Fachhochschule Köln
Cologne University of Applied Sciences
Campus Gummersbach
Fakultät für Informatik und Ingenieuerwissenschaften

Fachhochschule Dortmund
University of Applied Sciences and Arts
Fachbereich Informatik

Verbundstudiengang Wirtschaftsinformatik

Bachelor-Arbeit
(Vier-Monats-Arbeit)
zur Erlangung des Bachelorgrades
„Bachelor of Science“
in der Fachrichtung Informatik

Analyse und Realisierung von Optimierungsmöglichkeiten im Rahmen der Datenbankprogrammierung unter Berücksichtigung von Performance-Kriterien

Erstprüfer:	Prof. Dr. Birgit Bertelsmeier
Zweitprüfer:	Prof. Dr. Heide Faeskorn-Woyke
vorgelegt am:	9. Februar 2012

von cand.:	Miguel Cifuentes Perelló
Anschrift:	Am Büter 26 44269 Dortmund
E-Mail:	miguel.cifuentes@live.de
Matr.-Nr.:	7073096

Abstract

Diese Arbeit befasst sich mit Strategien zur Vermeidung von Perfomance-Problemen im ORACLE©-Datenbankumfeld, die auf ineffizienten Algorithmen oder Datenbankstrukturen basieren. In diesem Zusammenhang werden Lösungsalternativen hinsichtlich Effizienz und Zielabdeckung bewertet. Auf Grundlage der Erkenntnis, dass Ursachen und Lösungsoptionen für Performance-Probleme vielfältig sein können, entsteht die Konzeptidee für eine neue Rolle innerhalb des Entwicklungsteams - den Performance-Betreuer. Dem Leser vermittelt diese Arbeit mit Hilfe konkret geschilderter Aspekte der betrieblichen Softwareentwicklung einen praxisnahen Einblick in die Komplexität der nicht-funktionalen Anforderung Datenbankperformance. Trotz dieser Vielschichtigkeit werden möglichst einfache und verständliche Optimierungsmöglichkeiten aufgezeigt.

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung

Im Kapitel „Einleitung“ erfolgt die Einführung in das Themengebiet Datenbank-Performance im Umfeld des Teilprojektes IEA[1] (Internet-EMCS-Anwendung). Zu Beginn wird der grundsätzliche Aufbau dieser Arbeit beschrieben (Kapitel 1.1). In Kapitel 1.2 wird die Zielgruppe definiert, für die diese Arbeit ausgelegt ist. Anschließend erfolgt die Einordnung in das Projektumfeld (Kapitel 1.3). Anhand der Problemstellung (Kapitel 1.4) werden die Motive zur Ausarbeitung von Optimierungsmöglichkeiten im Rahmen der Datenbankentwicklung erläutert. Es erfolgt dabei ein Vorgriff auf den Begriff „Performance“, der im Anschluss definiert wird (Kapitel 2.1).

1.1 **Aufbau**

Diese Arbeit besteht neben dem einleitenden Kapitel, das die Rahmenbedingungen und einen Einstieg hinsichtlich der Problemstellung und der Motivation aufgreift, aus fünf Hauptkapiteln.

In Kapitel 2 werden die Informationsgrundlagen für das Verständnis dieser Arbeit hinsichtlich des Begriffs „Performance“ und die Einschränkung auf Teilaspekte dieses komplexen Themenfeldes aufgeführt.

In Kapitel 3 werden anhand der Kennzahlen, Auswertungen von Besprechungen und Fehlertickets aus dem Echtbetrieb des Teilprojektes IEA Indizien für erforderliche Maßnahmen im Problembereich „Performance“ benannt.

Das Kapitel 4 beschreibt unterschiedliche Lösungsvarianten und führt Vor- und Nachteile auf.

Als Ergebnis der Lösungsvarianten, die keine allumfassende Problembehebung ermöglichen, wird die Konzeptidee des Perfomance-Betreuers in Kapitel 5 erläutert.

Abschließend wird in Kapitel 6 das Fazit gezogen.

1.2 **Zielgruppe**

Diese Arbeit ist im ORACLE©-Datenbankumfeld entstanden. Deshalb werden beim Leser Datenbankgrundkenntnisse vorausgesetzt und nicht weitergehend

[1] Nähere Erläuterung erfolgt in Kap. 3.2.

erläutert. Neue Funktionalitäten oder Besonderheiten werden explizit erklärt. Neben allen interessierten Lesern ist diese Arbeit für Entscheidungsträger konzipiert, die eine technische und organisatorische Verantwortung in Datenbankprojekten übernehmen.

1.3 **Projektumfeld**

Diese Arbeit basiert auf den Erfahrungen und den Gegebenheiten, wie sie im Projekt EMCS[2] (Teilprojekt IEA) vorzufinden waren, beziehungsweise vorzufinden sind. Aus diesem Grund stehen Problemstellungen und Lösungsmöglichkeiten im direkten Zusammenhang mit dem Datenbanksystem ORACLE© (11G Release 1). Unter Berücksichtigung der gegebenen organisatorischen, strukturellen und technischen Einschränkungen und den vorhandenen Hilfsmitteln[3] werden Lösungsmöglichkeiten aufgezeigt und im Kontext der gesamten betrieblichen Softwareentwicklung bewertet.

1.4 **Problemstellung und Motivation**

Bei jedem Projekt handelt es sich um ein einmaliges Vorhaben, an dem mehrere Personen beteiligt sind[4]. Trotz der Einmaligkeit aus der Gesamtperspektive gibt es in der Softwareentwicklung Tätigkeiten und Aufgaben, die von ihren Rahmenbedingungen in jedem Projekt identisch sind. So wird zum Beispiel die Datenbank „die Mutter jeder großen Anwendung“[5] genannt und ist damit die zentrale Komponente in einer Vielzahl von Projekten. Jeder Entwickler profitiert somit von seinen persönlichen Erfahrungen aus den Vorgängerprojekten. Dadurch werden sein Fachwissen und seine Arbeitsleistung gesteigert. Fehler, die begangen wurden, sollten in Zukunft vermieden und die Vorgehensweise angepasst werden. Funktionale Fehler einer Software können oftmals unmittelbar durch Tests aufgedeckt werden. Dieser Umstand verstärkt das persönliche Problemempfinden der Softwareentwickler. Bei Fehlern, die eine schlechte Performance der Datenbank hervorrufen, treten die Auswirkun-

[2]Begriffserklärung erfolgt in Kap. 3.1.
[3]Vgl. [Ham08] S.158 (Kap. 5.1.2).
[4]Vgl. KARNOVSKY, HANS: Grundlagen des Projektmanagements: ein Leitfaden für die Projektpraxis. Paul Bernecker Verlag, 2002. zitiert in [Sch10] S.72 (Kap. 4.1).
[5][Bra08] S.13 (Kap. 1.2).

gen zeitversetzt auf. Oftmals fallen die Probleme erst im Lasttest oder im Echtbetrieb auf und gehen auf Versäumnisse oder fehlende Kenntnisse während der Design- und Realisierungsphase zurück[6]. Die Ursachenfindung ist meist ein aufwendiger Prozess, da die Symptome von Performance-Problemen zwar anhand langsamer Reaktionszeiten zu erkennen sind, die Gründe aber vielfältig sein können[7]. Aufgrund dieser speziellen Eigenschaften der Fehler im Performance-Bereich werden die Hauptursachen von den Entwicklern nicht bewusst wahrgenommen: Es handelt sich dabei in erster Linie um ineffizienten Datenbankstrukturen und -algorithmen.

In dieser Arbeit wird deshalb im Sinne agiler Softwareprojekte die Umgangsweise mit Performance-Problemen innerhalb des Teilprojektes IEA analysiert und Lösungsmöglichkeiten für eine optimierte Vorgehensweise ermittelt und bewertet[8]. Somit werden die persönlichen Erfahrungen der einzelnen Akteure um die im zeitlichen Verlauf gesammelten Projekterkenntnisse erweitert. Nach dieser Abschluss- bzw. Zwischenanalyse als ein Aspekt des geregelten Projektabschlusses wird auch das Nachfolgeprojekt durch seine Einmaligkeit gekennzeichnet bleiben. Allerdings können die Erkenntnisse aus dem Teilprojekt IEA mit einfließen und die Vorgehensweise kann hinsichtlich der Beachtung der Datenbankperformance optimiert werden.

[6]Vgl. [Cor93] S.20 (Kap. 2).
[7]Vgl. [Mil03] S.26f (Kap. 1).
[8]Vgl. [Hru09] S.95 (Kap. Agile Softwareentwicklung in großen Projekten).

2 Grundlagen

Es wird in Kapitel 2.1 der Begriff „Performance“ als Grundlage für diese Arbeit beschrieben und auf den Teilaspekt ineffiziente Algorithmen und Datenbankstrukturen eingegrenzt. Die Grundannahme, dass Performance als Softwaremerkmal gesehen wird (Kapitel 2.2), führt im Fehlerfall zu einer Bewertung als Softwarefehler. Auf dieser Einschätzung basierend werden abschließend die übergeordneten Ziele (Kapitel 2.3) definiert, an denen die in Kapitel 4 geschilderten Lösungsmöglichkeiten gemessen werden.

2.1 **Der Begriff „Performance“**

Primär wird unter dem Begriff „Performance“ das Antwortzeitverhalten eines Anwendungssystems verstanden. Darunter fallen auch die Teilaspekte Durchsatz und Kapazität. Der Durchsatz gibt an, wie viele Transaktionen oder Daten ein System pro Zeiteinheit verarbeiten kann. Den Umfang der zur Verfügung stehenden Ressourcen eines Systems (u. a. Festplattengröße, Prozessoranzahl, Netzwerkbandbreite) beschreibt der Begriff „Kapazität“. Da eine niedrige Kapazität und/oder ein niedriger Durchsatz für langsames Antwortverhalten sorgen, kann es vielfache Ursachen für eine schlechte Performance geben. In dieser Arbeit geht es in erster Linie um ineffiziente Algorithmen bzw. SQL-Abfragen. Diese sind meist der Hauptgrund für Performance-Probleme, die durch Fehler in der Datenbankprogrammierung oder –konzeptionierung verursacht werden. Zwar kann es auch durch ein fehlerhaft implementiertes Speichermanagement zu Speicherlecks und somit zu einer Verschlechterung der Antwortzeiten (Arbeitszeit + Wartezeit) bei längerer Laufzeit kommen, aber in diesen Fällen können die ausschlaggebenden Faktoren so unterschiedlich sein, dass sich keine allgemeingültigen Problemlösungen in dieser Arbeit aufstellen lassen[9]. Insofern beschreibt diese Arbeit die Suche nach allgemeingültigen Verbesserungsmöglichkeiten in der täglichen ORACLE©-Datenbankentwicklung mit dem Ziel, dauerhaft kurze Reaktionszeiten des Anwendungssystems zu erreichen.

[9]Vgl. [Sch07] S.251f.

2.2 Performance als Softwaremerkmal

Bei der Betrachtung der Performance eines Anwendungssystems bei identischem Softwarestand ist im Unterschied zu den implementierten Funktionen ausschlaggebend, unter welchen Voraussetzungen Antwortzeiten gemessen werden. In der Regel lässt sich durch Testverfahren die Funktionalität einer Software verifizieren. Zwar gibt es nie eine vollkommene Sicherheit, aber die Wahrscheinlichkeit, dass sich ein Testszenario in unterschiedlichen Umgebungen gleich verhält, ist entscheidend höher als bei der Analyse der Performance. So kann es vorkommen, dass aus Sicht der Entwicklung die Anwendung zügig und schnell arbeitet. Dies muss sich aber nicht zwangsläufig mit der Sicht des Kunden decken. Neben zahlreichen Faktoren wie der Netzauslastung oder der Anzahl der parallelen Zugriffe haben oftmals die Kennzahlen der unterschiedlichen Umgebungen einen maßgeblichen Einfluss auf die Reaktionszeiten der Anwendung. Gerade im Datenbankumfeld lässt sich ein eindeutiges Unterscheidungsmerkmal benennen: die Füllstände der Datenbanken. Jede Anwendung scheint performant, wenn die Datenbank nur einen Testdatensatz verarbeiten muss, aber welche Antwortzeiten lassen sich ermitteln, wenn der Füllstand 200.000 mal so hoch ist? Wird Performance als ein Merkmal der Software gesehen, muss daran genauso gearbeitet werden wie an den Funktionalitäten der Anwendung[10]. Da Störungen im Betriebsablauf immer auch mit Vertrauensverlust beim Kunden einhergehen, können schlechte Antwortzeiten als Softwarefehler eingestuft werden[11].

2.3 Zieldefinition

Gehen Performance-Probleme auf ineffiziente Algorithmen im Quellcode zurück, sind sie mit derselben Priorität einzustufen wie funktionale Fehler. Ein Fehler ist umso teurer, je später er im Projektverlauf erkannt wird[12]. Fällt dieser erst im Lasttest auf, arbeitet ein ganzes Expertenteam aufgrund begrenzter Ressourcen (z.B. begrenzte Rechnerverfügbarkeit) an einer schnellen Lösung. Bei Performance-Schwachstellen im Echtbetrieb schalten sich gleich die tech-

[10] Vgl. [iMil09] S.12 (Kap. 2.1).
[11] Vgl. [Frö08] S.1 (Kap. 1).
[12] Vgl. [Bra08] S.23 (Kap. 1.2).

nische und die allgemeine Projektleitung mit ein. Es ist das Ziel dieser Arbeit das Thema Performance stärker in den Blickpunkt der Entwicklung zu rücken und Möglichkeiten aufzudecken, Schwachstellen in dieser Hinsicht bereits während der Testphasen aufzufinden. Dadurch könnten Zeit- und Kostendruck minimiert und die Qualität der Software in Bezug auf die nicht-funktionale Anforderung Performance verbessert werden. Es gilt aus einem Pool von Lösungsmöglichkeiten die beste zu ermitteln[13].

