

# Konzeption und Entwicklung eines Auswertungstools für die Logdateien des JRockit Garbage Collectors

**Student:** *Raffael Schmid*  
**Studiengang:** *Informatik*  
**Zeitraum:** *Juli 2011 bis Dezember 2011*

## Aufgabenstellung

### Ausgangslage

Für die Ermittlung von Java Performance-Problemen braucht es Wissen über die Funktionsweise der Java Virtual Machine (JVM), deren Ressourcenverwaltung (Speicher, I/O, CPU) und das Betriebssystem. Die Verwendung von Tools zur automatisierten Auswertung der Daten kann in den meisten Fällen sehr hilfreich sein.

Die Auswertung von Garbage Collection Metriken kann im laufenden Betrieb durch Profiling (online) gemacht werden, sie ist aber bei allen JVMs auch via Logdatei (offline) möglich. Die unterschiedlichen Charakteristiken der Garbage Collectors bedingen auch unterschiedliche Auswertungs- und Einstellungsparameter.

JRockit ist die Virtual Machine des Weblogic Application Servers und basiert entsprechend auch auf anderen Garbage Collection Algorithmen als die der Sun VM. Aktuell gibt es noch kein Tool, welches die Daten der Logs sammelt und grafisch darstellt.

### Ziel der Arbeit

Ziel der Bachelorthesis ist die Konzeption und Entwicklung eines Prototypen für die Analyse von Garbage Collection Logdateien der JRockit Virtual Machine. Die Software wird mittels einer Java Rich Client Technologie implementiert. Zur Konzeption werden die theoretischen Grundlagen der Garbage Collection im Allgemeinen und der JRockit Virtual Machine spezifisch erarbeitet und zusammengestellt.

### Aufgabenstellung

Im Rahmen der Bachelorthesis werden vom Studenten folgende Aufgaben durchgeführt:

- Studie der Theoretischen Grundlage im Bereich der Garbage Collection (generell und spezifisch JRockit Virtual Machine)
- Stärken- / Schwächen-Analyse der bestehende Rich Client Frameworks (Eclipse RCP Version 3/4, Netbeans)
- Durchführung einer Anforderungsanalyse für einen Software-Prototyp.
- Auswahl der zu verwendenden Frameworks
- Konzeption und Spezifikation des Software-Prototypen (auf Basis des ausgewählten Rich Client Frameworks), der die ermittelten Anforderungen erfüllt.
- Implementation der Software
- Bewertung der Software auf Basis der Anforderungen

### Erwartete Resultate

Die erwarteten Resultate dieser Bachelorthesis sind:

- Detaillierte Beschreibung der Garbage Collection Algorithmen der Java Virtual Machine im Generellen und spezifisch der JRockit Virtual Machine.
- Analyse über Stärken und Schwächen der bestehenden (state of the art) Java Rich Client Technologien
- Anforderungsanalyse des Software Prototyps
- Dokumentierte Auswahlkriterien und Entscheidungsgrundlagen
- Konzept und Spezifikation der Software
- Lauffähige, installierbare Software und Source-Code
- Dokumentierte Bewertung der Implementation

