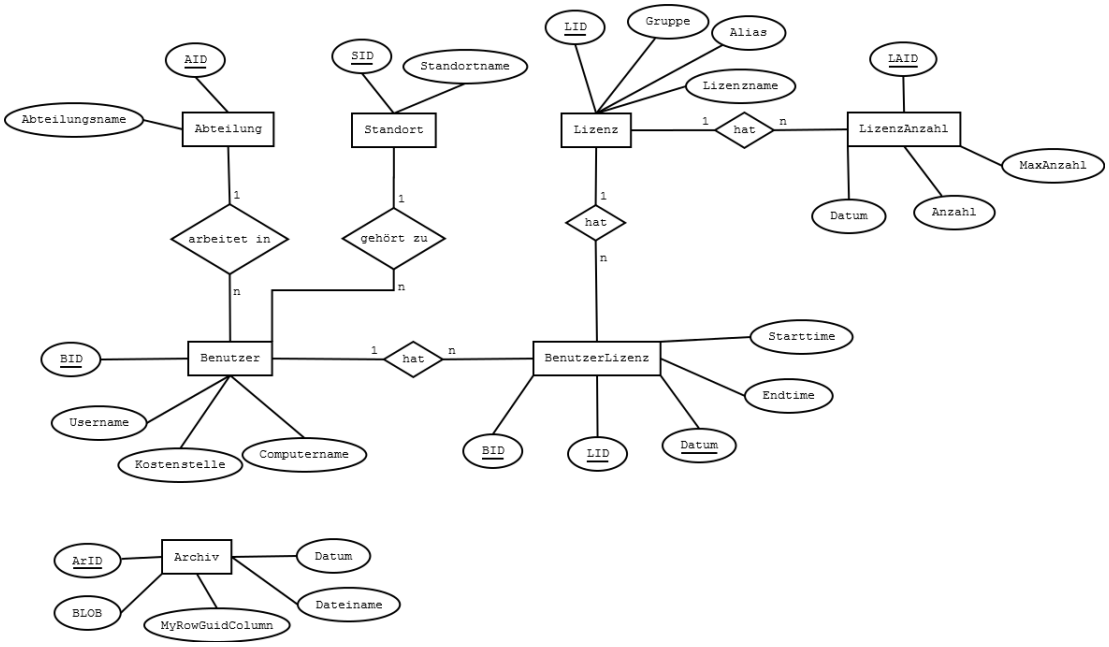

Abbildung 4.1: Entity-Relationship Diagramm

Der Entitätstyp *Benutzerlizenz* hat einen zusammengesetzten Schlüssel, bestehend aus *BID*, *LID* und *Datum*. Wobei die beiden erstgenannten Schlüssel auch Fremdschlüssel aus den Entitäten *Benutzer* und *Lizenz* sind. Das Attribut *Datum* ist das aktuelle Datum beim Eintragen eines Datensatzes in die Datenbank. Das Attribut *Starttime* wird aus der Auswertungsdatei ausgelesen. Das Attribut *Endtime* ist ein errechnetes Attribut. Es wird das *Datum* von der *Starttime* abgezogen, somit erhält man die *Endtime*. Diese Relation dient zur Abfrage und Erstellung von Reports. Das Attribut *Endtime* wird bei den aktuellen Reports nicht mehr genutzt, da es „ad-hoc“ in einer Abfrage aus den Attributen *Starttime* und *Datum* berechnet werden kann.

Der Entitätstyp *LizenzAnzahl* ist zur Speicherung der Daten, die die Auslastung der Lizenzen angibt. Die Relation besitzt einen künstlichen Schlüssel (*LAID*). In der Entität gibt es außerdem die Attribute *Anzahl*, *MaxAnzahl* und *Datum*. Diese zeigen an, zu welchem Zeitpunkt wie viele Lizenzen in Benutzung sind und wie viele maximal verfügbar sind.

Der Entitätstyp *Lizenz* speichert die unterschiedlichen Lizenzen mit den Attributen *Gruppe*, *Alias* und *Lizenzname* ab. Es gibt einen künstlichen Schlüssel (*LID*), der jede Lizenz identifiziert.

4.3 Dialoge

In diesem Abschnitt werden Mockups der grafischen Oberfläche vorgestellt. Mockups sind Nachbildungen des eigentlichen Objektes bzw. hier einiger Dialoge. Diese sollen einen Anhaltspunkt zum Erstellen der GUI bieten.

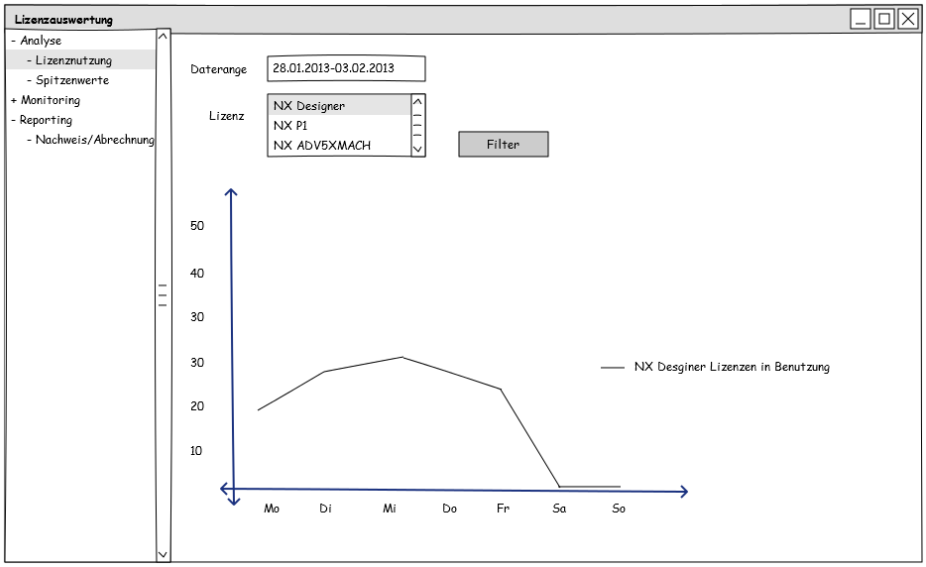

Abbildung 4.2: Mockup einer grafischen Oberfläche zur Lizenzauswertung (erste Version)

4.3.1 Allgemein

Auf der linken Seite des in Abbildung 4.2 dargestellten Mockups sieht man eine Baumstruktur, in der sich die verschiedenen Funktionen der Anwendung aufrufen lassen. Mit dieser Struktur lassen sich Funktionen des Programms gruppieren. Jeder Eintrag im Baum soll später auf einen eigenen Frame auf der rechten Seite verweisen. Durch erste Tests entstanden weitere Versionen der Mockups.

4.3.2 Lizenznutzung – erste Version

Im rechten Frame (Abbildung 4.2) ist die Lizenznutzung einer Lizenz dargestellt. Es kann ein Datum (von - bis) eingegeben werden und dazu eine Lizenz ausgewählt werden. Nach Bestätigung des Filter-Buttons wird das entsprechende Diagramm berechnet und anschließend dargestellt. Entsprechend des eingegebenen Zeitraums wird die X-Achse skaliert.

4.3.3 Lizenznutzung – zweite Version

Die erste Version des Mockups wurde in einem Workshop mit den Projektleitern analysiert und in der zweiten Version entsprechend den neuen Vorgaben /Anregungen angepasst.

In dieser zweiten Version (Abbildung 4.3) wurde die Auswahl des Datums in zwei Textfelder aufgeteilt, die mittels eines „Datepickers“ ausgewählt werden können. Ein Datepicker bietet die Möglichkeit, das Datum mithilfe eines Kalenders auszuwählen. Dies senkt die Fehlerquote, da keine händisch eingegebenen Datumseingaben mehr ausgewertet werden müssen.

Des Weiteren wurde eine Auswahlliste hinzugefügt, in der die Lizenzgruppe ausgewählt werden kann. Somit findet eine Vorfilterung der Lizenzen statt. Dieses Feld zieht auch Änderungen an der Datenbank und dem Anwendungskern nach sich.

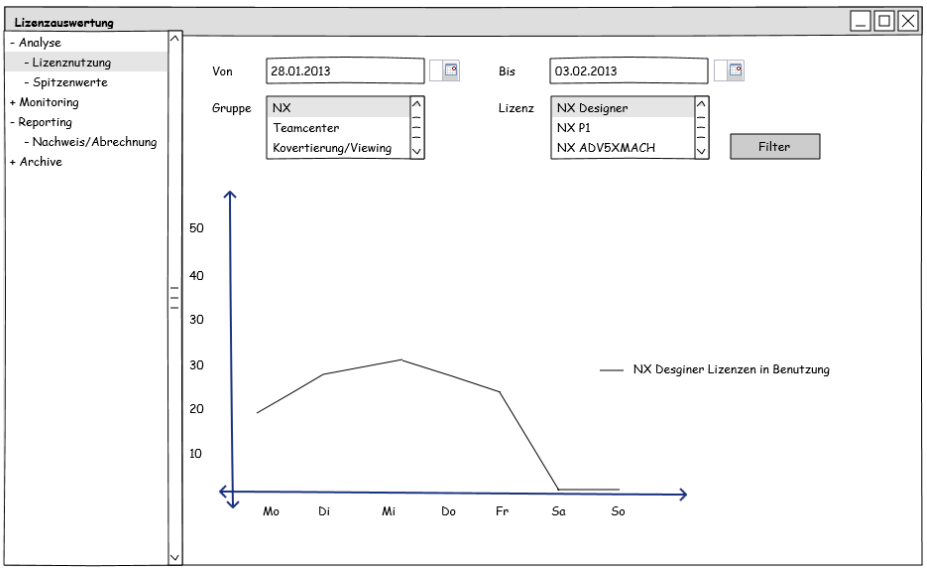

Abbildung 4.3: Mockup einer grafischen Oberfläche zur Lizenzauswertung (zweite Version)

4.3.4 Report Standort

Nachfolgend in der Abbildung 4.4 ist das Report-Mockup abgebildet. Das Datum kann wieder per „Datepicker“ gewählt werden. Die danach folgenden Auswahllisten lassen eine Mehrfachauswahl zu. So kann der Report flexibel nach Lizenzen und Standort zusammengestellt werden. Der eigentliche Report wird in tabellarischer Form dargestellt und kann gedruckt oder als PDF gespeichert werden.