Dabei sollen gewisse Kriterien erfüllt werden:

Es gilt eine effiziente Lösung anzustreben: Eine Lösung zu finden, die möglichst wenig an Ressourcen wie Zeit, Mitarbeitereinsatz und Geld kostet und einen möglichst hohen Ertrag liefert. Diese Betrachtungsweise ist im Prinzip eindimensional, da hierbei nur oberflächliche Kostengründe einfließen und die Bedeutung der Risiko-Einschätzung[14] außer Acht gelassen wird. Doch es liegen keine Informationen über mögliche Regressansprüche oder sonstige finanzielle Risiken und Rahmenbedingungen des Projekts vor. Es bleibt also primär das Ziel der Vermeidung von direkten Kosten. Hierzu zählen Schulungsaufwendungen und der generelle Ressourcenbedarf an Mitarbeitern oder Hardware.

Dies könnte sich gerade unter dem Aspekt der kreativen Entfaltung bei der Entwicklung von Lösungsmöglichkeiten[15] als Manko erweisen. Es wirkt oftmals reizvoller, beispielsweise neue Tools zu kaufen oder zu programmieren[16].

Daraus folgt das zweite formulierte Ziel: *Die vorhandenen Werkzeuge sollen optimal ausgenutzt werden*. Dabei gilt es zu überprüfen, ob wirklich alle Funktionen bekannt sind. Es ist möglich, dass nur ein Teil der Werkzeugfunktionalitäten genutzt wird oder dass Weiterentwicklungen gar nicht bemerkt werden. Wird dieses Ziel mit dem Streben nach Effizienz und der Anforderung, dass alle Entwickler und Mitarbeiter der Qualitätssicherung (QS) die Optimierungsstrategien auch verstehen, akzeptieren und nutzen können sollten, verknüpft, führt dies unweigerlich zu einer klaren Regel:

---

[13]Vgl. [iMil99] S.9 (Kap. 8).
[14]Vgl. [Frö08] S.4 (Kap. 1).
[15]Vgl. [Ham08] S.157 (Kap. 5.1).
[16]Vgl. [Bal09] S.59 (Kap. 6.1).

> „Vereinfache! Vereinfache! Vereinfache! (…) Weniger ist (manchmal) mehr! Einfache Strukturen sind leichter und günstiger realisierbar, einfacher verständlich, weniger fehleranfällig. Die zuverlässigste, preiswerteste und robusteste Komponente eines Systems ist diejenige, die erst gar nicht realisiert werden muss!“[17]

Die Lösung sollte außerdem entscheidend mit berücksichtigen, dass sich die Laufzeit älterer Abfragen durch Datenbankanpassungen jederzeit ändern kann[18]. Doch es geht nicht allein darum, Softwareanpassungen im Blick zu behalten, sondern vielmehr um die Tatsache, dass Erkenntnisse gewonnen werden, die für das gesamte Team und im besten Fall für die ganze Abteilung unabhängig vom Projekt nützlich sein können.

[17] [Ham08] S.166 (Kap. 5.2.1).
[18] Vgl. [Frö08] S.2 (Kap. 1).

3 IST-Analyse: Teilprojekt IEA

Um ein Verständnis für die Zusammenhänge und Hintergründe, vor denen diese Arbeit entstanden ist, zu gewinnen, werden in Kapitel 3.1 die grundlegenden Informationen zum Hauptprojekt EMCS (Excise Movement and Control System) aufgeführt. Anschließend wird die Internet-EMCS-Anwendung (IEA) als Teilprojekt von EMCS (Kapitel 3.2) vorgestellt. Es werden die Vorgehensweise (Kapitel 3.3) und technischen Gegebenheiten der Datenbankumgebungen (Kapitel 3.4) des Softwareprojekts erläutert. Zusammen mit den IEA-Kennzahlen und den Statistiken (Kapitel 3.5) bilden diese Informationen die Rahmendaten für die Problem-Analyse (Kapitel 3.6). Als abschließendes Resümee wird anhand der zuvor erworbenen Erkenntnisse das Handlungsfeld Performance (Kapitel 3.7) definiert.

3.1 **EMCS**

> „Das EMCS (Excise Movement and Control System) ist ein EDV-System für die Überwachung der Beförderung verbrauchsteuerpflichtiger Waren unter Steueraussetzung innerhalb der EU."[19]

Innerhalb der Europäischen Union (EU) ist es durch dieses EDV-gestützte, papierlose Verfahren möglich, den Transport von verbrauchsteuerpflichtigen Waren (z.B. Alkohol, Tabakwaren oder Energieerzeugnisse[20]) unter Steueraussetzung online anzumelden. Nach automatisierten Prüfungen und Validierungen (1) (z.B. Rechte oder Steuernummern des Empfängers/Versenders) wird im Gutfall (2) eine PDF-Datei versendet (e-VD), die dem LKW-Fahrer in ausgedruckter Form mit auf den Weg (3) zu einem EU-Mitgliedstaat gegeben wird. Dieses Dokument ist der Nachweis der Erlaubnis für die Entnahme und den Transport von Gütern von Lager zu Lager innerhalb der EU ohne die Zahlung der Verbrauchsteuern, die im Normalfall bei der Lagerentnahme zu entrichten wären. Wird ein LKW, mit Zigaretten beladen, auf der Autobahn angehalten und der Fahrzeugführer kann weder ein e-VD

[19] [iEUK09].
[20] Vgl. [iEUK09].

noch die Zahlung der Verbrauchsteuern nachweisen, sind die Behörden aller Wahrscheinlichkeit nach einem Betrugsfall auf der Spur[21].

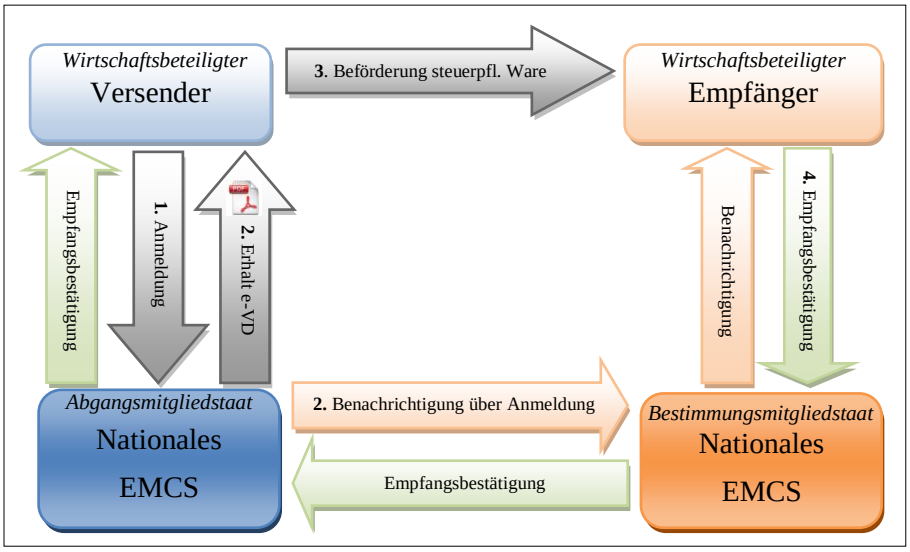

Abbildung 1: Warentransport nach EMCS-Anmeldung[22]

Kommt die Ware am Zielort an, versendet der Empfänger eine elektronische Mitteilung über den Erhalt der Ware (4) über sein nationales EMCS, welches die Nachricht an das EMCS des Versenders weiterleitet. Über frei wählbare Anbindungen (X.400, EDIFACT, FTAM[23] oder die Internet-EMCS-Anwendung) erhält der Empfänger einen Hinweis über den Eingang der Waren.

Jeder Mitgliedsstaat der EU ist verpflichtet, eine nationales EMCS zu installieren, das mittels spezifizierten XML-Nachrichten (s. EU-Vorgaben DDNEA[24]) mit den Systemen der anderen Mitgliedsstaaten kommuniziert. Die Kommunikation zwischen dem nationalen EMCS und den jeweiligen Wirtschaftbeteiligten erfolgt auf unterschiedlichen Kommunikationsstrecken. Stichworte wie X.400 oder FTAM sollen an dieser Stelle ausreichen, da die Verfahren im Detail Unterschiede aufweisen, die für den Teilnehmer relevanten Inhalte in

[21] Vgl. [iEUC10].
[22] Nach [iBMF10_1].
[23] Vgl. [iBMF10_2] (Kap. Zertifizierte Software).
[24] DDNEA abrufbar unter: http://ec.europa.eu/taxation_customs/taxation/excise_duties/circulation_control/emcs_computerisation_project/article_4403_en.htm.

digitaler Form aber identisch sind. Dem Teilnehmer stehen dafür kostenpflichtige Teilnehmersoftware oder die IEA zur Auswahl[25].

3.2 Das Teilprojekt IEA als Bestandteil des nationalen EMCS

Da die Teilnahme an diesem Verfahren verpflichtend ist, wurde von der EU gefordert, jedem Wirtschaftsbeteiligten eine kostenlose Zugangsmöglichkeit zu verschaffen. Das Bundesministerium für Finanzen hat sich deshalb entschieden, die Internet-EMCS-Anwendung (IEA) in Auftrag zu geben. Die Anwendung ermöglicht es allen Wirtschaftsbeteiligten, die verbrauchsteuerpflichtige Waren unter Steueraussetzung versenden oder empfangen, die hierzu vorgeschriebenen elektronischen Meldungen über das Internet zu versenden. Es wird hierzu ein Computer mit Internetzugang ohne spezielle Software benötigt[26].

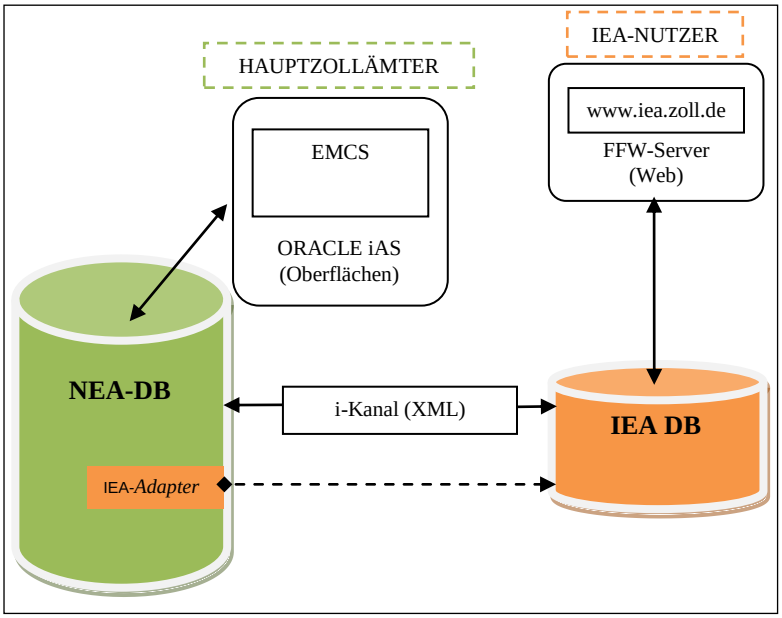

Abbildung 2: Anbindung der IEA an EMCS[27]

Abbildung 2 stellt den grundsätzlichen, schematischen Aufbau dar. Es handelt sich beim nationalen EMCS-System (NEA) um eine ORACLE©-Datenbank, die mittels ORACLE-Forms© administriert und den fachlichen Vorgaben entsprechend genutzt wird. Die EMCS-Oberflächen werden von den Mitarbei-

[25] Liste der Softwareanbieter abrufbar unter: http://www.zoll.de/DE/Fachthemen/Steuern/Verbrauchsteuern/EMCS/Teilnahme/Softwareanbieter/liste_der_softwareanbieter_release_2_0.html?nn=210052&view=render%5BStandard%5D.

[26] Vgl. [iBMF10_1].

[27] Nach interner Softwarespezifikation IEA (DokNr. 22075).

tern der Hauptzollämter (HZA) genutzt, um manuelle Prüfungen und Abgleiche durchführen zu können. Die Pfeilrichtung zeigt, dass XML-Dateien sowohl von der NEA zur IEA als auch in umgekehrter Richtung fließen. Bei der gestrichelten Linie handelt es sich um die Darstellung der turnusmäßigen Übertragung der Stammdaten von der NEA-DB zur IEA-DB. Das maßgebende System ist somit die NEA. Die Oberfläche wurde mit Hilfe der lizensierten Software FFW© (FormsForWeb) erzeugt. Mit dieser können Internetformulare erzeugt werden, die mit unterschiedlichen RDBMS-Systemen verknüpft werden können und die Datenzugriffe steuern. Bei der IEA und der NEA kommt die ORACLE©-Datenbank in der Version 11G R1 zum Einsatz.