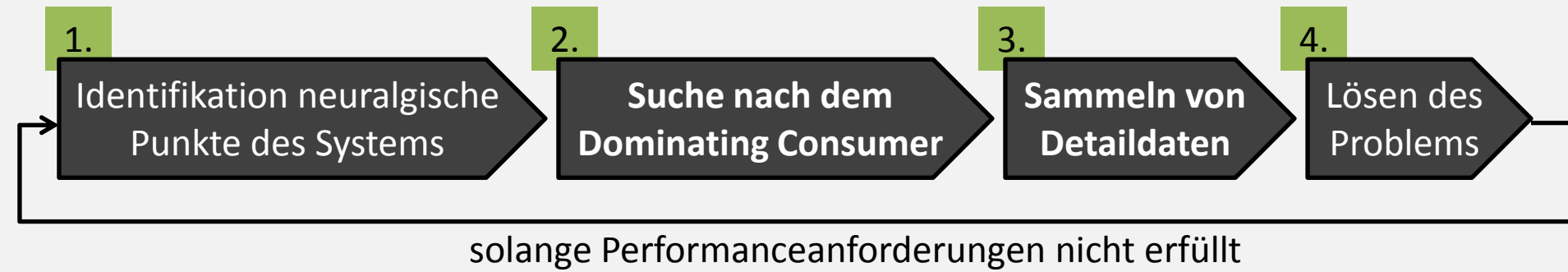
### Methodik

Die Evaluation von Rich Client Framework und Charting-Bibliothek sowie die Ermittlung der funktionalen Anforderungen ist getrieben durch eine methodenbasierte Anforderungsanalyse. Es standen drei Methoden zur Auswahl: Use Cases, Requirements Engineering nach IEEE 830 und User Stories. Auf Ebene der Customer Requirements wird die Methode der Use Cases eingesetzt. Sie zeichnet sich dadurch aus, dass man die Anforderungen nicht nur textuell sondern auch modellbasiert mit UML definieren kann. Für die Dokumentation der Development-Requirements, beinhaltet auch die Qualitätsanforderungen, wird der Standard IEEE 830 verwendet. Die damit definierten Anforderungen lassen sich gut in einzelne Entwicklungspakete verpacken. Auch die Struktur der Anforderungsanalyse wurde anhand dieser Definition aufgebaut.

## Grundlagen

### Vorgehen Performance Tuning

Die Performanceanalyse ist ein iterativer Prozess und dauert in der Regel so lange, bis die Anforderungen an das System in diesem Bereich erfüllt sind. Eine einzelne Iteration besteht aus den folgenden vier Schritten:



#### Suche nach dem Dominating Consumer

##### Hohe relative Systemlast

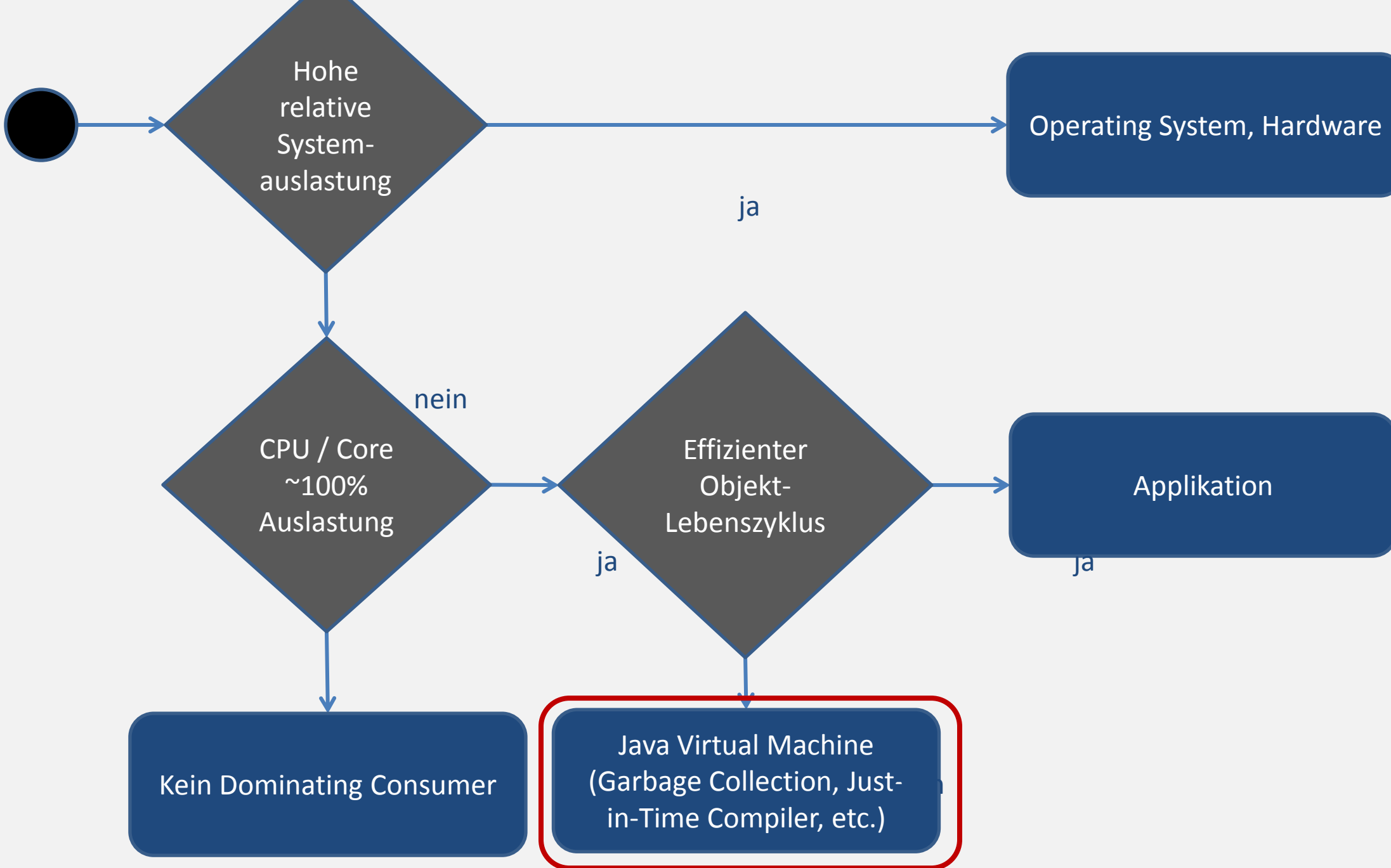
Die erste Frage bei der Suche nach dem Dominating Consumer ist, ob die hohe CPU-Auslastung durch die Applikation oder das System verursacht wird (hohe relative Systemlast). Dies kann unterschiedliche Gründe haben (exzessive Kontextwechsel, hohes I/O). Zur Analyse kann der Task Manager oder vmstat verwendet werden.

##### Hohe CPU- respektive Core-Last

Sofern die relative Systemlast nicht gross ist wird überprüft, ob die CPU Auslastung insgesamt gross ist. Man prüft also, ob es eventuell **keinen Dominating Consumer** gibt. Dies würde bedeuten, dass etwas die Threads daran hindert, CPU-Ressourcen zu bekommen (*Dead Locks, Applikation skaliert nicht, kleine Connection-, Thread-Pools, etc.*)

##### Effizienter Objekt-Lebenszyklus

Sofern die CPU-Auslastung, trotz kleiner relativer Systemlast durch die Applikation, hoch ist, muss **Java Virtual Machine** und evt. die **Garbage Collection** angeschaut werden.



### Garbage Collection Tuning

#### Ziele

Beim Tuning der Garbage Collection verfolgt man eines der drei folgenden Ziele:

- Verbesserung des Durchsatzes
- kleine und gleichmässige Pausenzeiten**
- geringerer Speicherverbrauch

Das einzige relevante Tuningziel ist in der Regel die Optimierung hinsichtlich gleichmässiger und kurzer Pausenzeiten. Dies ist insbesondere bei Applikationen wichtig, bei denen die Interaktion mit einem Benutzer im Vordergrund steht. Durchsatztuning ist meistens nicht sehr effektiv und der Austausch der CPU kostengünstiger. Arbeitsspeicher-Tuning ist mit der 64-Bit Architektur in den Hintergrund gerückt.