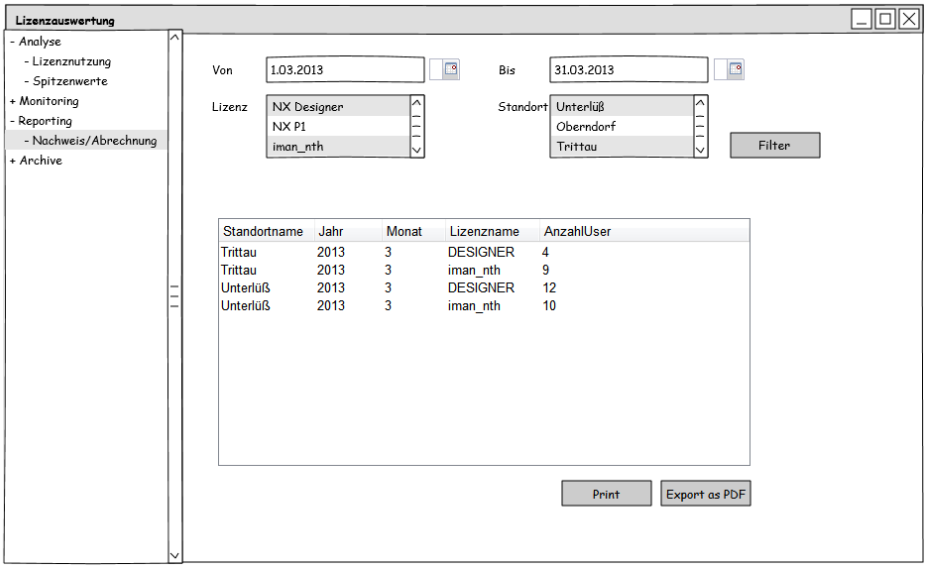

Abbildung 4.4: Mockup einer Oberfläche zum Reporting

5 Konzeption und Entwurf

In diesem Kapitel wird die Konzeption und der Entwurf der Anwendung dargestellt. Es werden die Architektur und die Komponentenstruktur dargestellt.

5.1 Schichten-Architektur

In diesem Kapitel wird die Drei-Schichten-Architektur der Anwendung näher beschrieben.

Es wurde eine Drei-Schichten-Architektur ausgewählt, weil diese Aufteilung der Schichten auf viele Informationssysteme passt. Die Architektur wird in der Praxis eingesetzt und hat sich dort bewährt [vgl. Siedersleben, 2005]. Da die hier zu entwickelnde Anwendung eine GUI hat und eine Datenbank für die Persistenz benutzt, bietet sich die Aufteilung in drei Schichten an (siehe Abbildung 5.1). Wenn die Anwendung z. B. auf ein Tablet portiert werden soll, kann die GUI-Schicht ausgetauscht werden. Der Anwendungskern mit der Applikationslogik und die Persistenzschicht bleiben davon unberührt. Des Weiteren muss bei einem Wechsel der Datenbank nur die Schicht des PersistenceManagers angepasst werden. Wenn der Dialekt gleich bleibt, muss lediglich der Connector zur physischen Datenbank angepasst werden. Der Anwendungskern kann flexibel erweitert werden, jedoch ziehen neue Funktion auch Änderungen in den anderen Schichten mit sich.

Die bei dieser Anwendung eingesetzte Drei-Schichten-Architektur bietet folgenden Vorteile: [nach Starke, 2008]

- Schichten sind voneinander unabhängig, sowohl bei der Erstellung als auch im Betrieb vom System
- Die Implementierung einer Schicht kann ausgetauscht werden, sofern die neue Implementierung die gleichen Dienste anbietet
- Schichtenbildung minimiert Abhängigkeiten zwischen Komponenten
- Schichtenbildung ist ein leicht verständliches Strukturkonzept

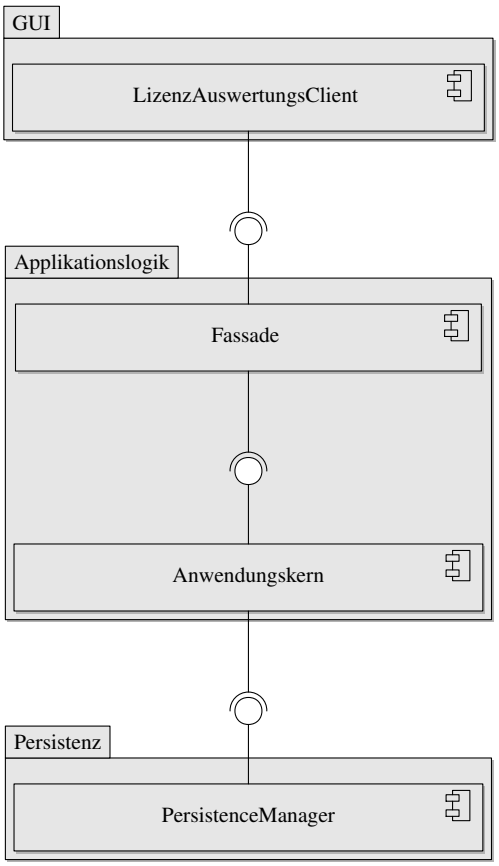

Abbildung 5.1: Drei-Schichten-Architektur

Jedoch werden manche Änderungen eines Systems von Schichten schlecht unterstützt. Ein neues Datenfeld, das sowohl gespeichert, als auch in der GUI angezeigt werden soll, zieht Änderungen in allen Schichten nach sich.

Schichtenbildung kann die Performance eines Systems beeinträchtigen, weil Anfragen unter Umständen durch mehrere Schichten durchgereicht werden, bis sie anschließend bearbeitet werden. [vgl. Starke, 2008]

Die hier entwickelte Anwendung ist jedoch nicht zeitkritisch, so dass die Vorteile einer Schichtung die Nachteile überwiegen.

5.2 Komponenten

Nachfolgend ist hier die Komponentenstruktur (Abbildung 5.2) abgebildet. In gelb ist die Benutzeroberfläche markiert. In hellblau ist die Komponente Anwendungskern mit der dazugehörigen Fassade dargestellt. In dunkelblau sind die innen liegenden Komponenten des Anwendungskerns festgehalten. In der Farbe rosa ist die Komponente des PersistenceManagers aufgeführt. In grün ist die Eintragungskomponente, die Daten in die Datenbank einträgt, und in rot die Alarmkomponente, die für die E-Mail Benachrichtigung bei der Überschreitung eines Schwellenwerts zuständig ist, dargestellt.

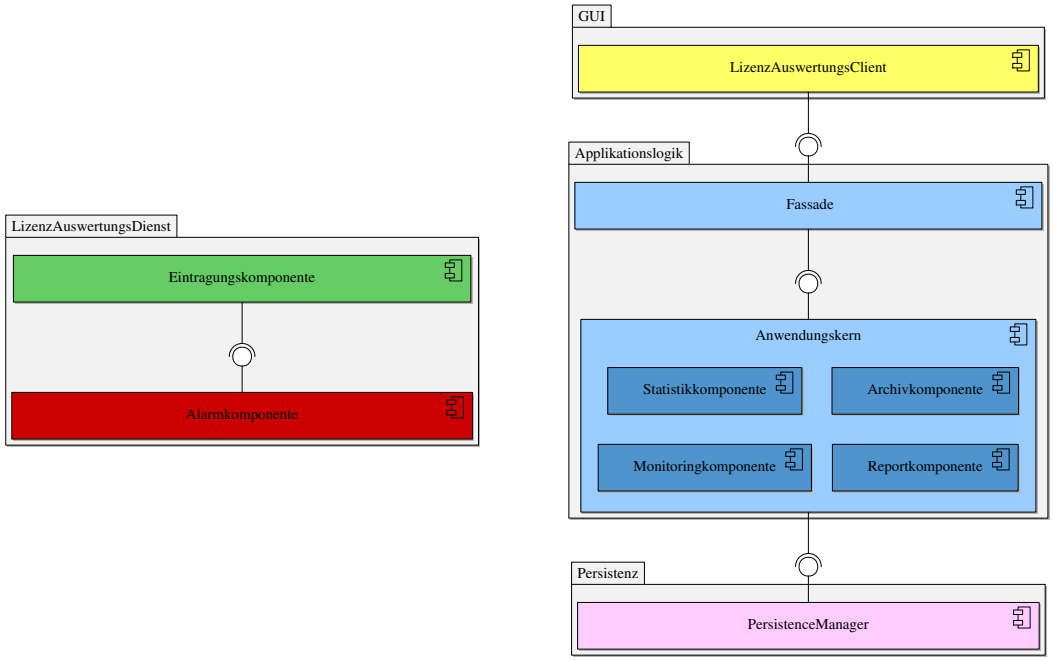

Abbildung 5.2: Komponentenstruktur

Die Alarmkomponente ist im *LizenzAuswertungsDienst* integriert, da beim Eintragen der Lizenzdaten in die Datenbank geprüft wird, ob eine Lizenz die 90 % Marke (Auslastung) überschritten hat und dann eine Nachricht per JavaMail (Bibliothek zum Versenden von Mails unter Java) versendet wird.

Die Statistikkomponente bereitet die Daten auf, die in der GUI als Diagramm angezeigt werden sollen. In der GUI werden diese Daten dann an *JFreeChart* übergeben und ein Diagramm erstellt und angezeigt. *JFreechart* ist eine Bibliothek zum Erstellen von Diagrammen in Java.

Die Archivkomponente kapselt die Fachlichkeit, die benötigt wird, um aus einem Eintrag in der Archiv-Tabelle eine Datei zu erzeugen. Die Datei wird unter *C:\Temp* zwischengespeichert. Die GUI öffnet diese Datei dann in einem Editor.

Die Reportkomponente bereitet die Daten für die verschiedenen Reports auf. In der GUI werden diese Daten dann an *Dynamicreports* (Bibliothek zum Erstellen von Reports unter Java) übergeben und der Report wird angezeigt.

Die Monitoringkomponente kapselt die Fachlichkeit, die benötigt wird, um Daten über die Server anzuzeigen. Es werden Daten über die CPU-Auslastung, Speicher-Auslastung, Festplattenplatz und weitere Werte angezeigt.

Die Fassade ist in der Applikationsschicht dargestellt. Dies folgt dem Entwurfsmuster „Fassade“, bei welchem es für alle Schnittstellen eines Subsystems eine einzige einheitliche Schnittstelle gibt. Dies löst Abhängigkeiten auf und vereinfacht in diesem System die Verteilung des Systems in Schichten.