3.3 **Softwareentwicklung der IEA aus Rollensicht**

Für eine erfolgreiche Softwareentwicklung sind angemessene Projektstrukturen erforderlich. Eine detaillierte Ausgestaltung ist ähnlich komplex wie die zu realisierende Softwarestruktur[28]. Abbildung 3 vermittelt daher in vereinfachter Form einen Eindruck über die Sichtweisen der wesentlichen Rollen auf die Umgebungen im Projektverlauf durch die der Lebenszyklus der Software maßgeblich beeinflusst und geprägt wird[29]. Innerhalb des Teilprojektes IEA sind für die fachliche Umsetzung die internen Rollen „Entwickler“, „Tester“ (Qualitätssicherung) und „Projektleiter“ (technische und fachliche aus Übersichtsgründen zusammengefasst) verantwortlich. Für den Lasttest und die Inbetriebnahme wird das Team durch Experten unterstützt (interne und externe Rollen). Sie stellen sicher, dass die Systemarchitektur die technischen Anforderungen der Software erfüllt. Durch einen Lasttest wird sichergestellt, dass die zu erwartende maximale Beanspruchung auch erfüllt werden kann. Im Wesentlichen wird die Software auf internen Systemen entwickelt und getestet. Sie besitzen aus Kostengründen eine viel niedrigere Kapazität als die Maschinen im Echtbetrieb. Bei Problemen stehen die Mitglieder der zuständigen technischen Abteilung (z.B. Datenbanken, Webserver) zur Verfügung („technische Experten“). Die Rolle „Entwickler“ realisiert die Funktionen in der „Entwicklungsumgebung“. In der Systemtestumgebung bescheinigt der „Tester“,

[28]Vgl. [Vig07] S.3 (Kap. 1).
[29]Vgl. [Ham07] S.154 (Kap. 6).

dass Funktionalitäten fehlerfrei ausgeführt werden können. Der „Auftraggeber“ ist für die Abnahme und die Prüfung zur Sicherstellung der Softwareanforderungen verantwortlich[30]. In der Echtbetriebsumgebung wird die Software dem Nutzer zur Verfügung gestellt.

Somit greift jede Rolle zu unterschiedlichen Zeitpunkten in voneinander unabhängigen Umgebungen auf die Software zu. Die Rollen „Tester“, „Kunde“ und „Projektleiter“ haben die Funktionalität der Software im Fokus. Durch den Lasttest steht zum Ende des Projektverlaufs die Performance im Blickpunkt. Diese nicht-funktionale Anforderung wird direkt von den „technischen Experten“ überprüft.

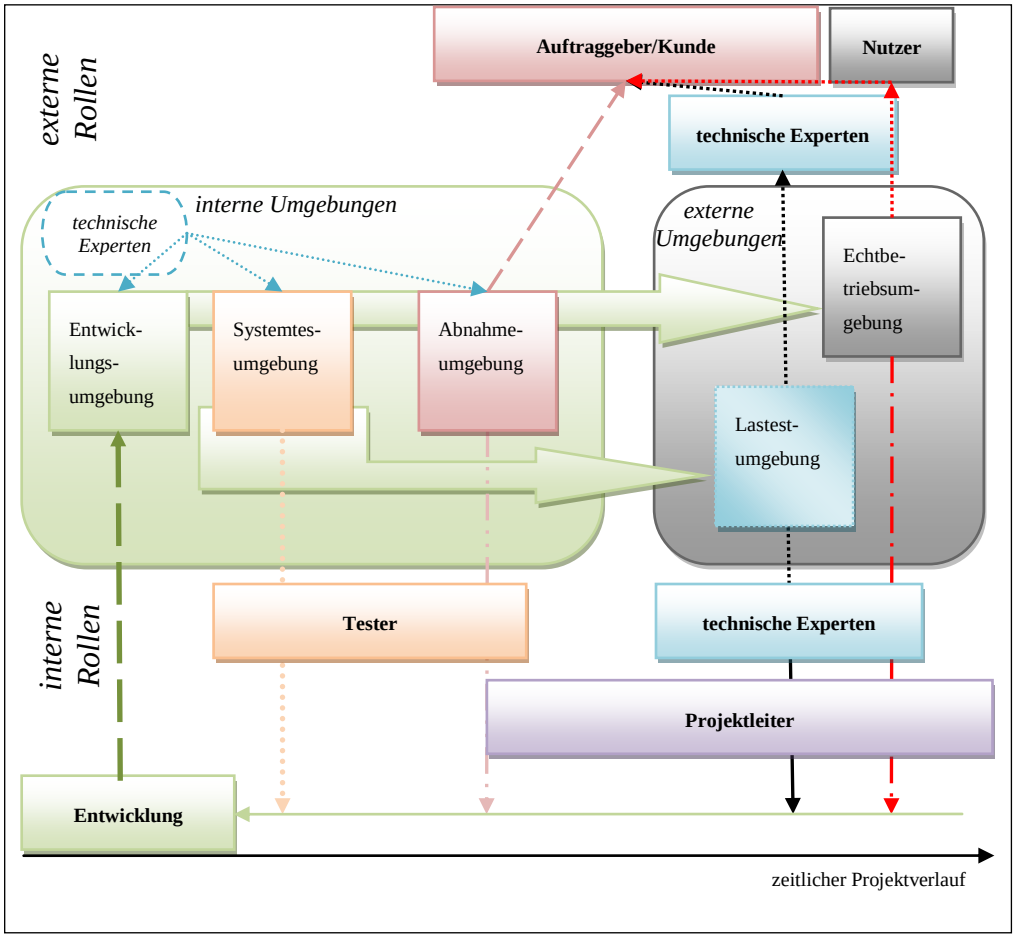

Abbildung 3: Softwareentwicklung aus Rollensicht[31]

Die Erkenntnisse werden an die „Entwickler“ übermittelt. Da die Lasttestumgebung nur eine begrenzte Zeit zur Verfügung steht, werden Anpassungen

[30] Vgl. [Bra08] S.19 (Kap. 1.2.6).
[31] Nach [Bra08] S.18 (Kap. 1.2.5).

nach einem Systemtest direkt in die Lasttestumgebung eingespielt. Dieser Vorgang wiederholt sich so oft, bis alle ineffizienten Algorithmen und Datenbankstrukturen durch die Entwicklung beseitigt sind und der Test erfolgreich absolviert werden kann. Werden Performance-Probleme vom Nutzer gemeldet, wird der Auftraggeber darüber informiert. Der „Projektleiter“ bespricht mit den „technischen Experten“ und den „Entwicklern“ mögliche Konsequenzen und Lösungsmöglichkeiten.

3.4 **Eigenschaften der Datenbankumgebungen**

Für die Realisierung des Teilprojektes IEA sind die internen Umgebungen für die Entwicklung, den Systemtest der QS und den Abnahmetest durch den Kunden im Einsatz. Sie werden nach gleichen Kriterien erzeugt und besitzen die gleichen technischen Merkmale. Ähnlich wird bei den extern gehosteten Umgebungen für den Lasttest und den Echtbetrieb vorgegangen. So wird das Risiko für Abweichungen minimiert und eine einheitliche Systempflege ermöglicht. Die Kapazität und das Leistungsvermögen der externen Umgebungen sind im Vergleich zu den internen um ein Vielfaches höher. Bis auf die Entwicklungsumgebung werden die restlichen Systeme auf Basis von Softwarepaketen aufgebaut. Dafür verantwortlich ist das Konfigurationsmanagement. Anders als bei Kopien werden durch einen Neuaufbau der Datenbanken zu Beginn leere Tabellenstrukturen erzeugt. Somit weisen die Tabellen der einzelnen Umgebungen unterschiedliche Füllstände auf. Sie sind abhängig von Anzahl der durchgeführten Testfälle.

Abfragen am 14.12.2011 ergaben, dass der Datenbestand in den Vorgangstabellen der IEA im Echtbetrieb bei ca. 200.000 Datensätzen lag. Diese Datenmengen wurden auch im Lasttest durch Automatisierungen erreicht[32]. Eine Auswertung des Lasttestdrehbuchs ergab, dass im Vergleich zur Spitzenlast im Echtbetrieb bei den Tests ein durchgängig doppelt so hoher Nachrichtendurchsatz pro Stunde (2.000) gewährleistet werden musste. Dieser wurde in Spezialtests noch übertroffen. Dies erklärt, warum der Füllstand der Tabellen in der

[32]Vgl. [Sch07] S.254f.

Lasttest-Datenbank nach nur 3 Wochen Einsatzzeit mit den Werten aus dem Echtbetrieb nach 180 Tagen vergleichbar ist.
In der Abnahmetestumgebung wurden 1.435 Datensätze ermittelt. Die Systemtestumgebung lieferte ein Ergebnis von 127 Datensätzen.

3.5 Zahlen und Statistiken

> „Kennzahlen beschreiben das IT-System mit charakteristischen Zahlwerten. Sie sind insbesondere eine wichtige Grundlage für die Architektur des Systems. Wenn mit einem Kundenverwaltungsprogramm 13 Millionen Kunden verwaltet werden sollen, ist das eine ganz andere Herausforderung, als wenn es 1.500 Kunden sind."[33]

Mithilfe von SQL-Abfragen wurden der IEA-DB Informationen über die Nutzeranzahl und die Aktivitäten entnommen. Der Zeitpunkt der Abfragen war der 19. Dezember 2011[34]. Im Folgenden wird der Begriff „Vorgang" verwendet. Er umschreibt den Austausch sämtlicher Nachrichten, die sich auf eine EMCS-Anmeldung beziehen. Jede EMCS-Anmeldung besteht, wenn sie beendet wurde, mindestens aus der Nachricht „e-VD" (Begleitdokument als positives Resultat des „Entwurf e-VDs") und der „Eingangsmeldung". Ein Vorgang ist abgeschlossen, wenn alle Wirtschaftsbeteiligten die Nachricht erhalten haben, dass die Ware am Zielort angekommen ist.

3.5.1 Einschränkungen durch automatisiertes Löschverfahren

Bei der Auswertung der Zahlen im Zusammenhang mit der IEA ist grundsätzlich zu beachten, dass Vorgänge nach Beendigung maximal 180 Tage gespeichert werden. Es handelt sich dabei um eine Anforderung, die im Gegensatz zur gängigen Datenbankpraxis das tatsächliche, physikalische Löschen verlangt. Alternativ könnten mit Hilfe eines Kennzeichens alte Datensätze nicht zur Anzeige gebracht werden. Dies ist aber keine fachliche Anforderung, da

[33] [Bra08] S.13 (Kap. 1.2).
[34] Die Auswertungen, die auf Bewegungsdaten basieren, variieren je nach Betrachtungszeitpunkt und Betrachtungszeitraum.

jeder steuerrechtlich relevante Austausch im nationalen EMCS unbefristet vorgehalten wird.

3.5.2 Gesamtanzahl der Nutzer

Es wurde zum Auswertungszeitpunkt eine Gesamtzahl von 6.796 unterschiedlichen Nutzern anhand ihrer Verbrauchsteuernummer ermittelt (IEA-Nutzer). Davon sind 1.157 in der Rolle „Versender“ (17%) und 5.639 (83%) als „Empfänger“ aktiv.

3.5.3 Erzeugte Vorgänge pro Arbeitstag

Die Mehrzahl der IEA-Nutzer (6.292) bearbeitete zum Auswertungszeitpunkt im Durchschnitt pro Arbeitstag (180 Tage, davon ca. 127 Arbeitstage) einen Vorgang. Zwei bis neun Vorgänge pro Arbeitstag wurden 467 IEA-Nutzern zugeordnet. Als Vielnutzer konnten 28 IEA-Nutzer mit durchschnittlich mehr als neun Vorgängen pro Arbeitstag ermittelt werden.

3.5.4 Vorgänge nach Empfangsergebnis

Abbildung 4 zeigt die Anzahl der Vorgänge pro Rolle unterschieden nach Empfangsergebnis.