#### Garbage Collection auf der JRockit Virtual Machine

Die Grundlage der JRockit Garbage Collection bildet der Tri-Coloring Mark & Sweep Algorithmus. Er wurde hinsichtlich besserer Parallelisierbarkeit und der optimalen Verwendung der Anzahl Garbage Collection Threads optimiert. Die Garbage Collection der JRockit VM arbeitet entweder mit oder ohne Generationen. Es gibt folgende Algorithmen:

- Generational Concurrent Mark & Sweep
- Single Concurrent Mark & Sweep
- Generational Parallel Mark & Sweep
- Single Parallel Mark & Sweep

##### Aufbau des Heaps (mit Generationen)

Die JRockit Virtual Machine kommt ohne Permanent-Bereich aus. JRockit verwendet kein Mark & Copy Algorithmus und benötigt deshalb auch keine Survivor-Regionen. Das Schema ist wie folgt:

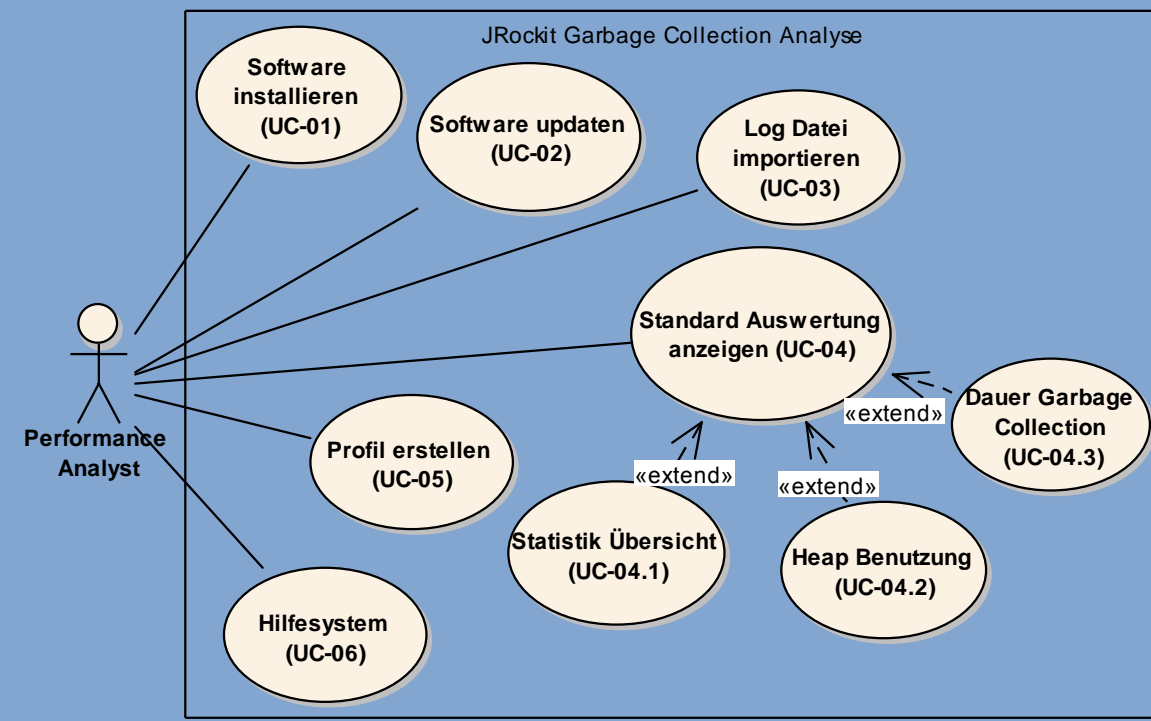


In der Young Generation werden die neuen Objekte angelegt, sie wird in thread-lokale Bereiche unterteilt, so dass die Allokation nicht synchronisiert werden muss. In die Keep Area kommen die Objekte nach einer, in die Old Generation nach einer weiteren Garbage Collection.