Die Persistenz-Schicht dient der dauerhaften Speicherung der Daten des Systems und abstrahiert technische Details der physischen Datenbank. Es wurde sich gegen den Einsatz eines Objekt-Relationalen Mappers (ORM) als Persistenz-Framework entschieden, weil ein Problem die größere Einarbeitungszeit wäre und zum anderen bei den Reports große Datenmengen abgefragt werden. Die Performance bei so einem Objekt-Relationalen Mapper ist oft schlechter als bei von Hand geschriebenen Abfragen über JDBC und SQL. [vgl. Scholten, 2011]

5.3 Technische Architektur

In der technischen Architektur (Abbildung 5.3) sieht man auf der linken Seite den *LizenzAuswertungsDienst*. Dieser wird in ein JAR-File kompiliert und auf einem separaten Server alle 10 Minuten zur Ausführung gebracht. Das Intervall ist frei konfigurierbar. Diese JAR-Datei ist ein ausführbares Java-Archiv. Auf eine Schichtung im *LizenzAuswertungsDienst* wurde verzichtet, da diese Anwendung schlank gehalten ist. Es werden die entsprechenden Daten in die Datenbank eingetragen. Es besteht ein direkter Zugriff von der Eintragungskomponente auf die Datenbank. Die Verbindung dieser Komponente zur Datenbank wird über JDBC hergestellt. Die Alarmkomponente kapselt die Funktionen zur E-Mail Benachrichtigung und stellt sie der Eintragungskomponente bereit.

Der Microsoft SQL Server läuft auf einem separaten Server und ist von beiden Anwendungen über JDBC erreichbar.

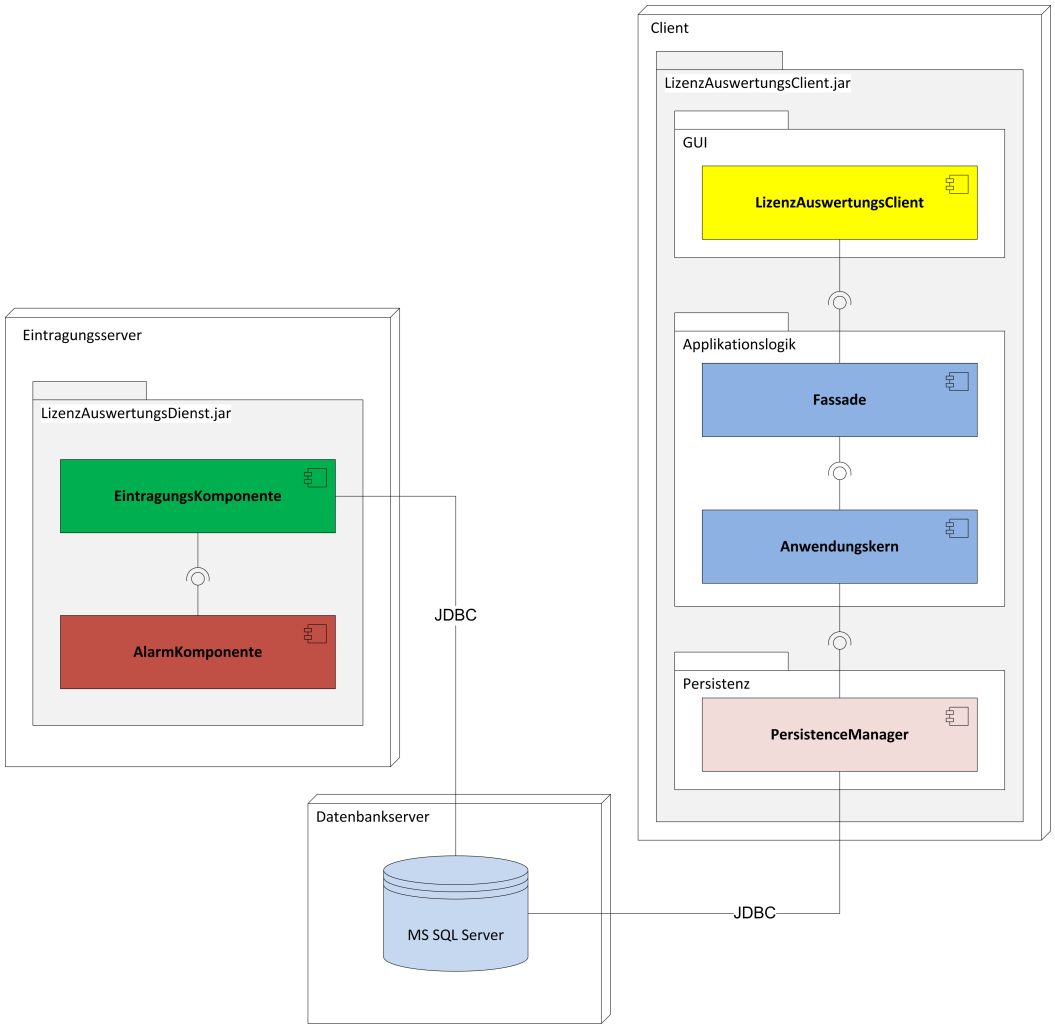

Abbildung 5.3: Technische Architektur

Auf der rechten Seite ist der *LizenzAuswertungsClient* dargestellt. Dieser Client ist in einer Drei-Schichten-Architektur aufgebaut. Er besteht aus der GUI, der Applikationslogik und der Persistenzschicht.

Die Anwendung kann auf beliebig vielen Clients zur Ausführung gebracht werden. Es wird auf dem Client ein JRE in der Version 7 benötigt. Dieses gehört im Regelfall zur Basisausstattung einer aktuellen Workstation.

5.4 Technische Infrastruktur

Die nachfolgende Tabelle 5.1 beinhaltet die Technische Infrastruktur des Systems. Dazu werden alle verwendeten Frameworks, Bibliotheken und andere Softwarekomponenten aufgeführt und deren Einsatzbereich beschrieben.

Software	Einsatzbereich
Java SE (Version 7)	Objektorientierte Programmiersprache für die Entwicklung der Anwendung
JFreeChart (Version 1.0.14)	Bibliothek zur Darstellung von Diagrammen in Java
DynamicReports (Version 3.1.0)	Bibliothek zur Erzeugung von Reports in Java
JUnit (Version 4)	Framework zum Testen des Systems
Microsoft JDBC (Version 4)	Bibliothek um den Zugriff auf die Datenbank zu ermöglichen
Microsoft SQL Server 2008 R2 Express	Datenbank zur Speicherung der Lizenzdaten
JavaMail (Version 1.4.7)	Bibliothek um mit Java E-Mails zu versenden. Zur Alarmierung bei Erreichen eines Schwellenwerts.
Mockito (Version 1.9.5)	Bibliothek zum Erstellen von Mock-Objekten in den Tests

Tabelle 5.1: Verwendete Software von Drittanbietern

6 Realisierung und Test

In diesem Kapitel wird schrittweise die Realisierung und der Test der Anwendung beschrieben. Begonnen wurde mit dem Entwurf und der Erstellung der Datenbank. Anschließend wurde die Eintragungskomponente realisiert und danach der LizenzAuswertungsClient mit GUI, Anwendungskern und PersistenceManager.

6.1 Normalisierung der Datenbank

Hier folgt die Normalisierung. Ziel ist es, die Datenbank in die dritte Normalform zu bringen bzw. dies sicherzustellen.

6.1.1 Erste Normalform (1NF)

Die erste Normalform besagt, dass mehrwertige und zusammengesetzte Attribute und ihre Kombinationen nicht vorkommen dürfen. Die Domäne eines Attributs darf nur atomare (einfache, unteilbare) Werte beinhalten und dass der Wert eines Attributs in einem Tupel ein Einzelwert aus einer Domäne dieses Attributs sein darf. Anders gesagt verhindert die 1NF, dass Relationen innerhalb von Relationen oder Relationen als Attribute von Tupeln vorkommen. [vgl. Elmasri und Navathe, 2005]

Die Attribute aus dem Entity-Relationship-Modell (Abbildung 4.1) wurden atomar gewählt, so dass die Datenbank sich in erster Normalform befindet.

6.1.2 Zweite Normalform (2NF)

Ein Relationsschema R ist in 2NF, wenn jedes Nicht-Prime-Attribut A in R funktional voll vom Primärschlüssel in R abhängig ist und die 1NF erfüllt ist. Alle Nichtschlüsselattribute müssen also von allen Schlüsselattributen vollständig abhängen. [vgl. Elmasri und Navathe, 2005]

Die Tabelle *BenutzerLizenz* besitzt einen zusammengesetzten Primärschlüssel aus *PK_ID_User*, *PK_ID_Lic* und *PK_Date*. Das Attribut *starttime* ist vollständig abhängig von dem gesamten Primärschlüssel. Eindeutig wird der Schlüssel erst mit der Aufnahme des Datums. Ebenso ist die *endtime* (ein errechnetes Attribut) vollständig abhängig von dem zusammengesetzten Primärschlüssel. Die zweite Normalform ist somit für diese Tabelle erfüllt. Bei Tabellen, die nur einen Primärschlüssel und ein weiteres Attribut beinhalten, ist die zweite Normalform automatisch erfüllt, solang auch die erste Normalform erfüllt ist. Dies trifft bei dieser Datenbank auf die Tabellen *Standort* und *Abteilung* zu.

Durch die 2NF werden Redundanz und die damit einhergehende Gefahr von Inkonsistenzen reduziert. Nur noch logisch /sachlich zusammengehörige Informationen finden sich in einer Relation.

PK_ID_User	PK_ID_Lic	PK_Date	endtime	starttime
5	1	2013-04-24 10:40:01.8500000	0d2h9m1s	2013-04-24 08:31:00.0000...
5	1	2013-04-24 10:50:01.1190000	0d2h19m1s	2013-04-24 08:31:00.0000...
5	1	2013-04-24 11:00:01.5570000	0d2h29m1s	2013-04-24 08:31:00.0000...
5	19	2013-04-22 13:30:01.9530000	0d0h0m1s	2013-04-22 13:30:00.0000...
5	19	2013-04-22 13:40:01.2920000	0d0h10m1s	2013-04-22 13:30:00.0000...
5	19	2013-04-22 13:50:01.1830000	0d0h20m1s	2013-04-22 13:30:00.0000...