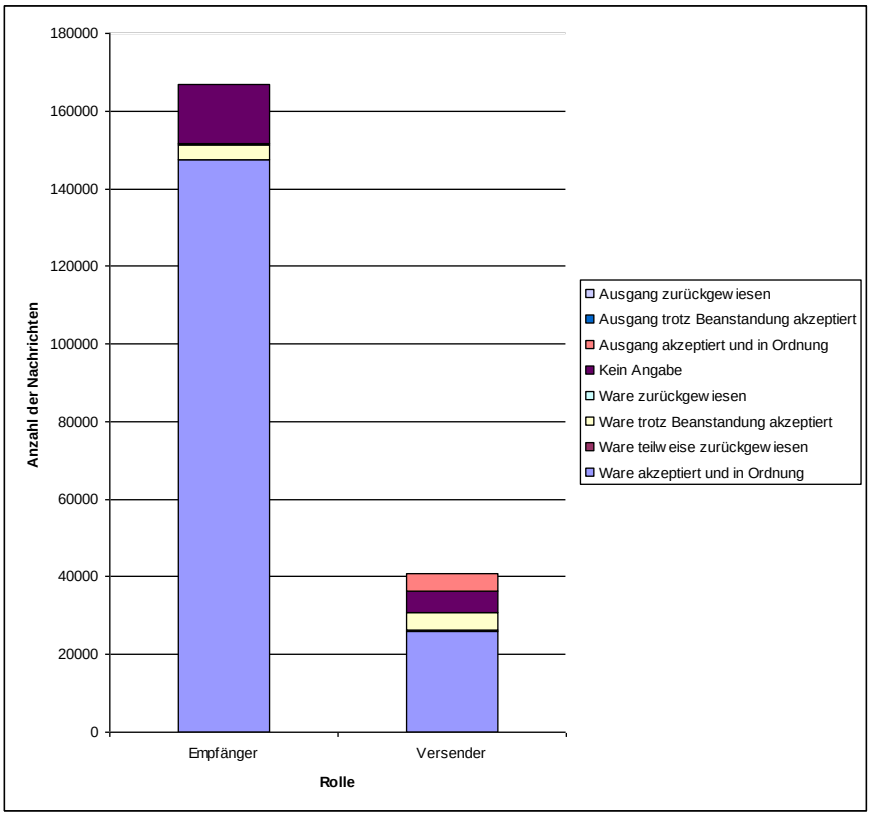

Abbildung 4: Vorgänge nach Empfangsergebnis

Im betrachteten Zeitraum wurden insgesamt 207.782 Vorgänge (Empfänger 166.989; Versender 40.793) mit Hilfe der IEA bearbeitet. Die Ware wurde 186.331 mal akzeptiert[35] (Empfänger 151.264; Versender 35.067). Bei 486 Vorgängen (0,23%) wurde die Ware zurückgewiesen bzw. teilweise zurückgewiesen (Empfänger 406; Versender 80). Vorgänge, zu denen es keine Angaben gibt, befinden sich im Status „akzeptiert“ oder im Status „Entwurf“. Die Ware war somit zum Auswertungszeitpunkt noch nicht am Zielort angekommen. Insgesamt waren somit 20.965 Vorgänge (Empfänger 15.319; Versender 5.646) nicht abgeschlossen.

3.5.5 Nachrichtendurchsatz pro Stunde

Abbildung 5 zeigt die Auslastung der IEA-Datenbank pro Stunde im Betrachtungszeitraum einer Woche (21.11.2011 - 27.11.2011). Gezählt wurden eingehende und ausgehende Nachrichten. Die Aktivitäten um 0.00 Uhr sind auf automatisiert von EMCS versendete Erinnerungsmeldungen zurückzuführen. Die Spitzenlast wurde am Dienstag, den 22.11.2011, mit 1.000 Nachrichten pro Stunde erreicht. Während der Kernarbeitszeit (Montag bis Freitag zwischen 8.00 Uhr und 17.00 Uhr) wurden im Durchschnitt 569 Nachrichten übermittelt.

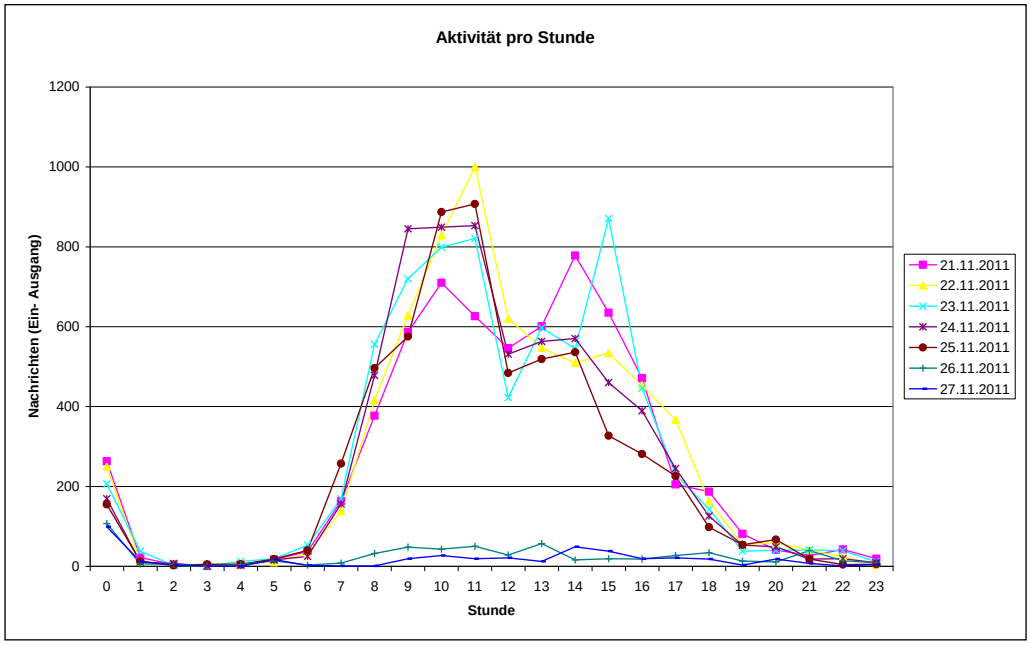

Abbildung 5: IEA-Nachrichtendurchsatz pro Stunde

[35] Dazu zählen auch trotz Beanstandungen akzeptierte Waren.

3.6 **Problemfelder**

Die Entwicklung des Teilprojektes IEA wurde für das Release 1.0.3 und das Release 2.0.2 erfolgreich abgeschlossen. Das Release IEA 1.0.3 war von April 2010 bis November 2011 im Echtbetrieb installiert. Die IEA 2.0.2 ist seit dem 3. Januar 2011 als Nachfolgerelease im Einsatz. Als Informationsgrundlage für die Problem-Analyse wurden folgende Teilbereiche ausgewertet:

Kapitel 3.6.1: Besprechungen mit hoher Priorität

Kapitel 3.6.2: IEA-Tickets aus dem Echtbetrieb

3.6.1 Besprechungen mit hoher Priorität

Die Terminanfragen im Teilprojekt IEA im Zeitraum von Januar 2009 bis Januar 2011 wurden ausgewertet. Es wurde zwischen kurzfristig einberufenen Terminen und langfristig geplanten Terminen unterschieden. Analysiert wurden nur Termine, bei denen auch die Projektleitungsebene mit involviert war. Von 16 als dringlich einzustufenden Terminen, wurden 13 Besprechungen für die Behebung von Performance-Problemen anberaumt. Bei sieben Terminen waren zudem Technikexperten des Auftraggebers telefonisch zugeschaltet.

3.6.2 IEA-Tickets aus dem Echtbetrieb

Bei der Auswertung wurden Problemfälle aus dem Echtbetrieb der IEA [Release 1.0.1] betrachtet (April 2010 bis November 2011). Insgesamt wurden 12 Tickets[36] erfasst, fünf der Fehlerklasse III (unkritisch), fünf der Fehlerklasse II (kritisch) und zwei der Fehlerklasse I (schwerwiegend). Diese zwei unterschiedlichen Problemfälle basierten auf Kundenaussagen, die ein viel zu langsames Verhalten der Software meldeten - in diesem Sinne also um Performance-Probleme. Wegen der kritischen Auswirkungen wurde beschlossen, diese Probleme durch Hotfixe zu beheben, die außerhalb der regulären Wartungsfenster lagen.

3.6.3 Einschränkungen bei der Erkennung von Performance-Problemen

Die Analyse der Statistiken aus dem Echtbetrieb hat gerade vor dem Hintergrund dieser Arbeit und der kritischen Auseinandersetzung mit dem Nutzerverhalten im Echtbetrieb zu folgender Erkenntnis geführt: Für tiefgründige

[36] *Bezeichnung für Problemfälle, die in Fehler- und Änderungsmanagementwerkzeugen festgehalten werden.*

Performance-Analysen und -Beobachtungen in Hinblick auf Spitzenlasten und Wachstumsverhalten, ist die IEA-Datenbank nur eingeschränkt tauglich. Dies liegt zum einen an der Tatsache, dass als IEA-Nutzer nur ein begrenzter Kreis an Personen in Frage kommt (s. Kapitel 3.5.2) und große Wachstumssprünge nicht zu erwarten sind. Zum anderen sorgt das automatisierte Löschverfahren (s. Kapitel 3.5.1) dafür, dass die Datenbanktabellen gleichbleibende Füllstände aufweisen. Deshalb ist ein stetig starker oder gar sprunghafter Anstieg nur bei einer fehlerhaft oder eingeschränkt implementierten Löschfunktion zu erwarten. Das mindert das Risiko für plötzlich auftretende Performance-Probleme[37]. Unabhängig davon sind aber die Füllstande der Datenbanken (s. Kapitel 3.4) ausreichend hoch, um die Auswirkungen ineffizienter Abfragen und Zugriffshilfen im Echtbetrieb anhand schlechter Antwortzeiten objektiv messen und subjektiv erkennen zu können. Dass ein Großteil der IEA-Nutzer durchschnittlich einmal am Tag Vorgänge bearbeitet (s. Kapitel 3.5.3), spielt deshalb nur eine untergeordnete Rolle. Dennoch sollte festgehalten werden, dass ein etwas geringerer Anspruch an das Reaktionsverhalten der Internet-Anwendung durch die Tatsache begünstigt werden könnte, dass es sich um eine kostenlose Software handelt (s. Kapitel 3.2). Gestützt wird diese These durch Beobachtungen während der Echtbetriebsaufnahme der IEA 2.0.2. Obwohl zahlreiche Neustarts der Webserver protokolliert wurden und interne Tests vielfach schlechtere Antwortzeiten als bei den Referenzwerten der IEA 1.0.3 nachweisen konnten, gab es keine großartigen Beschwerdemails oder sonstige Hinweise der Teilnehmer. Gerade deshalb wäre es mit Sicherheit reizvoll, die Vielnutzer, die mehr als neun Vorgänge pro Tag mit Hilfe der IEA bearbeiten, zu befragen[38]. Diese Erfahrungen könnten gerade in Hinblick auf die Implementierung einer komfortableren Bedienung nützlich sein. Doch aus Kosten-Nutzen-Sicht ist eine Befragung dieser Art nicht unbedingt vielversprechend, da 83% der IEA-Nutzer in der Rolle „Empfänger“ aktiv sind (s. Kapitel 3.5.2). Überdies werden mehr als 99% der Waren akzeptiert (s. Kapitel 3.5.4). Konkret bedeutet das für den Großteil der Nutzer, dass sie für die empfangenen e-VDs nur einen

[37]Vgl. [Frö08] S.7 (Kap. 2.1).
[38]Vgl. [Mil03] S.40 (Kap. 2).

Lesezugriff besitzen. Ein aktiver Zugriff auf drei bis vier Felder erfolgt bei der Erfassung der Eingangsmeldung. Bei insgesamt mehr als 60 zur Verfügung stehenden Feldern lässt sich festhalten, dass das gesamte Spektrum der Softwarefunktionalitäten nur in ganz wenigen Ausnahmefällen vollständig Verwendung findet und die Anwendung größtenteils passiv genutzt wird.

3.6.4 Ursachen für die Häufung von Performance-Problemen

Die in Kapitel 3.6.1 ermittelten Besprechungen zum Thema „Performance-Probleme in der IEA“ lassen sich aufgrund unterschiedlicher Gegebenheiten begründen. In erster Linie wurden sechs Termine im Zuge der Lasttests der IEA (Release 1.0.3 und 2.0.2) durchgeführt. Da der Lasttest in Zusammenarbeit mit den internen und externen Experten durchgeführt wurde, übernahm die Projektleitung die Moderation. Die Häufigkeit ist in diesen Fällen mit Verzögerungen in der Testdurchführung zu begründen. In der Mehrzahl der Fälle handelte es sich um ineffiziente Datenbankstrukturen und Algorithmen. Die Verzögerungen, die dadurch entstanden, lassen sich auf insgesamt sechs Tage beziffern. Die anderen sechs Termine wurden aufgrund von Schwierigkeiten beim Release-Wechsel anberaumt. Es wurde ein vom Lasttest abweichendes Antwortverhalten in der Echtbetriebsumgebung registriert. Als mögliche Ursache wurde eine eingeschränkte funktionale Lasttestabdeckung in Betracht gezogen. Sie wurde mit unzureichenden Testfällen oder technisch eingeschränkten Testautomaten begründet, die keinen Datenimport/ -export simulieren konnten. Als weitere Problemursache wurden mögliche Softwareanpassungen nach dem Lasttest vermutet. Diese Problematik verdeutlichen auch die Tickets aus dem Echtbetrieb (s. Kapitel 3.6.2), bei denen Bugfixeinspielungen Performance-Probleme verursachten. Als Sonderfall einzustufen war eine Besprechung hinsichtlich eines „Langläufers“ beim Einspielen eines Bugfixes. Es wurden durch die Softwareanpassung komplizierte Views durch redundante Tabellenstrukturen ersetzt. Bei der initialen Befüllung der Tabellen kam es aufgrund von ca. 1 Mio. Neuberechnungen zu erheblichen Verzögerungen, die aufgrund der niedrigen Füllstände in der Abnahmetestumgebung nicht aufgefallen waren.

