Monitoring von Perl-basierten Webanwendungen mittels Kieker

Proposal zur Bachelor-Arbeit

Nis Börge Wechselberg

11. März 2013

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Arbeitsgruppe Software Engineering

Betreut durch: Prof. Dr. Wilhelm Hasselbring
Dipl.-Inf. Peer Brauer

Inhaltsverzeichnis

Kapitel 1

Einleitung

Dieses Proposal soll zur genauen Definition des Themas und des Umfangs der Bachelorarbeit dienen. Hierzu wird zunächst die Zielsetzung der Arbeit und eine Einordnung in das umgebende Themenfeld vorgestellt. Anschließend werden die benötigten und verwendeten Technologien und Softwaresysteme vorgestellt.

Hierauf aufbauend wird dann das geplantes Vorgehen für die Lösung der Problemstellung aufgezeigt.

Die Bachelorarbeit trägt den Arbeitstitel *Monitoring von Perl-basierten Webanwendungen mittels Kieker*. Somit soll in der Arbeit zunächst ein Monitoringverfahren für Perl-Anwendungen im Allgemeinen und Webanwendungen im speziellen etabliert werden. Das Monitoring soll hierbei mit dem Kieker Framework geschehen, welches bisher primär auf Java-Anwendungen ausgerichtet war. In mehreren Abschlussarbeiten wurden unter anderem bereits Erweiterungen für .NET oder COBOL sowie COM entwickelt, allerdings ist bisher noch kein Monitoring von Perl-Anwendungen möglich. Somit muss zunächst ein System entwickelt werden um Perl-Anwendungen mit *monitoring probes* zu versehen und diese Daten anschließend an Kieker zu übergeben. Hierbei liegt der Fokus auf dem *performance monitoring*, also der Erfassung des zeitlichen Verhaltens der Anwendung.

Mit diesen *monitoring probes* soll dann zunächst eine isolierte Testanwendung instrumentiert werden und das Verhalten der Anwendung ausgewertet werden. Hierbei können erste Ergebnisse zu dem Einfluss der *monitoring probes* auf die Laufzeit der Anwendung und das korrekte Verhalten der Probes ermittelt werden.

Sobald diese Tests erfolgreich abgeschlossen sind, sollen dann die Probes in die Anwendung Kielprints eingeführt werden. Diese Plattform wird vom Geomar Helmholtz-Zentrum betrieben und in der letzten Zeit sind hier einige Probleme mit dem Laufzeitverhalten beobachtet worden. Durch das Monitoring kann dann das Verhalten der Anwendung im Betrieb beobachtet werden. Es soll versucht werden Ursachen für Leistungseinbußen zu lokalisieren und Optimierungspotential in der Anwendung aufgezeigt werden.

Bei ausreichender Zeit können dann noch weitere Funktionen für die Perl-Schnittstelle implementiert werden wie Optionen zur automatischen oder teil-automatischen Instrumentierung der Anwendung oder andere Monitoringoptionen.

Kapitel 2

Ziele & Technologien

Im folgenden werden kurz die wichtigsten Technologien und Softwaresysteme vorgestellt, welche im Rahmen dieser Bachelor-Arbeit eingesetzt werden sollen. Zuvor wird auf das Verfahren des Performance Monitorings (2.1) eingegangen um den Zweck dieser Systeme zu verdeutlichen. Die verwendeten Softwaresysteme umfassen neben der Programmiersprache Perl (2.3) primär das Monitoring Framework Kieker (2.2). Das zu untersuchende System ePrints wird ebenfalls kurz vorgestellt (2.4) um die Rahmenbedingungen zu erläutern. Zum Abschluss werden dann die Ziele der Arbeit aufgelistet (2.5).

2.1. Performance Monitoring

Beim Monitoring, auch dynamische Analyse genannt, werden während der Ausführung einer Anwendung Messdaten erzeugt und anschließend anhand dieser Messdaten das Verhalten der Anwendung analysiert. Dieses Verfahren steht als ergänzendes Mittel zur statischen Analyse, bei welchem die Software anhand ihres Quelltextes analysiert wird. Allerdings ist die statische Analyse für umfangreichere Softwaresysteme nur schwierig durchführbar. Mit der dynamischen Analyse können die tatsächlich ausgeführten Programmpfade ermittelt werden und direkt das Verhalten der Anwendung nachvollzogen werden.(vgl. auch Schmalenbach 2007) Mit verschiedenen Techniken können dann auch bei Bedarf nur Teile der Anwendung beobachtet werden und der Fokus auf bestimmte Programmteile gelegt werden.