Abbildung 6.1: Auszug aus der Tabelle BenutzerLizenz

6.1.3 Dritte Normalform (3NF)

Die dritte Normalform besagt, dass eine Relation kein Nichtschlüsselattribut enthalten darf, das funktional von einem anderen Nichtschlüsselattribut (oder von einer Menge von Nichtschlüsselattributen) bestimmt wird. Das heißt, es sollte keine transitive Abhängigkeit eines Nichtschlüsselattributs vom Primärschlüssel bestehen. Außerdem muss die Relation die zweite Normalform erfüllen. [vgl. Elmasri und Navathe, 2005]

In der Tabelle *BenutzerLizenz* (Abbildung 6.1) hängen beide Nichtschlüsselattribute voll vom zusammengesetzten Primärschlüssel ab. Keines der beiden Nichtschlüsselattribute hängt von dem jeweils anderen transitiv ab, die 3NF ist somit gegeben.

Verbliebene thematische Durchmischungen in der Relation werden behoben, nach der 3NF sind die Relationen des Schemas zuverlässig monothematisch.

6.2 Datenbank-Erstellung

Im nachfolgenden Listing 6.1 wird die Datenbank mittels DDL (Data Definition Language) erstellt. Grundlage für die Anfrage war das vorher unter der Abbildung 4.1 konzipierte Entity-Relationship-Modell. Die Anfrage ist im SQL-Server Dialekt geschrieben.

```
1  CREATE TABLE Abteilung (
2      PK_ID_Abteilung int IDENTITY(1,1) PRIMARY KEY,
3      Abteilungsname nvcarchar(75)
4  )
5
6  CREATE TABLE Lizenz (
7      PK_ID_Lic int IDENTITY(1,1) PRIMARY KEY,
8      Lizenzname nvarchar(75),
9      Alias nvarchar(75),
10     Gruppe nvarchar(75)
11 )
12
13 CREATE TABLE Standort (
14     PK_ID_Standort int IDENTITY(1,1) PRIMARY KEY,
15     Standortname nvarchar(75)
16 )
17
18 CREATE TABLE Benutzer (
19     PK_ID_User int IDENTITY(1,1) PRIMARY KEY,
20     Username nvarchar(75),
21     Computername nvarchar(75),
22     Kostenstelle nvarchar(30),
23     FK_ID_Abteilung int REFERENCES Abteilung(PK_ID_Abteilung),
24     FK_ID_Standort int REFRENCES Standort(PK_ID_Standort)
25 )
26
27 CREATE TABLE BenutzerLizenz (
28     PK_ID_User int REFRENCES Benutzer(PK_ID_user),
29     PK_ID_Lic int REFERECNES Lizenz(PK_ID_lic),
30     PK_Date datetime2,
31     PRIMARY KEY (PK_ID_user, PK_ID_lic, PK_Date),
32     Starttime datetime2,
33     Endtime nvarchar(75)
34 )
35
36 CREATE TABLE LizenzAnzahl (
37     PK_ID_Liccount int IDENTITY(1,1) PRIMARY KEY,
38     FK_ID_Lic int REFRENCES Lizenz(PK_ID_Lic),
39     Datum datetime2,
40     Anzahl int,
```

```
41      MaxAnzahl int
42  )
```

Listing 6.1: Data Definition Language zur Datenbankerstellung

6.3 Eintragungskomponente

Die Eintragungskomponente wird per „Geplantem Task“ alle 5 - 15 Minuten (je nach gewünschter Genauigkeit) ausgeführt und trägt Benutzer, Lizenzen und Lizenzdaten in die Datenbank ein. Des Weiteren erstellt diese Anwendung die Logdateien, die in das Archiv geschrieben werden. Die Anwendung wurde ebenfalls mit Java SE realisiert und wird separat gestartet. Um das Programm als Task ausführen zu können, wurde ein JAR-Archiv erstellt. Der Server, der diesen Job später ausführt, benötigt mindestens ein JRE in der Version 7.

6.4 LizenzAuswertungsClient

In der nachfolgenden Abbildung 6.2 wird die Lizenz *iman_nth* ausgewertet. Es wurde in JFreeChart konfiguriert, dass ein Fadenkreuz benutzt werden kann. Mit diesem kann auf der Diagrammlinie ein genauer Wert angewählt werden. Somit ist es einfacher, diesen abzulesen. Zusätzlich wird kontextsensitiv auf den gezeigten Wert ein Infofenster angezeigt.

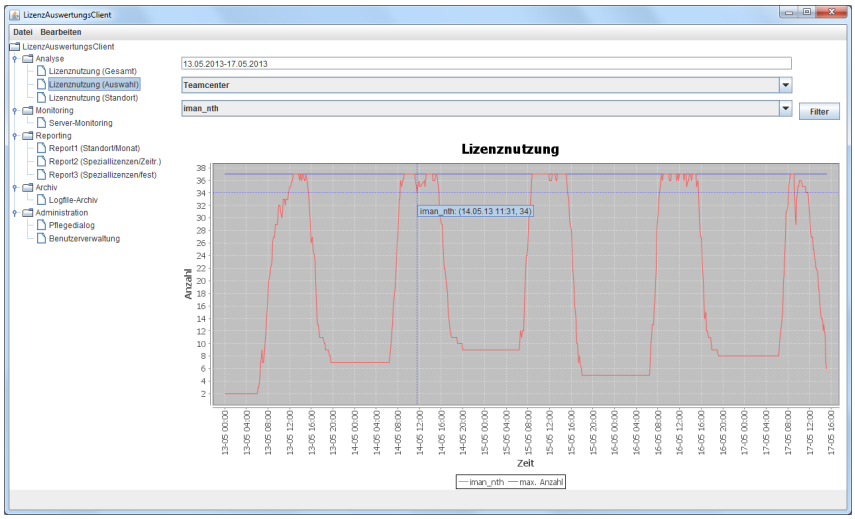

Abbildung 6.2: Lizenzauslastung der Lizenz „iman_nth“

In der nachfolgenden Abbildung 6.3 ist ein Balkendiagramm zu sehen, das die Lizenznutzung pro Standort und Monat zeigt. Es werden hier alle an dem jeweiligen Tag gezogenen Lizenzen berücksichtigt.

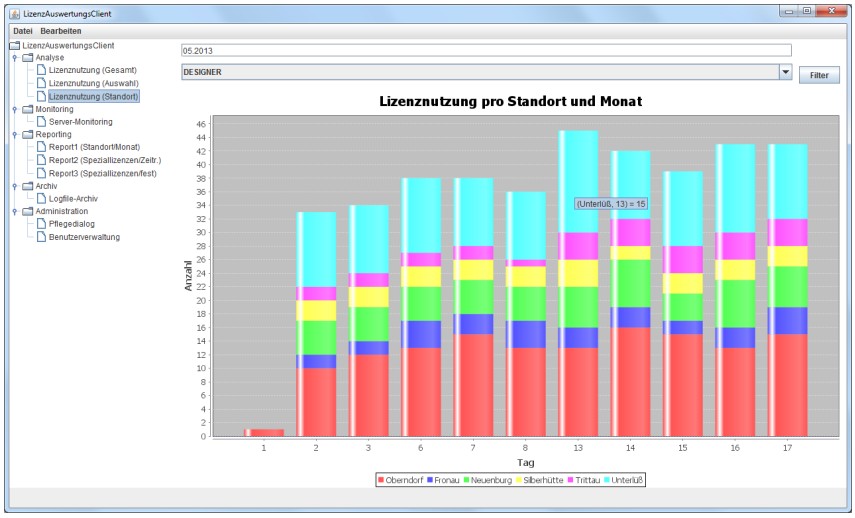

Abbildung 6.3: Lizenznutzung pro Standort und Monat

In Abbildung 6.4 ist ein Monatsreport zu sehen, der die Anzahl der Benutzer, sowie die Benutzungsdauer pro Lizenz beinhaltet. Über die Auswahllisten können mehrere Lizenzen und Standorte selektiert werden.

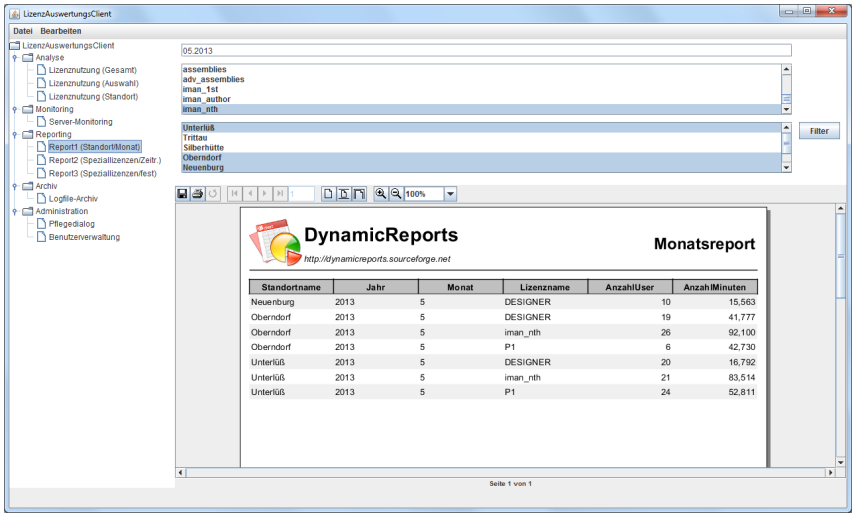

Abbildung 6.4: Monatsreport nach Auswahlkriterien

In der Abbildung 6.5 ist die Archivfunktionalität dargestellt. Es kann im Textfeld ein Zeitraum angegeben werden. Eine ausgewählte Datei kann dann angezeigt werden, dazu wird Notepad benutzt.