3.7 **Handlungsfeld Performance**

Die Auswertungen der Problem-Analyse zeigen anhand der dringlichen Besprechungen und der Fehlertickets der Klasse I, dass Performance-Problemen im Teilprojekt IEA eine besondere Beachtung geschenkt wird. Es zeigt sich, dass im Idealfall jede Softwareanpassung erst den Lasttest bestehen müsste, bevor sie in den Echtbetrieb gelangt. Aus Kostengründen wird eine Lasttestumgebung aber nur einmal pro Releaseentwicklung betrieben. Somit müssen gerade nach dem Lasttest die Anstrengungen intensiviert werden, ineffiziente Algorithmen oder Tabellenstrukturen zu vermeiden. Hinzu kommen die Testverzögerungen zu Beginn der Lasttests. Es sollte sichergestellt werden, dass die Performance zu jedem Zeitpunkt der Softwareerstellung überprüft werden kann. Somit lassen sich Unsicherheiten minimieren und Kosten reduzieren. Außerdem kann die Gefahr eingeschränkt werden, dass Funktionen, die nicht Gegenstand des Lasttests sind, erst im Echtbetrieb als Performance-Schwachstellen identifiziert werden.

4 Lösungsmöglichkeiten

Auf Grundlage der IST-Analyse des Teilprojekts IEA (Kapitel 3), die den Problembereich Performance als Handlungsfeld herausstellt, werden in diesem Kapitel Lösungsmöglichkeiten für die Vermeidung ineffizienter Algorithmen (SQL-Abfragen) und Tabellenstrukturen beschrieben und bewertet. Auf der Suche nach Lösungsmöglichkeiten spielen die zur Verfügung stehenden Werkzeuge eine entscheidende Rolle. Kapitel 4.1 beschreibt einleitend in groben Zügen die Eigenschaften und die Nutzungsart der verwendeten Software. Anschließend werden die einzelnen Lösungsmöglichkeiten aufgeführt. In den Unterkapiteln werden die Vor- und Nachteile geschildert.

Kapitel 4.2: I: REST-Schnittstelle

Kapitel 4.3: II: Verstärkte Beachtung des Ausführungsplans

Kapitel 4.4: III: Automatische Datengenerierung

Kapitel 4.5: IV: Lasttestkopie als Performance-Test-Umgebung

Kapitel 4.6: V: Performance-Tests auf Echtbetriebsabzug

Zusammenfassend wird eine abschließende Bewertung der Lösungsmöglichkeiten durchgeführt (Kapitel 4.7).

4.1 **Werkzeuge**

> „Die Verwendung von Werkzeugen ist ein wesentliches Kennzeichen des ingenieurmäßigen Vorgehens, das in der Softwaretechnik angestrebt wird. Sie sollen die Effektivität und Effizienz von Entwicklern erhöhen.“[39]

Folgende Werkzeuge stehen im Teilprojekt IEA für die Datenbankentwicklung zur Verfügung:

- CASE/4/0©
- PL/SQL-Developer© (Version 9.0.1)
- ORACLE SQLDeveloper© (Version 3.0)

Bei CASE/4/0 handelt sich es sich um ein CASE-Werzeug (Computer-Aided Software Engineering). Mit Hilfe von CASE/4/0© werden die Relationen

[39][Bal09] S.59 (Kap. 6.1).

angelegt. Die Ausgabedateien beinhalten nicht nur einfache CREATE TABLE Statements, sondern liefern INSERTS für einen unternehmenseigenen, so genannten „Installer“, der die Fähigkeit besitzt, Tabellen und Indizes in Abhängigkeit von den bereits vorhandenen Objekten zu aktualisieren bzw. neu anzulegen. Auf Grundlage der Attribute der Relationen können Views erzeugt werden. PL/SQL-Code wird ebenfalls mittels Modulen aus CASE/4/0© erzeugt. Hier sorgen schablonenartige Vorgaben für den einheitlichen Einsatz von Ausnahmebehandlungen. Der PL/SQL-Developer© ist das Hauptentwicklungswerkzeug. Meistens werden zu erstellende Datenbankprozeduren und -funktionen mit diesem Werkzeug vorab kompiliert und getestet, bevor sie mit Hilfe von CASE/4/0© einheitlich erfasst werden. Der ORACLE SQLDeveloper© ist bei einzelnen Entwicklern im Einsatz. Hauptsächlich wird er auch für SQL-Abfragen und PL/SQL-Programmierung genutzt. Die Vorgehensweise ist dabei identisch wie beim PL/SQL-Developer©.

4.2 I: REST-Schnittstelle für Anweisungstests

Bei der Entwicklung stellt sich oftmals die Frage, wie die Laufzeit des formulierten SQL-Statements im Echtbetrieb bzw. auf Datenbanken mit realitätsnahen Tabellenfüllständen ist. Die Idee lautet eine Schnittstelle zu implementieren die per RESTful-Service[40] auf jeder beliebigen Datenbank Statements entgegennimmt und sie nachts ausführt. Die Antwortzeiten werden auf einem Repository-Server hinterlegt und ausgewertet. Dies soll möglichst automatisch geschehen und durch die Verwendung eines Art Monitoring-Dienstes dem Entwickler die Rückmeldung geben, falls seine formulierten Statements imperformant sind.

4.2.1 Vorteile

Durch dieses Verfahren könnte immer garantiert werden, dass ein ausführbares SQL-Statement im Echtbetrieb performant ist. Die Erfahrungen, die beim Aufbau einer solchen Schnittstelle gesammelt werden, können eine gute Grundlage für die Implementierung weiterer Webservices sein. Durch die

[40] *Hiermit ist das Programmierparadigma gemeint, dass mittels einer konkreten URL Anweisungen ausgeführt werden. Die Ergebnisse werden anschließend als Webseiten angezeigt. Vergleichbar mit dem Aufbau des Onlineshops amazon©.*

universellen Möglichkeiten bei der Verwendung des HTTP können einfache Zugriffe und Administrationsvorgänge über Webseiten erzeugt werden.

4.2.2 Nachteile

Es müsste sichergestellt werden, mit welchen Parametern die Statements ausgeführt werden. Beispielsweise macht es einen Unterschied, ob in einer Datenbank eines Mädchengymnasiums nach dem einzigen Mann gesucht wird oder nach einem ganz bestimmten Mädchen - die so genannte „Selektivität[41]„. So können bei dem gleichen Statement die Antwortzeiten je nach Parameter abweichen. Gerade die Benutzerdaten unterscheiden sich im Echtbetrieb immer von den Daten im Testbetrieb. Die Problematik liegt im Aufwand bei der Konfiguration repräsentativer Daten. Des Weiteren können Abfragen basierend auf neuen Attributen oder Objekten auf Datenbanken mit einem veralteten Softwarestand nicht ausgeführt werden. Insofern klingt dieser Lösungsansatz visionär und die zukünftige Nutzung einer REST-Schnittstelle über das Intranet ist für Auswertungen oder sonstige Prüfungen mit Sicherheit reizvoll, dennoch steht unter den Gesichtspunkten der Einfachheit und Effizienz die Lösung im Gegensatz zu den in Kapitel 2.3 formulierten Zielen.

4.3 II: Verstärkte Beachtung des Ausführungsplans

Im Zuge der Recherche für diese Arbeit gab der QS-Teamleiter im das Projekt EMCS am 08.12.2011 bei einer direkten Befragung die Antwort, dass jeder Entwickler einfach den Ausführungsplan seiner Statements prüfen müsse und daran erkenne, ob FULL-TABLE-Scans durchgeführt würden oder nicht. Da das Hauptentwicklungswerkzeug der PL/SQL-Developer© sei, könne sich jeder mit Hilfe der Funktionstaste F5 den Ausführungsplan der Abfrage anzeigen lassen.

4.3.1 Vorteile

Bei dieser Lösung trägt jeder Entwickler selbst die Verantwortung. Es gäbe keine Änderungen in der bisherigen Vorgehensweise. Eine Alternative könnte durchaus sein, nichts zu verändern, sondern nur die Entwickler verstärkt darum zu bitten, sich mit den Ausführungsplänen der Abfragen im Vorfeld auseinan-

[41] Vgl. [Ala09] S.1065f (Kap. 19).

der zu setzen. Dafür müsste nur der Hinweis gegeben werden, dass dies im PL/SQL-Developer© durch das drücken der Taste F5 problemlos möglich ist.

4.3.2 Nachteile

Ein Ausführungsplan gibt viele nützliche Hinweise, allerdings müssen diese interpretiert werden können. Bei dieser Vorgehensweise wird vorausgesetzt, dass alle Entwickler gewissenhaft jedes einzelne Statement prüfen und die richtigen Schlüsse aus den Hinweisen des Ausführungsplans ziehen. Außerdem stellt sich die Frage, ob Statements, die zum Entwicklungszeitpunkt nach einem optimalen Ausführungsplan ablaufen, auch nach Datenbankanpassungen noch effizient sind. Es wird nicht der Möglichkeit Rechnung getragen, dass sich auch verwendete Indizes oder implementierte Funktionen im Laufe der Zeit ändern können. Gerade in der Entwicklung handelt es sich um einen ständigen Prozess, bei dem das endgültige Resultat zu Beginn nicht eindeutig zu erkennen ist. Außerdem kommt es oftmals zu Anpassungen der Abfragen, um die Kompilierung des Paketes sicherzustellen. Hierbei steht eine schnelle Problembeseitigung im Vordergrund, was wiederum zu Fehlern führen kann. Diese Lösung vernachlässigt, dass trotz aller Hinweise ineffiziente Abfragen aus unterschiedlichen Gründen in der Datenbank implementiert sein können.

4.4 III: Automatische Datengenerierung

Es könnten Skripte automatisiert in der Entwicklungsumgebung ablaufen, die nach bestimmten Regeln die Tabellen automatisch füllen, da alle Informationen zu Datenbanktabellen und ihren Attributen dem Data-Dictionary entnommen werden können. Hierfür liefert die View `dba_tab_columns` alle notwendigen Informationen.

4.4.1 Vorteile

Schon während der Entwicklung von SQL-Abfragen könnten direkt Rückschlüsse auf das Laufzeitverhalten gezogen werden, da jedes Teststatement auf Tabellen ausgeführt wird, die einen gewissen Füllgrad besäßen. Auch bei den Systemtests über die Oberflächen könnten imperformante Abfragen unmittelbar erkannt werden.

4.4.2 Nachteile

Da es sich um ein relationales Datenmodell handelt, gibt es nur wenige Tabellen, die unabhängig sind. Somit liegen auf zahlreichen Tabellen Fremdschlüs-

sel-Zwangsbedingungen (ForeinKey-Constraints). Dadurch reichen einfache INSERTS für die untergeordneten Kind-Tabellen nicht mehr aus. Hierbei ist davon auszugehen, dass Spalten wie z.B. die Primärschlüssel-Spalte im Teilprojekt IEA sehr selten mit Hilfe eines Triggers automatisch gefüllt werden, sondern die Sequenzen explizit beim IINSERT aufgerufen werden. Daher gibt es eine Vielzahl an Konstellationen zu beachten. Diese automatisch zu erzeugen wäre in gewissen Grenzen möglich. Die Lösung projektübergreifend einzusetzen gestaltet sich allerdings schwierig. Hier könnte das CASE/4/0©-System nützlich sein, da INSERT-Anweisungen auf Grundlage der modellierten Relationen erstellt werden können. Die Schwierigkeit mit den Zwangsbedingungen bleibt allerdings bestehen. Deshalb scheint es bei der Verfolgung dieser Lösung am sinnvollsten, je nach Bedarf die INSERTS halbautomatisch zu generieren. Hier kann auch der ORACLE SQL Developer© hilfreich sein, bei dem alle Abfragearten per Drag und Drop der Tabelle in das SQL-Arbeitsblatt erzeugt werden können. Es stellt sich allerdings die Frage, ob diese Lösungsmöglichkeit wirklich den Grundsatz der Einfachheit verfolgt und in welchem Verhältnis der Aufwand für das Füllen der Tabellen zum Nutzen steht, da optimale Testdaten nicht aus wahllosen Buchstaben und Zahlen bestehen.

4.5 IV: Lasttestkopie als Performance-Test-Umgebung

Als Alternative zur Lösungsmöglichkeit III (s. Kapitel 4.4) könnten durch eine Kopie der Lasttestumgebung mit samt der automatisiert erzeugten Inhalte eine eine Art PTU (Performance-Test-Umgebung) dauerhaft vorgehalten werden.

4.5.1 Vorteile

Alle Oberflächen werden so konfiguriert, dass sie in den Entwicklungs- oder Systemtests auf eine Datenbank mit gefüllten Tabellen zugreifen. Hierbei könnte ein echtbetriebsnahes Verhalten simuliert werden.

4.5.2 Nachteile

Es ist erforderlich, dass der Softwarestand der PTU dem der Systemtestumgebungen entspricht. Dies kann nur erfolgen, wenn während des Lasttests sicher-

gestellt wird, dass Änderungen nur mittels Paketinstallationen[42] durchgeführt werden. Wegen des hohen Zeitdrucks und der vorherrschenden „try and error"-Vorgehensweise kann diese Vorgabe bislang nicht erfüllt werden. Die Möglichkeit, die Daten aus dem Lasttest mit Hilfe der Transportable Tablespaces (TTS) oder eines Exports zu übernehmen, zieht die Problematik nach sich, dass es nicht ausreicht, Daten zu transportieren, sondern dass auch Sequenzen übertragen und angepasst werden müssen. Dies wird im Projekt EMCS unter dem Stichwort „Migration" zusammengefasst. Somit ist eine einheitliche Vorgehensweise auch bekannt und wird bei Release-Wechseln erfolgreich praktiziert[43]. Jeder Transport von Daten ist allerdings nur problemlos möglich, wenn die Tabellenstrukturen der Quell- und der Zieldatenbanken deckungsgleich sind. In den anderen Konstellationen muss berücksichtigt werden, ob sich Spalten oder deren Bezeichnungen verändert haben, ob neue Zwangsbedingungen alte Testdatensätze eventuell als ungültig abweisen oder ob neue Tabellen hinzugekommen sind. Weitergehend gibt es das Problem, dass gefüllte Tabellen weiterhin erst nach dem Lastest vorhanden sind. Das steht im Widerspruch zum Ziel, das Thema „Performance" stärker in den Fokus der frühen Entwicklungsphasen zu rücken.