Die dynamische Analyse wird auch häufig eingesetzt um das Verhalten eines existierenden Softwaresystems in Modernisierungsprozessen zu erfassen. Da Anwendungen häufig im Betrieb modifiziert und weiterentwickelt werden und hierbei mitunter die Dokumentation der Änderungen nur untergeordneten Charakter gegenüber der Funktionalität geniest, weicht häufig das dokumentierte Verhalten von dem tatsächlich beobachteten ab. Diese Differenzen können durch die dynamische Analyse erkannt werden und in den Modernisierungsprozess eingebunden werden.

Zur Durchführung des Monitorings wird der Quelltext des Programms mit sogenannten Probes versehen, welche verschiedene Ereignisse im Programmablauf protokollieren. Hierbei sind häufig Verzweigungen, Sprünge oder Funktionsaufrufe besonders von Interesse, aber auch andere Ereignisse wie Anfragen an Datenbanken können protokolliert werden.

Mit diesen Messdaten kann dann eine Analyse und Visualisierung durchgeführt werden

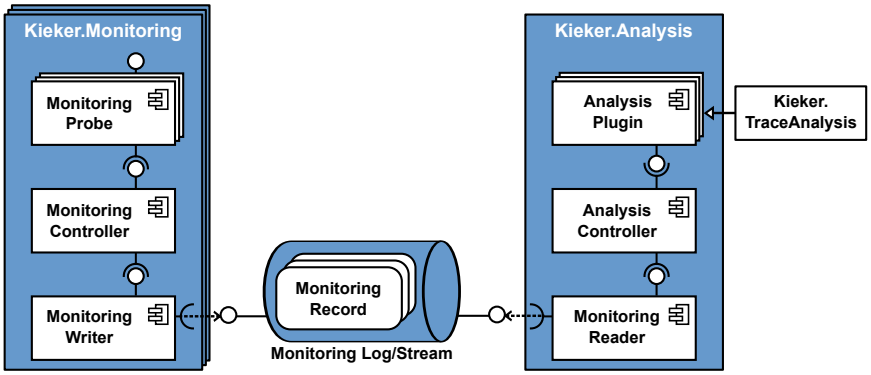

Abbildung 2.1. Kieker Komponentendiagramm [3]

um das Verhalten der Anwendung zu dokumentieren. Hierbei werden dann zum Beispiel UML Sequenz Diagramme erstellt oder das zeitliche Verhalten der Anwendung detailliert untersucht.

Der Schwerpunkt dieser Arbeit wird auf dem *performance monitoring* liegen. Hierbei wird eine möglichst präzise Zeitmessung im Programmablauf durchgeführt und dieser Zeitcode in den *monitoring records* eingefügt. Somit kann nicht nur die Aufrufreihenfolge sondern auch der zeitliche Ablauf, also konkret die Dauer zur Ausführung einer Funktion und auch die Häufigkeit der Ausführung, festgestellt werden. Somit können dann besonders lang und/oder häufig laufende Funktionen identifiziert werden um Ansatzpunkte für Performanceprobleme aufzuzeigen. Neben der Zeit können bei Bedarf auch noch weitere Systemdaten wie CPU-Auslastung oder Speicherverbrauch protokolliert werden.

2.2. Kieker Monitoring Framework

Kieker[1] ist ein System zum Monitoring und zur Analyse des Laufzeit-Verhaltens einer Software. Es wird seit 2006 in der Arbeitsgruppe Software Engineering an der CAU Kiel entwickelt. Frühere Versionen von Kieker waren primär auf das Monitoring von Java-Anwendungen ausgerichtet, doch wurden bereits ergänzende Module entwickelt oder befinden sich zur Zeit in der Entwicklung um zum Beispiel .NET oder COBOL mit Kieker analysieren zu können. Seit 2011 wird Kieker von der SPEC Research Group als empfohlenes Tool im SPEC RG Software Verzeichnis[2] angeboten.