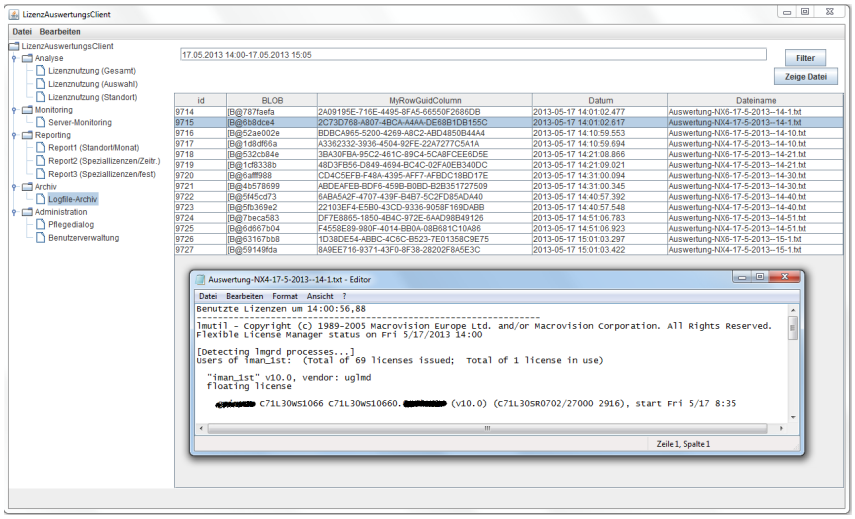

Abbildung 6.5: Archivfunktionalität

6.5 Reports

Bei den Reports wurde auf ein externes Reporting-Tool verzichtet. Die Abfragen, welche die Daten bereitstellen, sind von Hand konstruiert und können so genau auf die Anforderungen und Wünsche des Unternehmens angepasst werden. Der Firma Rheinmetall war es wichtig, alles in einer Applikation vereint zu haben und nicht ein extra Tool für das Reporting benutzen zu müssen.

Vorteil einer externen Anwendung wäre die höhere Flexibilität, da sich der Anwender basierend auf der vorhanden Datenbasis eigene Reports generieren lassen kann. Ein neuer Report in der jetzigen Anwendung müsste geschrieben und integriert werden. Die Vorgehensweise bei der Abrechnung ist im Vertrag festgeschrieben und wird nicht geändert.

Report Lizenzen nach Standort und Minuten

Die nachfolgende Abfrage im Listing 6.2 dient als Datenbasis für den Report. Es handelt sich um drei verschachtelte Abfragen. Das besondere an dieser Abfrage ist die *RANK()*-Funktion. In die Datenbank werden die *starttime* und das Datum (*pk_date*) eingetragen. Um die Nutzungsdauer

der Benutzergruppe herauszufinden, muss von diesen beiden Werten die Differenz gebildet werden. Jedoch muss immer nur die höchste Differenz aus Startzeit und Eintragungsdatum für die Nutzungsdauer akkumuliert werden. Mit der *RANK()*-Funktion kann genau diese Methodik erreicht werden. Es wird eine Partition über die *starttime* gebildet und nach Eintragungsdatum (*pk_date*) aufsteigend sortiert. Für die Berechnung herangezogen werden nur die Rank=1 Zeilen.

Nachfolgend ist die Syntax der *RANK()*-Funktion dargestellt, diese basiert auf *TRANSACT-SQL*.

```
RANK ( ) OVER ( [ partition_by_clause ] order_by_clause )
```

In der *WHERE*-Klausel kann dann der entsprechende Rank ausgewählt werden.

```
SELECT
 Standortname,
 Jahr,
 Monat,
 Lizenzname,
 COUNT(DISTINCT username) AS AnzahlUser,
 SUM(DISTINCT RuntimeMinute) AS AnzahlMinuten
 FROM(
    SELECT * FROM
    (
     SELECT DISTINCT Standortname,
            DATEPART(YEAR,PK_Date) AS Jahr,
            DATEPART(month,PK_Date) AS Monat,
            Lizenzname,
            COUNT(DISTINCT username) AS AnzUser,
            SUM(DISTINCT DATEDIFF(MINUTE,starttime ,pk_date))
                  AS RuntimeMinute,
            endtime,
            starttime,
            pk_date,
            username,
            RANK() Over (PARTITION BY starttime ORDER BY
                  pk_date DESC) AS Rank

            FROM BenutzerLizenz,Benutzer,Abteilung,Lizenz,Standort
            WHERE
            BenutzerLizenz.PK_ID_user=Benutzer.PK_ID_user
            AND BenutzerLizenz.PK_ID_lic=Lizenz.PK_ID_lic
            AND PK_ID_standort=FK_ID_standort
            AND DATEPART(MONTH,PK_Date) = '04'
            AND DATEPART(YEAR,PK_Date) = '2013'
            AND Lizenzname IN ('DESIGNER','iman_nth')
            AND Standortname = 'Unterlüß'

            GROUP BY
            Standortname,
```

```
37                  DATEPART(YEAR, PK_Date),
38                  DATEPART(MONTH, PK_Date),
39                  Lizenzname,
40                  starttime,
41                  endtime,
42                  pk_date,
43                  username
44                  )tmp WHERE Rank=1
45      )tmp2 GROUP BY Standortname, Jahr, Monat, Lizenzname
```

Listing 6.2: Query für den Report

6.6 Logfile Archiv

Um Dateien in der Datenbank abzulegen, muss ein *FILESTREAM* erstellt und für die Datenbank aktiviert werden. Es wird dazu ein Verzeichnis angelegt, unter dem die Daten abgelegt werden. Die Speicherung der Dateien findet in der Spalte *BLOB* statt.

Das Konzept von FILESTREAM integriert das SQL-Server Datenbankmodul in ein NTFS-Dateisystem: BLOB-Daten vom Typ varbinary(max) werden im Dateisystem gespeichert. Dadurch greift nicht die für varbinary(max) definierte Größenbeschränkung auf 2 GB, sondern die Größe der Dateien ist lediglich durch den verfügbaren Speicherplatz auf dem Laufwerk begrenzt. Ansonsten können FILESTREAM-Daten ganz normal mit Transact-SQL-Anweisungen abgefragt, eingefügt, geändert oder gelöscht werden – das inklusive Transaktionen und Datenbanksicherheit. Des Weiteren können alle Verwaltungsfunktionen genutzt werden, z. B. zum Sichern und Wiederherstellen. [vgl. Riemenschneider, 2008]

```
1  ALTER DATABASE LizenzAuswertung
2  ADD FILEGROUP FileStreamGroup1
3    CONTAINS FILESTREAM
4  GO
5
6  ALTER DATABASE LizenzAuswertung
7  ADD FILE
8    (NAME = 'FS_Group'
9     , FILENAME = 'c:\Data\ArchiveFS'
10    )
11 TO FILEGROUP FileStreamGroup1
12 GO
13
14
15
16 CREATE TABLE Archive
17 (
```

```
18     id int IDENTITY(1,1) PRIMARY KEY,
19     BLOB varbinary(max) FILESTREAM NULL,
20     MyRowGuidColumn uniqueidentifier ROWGUIDCOL
21             NOT NULL UNIQUE DEFAULT NEWID(),
22     Datum datetime2,
23     Dateiname nvarchar(75)
24 )
25 GO
```

Listing 6.3: Abfrage für die Erstellung der Archiv-Datenbank und -Tabelle

6.7 Alarmfunktion

In dem nachfolgenden Listing 6.4 ist ein Ausschnitt aus der Alarmfunktion zu sehen. In der Variablen *keylic* befindet sich die aktuelle Lizenz-ID beim Eintragen in die Datenbank. Für die drei am häufigsten benutzten Lizenzen DESIGNER, P1 und iman_nth (NX und Teamcenter) wird die Alarmfunktion angewendet. Es wird bei Überschreitung der Auslastung von 90 % eine E-Mail an den verantwortlichen Mitarbeiter der Technischen Datenverarbeitung geschickt. Wenn die Lizenzauslastung ansteigt werden weitere E-Mails versendet, sinkt die Auslastung hingegen kommt es nicht zu einer E-Mail Benachrichtigung. Mit der SQL-Afrage *querySelLicLast* wird der vorletzte Stand der Auslastung abgerufen und in der Variablen *lastvalue* abgelegt. Die Variablen *max* und *belegt* werden mit den aktuellen Werten bei Eintragung belegt. In der nächsten SQL-Abfrage wird der Lizenzschlüssel in den Lizenznamen umgesetzt. Dieser wird bei der Benachrichtigung benötigt. Bei der Variablen *prozent* wird auf 90 % vom *max*-Wert abgerundet. Anschließend wird die Methode *sendMail* der Alarmkomponente aufgerufen und die E-Mail mit den Parametern Lizenzename, Maximum und Belegung versandt.

```
1  [...]
2  //Lizenzen: 1=DESIGNER, 2=P1, 23=iman_nth
3  if (keylic == 1 || keylic == 2 || keylic == 23) {
4    //Wert vor dem letzten Eintrag
5    int lastvalue=0;
6    Statement statementLicLast = con.createStatement();
7    String querySelLicLast = "SELECT * FROM ( SELECT *, ROW_NUMBER()
8                              OVER (ORDER BY Datum DESC) AS ROW FROM
9                              LizenzAnzahl WHERE FK_ID_Lic =
10                             '"+String.valueOf(keylic)+"' ) a WHERE
11                             a.ROW > 1 AND a.ROW <= 2";
12   ResultSet rsLast = statementLicLast.executeQuery(querySelLicLast);
13
14   if(rsLast.next()){
15     lastvalue = rsLast.getInt(4);
16   }
```

```
17
18   //Aktuelle Werte belegt und max
19   int max = Integer.valueOf(m1.group(2));
20   int belegt = Integer.valueOf(m1.group(3));
21   //Lizenznamen holen
22   Statement statementLicId = con.createStatement();
23   String querySelLicId = "SELECT Lizenzname FROM Lizenz WHERE
24                           PK_ID_Lic = '"+String.valueOf(keylic)+"'";
25   ResultSet rsId = statementLicId.executeQuery(querySelLicId);
26
27   if(rsId.next()){
28     //Maximalen Wert auf 90% abrunden
29     double prozent = Math.floor((double)max*90/100);
30
31     //Mail wird versandt, wenn belegt >=90% ist und belegt ansteigt
32     if (belegt>=prozent&&belegt>lastvalue) {
33        sendMail(rsId.getString(1), String.valueOf(max),
34                 String.valueOf(belegt));
35     }
36   }
37 }
38 [...]
```

Listing 6.4: Alarmfunktion

6.8 Test

In diesem Kapitel wird das Vorgehen und die Durchführung des Testens beschrieben. Es gibt einige Prinzipien, die sich in den letzten 40 Jahren als generelle Regeln für das Testen etabliert haben. Drei davon werden nachfolgend aufgeführt und beschrieben.