## Realisierung

### Anforderungsanalyse

#### Use Cases



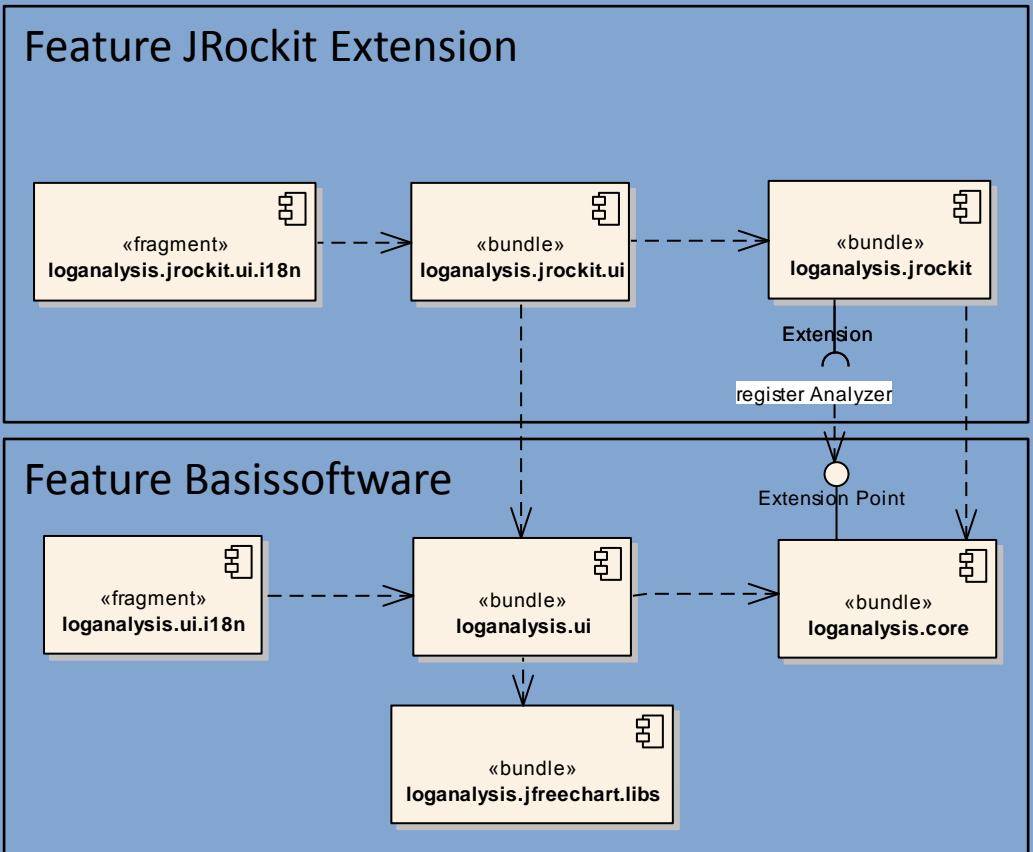
### Konzept

#### Verwendete Frameworks

Auf der Basis der Anforderungen wurde eine Evaluation von Bibliotheken gemacht. Aufgrund dieser ergab sich, welche Bibliotheken für die Realisierung verwendet werden: als Rich Client Framework wird Eclipse 3.x verwendet, zur Generierung der Charts hat sich die Verwendung von JFreeChart als leichtgewichtig und mächtig erwiesen.