Wie in Grafik 2.1 dargestellt besteht Kieker aus den zwei zentralen Komponenten Kieker.Monitoring und Kieker.Analysis. Die Monitoring Komponente stellt Klassen und Methoden zum Instrumentieren und Überwachen der Software zur Verfügung. Hierzu

[1] http://www.kieker-monitoring.net
[2] http://research.spec.org/projects/tools.html

werden von *monitoring probes* Messungen vorgenommen und hierzu *monitoring records* erstellt. Diese *monitoring records* werden dann an einen *monitoring writer* übergeben, der die Informationen in eine Log-Datei schreibt oder mit einem Stream an die Analysis Komponente übergibt. Zum Monitoring stehen manuelle Methoden aber auch automatische Mechanismen zur Verfügung die, zum Beispiel mittels *AspectJ*, die Programmstruktur analysieren und automatisch die benötigten Messpunkte erzeugen.

Die Analyse der erhaltenen Daten wird mittels Kieker.Analysis durchgeführt. Diese Komponente liest die *monitoring records* ein und stellt einen konfigurierbaren Ablauf zum Filtern, Analysieren und Visualisieren der Daten bereit. Hierbei können zum Beispiel Sequenz Diagramme oder Abhängigkeitsbäume erzeugt werden um das Verhalten der Anwendung zu beschreiben.

Die Kieker.Monitoring Funktionen werden in dieser Arbeit nicht oder nur teilweise eingesetzt werden, da sie primär auf Java-Anwendungen ausgerichtet sind. Allerdings werden die Kieker.Analysis Funktionen benutzt werden um die weitere Analyse durchzuführen.

2.3. Perl

Die Programmiersprache Perl[4] ist eine imperative, plattformunabhängige, interpretierte Programmiersprache. Die erste Version wurde 1987 vorgestellt und hat seine Wurzeln primär in C und awk. Die aktuelle Version ist Perl 5.16, die im Mai 2012 veröffentlicht wurde. Perl ist eine interpretierte Programmiersprache, also werden die Programme nicht zu einer nativen Anwendung übersetzt sondern mit einem Interpreter ausgeführt. Der Perl-Interpreter ist für alle relevanten Betriebssysteme verfügbar und in vielen Systemen bereits integriert.

In Perl werden verschiedenste Programmierparadigmen umgesetzt, allerdings ist es dem Programmierer freigestellt, welche dieser Möglichkeiten er umsetzen will. So ist zum Beispiel die Objektorientierung ein Teil der Sprache, allerdings nicht wie in Java erzwungen. Auch stellt Perl häufig mehrere Optionen für die selben Probleme bereit, so können häufig Befehle verkürzt werden oder es stehen mehrere äquivalente Befehle zur Verfügung. Dies führt allerdings sehr schnell zu schlechter Lesbarkeit der Programme, da Perl auch eine sehr freie Syntax ermöglicht. Zeilenumbrüche und Leerzeichen sind weitgehend bedeutungslos und führen zu einem sehr unterschiedlichen Code-Format und sehr persönlichen Gestaltungen der Programme.

Eines der stärksten Features von Perl sind Reguläre Ausdrücke. Bereits in frühen Versionen wurden diese unterstützt und aufgrund der hohen Verbreitung von Perl wurden die Perl Syntax de facto Standard für Reguläre Ausdrücke. Auch aufgrund dieser Stärke bei der Verarbeitung von Regulären Ausdrücken und der guten Verbindung mit anderen Softwaresystemen wird Perl in vielen Websystemen eingesetzt. Die meisten Webserver

[3] Grafik entnommen aus [Kieker Project 2012]

[4] `http://www.perl.org`

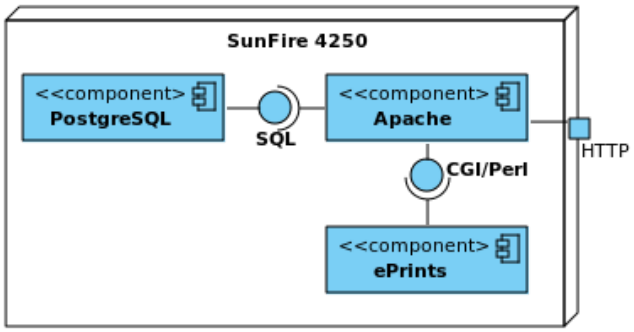

Abbildung 2.2. Deployment der Kielprints-Anwendung

bieten Schnittstellen um den Perl-Interpreter einzubinden und Perl-Anwendungen somit in die Webpräsenz einzubinden.