Prinzip 1

„Testing shows the presence of defects, not their absence.“ [Spillner u. a., 2011, S. 33]

Prinzip 2

„Exhaustive testing is not possible.“ [Spillner u. a., 2011, S. 33]

Prinzip 3

„Testing activities should start as early as possible.“ [Spillner u. a., 2011, S. 34]

Diese Prinzipien zeigen, dass auch eine sehr umfangreiche Durchführung von Tests nicht garantieren kann, dass die Anwendung keine Fehler mehr besitzt. Allerdings bieten Testfälle eine gewisse Art der Vorsorge, sodass Fehler im Regelbetrieb der Software kaum auftreten

sollten. Da eine Anwendung mit dem Fortschreiten der Entwicklung immer komplexer wird, ist frühzeitiges Testen besonders wichtig, um Fehler zu eliminieren.

6.8.1 Testvorgehen

In dem in Abbildung 6.6 dargestellten V-Modell wird gezeigt, dass Entwicklungsaufgaben und Testing zwei gleichberechtigte Aktivitäten sind. Die linke Seite des V-Modells zeigt die Schritte während des Entwicklungsprozesses. Die rechte Seite des Modells beschreibt die Test- und Integrationsprozesse, welche die korrespondierenden Schritte in der Entwicklung verifizieren. Nachfolgend werden die Teststufen des Modells erklärt.

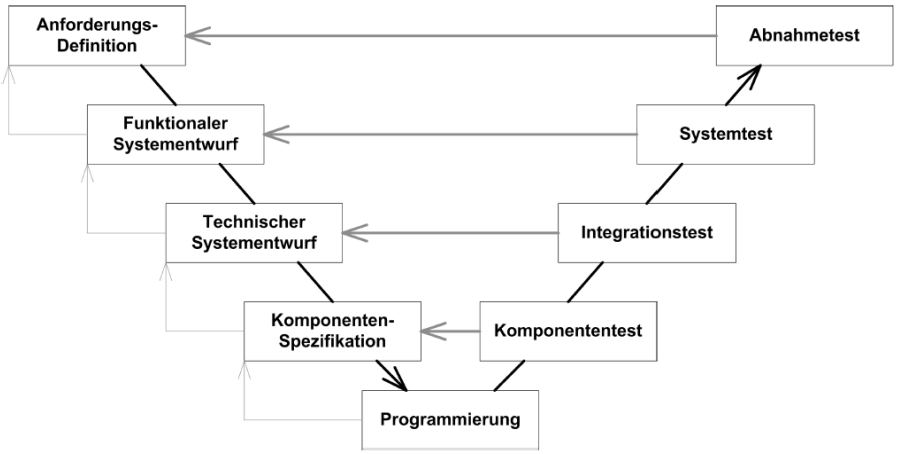

Abbildung 6.6: Aufbau des generellen V-Modells [Grafik entnommen aus Spillner u. a., 2011]

Komponententest

Die Durchführung von Komponententests erfolgt durch die Entwickler der Software. Getestet werden Module, wie beispielsweise Klassen. Dieser Vorgang dient zur Sicherstellung und Bewertung der technischen Lauffähigkeit, sowie korrekte fachliche Ergebnisse durch diese Module zu gewährleisten.

Integrationstest

Ein Integrationstest prüft die Korrektheit der Interaktion zwischen verschiedenen Komponenten. Es werden außerdem die Schnittstellen der Komponenten getestet.

Systemtest

Bei diesem Test wird das Gesamtsystem auf realisierte funktionale und nicht-funktionale Anforderungen überprüft.

Abnahmetest

Die letzte Stufe des Tests erfolgt meist durch den Auftraggeber. Dieser testet die Anwendung auf die vorgegebenen Anforderungen hin. Es wird lediglich das Verhalten des Systems getestet, also ein "BlackBox“-Test. Es wird nicht direkt die Implementierung der Software überprüft sondern die spezifizierten Handlungen der Software beim Gebrauch.

6.8.2 Ziele

Der Ablauf der Tests orientiert sich am vorhergehend vorgestellten V-Modell.

Ziel der Tests ist, es die funktionalen- und nicht-funktionalen Anforderungen auf Erfüllung und Qualität zu prüfen. Begonnen wurde mit den im Anwendungskern liegenden Komponenten. Es wurde für jede Komponente ein isolierter Komponententest durchgeführt. Hierbei entstehen keine Einflüsse durch andere Komponenten und ein eventueller Fehler kann direkt an der getesteten Komponente fest gemacht werden. Die Komponententests wurden mit dem Test-Framework JUnit4 durchgeführt.

Des Weiteren wird die Integration der Module durch manuelle Schnittstellentests durchgeführt.

Ein Systemtest wird durch Verwendung der Software und Überprüfung der funktionalen Anforderungen sichergestellt.

Ein Abnahmetest wird mit drei Mitarbeitern der Technischen Datenverarbeitung der Firma Rheinmetall durchgeführt. Die Mitarbeiter erhalten den Client und probieren die Funktionen der Anwendung aus und protokollieren eventuelle Schwierigkeiten beim Umgang mit der Anwendung.

6.8.3 Komponententest

Hier wird an zwei Testfällen beispielhaft gezeigt, wie die Alarmkomponente getestet wurde. Beim ersten Test (Listing 6.5) gibt die Methode *sendMail* bei erfolgreichen Durchlauf den Wert *true* zurück. Es wurde ebenfalls manuell überprüft, ob die Mail korrekt beim Empfänger angekommen ist und der Betreff und der E-Mail-Text korrekt war.

```
1 @Test
2 public void testSendMail() {
3       Alarm a = new Alarm();
4       assertTrue(a.sendMail("DESIGNER", "43", "43"));
5 }
```

Listing 6.5: Test der Alarmkomponente

Beim zweiten Test wurde ein falscher Port für den Mailserver gesetzt. Über diesen Port kann die E-Mail nicht zugestellt werden und die Methode wirft eine *RuntimeException* aus.

```
1 @Test(expected = RuntimeException.class)
2 public void testSendMailError() throws RuntimeException {
3       Alarm a = new Alarm();
4       //richtiger Port ist 25
5       a.setPort("1024");
6       a.sendMail("DESIGNER", "43", "43");
7 }
```

Listing 6.6: Test der Alarmkomponente mit RuntimeException

6.8.4 Integrationstests

Die Integrationstests für die entwickelte Anwendung erfolgten parallel zur Realisierung dieser. Begonnen wurde mit dem *LizenzAuswertungsDienst*. Es wurde mit dem Microsoft SQL Server Management Studio verifiziert, ob die korrekten Werte aus der Auswertungsdatei auch in die Datenbank eingetragen werden. Bei dem Test fiel auf, dass es beim Eintragen der Anmeldenamen in die Datenbank ein Problem gibt. Wenn die Anwender bei der Anmeldung ihr Anmeldenamen unterschiedlich schreiben (Groß- und Kleinschreibung) entstehen mehrere User in der Datenbank, die jedoch gleich sind. Um dieses Problem zu lösen wurden der eingetragene Benutzername von der Eintragungskomponente in Großbuchstaben umgewandelt, so ist in der Datenbank die Zusammensetzung aus Anmeldename und Computername immer eindeutig und wird nicht mehrfach vergeben. Bei der späteren Abrechnung über die Reports kommt es nicht zu falschen Zählungen der Benutzer.

Beim *LizenzAuswertungsClient* wurde für eine Anforderung zuerst das Zusammenspiel von Persistenz und Anwendungskern getestet und danach das Zusammenspiel von GUI und Anwendungskern. Die visuelle Darstellung z. B. der Diagramme in der GUI konnte wiederum mit den Daten in der Datenbank abgeglichen werden.

6.8.5 Systemtests

Die Systemtests erfolgten während der Entwicklung der Applikation iterativ. Bei Fertigstellung eines Use-Cases wurden die funktionalen und nicht-funktionalen Anforderungen an die Anwendung manuell geprüft. Durch die iterative Vorgehensweise beim Testen werden frühzeitig etwaige Fehler in der Realisierung aufgezeigt. Durch die Implementierung einer Anforderungen über alle Schichten hinweg, sodass diese Funktion für sich genommen implementiert ist, konnten Fehler im System ebenfalls früh entdeckt werden. Durch die Verwendung der Software konnten Anforderungen evaluiert und neue hinzugefügt oder bestehende Anforderungen angepasst werden. Da der Entwickler die GUI selbst realisiert hat, weiß er auch was diese unterstützt und was nicht. Der Entwickler geht nicht so unvoreingenommen an den Test der GUI, wie ein Anwender der das Programm zum ersten mal benutzt. Trotzdem kann der Entwickler beim Umgang mit der GUI erste Probleme und Fehlerfälle ausmachen, beseitigen und die Usability verbessern.

6.8.6 Abnahmetest / Nutzertest

Für den Abnahme- bzw. den Nutzertest wurde der *LizenzAuswertungsClient* drei Mitarbeitern der Technischen Datenverarbeitung der Firma Rheinmetall zur Verfügung gestellt. Die Anwendung wurde den Testern vorgestellt und erklärt. Der Nutzertest erlaubte die Durchführung eines „BlackBox“-Tests, da die Nutzer die Implementierung der Software nicht kannten. Weiterhin konnte durch die Protokolle der Anwender und Gespräche mit diesen Probleme im Bereich Usability und User-Experience identifiziert werden. Das Feedback der Nutzer konnte des Weiteren dafür genutzt werden, Erweiterungen in Form von fachlichen Anforderungen festzustellen und diese in die Applikation einfließen zu lassen.

6.8.7 Testergebnisse und Testauswertung

Die Testergebnisse der verschiedenen Testphasen wurden bei der Entwicklung und Anpassung der Applikation berücksichtigt. Durch die Unit-Tests für den LizenzAuswertungsDienst (Eintragung) und LizenzAuswertungsClient (Auswertung) wird die Korrektheit der Module validiert.

Das Testen der Integration der einzelnen Module in das Gesamtsystem, sowie die interaktiven Systemtests ermöglichen eine Abstimmung der gestellten Anforderungen mit den Realisierungen. Des Weiteren kann dadurch die GUI optimiert werden.

Der Nutzertest bzw. Abnahmetest ermöglichte es, das Nutzerinterface sowie die Nutzerakzeptanz der Applikation zu überprüfen. Einige dieser neuen Anforderungen, die durch das Feedback gewonnen wurden, konnten noch umgesetzt werden, andere dienen als Ausblick. Beispielsweise der automatisierte Abgleich der Benutzerdaten mit einer Excel-Tabelle aus der Personalabteilung konnte nicht mehr umgesetzt werden.