4.6 **V: Performance-Tests auf Echtbetriebsabzügen**

In Anlehnung an Lösungsmöglichkeit IV gibt es die Option Performance-Tests auf Kopien der Echtbetriebs-Datenbanken durchzuführen.

4.6.1 Vorteile

Es könnten Langläufer-Installationen (s. Kapitel 3.6.4) im Vorfeld erkannt werden und die Vorgehensweise entsprechend angepasst werden (z.B. durch eine nächtliche Initialisierung etc.). Auch neue Funktionen könnten echtbetriebsnah getestet werden. Durch die hohen Tabellenfüllstände fielen ineffiziente Algorithmen direkt bei den Oberflächentests auf.

---

[42] *Der Begriff „Paketinstallation" beschreibt einen Installationsvorgang, bei dem Anweisungen im Dateiformat in die Datenbank nach einer festgelegten Reihenfolge eingespielt werden. Mit Hilfe eines unveränderbaren „Daten-Paketes" kann sichergestellt werden, dass nach der jeweiligen Installation ein klar abgrenzbarer Softwarestand erreicht wird.*

[43] Aufgrund der zeitlichen Begrenzungen kann dieses Thema in dieser Arbeit nicht vertieft werden, dennoch stellt sich die Frage, ob diese Kenntnisse mit einfachen Mitteln für Performance-Tests zur Anwendung kommen können.

4.6.2 Nachteile

Diese Vorgehensweise ist nur geeignet für Systeme, die bereits länger im Betrieb sind und über gefüllte Tabellen verfügen. Außerdem ist ein ständiger Austausch mit dem Echtbetrieb erforderlich, um die Datenbank auf dem neuesten Stand zu halten. Diese Lösung kann deshalb erst NACH Echtbetriebsbeginn verfolgt werden.

4.7 Gesamtbewertung der Lösungsmöglichkeiten

Bei näherer Betrachtung der aufgeführten Lösungsmöglichkeiten (Kapitel 4.2-4.5) ist erkennbar, dass keine Idee als die mit Abstand beste Lösung angesehen werden kann. Die Gründe sind vielfältig. So scheint bei Lösungsmöglichkeit I (s. Kapitel 4.2) der Aufwand unverhältnismäßig hoch dafür, dass Abfragen basierend auf strukturellen Änderungen gar nicht getestet werden können. Lösungsmöglichkeit II (s. Kapitel 4.3) kann hingegen als problemlos umsetzbar eingestuft werden. Die Alternative, die Entwickler stärker in die Verantwortung zu ziehen, indem verstärkt auf die Betrachtung der Ausführungspläne hingewiesen wird, zeugt zwar nicht von Aktionismus, dennoch darf auch die Option im Prinzip „nichts zu verändern“ nicht pauschal ausgeschlossen werden. Doch dieses Vorgehen schafft keine größere Sicherheit bei Bugfix-Einspielungen. Und die Problematik der „Langläufer“-Besprechung zeigt (s. Kapitel 3.6.4), dass auch günstige Ausführungspläne nicht immer ein Indiz für die Vermeidung von Performance-Problemen sein müssen. Wäre bei den Entwicklungstests bereits aufgefallen, dass bei der Neuberechnung hinsichtlich der Tabellenrestrukturierung längere Laufzeiten zu erwarten waren, hätten im Vorfeld Überlegungen für eine Strategieoptimierung stattfinden können. Diesem Thema widmen sich die Lösungsmöglichkeiten III - V (s. Kapitel 4.4 - 4.6). Auch wenn sie an unterschiedliche Umgebungen anknüpfen, sind die übergeordneten Ziele, die Erhöhung der Tabellenfüllstände und echtbetriebsähnliche Testumgebungen. Hierbei ist das Hauptproblem der Zeitpunkt, ab wann die Umgebungen zur Verfügung stehen. Bei dem Ansatz der automatischen Datengenerierung stellt sich als größte Schwierigkeit der zu erwartende Aufwand dar. Somit erweist sich keine Lösung als universell einsetzbar. Dies führt zu der Erkenntnis, dass aufgrund der Vielzahl an möglichen Ursachen differenzierte Lösungen für die Problemvermeidung erforderlich sind.

5 Die Rolle des Performance-Betreuers als Lösungskonzept

In diesem Hauptkapitel wird das Konzept für die Einrichtung eines Performance-Betreuers als neue teaminterne Rolle vorgestellt. In Kapitel 5.1 wird die Idee vorgestellt. Anschließend werden die Anforderungen zusammengetragen (Kapitel 5.2 - Kapitel 5.3). Auf Basis der Anforderungen an die Organisation werden unterschiedliche Betreuungsphasen definiert, denen Aufgaben und erforderliche Kompetenzen zugeordnet werden. Dieses Unterkapitel dient als Leitfaden für einen Aufgabenkatalog für die Rolle des Performance-Betreuers.

Kapitel 5.4: Einstiegsphase - Informieren und Begleiten

Kapitel 5.5: Überwachungsphase - Kontrollieren und Prüfen

Kapitel 5.6: Revisionsphase - Optimieren und Experimentieren

5.1 **Konzeptidee**

Die Erfahrungen im Projekt EMCS zeigen, dass der Entwicklung viele grundsätzliche Entscheidungen im Datenbankumfeld bereits abgenommen werden wie die Verwendung des Tabellentyps, der Tablespaces oder der Indexarten. Das liegt zum einen daran, dass das Team „Datenbanken“ für die Installation der Umgebungen verantwortlich ist und zum anderen viele Rahmenbedingungen wie die Ausnahmebehandlung oder das Tracing (Protokollierung der Ausführungen in einer unternehmensintern einheitlichen Tabelle) durch den Einsatz von CASE/4/0© bzw. durch globale Pakete vorgegeben wird. Somit profitiert das Entwicklungsteam von der langjährigen Erfahrungen und Vorarbeiten der Abteilung. Das Realisierungsteam kann gerade in Hinsicht auf Performance-Probleme als interner Kunden des technischen Teams „Datenbanken“ gesehen werden. In den Entwicklungs-, System- und Abnahmentestphasen stehen Ansprechpartner für alle festgestellten Datenbankprobleme zur Verfügung. Diese Personen sind größtenteils dieselben Experten wie im Lasttest (s. Kapitel 3.3). Der einzige Unterschied ist, dass der Austausch während der Entwicklungsphasen bilateral erfolgt und eine durch die niedrige Dringlichkeitsstufe die Projektleitung nicht involviert ist. Es sollte mit Hilfe der Rolle des Performance-Betreuers eine teaminterne Instanz geschaffen werden, die als erster Ansprechpartner für alle Themen in diesem Umfeld zur Verfügung steht.

Teaminterne Koordination von Performance-Problemen

Der direkte Austausch der Entwicklung mit den Datenbankexperten führt zu Informationsverlusten innerhalb des Teams. Durch eine zentrale Koordination könnten wiederkehrende Probleme erkannt werden und der Aufbau teaminternen Wissens gefördert werden. Dies würde die Expertenteams entlasten und zu schnelleren Antworten interner Problemstellungen führen. Dies könnte zum einen die Abhängigkeit von den technischen Teams reduzieren und zum anderen zu projektspezifischen Lösungen führen.

Hohe Motivation für Informationstransfer

Da es sich bei der Performance-Betreuung um eine Nebentätigkeit handeln würde, ist die Motivation der ausführenden Person groß, die Nachfragen des Entwicklers durch Hilfestellungen so zu beantworten, dass zukünftige Anfragen vermieden werden können. Dadurch wird das Verantwortungsbewusstsein für eine verstärkte Betrachtung der Performance innerhalb des Projektes gestärkt.

Fachspezifische Wissensgrundlage

Die Person, die diese Rolle ausführt, erlangt im Themenfeld Performance einen Wissenszuwachs hinsichtlich eines allgemeinen Datenbankverständnisses. Tuning-Maßnahmen (Optimierungen hinsichtlich einer Verbesserung der Datenbank-Performance) könnten auf Basis einer größeren fachlichen Wissensgrundlage anwendungsspezifisch ausgestaltet werden. Somit wäre das Team stärker entlang der Geschäftsfunktionalitäten strukturiert und einer selbst-organisierenden, eigenverantwortlichen Arbeitsweise Rechnung getragen. Bei auftretenden Problemen können Schuldzuweisungen zwischen den technischen und fachlichen Teams gar nicht erst auftreten[44].

5.2 **Anforderungen an die Organisation**

Aus dem in Kapitel 4 vorgestellten Pool an Lösungsmöglichkeiten wird durch die Installation eines Performance-Betreuers nicht auf eine einzelne zurückgegriffen, sondern eine neue Rolle innerhalb des Projektteams geschaffen, die mehrere Lösungsmöglichkeiten kennen und über ein gewisses Repertoire an

[44]Vgl. [Bal09] S.92f (Kap. Agile Softwareentwicklung in großen Projekten).

Werkzeugen verfügen sollte. Es handelt sich somit um eine organisatorische Anpassung der bisherigen Vorgehensweise. Deshalb muss klar definiert werden, welchen Aufgaben der Inhaber dieser Rolle nachzugehen hat. Sie sollten angemessen schwer und herausfordernd sein. Es sollte Möglichkeiten für Wiederholungen und Korrekturen von Fehlern geben[45]. Im Idealfall können die Vorgaben so gestaltet werden, dass es einem Anfänger leicht fällt, diese Rolle direkt auszufüllen und anschließend auf Grundlage einer wachsenden Informationsbasis eigene kreative Ideen und Umsetzungspläne zu verfolgen. Der Mehraufwand für die Ausübung der Tätigkeiten muss berücksichtigt werden. Außerdem muss eine längerfristige Betreuung sichergestellt werden, um von diesem Wissen profitieren zu können. Des Weiteren ist eine Gewöhnungsphase erforderlich. Eine Prozessanpassung muss von den Mitarbeitern akzeptiert und anschließend praktiziert werden.

5.3 **Anforderung an die Person**

Die Aufgaben des Performance-Betreuers kann von einem Mitglied der Entwicklung oder einem Mitglied der Qualitätssicherung mit Datenbankgrundkenntnissen wahrgenommen werden. Da der Fokus auf der Optimierung der Datenbankentwicklung liegt, kann ein Performance-Betreuer aus dem Realisierungsteam von den erlernten Kenntnissen und Fähigkeiten in der täglichen Arbeit profitieren. Für die Mitglieder der Qualitätssicherung wären die Erkenntnisse nicht direkt nutzbar. In den frühen Entwicklungsphasen wären sie dennoch als Ansprechpartner gut geeignet, da sie nicht unter dem Zeitdruck stehen, selbst Softwareanteile fertigzustellen[46]. Wichtig ist ein Interesse für das Themengebiet „Performance“, das durch den Austausch mit den Datenbankexperten und mit Hilfe von Fachliteratur erarbeitet werden muss. Außerdem sollte die Person bereit sein, Wissen zu vermitteln und die Teammitglieder für das Themengebiet „Performance“ zu sensibilisieren. Dies erfordert ein gewisses Maß an Kommunikationsfähigkeiten. Diesen Kriterien untergeordnet ist die Fähigkeit wichtige von den unwichtigen Themen unterscheiden zu können[47].

[45]Vgl. [Hun09] S.28 (Kap. 2).
[46]Vgl. [Gad08] S.1ff (Kap. 4).
[47]Vgl. [Mil03] S.22 (Kap. 1).

„Eine Abfrage um den Faktor 2 zu optimieren, die 1% der Laufzeit beansprucht, macht das System maximal 0,5% schneller."[48]

5.4 Einstiegsphase - Informieren und Begleiten

„In den frühen Projektphasen wird der Grundstein für die Qualität gelegt."[49]

Wer sich mit Performance beschäftigt, muss sich zu Beginn mit Zugriffshilfen beschäftigen - besser bekannt als Indizes. In der Literatur wird darauf hingewiesen, dass Performanceprobleme meistens aufgrund fehlender oder unzureichender Indizes entstehen, die FULL-TABLE-Scans nach sich ziehen. Diese These lässt sich durch Erfahrungen in der Praxis bestätigen. Es darf daraus aber nicht die Schlussfolgerung gezogen werden, alle FULL-TABLE-Scans zu eliminieren führe automatisch zu einer besseren Performance. Es muss beachtet werden, dass zu viele Indizes gleichzeitig auch einen zusätzlichen Zeitaufwand bedeuten, da beim Füllen und Aktualisieren der Haupttabellen die Indextabellen ebenfalls gepflegt werden müssen. Insofern ist eine fachliche Abschätzung unablässig[50].