2.4. ePrints / Kielprints

Das System ePrints ist eine quelloffene Webplattform zur Veröffentlichung von Publikationen, Forschungsberichten oder anderen Dokumenten. Die Plattform implementiert dabei das OAI-PMH[5] Protokoll und ist somit ein Projekt zur Umsetzung von Open Access. ePrints wurde initial von der University of Southampton entwickelt und unter der [*GNU General Public License, version 3*] lizenziert. Die Plattform wurde in Perl implementiert und steht zur Zeit in der Version 3.3 zum Download[6] bereit.

An der Universität Kiel wird die Software Kielprints verwendet, was einen angepasste Version von ePrints ist. Betrieben wird diese Plattform von dem Kieler Datenmanagement Team am Geomar Helmholtz-Zentrum für Ozeanforschung Kiel. Das Deployment der Anwendung ist in der Abbildung 2.2 grob skizziert.

Für die Plattform wurden bereits einige Erweiterungen entwickelt und in die Plattform integriert. Mit der Öffnung der Plattform für alle Institute der Universität sind unerwartete Performanceprobleme aufgetreten. Diese sollen mit dieser Arbeit und den entwickelten Modulen aufgespürt werden. Diese Probleme treten einerseits für die Benutzer beim Datenabruf auf, wobei unerwünscht lange Antwortzeiten, besonders bei der Suche nach Autoren oder Forschungsbereichen, beobachtet werden können. Aber auch die Administration und Datenpflege leidet unter den Problemen, da die Oberflächen für das Eintragen oder Editieren von Publikationen teilweise Wartezeiten von über 10 Sekunden aufweisen. Diese Zeiten sind bei einer Datenbankgröße von etwa 14000 Publikationen und 1000 Autoren nicht angemessen und weisen auf Probleme in der Aufbereitung der Seiten hin.

[5] Open Archives Initiative Protocol for Metadata Harvesting
[6] `http://www.eprints.org/software/`

2.5. Zielsetzung

Folgende Punkte sollen in der Bachelorarbeit konkret umgesetzt werden:

- Implementierung von monitoring probes und einem monitoring writer in Perl.
- Verwendung eines monitoring logs in Dateiform.
- Untersuchung des Einflusses der monitoring probes auf die Laufzeit von Anwendungen.
- Instrumentierung der Kielprints-Anwendung.
- Lokalisierung von Performanceproblemen in der Kielprints-Anwendung.

Kapitel 3

Geplante Umsetzung

Wie bereits in der Einleitung skizziert muss für die Umsetzung der Arbeit zunächst eine Möglichkeit des Monitorings der Perl-Anwendungen geschaffen werden (3.1). Anschließend soll das Verhalten zunächst getestet werden (3.2) und schließlich das Projekt Kielprints untersucht werden (3.3).

3.1. Monitoring-Probes in Perl

Das Kieker Framework unterstützt zur Zeit noch kein Monitoring von Perl-Anwendungen. Im Rahmen von früheren Abschlussarbeiten wurden bereits Verfahren zum Monitoring von zum Beispiel .NET- [Magedanz 2011] oder COBOL-Anwendungen [Richter 2012] realisiert. Aufbauend auf den bisherigen Entwicklungen soll in dieser Arbeit nun das Monitoring von Perl-Anwendungen ermöglicht werden. Hierzu soll ein Perl-Modul implementiert werden, welches die Aufgaben des Kieker.Monitoring übernimmt. Dieser Ansatz ist in Abbildung 3.1 dargestellt.

Dieses Modul muss zunächst Funktionen zur Anreicherung des Codes mit Probes bereitstellen. Diese Funktionen müssen das Verhalten der Anwendung protokollieren und an eine (zu implementierende) Funktion weiterreichen, welche die persistente Sicherung der Daten übernimmt. Hierbei ist abzuwägen, ob von diesem Perl-Modul direkt *monitoring records* erzeugt werden sollen, die von Kieker.Analysis verarbeitet werden können, oder ob ein Ansatz gewählt werden soll, der zunächst eigene Records erstellt und anschließend