Der Test der Anwendung erfolgt zur Zeit größtenteils manuell. Lediglich die implementieren Unit-Tests können automatisiert ausgeführt werden. Um jedoch eine gute Testabdeckung über den gesamten Entwicklungszeitraum zu gewährleisten, ist eine Automatisierung der Tests anzustreben.

7 Evaluierung und Bewertung

In diesem Abschnitt werden die Inhalte der einzelnen Kapitel bewertet. Des Weiteren wird am Ende eine Gesamtbewertung vorgenommen.

7.1 Bewertung des Analyseteils

Im Analyseteil wurden die Anforderungen in einem Workshop mit zwei Projektverantwortlichen der Technischen Datenverarbeitung erhoben und verabschiedet. Als Grundlage dienten die zwei getesteten Lizenzmanagement-Tools.

Bei einem größeren Budget sowie Zeitrahmen wären umfangreichere Tests weiterer Lizenzmanager und Auswertungs-Tools möglich gewesen.

7.2 Bewertung der Spezifikation

Bei der Spezifikation wurde zuerst die Auswertungsdatei beschrieben. Diese Datei stellt die einzige Datenbasis für die Anwendung dar. Diese wird durch eine Lizenzserver-Abfrage alle n Minuten erzeugt und die Daten in die Datenbank geschrieben.

Der Master-Lizenzserver erstellt jedoch auch eine weitere Logdatei auf dem Server, in der bei Änderungen in der Lizenznutzung sofort ein neuer Eintrag in dieser Logdatei erscheint.

Es wurde sich frühzeitig für die Methode mit der Auswertungsdatei, die mittels Batchskript erstellt wurde, entschieden, weil das Parsing für die Server-Logdatei „on the fly“ gemacht werden muss und auch eine Vielzahl unterschiedlicher Pattern erstellt werden müssen. Vorteil bei der Auswertung der Server-Logdatei wäre die erhöhte Genauigkeit, da die Daten „live“ wären und nicht alle n Minuten abgefragt bzw. eingetragen werden müssten. Problematisch wäre der Zugriff auf die Server-Logdateien, da diese Server nicht in der Domäne sind und eine Freigabe somit nicht möglich ist. Außerdem wird ein spezieller Nutzer für den Zugriff auf die Server benötigt.

Weiterhin wurde das fachliche Entitiy-Relationship Model beschrieben, welches auf die Auswertemethode angepasst ist. Auf die Anpassungsfähigkeit des Modells wurde Wert gelegt, da hierdurch während der Entwicklung neue Anforderungen leichter umgesetzt werden können.

Die Dialogerstellung mittels eines Mocking-Werkzeugs half dabei, eine erste Diskussionsbasis für den späteren Aufbau der GUI zu haben.

7.3 Bewertung des Entwurfs

Bei dem Entwurf wurde zuerst die Architektur der Anwendung beschrieben. Es wurde eine Drei-Schichten-Architektur für den *LizenzAuswertungsClient* ausgewählt. Diese Architektur ist eine Standardarchitektur für Informationssysteme und bot sich auch für diese Anwendung an.

Die Trennung der Applikation in zwei separate Programme wurde realisiert, um die Eintragung der Daten in die Datenbank auch auf einem eigenen Server durchzuführen. Des Weiteren herrscht dadurch auch eine klare Trennung zwischen Eintragung und Auswertung der Daten.

7.4 Bewertung der Realisierung

Hier wird die Umsetzung der funktionalen und nicht-funktionalen Anforderungen überprüft. In der Applikation wurden alle Anforderungen bis auf zwei umgesetzt (siehe nachfolgende Tabelle 7.1). Diese beiden werden nachstehend kurz erläutert.

Die funktionale Anforderung A-7 beschreibt, dass Lizenzen auch temporär vergeben bzw. ausgeliehen werden können, so genanntes Borrowing. Dabei kann z. B. eine Lizenz für einen Laptop ausgeborgt werden, sodass die Lizenz auch offline verfügbar ist und ohne den Zugriff zum Lizenzserver benutzt werden kann. In der Auswertungsdatei wird dies über ein zusätzliches Schlüsselwort (linger) angezeigt. Zusätzlich wird die Dauer der Ausleihe in Minuten angegeben. Um diese Anforderungen umzusetzen, ist es denkbar eine neue Tabelle zu erstellen, die nur diese Ausleihen speichert. Dazu müsste außerdem in der *Eintragungskomponente* zusätzlich ein Pattern geschrieben werden, welches dieses Muster erkennt. Weiterhin müsste der *LizenzAuswertungsClient* auf allen Schichten angepasst werden. Da diese Funktion nur selten genutzt wird wurde die Priorität auf die anderen Anforderungen gelegt.

Die funktionale Anforderung A-8, in der es darum geht, die Lizenzserver bezüglich Diensten, Laufzeit, CPU und Speicher zu überwachen, wurde nicht umgesetzt. Es war bei der Recherche

schwierig, eine geeignete Library zu finden, die so eine Funktionalität gewährleistet. Die Entwicklung einer eigenen Client/Server-Anwendung wäre denkbar, würde jedoch mit Thema dieser Bachelorarbeit wenig zu tun haben und auch zeitlich schwer umsetzbar sein. Eine weitere denkbare Lösung wäre die Installation eines kommerziellen Tools zum Server-Monitoring auf den Lizenzservern und dann die Abfrage dieser Daten per Java bzw. mit Java über die Kommandozeile.

Funktionale Anforderung	Kurzbeschreibung	Realisierung
A-1	Überwachung und Auswertung der Lizenzen Teamcenter und NX	✔
A-2	GUI als Frontend, die ohne Einweisung benutzt werden kann	✔
A-3	Monatliche Reports zur Abrechnung	✔
A-4	Vergleich der Lizenznutzung zwischen Tag, Woche, Monat und Jahr	✔
A-5	Analyse und Auswertung der Spitzenwerte	✔
A-6	Schwellenwerteinstellung mit Warnnachricht ins Mail-System	✔
A-7	Überwachung temporär vergebener Lizenzen	✗
A-8	Monitoring der Lizenzserver (Dienste, Laufzeit, CPU, Speicher)	✗
A-9	Archivierung der Logdateien/Auswertungsdateien für einen späteren Nachweis	✔
A-10	Archivierung der Logdateien/Auswertungsdateien in Form von BLOBs	✔
Nicht-funktionale Anforderung	**Kurzbeschreibung**	**Realisierung**
A-11	Konfigurationsmöglichkeiten für weitere Lizenzen und Lizenzserver aus dem Teamcenter und NX Umfeld	✔
A-12	Portierbarkeit nach Unix	✔
A-13	Handbuch für die Migration und Konfiguration weiterer Lizenzen	✔

Tabelle 7.1: Umsetzung der Anforderungen

7.5 Gesamtbewertung

Wie im Abschnitt 6.8.7 Testergebnisse und Testauswertung kurz beschrieben wurde, werden zurzeit die Daten der Benutzer über einen Pflegedialog gepflegt. Diese Daten müssen dann vor dem aktuellen Monatsreport jeweils abgeglichen und gegebenenfalls korrigiert werden. In einem Unternehmen können sich Kostenstellen, Abteilungen und andere Daten ständig ändern. Dies kann in der jetzigen Anwendung nur manuell nachgepflegt werden.

Diese Tatsache bedeutet einen nicht zu unterschätzenden Aufwand, da im späteren System ungefähr 250 Benutzer eingetragen sind. Ziel wäre es, die Daten bei jedem anfallenden Report automatisch mit einer Datei von der Personalabteilung abzugleichen und somit den manuellen Pflegeaufwand zu sparen.

Wie bereits im Abschnitt 6.5 beschrieben wurde, sind die Reports von Hand geschrieben und es wurde auf ein externes evtl. flexibleres Reporting-Tool verzichtet. Um ein geeignetes Reporting-Tool auswählen zu können, müsste wiederum eine genauere Analyse und Test dieser Tools vorgenommen werden. Weitere Gründe, warum die Reports manuell erstellt wurden, sind vorhergehend in Abschnitt 6.5 aufgeführt.

Das Hinzufügen und Auswerten weiterer Lizenzserver ist in der Applikation integriert. Dazu muss lediglich die Batch-Routine, welche die Auswertungsdatei erstellt, und die Konfigurationsdatei angepasst werden. Eine kurze Anleitung dazu wird im Anhang A gegeben.

Es wurden in dieser Applikation elf von dreizehn Anforderungen umgesetzt. Der Systemtest und der Abnahmetest bestätigten auch die geforderte Qualität der Umsetzung dieser Anforderungen.

8 Zusammenfassung und Ausblick

In diesem Kapitel werden abschließend die erreichten Ergebnisse zusammengefasst, sowie ein Ausblick über mögliche Weiterentwicklungen und Verbesserungen des Systems der Arbeit gegeben.

8.1 Zusammenfassung

In dieser Arbeit wurde eine Applikation entworfen und realisiert, die es ermöglicht, die Lizenznutzung von standortübergreifend-nutzbaren Applikationen auszuwerten, abzurechnen, zu überwachen und einen Nachweis darüber zu führen.

Um dies zu erreichen wurde zuerst die Lizenzumgebung der Firma Rheinmetall analysiert. Anschließend wurden einige am Markt verfügbare Produkte getestet. Des Weiteren wurde das Verfahren der Lizenzvergabe und -verwaltung des im Betrieb befindlichen Lizenzmanagers FlexLM untersucht.

Im Analyseteil wurde basierend auf den getesteten Applikationen und den Anforderungen der Projektverantwortlichen das System abgegrenzt. Basierend auf der Analyse wurde das System durch die Aufschlüsselung der Auswertungsdatei, ein fachliches Entity-Relationship Model, Anwendungsfälle und Dialoge spezifiziert.

Im nächsten Schritt wurde das System entworfen. Es wurden Komponenten gesucht und die Architektur festgelegt.

Entsprechend der Spezifikation und des Entwurfs wurde das System dann realisiert. Es wurde hierbei mit der Datenbank begonnen, darauf folgte die Eintragungskomponente und anschließend der Client zur Analyse und Auswertung der Daten.

Gemäß dem V-Modell wurde durch Tests die Qualität der Anwendung über den ganzen Entwicklungszyklus hinweg sichergestellt.