[48][Sha92] S.2 (Kap. 1.2.1).
[49][Sto07] S.316 (Kap. 14).
[50]Vgl. [Mil03] S.35 (Kap. 2).

Der Performance-Betreuer sollte sich anfangs bewusst machen, welche Indizes durch die von CASE/4/0© generierten Dateien automatisiert angelegt werden und welche nicht. Dabei fällt auf, dass das Verfahren vom bekannten CREATE TABLE-Statement zwar abweicht, das Resultat aber dasselbe ist. Es werden automatisch Indizes für Primärschlüssel-Attribute und für Unique-Constraints angelegt[51]. Somit wird klar, dass Hilfswerkzeug CASE/4/0© zwar für eindeutige Namenskonventionen sorgt (`IDX_PK_REF_IEA_BAZ`, `IDX_PK_REF_IEA_BAZ` etc.), aber jede Spalte vom Entwickler dahingehend hinterfragt werden muss, ob es Abfragen nach diesen Kriterien geben könnte. Beispiele sind in der Vorgangstabelle (`TBL_IEA_Vorgang`) in Abbildung 6 der Bearbeitungszustand (`IEA_BAZ`) oder die Identifikationsnummer (`user_id`).

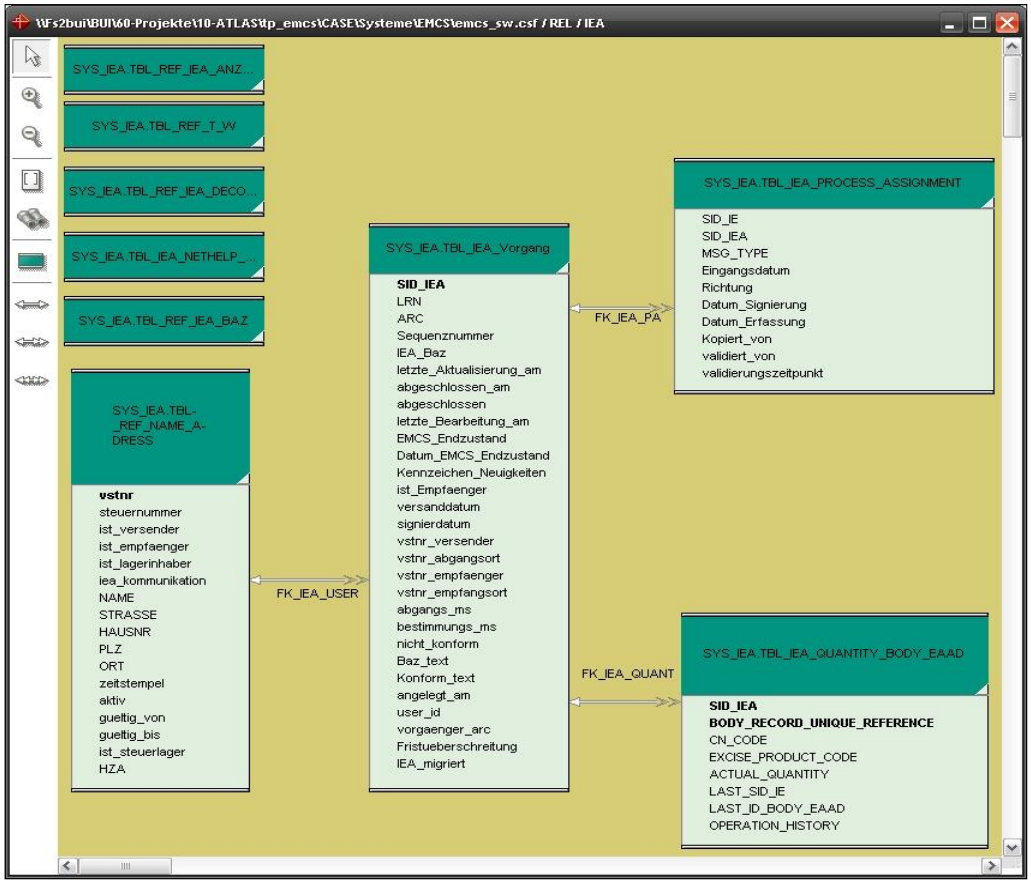

Abbildung 6: Screenshot CASE/4/0© IEA-Relationen

Für diese Überlegungen ist zwingend erforderlich, die Fähigkeiten für die Bedienung des CASE-Hilfswerkzeuges zu erlangen. Es hat sich im Projekt

[51] Vgl. [iORA05].

EMCS herausgestellt, dass durch das anfänglich unbekannte Werkzeug die Mitglieder der Entwicklung besonders mit der Bedienung und den zahlreichen Installationsversuchen der Ausgabedateien beschäftigt waren und dadurch konzeptionelle Überlegungen in den Hintergrund traten. Deswegen reicht es nicht aus, sich mit den optimalen Zugriffshilfen zu beschäftigen, sondern diese sollten auch in CASE/4/0© korrekt erzeugt werden können. Wer es gewohnt ist, Tabellen direkt in ORACLE© anzulegen, der stellt die Nützlichkeit dieses Hilfswerkzeuges schnell in Frage, da der direkte Vorteil oftmals nicht zu erkennen ist und das Werkzeug als veraltet und nicht zeitgemäß wahrgenommen wird. Im Vorfeld sollten die ersten Gehversuche begleitet und unmittelbar auch in Hinblick auf die Performance geprüft werden. Je eher die Entwickler die Sicherheit gewinnen, schnell und problemlos Indizes anzulegen, desto leichter fällt der Übergang von der funktionalen zur performanten Datenbankentwicklung. Somit könnte der Begriff Performance-Betreuer auch in Hinblick auf die Leistungsfähigkeit jedes einzelnen Entwicklers zum Tragen kommen. Folgendes wären die Primärziele eines Performance-Betreuers in der ersten Stufe:

- *Für Akzeptanz des Hilfswerkzeuges CASE/4/0 in der Entwicklung sorgen: Entscheidende Vorteile des einheitlichen Installers (z.B. bei der Aktualisierung von Spalten etc.) herausstellen.*
- *Die Vorgehensweise erklären und konkrete Problemstellungen begleiten: Die Relationen im Vorfeld auf sinnvolle Indizes prüfen und dem jeweiligen Entwickler eine Rückmeldung geben.*

Eine intensive Begleitung der Designphase könnte durch das Vorgehen nach dem „Vier-Augen-Prinzip“ erreicht werden. Dennoch kann es fehlende oder unzureichende Indizes durch später folgende Anpassungen nicht verhindern. Deshalb sollte in erster Linie sichergestellt werden, dass die Entwickler selbst in der Lage sind, FULL-TABLE-Scans zu erkennen. Den Ausführungsplan einer jeden Abfrage kann mit Hilfe des PL/SQL-Developers© durch Drücken der Taste F5 anzeigt werden. Es kann von Vorteil sein, zu Beginn von Entwicklungsphasen grundsätzliches Wissen abzufragen und fehlende Informationen zu vermitteln. Was für einen langjährigen Datenbankentwickler selbstverständlich ist, kann für einen Entwickler aus dem Web-Umfeld nach einer

einzigen absolvierten PL/SQL-Schulung ein hilfreicher Hinweis sein, der von Anfang an eine optimierte Arbeitsweise ermöglicht.
Zusammengefasst sollten zu Beginn Kompetenzen in folgenden Bereichen aufgebaut werden[52]:

CASE-Werkzeuge:
Vorgehensweise für das Anlegen von Tabellenstrukturen.

Indizes:
Eine gute Performance kann nur mit Hilfe angemessener Indizes erreicht werden[53].

Nutzung von Indizes:
Es sollte darüber informiert werden, in welchen Fällen Indizes nicht greifen.
Dies gilt es u.a. zu beachten:

- *Keine Ungleich oder Not-In-Bedingungen*
- *Wertebereiche nur mit (R%)*
- *Funktionsaufrufe erfordern FUNCTION BASED-Indizes[54]*
- *Vermeidung von DISTINCT und UNION[55]*

Ausführungspläne:
Es sollte erklärt werden, wie Ausführungspläne zu interpretieren sind. Dazu gehören Merkmale wie FULL-TABLE-Scans oder Sortiervorgänge.

5.5 Überwachungsphase - Kontrollieren und Prüfen[56]

„Manchmal kann im Vorfeld nicht alles bekannt sein."[57]

Da die Datenbank einem permanenten Veränderungs- und Anpassungsprozess unterliegt und das hauptsächlich in der Entwicklungsphase, reicht es nicht aus, in der Designphase der Datenbanktabellen und -attribute für ausreichend Indizes zu sorgen. Es kann Spalten geben, die anfangs noch nicht als Abfragekrite-

[52]Hierbei handelt es sich um eine Auswahl, die je nach Bedarf vervollständigt werden sollte.
[53]Vgl. [Sha92] S.53 (Kap. 3.1).
[54]Vgl. [Lon00] S.76f (Kap. 3.8).
[55]Vgl. [Sha92] S.111f (Kap. 4.6).
[56]Diese Phase ist Gegenstand der Projektarbeit, in der Überwachungsmöglichkeiten in der Praxis beschrieben werden.
[57][iMil08] S.1.

rium identifiziert wurden. So wird beispielsweise in der Entwicklungsphase festgestellt, dass eine Funktion kein eindeutiges Ergebnis zurückliefert, da die Abfrage zu wenig eingegrenzt ist. Anschließend wird auf die Schnelle eine zusätzliche Bedingung eingebaut und die Funktion liefert das erwartete Resultat. Somit ist der „Entwicklertest“ bestanden. Die Lehre von der Softwareentwicklung bestätigt dabei die Erfahrung, dass Fehler niemals vollständig vermieden werden können[58]. Deshalb stellt sich die Frage, wie diese Fehler schon vor dem Lasttest oder dem Echtbetrieb aufgespürt werden können. ORACLE© stellt mit der Version 11G Werkzeuge zur Verfügung, die ausreichende Überwachungsmöglichkeiten bieten. Und da es meistens kostengünstiger ist, Arbeiten vom Mensch auf die Maschine zu übertragen[59], sollte in erster Linie geprüft werden, wie die so genannten ORACLE©-Funktionen am sinnvollsten zum Einsatz kommen können und Informationen bestmöglich ausgewertet werden können. Denn Folgendes lässt bereits die Liste der neuen Funktionen der ORACLE©11G Version auf den ersten Blick erkennen: Auch ORACLE© nimmt sich dem Thema Performance mit dem Ziel an, Schwachstellen nicht nur aufzudecken, sondern gleichzeitig auch Lösungsmöglichkeiten anzubieten. Dabei ist es erforderlich die Vorschläge immer zu hinterfragen, da der SQL Performance Analyser© (SPA) menschliche Intelligenz nicht ersetzen kann[60]. Dennoch kann er Hinweise für weitere Nachforschungen liefern[61].

Unabhängig von der Verwendung des ORACLE Adivsors© sind für das Verständnis von SQL-Anweisungen folgende Kriterien entscheidend. Sie liefern Ansatzpunkte für Kontroll- und Prüfmechanismen[62]:

Phasen einer SQL-Anweisung

Wie wird eine SQL-Anweisung verarbeitet?

Histogramme und Statistiken

Welche Rolle spielen die Statistikinformationen in der ORACLE©-Tabellen und welchen Nutzen können aus Ihnen gezogen werden?

---

[58]Vgl. [Bra08] S.22 (Kap. 1.2.7).
[59]Vgl. [Lon00] S.22 (Kap. 2.3.1).
[60]Vgl. [Ala07] S.168 (Kap. 4).
[61]Vgl. [Bal09] S.69 (Kap. 6.2).
[62]Hierbei handelt es sich um eine Auswahl, die je nach Bedarf vervollständigt werden sollte.

Cursor und Bind-Variablen

Welche Auswirkungen hat die Verwendung unterschiedlicher Bind-Variablen?

Optimizer

Was leistet der kostenbasierte Optimizer? Was sind seine Vorteile und wie kann dieser durch erweiterte Statistiken noch besser an die Anwendung ausgerichtet werden?

AWR - Automatic Workload Repository[63]

Wie können die Informationen aus dem Workload Repository verwertet werden? Inwiefern kann die Database-Replay©-Funktion nützliche als Testszenario dienen?

Tracing[64]

Wie können Informationen aus dem Datenbank-Tracing entnommen und bewertet werden?

5.6 **Revisionsphase - Optimieren und Experimentieren**

Revisionsphasen können zu jedem beliebigen Zeitpunkt erfolgen. Es sollten ausreichend Informationen für die Auswertungen vorliegen. Deshalb bietet es sich an, Meilensteine wie Softwareabgaben als Anlass zu nehmen, den Blick von konkreten Problemen auf allgemeine Fragestellung zu lenken. Anhand des gesammelten Wissens kann die Vorgehensweise hinterfragt und im besten Fall optimiert werden. Dafür bietet es sich an, folgende Fragestellungen zu beantworten[65]:

- Welche Rückmeldungen der Entwicklung gibt es hinsichtlich der Rolle des Performance-Betreuers?
- Werden die Hilfestellungen angenommen?
- Wie funktioniert die Abstimmung mit den Datenbankexperten?
- An welchen Stellen wird Performance-Wissen gesammelt?
- Wie werden die Informationen zur Verfügung gestellt?