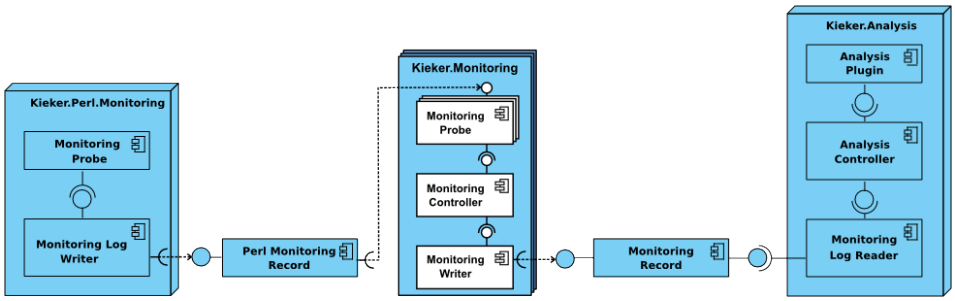

Abbildung 3.1. Entwurf der Systemarchitektur

mit einer neuen Kieker Komponente die Übersetzung in Kieker *monitoring records* realisiert. Dieser zweistufige Ansatz wurde bereits für COBOL-Anwendungen realisiert, allerdings ist auch die Erzeugung fertiger Records in Perl möglich. Für die erste Implementierung soll aber der Ansatz mit einer zwischengelagerten Kieker-Ausführung gewählt werden.

Da in Perl auch recht mächtige Funktionen zur Anbindung anderer Programmiersprachen unterstützt werden könnte auch eine Integration der bestehenden Kieker Klassen in Betracht gezogen werden. Da aber hierbei eine Verbindung von vorcompiliertem Java-Code und interpretiertem Perl-Code realisiert werden muss, kann bei dieser Veriante das Laufzeitverhalten stärker als in dem oben skizzierten Ansatz beeinflusst werden.

3.2. Instrumentierung und Monitoring einer Testanwendung

Das fertige Perl-Modul wird anschließend auf eine isolierte Testanwendung angewendet. Hierbei wird es sich voraussichtlich um eine Kopie des ePrints-Systems handeln. In dieser Testanwendung kann dann die Instrumentierung durchgeführt werden und verschiedenen Testszenarien untersucht werden. Durch die Simulation von verschiedenen Anfragen kann neben dem Verhalten der Anwendung auch der Einfluss des Monitorings auf die Anwendung untersucht werden. Es sollte versucht werden, den zusätzlichen Aufwand der Anwendung möglichst gering zu halten um das tatsächliche Verhalten der Anwendung nicht zu stark zu verzerren.

Mit den erzeugten Daten soll die Anbindung an Kieker.Analysis getestet werden und eine geeignete Analysemethode für die späteren Tests festgelegt werden. Sobald diese Tests zufriedenstellend ausfallen kann dann die Instrumentierung von Kielprints im produktiven Betrieb durchgeführt werden.

3.3. Instrumentierung von Kielprints

In Zusammenarbeit mit dem Geomar Helmholtz-Zentrum für Ozeanforschung Kiel soll dann Kielprints instrumentiert werden. In dem momentanen Zustand der Anwendung sind langsame Reaktionszeiten aufgetreten, obwohl bisher mit knapp 14000 Publikationen die Auslastung der Datenbank noch in einem üblichen Rahmen liegt. Somit soll die Anwendung instrumentiert werden und Engpässe in der Anwendung identifiziert werden. Hierzu werden die Erkenntnisse der vorherigen Testläufe verwendet um Analysemethoden festzulegen und Vergleichswerte zu erhalten. Mit den erhaltenen Analysedaten soll dann versucht werden Optimierungspotential der Anwendung aufzuzeigen.

Kapitel 4

Organisatorischer Rahmen

▷ Organisation

Dauer der Arbeit 1. Oktober 2012 bis 31. März 2013

Bearbeiter Nis Börge Wechselberg, Mat-Nr. 774996, nbw@informatik.uni-kiel.de

Berater Dipl.-Inf. Peer Brauer, pcb@informatik.uni-kiel.de

▷ Entwicklung

Hardware Die Kielprints-Plattform wird im Geomar auf einer SunFire 4250 betreiben. Das Testsystem wird auf einem Blade des Lehrstuhls bereitgestellt.

System Solaris 10

Software ePrints 3.3 unter Perl 5.10

Entwicklungsumgebung Für den Betreib von Kielprints werden PostgreSQL 8.3 sowie Apache 2.2.22 eingesetzt. Die Pakete wurden von OpenCSW.org angepasst.