8.2 Ausblick

Das entwickelte System stellt eine Basis für viele Erweiterungsmöglichkeiten dar. Es können weitere Diagrammarten sowie Reports integriert werden.

Es können weitere Lizenzserver und damit Lizenzen aus dem FlexLM-Umfeld mit wenigen Anpassungen in der Konfigurationsdatei aufgenommen werden.

Eine sinnvolle Erweiterung würde eine Berechtigungskomponente darstellen. Beim Start der Anwendung hat der Benutzer Zugriff auf alle Funktionen des Programms – insbesondere auch auf die Pflegedialoge. Über diese können für die Abrechnung benötigte Daten angepasst werden. Wenn diese Anwendung in der Abteilung oder an anderen Standorten verteilt wird, sollte der Zugriff auf diese Funktion reglementiert werden, so dass nicht jeder Anwender die abrechnungskritischen Daten verändern kann.

Der Report nach Standort und Monat dauert zur Zeit in der Erstellung etwa fünf Minuten. Sollten später noch größere Zeiträume abgerechnet werden müssen – z. B. ein Quartal oder ein Jahr – ist hier eine Optimierung anzustreben. Denkbar wäre eine Vorberechnung nach Programmstart und Zwischenspeicherung dieser Ergebnisse, um bei einer konkreten Anfrage die Daten schneller zur Verfügung stellen zu können.

Weiterhin sollte der Schwellenwert, bei dem eine E-Mail generiert wird, konfigurierbar sein. Da bereits eine Konfigurationsdatei besteht, wäre eine Aufnahme des Schwellenwerts in diese sinnvoll.

Anhang A

Installations- und Konfigurationsanleitung

In diesem Kapitel folgt eine kurze Anleitung zur Aufnahme weiterer Lizenzen / Lizenzserver in die Anwendung. Des Weiteren wird beschrieben, wie die Datenbank auf einen anderen Server migriert werden kann.

A.1 Konfiguration weiterer Lizenzen und Lizenzserver

Um weitere Lizenzen in der Eintragungskomponente zu definieren, muss die Datei *Configuration.properties* angepasst werden. Zurzeit sind in der Datei zwei Einträge. Diese Einträge werden in einer Schleife abgearbeitet und die .bat-Dateien werden ausgeführt. Die Ergebnisdateien müssen so heißen wie der Schlüssel der Properties Einträge. Also bei der Datei *Lizenz-nx4-aktuell.bat* ist der Schlüssel *NX4* und die Datei die die .bat Datei erstellt heißt *NX4.txt*. Diese Datei wird dann ausgewertet und die Ergebnisse in die Datenkbank eingetragen. Anschließend wird die Datei in das Archiv geschrieben und gelöscht.

```
NX4=S:\\LizenzAuswertungsDienst\\dist\\Lizenz-nx4-aktuell.bat
NX6=S:\\LizenzAuswertungsDienst\\dist\\Lizenz-nx6-aktuell.bat
```

Listing A.1: Configuration.properties

```
echo off
set TEMP=S:\Praktikum3\LizenzAuswertungsDienst\dist\
cd /d %TEMP%
set TIME_1=%TIME%
for /f "tokens=1,2,3* delims=. " %%i in ('date /t') do set DAT_TAG=
%%i&set DAT_MONAT=%%j&set DAT_JAHR=%%k

for /f "tokens=1,2* delims=: " %%i in ('time /t') do set STUNDE=%%i&set MINUTE=%%j

set STUNDE00=%STUNDE%
if %STUNDE% LSS 10 (set STUNDE00=0%STUNDE%)
if %STUNDE% LSS 1 (set STUNDE00=00)

set CHECK_DATE=%DAT_JAHR%-%DAT_MONAT%-%DAT_TAG%_%USERNAME%
```

```
15 set LIC_FILE=%TEMP%NX4
16
17 echo Benutzte Lizenzen um %TIME%  >> %LIC_FILE%.txt
18 echo ------------------------------------------------------------------------ >> %LIC_FILE%.txt
19 lmutil lmstat -f iman_1st -c 27000@C71l17SR0703 >> %LIC_FILE%.txt
20 echo ------------------------------------------------------------------------ >> %LIC_FILE%.txt
21 lmutil lmstat -f iman_author -c 27000@C71l17SR0703 >> %LIC_FILE%.txt
22 echo ------------------------------------------------------------------------ >> %LIC_FILE%.txt
23 lmutil lmstat -f iman_nth -c 27000@C71l17SR0703 >> %LIC_FILE%.txt
24 echo ------------------------------------------------------------------------ >> %LIC_FILE%.txt
25 echo ------------------------------------------------------------------------ >> %LIC_FILE%.txt
```

Listing A.2: Lizenz-nx4-aktuell.bat

A.2 Migration der Datenbank

Durch die im Microsoft SQL Server Management Studio eingebaute *Sichern* und *Wiederherstellen* Funktion kann hiermit die Datenbank auf einen neuen Server migriert werden. In dem Artikel von Microsoft unter http://support.microsoft.com/kb/314546 wird beschrieben, wie so eine Migration durchzuführen ist.

Anhang B

Inhalte der beigefügten CD

- Sourcecode des LizenzAuswertungsDienst
- Sourcecode des LizenzAuswertungsClient
- Bachelorarbeit als PDF

Glossar

CAD

Computer Aided Drafting oder Computer Aided Design, also computerunterstützte Zeichnungserstellung oder Konstruktion.

CAE

Computer Aided Engineering wird für computergestützte Berechnung und Simulation im Ingenieurwesen verwendet. Daneben wird die Abkürzung vielfach auch für CAD in der Elektrotechnik gebraucht.

CAM

Computer Aided Manufacturing. Erzeugung von Programmen zur Ansteuerung von Maschinen beispielsweise zum Fräsen, Drehen, Bohren oder Stanzen. Moderne Installationen gestatten die weitgehend automatische Erstellung solcher Programme auf Basis von CAD-Modellen.

Compliancecheck

Beschreibt den Vorgang, die im Rahmen des Lizenzmanagement notwendigen kaufmännischen und technischen Daten abzugleichen, mit dem Ziel, die dauerhafte und umfassende Einhaltung der gesetzlichen Regelungen für den Umgang mit Software und deren Lizenzbestimmungen im Unternehmen zu gewährleisten. [vgl. Groll, 2009]

PDM

Product Data Management dient dazu, Informationen über Produkte und deren Entstehungsprozesse bzw. Lebenszyklen konsistent zu speichern, zu verwalten und allen relevanten Bereichen eines Unternehmens bereitzustellen.

PLM

Product Lifecycle Management bezeichnet ein strategisches Konzept mit dem Ziel, den gesamten Produktlebenszyklus (von der Konzeption bis zur Entsorgung) durchgängig zu

unterstützen. Fokussiert wird hierbei auf die Erarbeitung, Verwaltung, Kommunikation und Nutzung von Informationen zur Produktdefinition in einem Unternehmen.

Unterlizenzierung

Es gibt weniger Lizenzen als Software-Bereitstellungen oder -Installationen.

Überlizenzierung

Es gibt mehr Lizenzen als Software-Bereitstellungen oder -Installationen.

Abkürzungsverzeichnis

BLOB	Binary Large Object
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CPU	Central Processing Unit
DDL	Data Definition Language
GUI	Graphical User Interface
JAR	Java Archive
JDBC	Java Database Connectivity
JRE	Java Runtime Environment
ORM	Object-Relational Mapping
PDM	Product Data Management

PLM	Product Lifecycle Management
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[Acresso 2008] Acresso: *License Administration Guide.* 2008

[Bürkner 2003] Bürkner, Reimer M.: *Erfolgreiche Software-Lizenzierung: Electronic License Management.* Berlin, Heidelberg : Springer Verlag, 2003

[Elmasri und Navathe 2005] Elmasri, Ramez ; Navathe, Shamkant B.: *Grundlagen von Datenbanksystemen – Ausgabe Grundstudium 3. Auflage.* München : Pearson Studium, 2005

[Graf u. a. 2012] Graf, Stefan ; ; Corban, Michael: *PLM-Jahrbuch 2013 – Der Leitfaden für den PLM-Markt.* Darmstadt : WEKA BUSINESS MEDIEN GmbH, 2012

[Groll 2009] Groll, Torsten: *1 × 1 des Lizenzmanagements: Praxisleitfaden für Lizenzmanager.* München : Hanser, 2009

[IBM 2013] IBM: *LUM License Use Management.* 2013. – URL http://www-01.ibm.com/software/awdtools/lum/about.html. – Zugriffsdatum: 04.04.2013

[Klette u. a. 2011] Klette, Guido ; Nulsch, Michael ; Vajna, Sándor: *NX7.5 – kurz und bündig.* Wiesbaden : Vieweg+Teubner Verlag, 2011

[NovaTec 2012] NovaTec: *.inCharge.* 2012. – URL http://www.incharge.eu. – Zugriffsdatum: 10.12.2012

[Riemenschneider 2008] Riemenschneider, Dorrit: *Neuerungen in SQL Server 2008: FILESTREAM Storage.* 2008. – URL http://www.communardo.de/home/techblog/2008/12/25/neuerungen-in-sql-server-2008-filestream-storage/. – Zugriffsdatum: 23.04.2013

[Scholten 2011] Scholten, Anne-Katrin: *Objektrelationales Mapping am Beispiel von Hibernate.* 2011. – URL https://www.matse.rz.rwth-aachen.de/dienste/public/show_document.php?id=7095. – Zugriffsdatum: 24.05.2013

[Siedersleben 2005] Siedersleben, Johannes: *Moderne Software-Architektur: umsichtig planen, robust bauen mit Quasar.* Heidelberg : dpunkt-Verlag, 2005

[Siemens 2012] Siemens: *Siemens PLM Licensing User Guide.* 2012

[Spillner u. a. 2011] Spillner, Andreas ; Linz, Tilo ; Schaefer, Hans: *Software Testing Foundations.* Santa Barbara : Rocky Nook Inc., 2011

[Starke 2008] Starke, Gernot: *Effektive Software-Architekturen – Ein praktischer Leitfaden.* München : Hanser, 2008

[Wikipedia 2012] Wikipedia: *Siemens NX.* 2012. – URL http://de.wikipedia.org/wiki/NX_(Siemens). – Zugriffsdatum: 08.01.2013

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 28. Mai 2013 Jan Henrik Ohlhoff