---

[63] *Informationen über die Servicezeiten und Wartezeiten, die in ORACLE©-Tabellen gesammelt werden.*

[64] *Spezielle Methode zum Sammeln von Ausführungsinformationen. Bei Datenbanken werden u.a. die SQL-Anweisungen im Textformat protokolliert samt ihrer Häufigkeit und Ausführungszeiten.*

[65] Hierbei handelt es sich um eine Auswahl, die je nach Bedarf vervollständigt werden sollte.

- Auf welche Art und Weise werden Informationen weitergegeben?
- Kann das Arbeitspensum des Performance-Betreuers mit den sonstigen Projekttätigkeiten vereinbart werden?
- Welche Performance-Probleme wurden am häufigsten festgestellt und was sind mögliche Ursachen?
- Gibt es neue Funktionalitäten in den Hilfswerkzeugen, die bislang nicht genutzt werden?
- Hat sich die ORACLE©-Version geändert? Falls ja, welche neuen Funktionalitäten können die Arbeit des Performance-Betreuers unterstützen?

Vor dem Hintergrund dieser Fragen wird an dieser Stelle festgehalten, dass auch die Rolle des Performance-Betreuers sich ständig den Gegebenheiten anpassen sollte. Es können im Laufe der Zeit Veränderungen der internen Rahmenbedingung auftreten. Dazu zählt die Teamstruktur, die sich durch Zugänge oder Abgänge verändern kann, oder eine Verschiebung der Aufgabenprioritäten. Es ist auch möglich, dass die Maßnahmen der Einstiegsphase (s. Kapitel 5.4) die Ergebnisse der Überwachungsphase insofern beeinflussen, dass es ineffektiv ist, regelmäßige Kontroll-Abfragen durchzuführen. In diesem Fall können Forschungen und Experimente weiter vorangetrieben werden. Diese sollten mit einbeziehen, dass sich die externen Rahmenbedingungen auch ständig ändern können. Dazu zählen die Datenbankfunktionalitäten und die Hilfswerkzeuge.

6 Fazit

Die Untersuchungen im Rahmen der Bachelorarbeit „Analyse und Realisierung von Optimierungsmöglichkeiten im Rahmen der Datenbankprogrammierung unter Berücksichtigung von Performance-Kriterien“ wurden innerhalb der organisatorischen und technischen Vorgaben des Teilprojektes IEA durchgeführt. Sie zeigten, dass es lohnenswert ist, trotz einer relativ niedrigen Fehleranfälligkeit der Software die Vorgehensweise in der Datenbankentwicklung rückblickend zu hinterfragen und Handlungsfelder für Optimierungsmöglichkeiten zu benennen. Anhand der nicht-funktionalen Anforderung Performance konnte aufgedeckt werden, dass vielschichtige Abläufe und Verantwortlichkeiten die Problematik nach sich ziehen, dass gewisse Bereiche aus dem Blickfeld der einzelnen Akteure innerhalb des Projektteams verschwinden können. Zu begründen ist es in diesem Fall unter anderem durch Testumgebungen, die einer anderen Last ausgesetzt sind als der Echtbetrieb. Einfache Oberflächentests reichen deshalb für die Ermittlung von imperformanten Datenbankstrukturen im Regelfall nicht aus. Somit ist ein erfolgreich absolvierter Lasttest unabdingbar für die Sicherstellung einer echtbetriebstauglichen Software. Da dieser spezielle Test nur einmal in der Projektentwicklungsphase durchgeführt wird, ist das größte Manko, dass Softwareanpassungen, die im Anschluss folgen, eine große Gefahr für Performance-Einbrüche darstellen. Durch den großen Bedarf an Ressourcen in technischer und personeller Hinsicht ist die Terminierung von Lasttests mit großen organisatorischen Schwierigkeiten verbunden. Somit kann die Absicht einen Lasttest so spät wie möglich durchzuführen zwar als Ziel festgehalten werden, aber als allgemeingültiges Optimierungsergebnis ist es untauglich, da Softwareprojekte meistens innerhalb terminlicher, organisatorischer und technischer Grenzen durchzuführen sind. Die kurzen Zeiträume, die durch Terminplanungen oder Auswirkungen in den Betriebsabläufen entstehen, sorgen gerade bei Performance-Problemen für eine hohe Dringlichkeitseinstufung. Zeigt sich bei der anschließenden Problemanalyse wie im Fall dieser Arbeit, dass ein Großteil der Performance-Schwachstellen vorhersehbar war, darf die Frage nach Verbesserungsmöglichkeiten nicht ungestellt bleiben. Aus Kosten- und Qualitätssicht sollten Fehler so schnell wie möglich erkannt werden. Ein schlechtes Antwortverhalten kann als Softwarefehler eingestuft werden, wenn es auf ineffiziente Algorithmen (SQL-

Abfragen) und nicht-optimale Tabellenstrukturen zurückzuführen ist. Neue ORACLE©-Funktionen wie der SQLTuning-Advisor© greifen genau diese Problematik auf und bieten eine direkte Hilfestellung zum Thema Performance. Diese Fortschritte fließen auch in die Entwicklungswerkzeuge ein. Dadurch ist für unerfahrene Programmierer direkt möglich, sich diese Hilfestellungen zu Nutzen zu machen. Dafür könnte innerhalb der Entwicklung die Rolle des Performance-Betreuers geschaffen werden. Diese kann funktionale Neuerungen bei den Hilfswerkzeugen im Blick behalten und im Vorfeld Hinweise auf mögliche Konsequenzen von unzureichendem Datenbankdesign geben. Durch einen direkten Ansprechpartner werden Unsicherheiten innerhalb des Entwicklungsteams reduziert und ein expertenunabhängiges Fachwissen innerhalb des Projektteams aufgebaut. Auch für den Lasttest können vermeidbare Probleme reduziert und eine effizientere Testdurchführung gefördert werden. Es können in der begrenzten Zeit auch Informationen über Spezialkonstellationen gesammelt werden. Eine Fragestellung könnte demnach auch sein, was die Minimalanforderungen an die Hardware sind. In Zukunft sollte es trotz sinkender Hardwarekosten weiterhin das Ziel sein, Software so effizient wie möglich zu gestalten. Somit könnte im Idealfall nicht nur die Anwenderzufriedenheit gesteigert werden, sondern auch der eine oder andere Server abgeschaltet werden und Energiekosten eingespart werden[66]. Dadurch wird Performance nicht nur als Randthema angesehen, für das die Problemvermeidungsstrategie gilt, sondern als aktiv zu überwachende Funktionalität schon während der Softwareentwicklung. Es hat sich allerdings gezeigt, dass allgemeingültige Lösungen schwer zu definieren sind. Deshalb versteht sich diese Arbeit in erster Linie als Denkanstoß wie durch die Benennung von Verantwortungsbereichen ein geregelter Fortschrittsprozess initiiert werden kann. Dieser ist stark durch die Handlungsweisen einzelner Akteure und ihre subjektiven Sichtweisen geprägt. Innerhalb dieses abstrakten Rahmens müssen Ideen deshalb ausprobiert und weiterentwickelt werden. Dies ist ein ständiger Prozess, bei dem die Zielvorgaben schwer zu definieren und zu messen sind. Es besteht die Gefahr, dass zeitintensive Forschungen in diesem Bereich aus betriebswirtschaftlicher

[66]Vgl. [Cor93] S.4f (Kap. 1).

Sicht als ineffektiv bewertet werden. Die Hauptaufgaben dürfen nicht in den Hintergrund treten. Es muss deshalb ein Gespür dafür entwickelt werden, wie sich ein tatsächlicher Nutzen aus den Erkenntnissen ergeben kann. Davon kann im Gesamten nur profitiert werden, wenn das Wissen auch kommuniziert wird. Als Beispiel dafür dient in dieser Arbeit die Sicherstellung der korrekten Handhabung der Werkzeuge. Wird dieser Aspekt weiter verfolgt, bietet der Themenbereich Performance als Synonym für Arbeitsleistung noch weitere vielversprechende Ansätze für Untersuchungen von Verbesserungsstrategien.

7 Literaturverzeichnis

[Ala09] Alapati, SR 2009, 'Improving Database Performance: SQL Query Optimization', in *Expert Oracle Database 11G Administration*, Apress.

[Ala07] Alapati, SR & Kim, C 2007, *Oracle Database 11g*, Apress.

[Ala11] Alapati, SR, Kuhn, D & Padfield, B 2011, , in *Oracle Database 11G Performance Tuning Recipes*, Apress.

[Bal09] Balzert, H 2009, 'Werkzeuge', in *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*, Spektrum Akademischer Verlag.

[Bra08] Brandt-Pook, H & Kollmeier, R 2008, 'Der Prozess der Softwareentwicklung', in *Softwareentwicklung kompakt und verständlich*, Vieweg+Teubner.

[Cor93] Corrigan, P & Gurry, M 1993, *ORACLE Performance Tuning*, 1. Edition, O'Reilly & Associates, Inc.

[Fra07] Franz, K 2007, 'Tests zur Effizienz und Zuverlässigkeit', in *Handbuch zum Testen von Web-Applikationen*, Springer , Berlin Heidelberg.

[Frö08] Fröhlich, L 2008, *Oracle 11G Performance Forecast*, Carl Hanser Verlag.

[Gad08] Gadatsch, A 2008, 'Personen und Rollen in IT-Projekten', in *Grundkurs IT-Projektcontrolling*, Vieweg+Teubner.

[Ham07] Hamilton, P 2007, 'Von der Krisis zur Katharsis', in *Dynaxity - Management von Dynamik und Komplexität im Softwarebau*, Springer, Berlin Heidelberg.

[Ham08] Hamilton, P 2008, 'Wege zum Softwarebau von morgen', in *Wege aus der Softwarekrise*, Springer Berlin Heidelberg.

[Hru09] Hruschka, P, Rupp, C & Starke, G 2009, 'Agile Softwareentwicklung in großen Projekten', in *Agility kompakt*, Spektrum Akademischer Verlag.

[Hun09] Hunt, A 2009, *Pragmatisches Denken und Lernen*, Carl Hanser Verlag.

[Krc05] Krcmar, H 2005, 'Die Aufgaben des Informationsmanagements', in *Informationsmanagement*, Springer Berlin Heidelberg.

[Lon00] Loney, K & Koch, G 2000, *ORACLE 8i: Die umfassende Referenz*, Oracle Press.

[Mil03] Millsap, C & Holt, J 2003, *Optimizing Oracle Performance*, 1. Edition, O'Reilly & Associates, Inc.

[Sch10] Schatten, A, Demolsky, M, Winkler, D, Biffl, S, Gostischa-Franta, E & Östreicher, T 2010, 'Software-Projektmanagement', in *Best Practice Software-Engineering*, Spektrum Akademischer Verlag.

[Sch07] Schlimm, N, Novakovic, M, Spielmann, R & Knierim, T 2007, 'Performance-Analyse und -Optimierung in der Softwareentwicklung', *Informatik-Spektrum* (30_4_2007), S. 251-258.

[Sha92] Shasha, DE 1992, *Database Tuning: A Principled Approach*, Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632.

[Sto07] Stoyan, R 2007, 'IT', in *Management von Webprojekten*, Springer Berlin Heidelberg.

[Vig07] Vigenschow, U, Schneider, B 20??, *Softskills für Softwareentwickler*, dpunkt.verlag.

Internetquellen

[iBMF10_1] Bundesministerium der Finanzen 2010, *www.zoll.de*, Zugriff: 10.12.2011, <http://www.zoll.de/DE/Fachthemen/Steuern/Verbrauchsteuern/EMCS/Einfuehrung/einfuehrung_node.html>.

[iBMF10_2] Bundesministerium der Finanzen 2010, *www.zoll.de*, Zugriff: 10.12.2011, <http://www.zoll.de/DE/Fachthemen/Steuern/Verbrauchsteuern/EMCS/Teilnahme/Teilnahmevoraussetzungen/teilnahmevoraussetzungen_node.html>.

[iEUC10] Communication department of the European Commission 2010, *Europa.eu*, Zugriff: 02.01.2012, <http://europa.eu/rapid/pressReleasesAction.do?reference=IP/10/401&format=HTML&aged=0&language=de&guiLanguage=de>.

[iEUK09] Europäische Kommission 2009, *ec.europa.eu*, Zugriff: 09.01.2012, <http://ec.europa.eu/taxation_customs/taxation/excise_duties/circulation_control/index_de.htm>.

[iMil99] Millsap, C 1999, *Performance Management Myths & Facts*, Zugriff: 02.01.2012, <http://method-r.com/papers/doc_download/17-performance-management-myths-a-facts>.

[iMil08] Millsap, C 2008, *Measure Once, Cut Twice (No, Really)*, Zugriff: 01.12.2011, <http://method-r.com/papers/doc_download/5-measure-once-cut-twice-cary-millsap>.

[iMil09] Millsap, C 2009, *For Developers: Making Friends with the Oracle Database*, Zugriff: 05.12.2011, <http://method-r.com/papers/doc_download/10-for-developers-making-friends-with-the-oracle-database-cary-millsap>.

[iORA05] ORACLE 2005, *docs.oracle.com*, Zugriff: 03.01.2012, <http://docs.oracle.com/cd/B19306_01/server.102/b14200/clauses002.htm>.

8 Abbildungsverzeichnis

9 Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Dortmund, den 09.02.2012 _____