Kapitel 5

Zeitplan

In der folgenden Tabelle ist das geplante Vorgehen skizziert:

Tabelle 5.1. Zeitplan

KW	Projektphase	Meilensteine
47	Vorbereitung	Literaturrecherche
48	Perl-Modul	Analyse der benötigten Funktionen, Festlegung des Datenformats
48		Implementierung der Probes
49		Implementierung der Probes
50		Lokale Tests und Fehlerbehebung
51	Konsolidierung	Schreiben der Arbeit
52		Schreiben der Arbeit
1	Test-Instrumentierung	Einrichtung von Testanwendungen, Instrumentierung
2		Feststellung von Monitoring-Overhead
3	Konsolidierung	Schreiben der Arbeit
4	Kielprints	Instrumentierung der Anwendung
5		Ablauf des Monitorings zur Laufzeit
6		Analyse der Traces
7		Evtl. Langzeit Monitoring
8		
9		
10	Konsolidierung	Schreiben der Arbeit
11	Abschluss der Arbeit	Fertigstellung, Kolloqium
12		
13		

Dieser Verlauf ist auch aus dem folgenden Gantt-Diagramm (Abbildung 5.1) ersichtlich.

5. Zeitplan

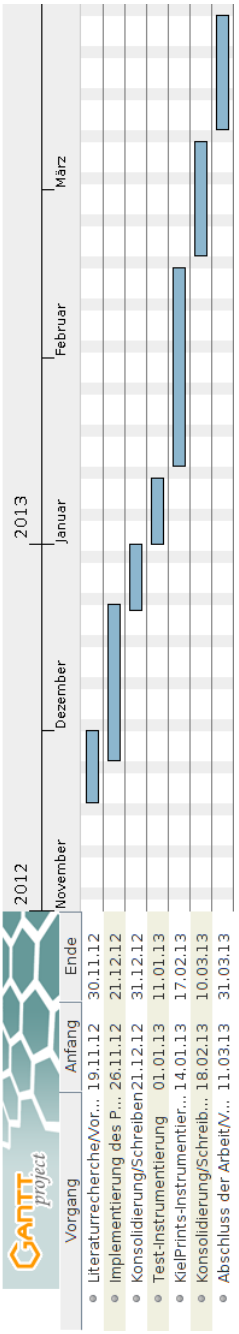

Abbildung 5.1. Gantt-Chart

Glossar

E

ePrints
: Open-Source Plattform zur Veröffentlichung von Dokumenten, `http://www.eprints.org/software`. 3, 6

K

Kieker
: Software Framework zur Durchführung von Monitoringaufgaben, `http://www.kieker-monitoring.net`. 1, 3, 4, 9, 10

M

Monitoring
: Überwachung des Verhalten eines Programmes während der Laufzeit. 1, 4, 5, 9, 13

monitoring probe
: Eingefügter Messpunkt in einem Programm zur Erzeugung von monitoring records. 1, 5, 7, 15

monitoring record
: Bei der Ausführung einer monitoring probe erzeugter Protokoll-Eintrag einer Messung. 4, 5, 9, 10, 15

R

Rekursion
: siehe Rekursion. 15

Literaturverzeichnis

[*GNU General Public License, version 3*]. GNU General Public License, version 3. 2007. URL: http://www.gnu.org/licenses/gpl.html. (Siehe Seite 6)

[Kieker Project 2012] Kieker Project. Kieker 1.6 User Guide. Software Engineering Group, Kiel University, Kiel, Germany. Okt. 2012. URL: http://kieker-monitoring.net/documentation/. (Siehe Seite 5)

[Magedanz 2011] F. Magedanz. Dynamic analysis of .NET applications for architecture-based model extraction and test generation. Diplomarbeit. Department of Computer Science, University of Kiel, Germany, 2011. URL: http://eprints.uni-kiel.de/15486/. (Siehe Seite 9)

[Richter 2012] B. Richter. Dynamische Analyse von COBOL-Systemarchitekturen zum modellbasierten Testen. Diploma Thesis, University of Kiel (work in progress). Diplomarbeit. Department of Computer Science, University of Kiel, Germany, 2012. URL: http://eprints.uni-kiel.de/15489/. (Siehe Seite 9)

[Schmalenbach 2007] C. Schmalenbach. Performancemanagement für serviceorientierte Java-Anwendungen: Werkzeug- und Methodenunterstützung im Spannungsfeld von Entwicklung und Betrieb. Springer, 2007. URL: http://books.google.de/books?id=IbaEtgAACAAJ. (Siehe Seite 3)