#### Architektur

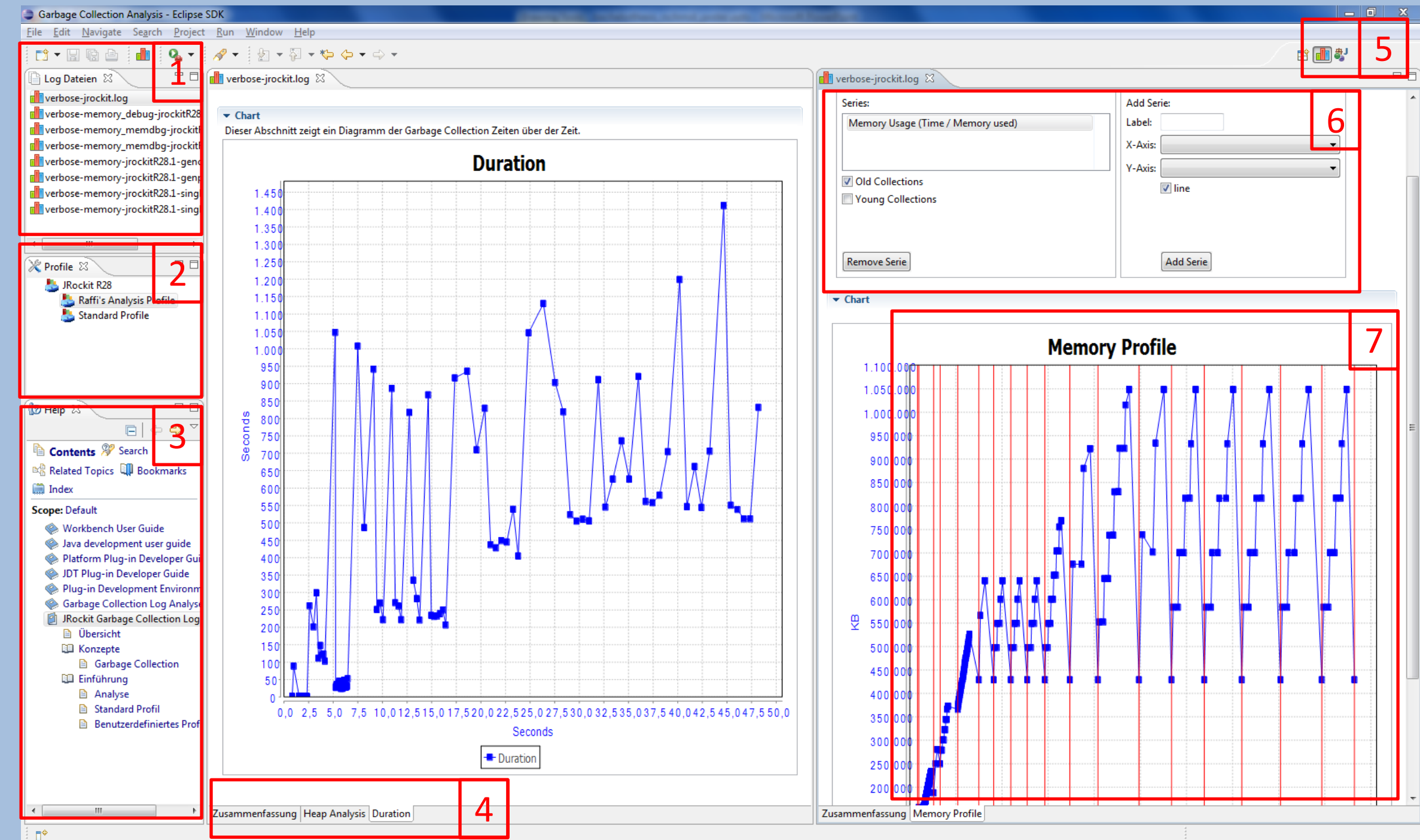
Aufgrund der Anforderungen ist die Applikation so zu konzipieren, dass auch weitere Logformate hinzugefügt werden können. Die Applikation wird deshalb initial (später können andere Erweiterungen hinzu kommen) in die Teile *Basissoftware* und *Erweiterung JRockit* aufgeteilt. Die Kommunikation dazwischen findet über den Eclipse Extension-Point-Mechanismus statt. Die Verantwortlichkeiten sind folgendermassen:



#### JRockit Extension (JRockit Extension Feature)

- Garbage Collection Log parsen
- Daten in eigenem Domänenmodell speichern
- Laden der Daten aus dem Domänenmodell

## Proof of „Konzept“



### Review

#### Was das Tool leistet

Die Analysesoftware kann dann verwendet werden, wenn man Auswertungen auf Basis von Garbage Collection Logs machen will. Andere Tools benötigen zur Analyse oft eine Verbindung mit der entsprechenden Virtual Machine, dies ist in vielen Unternehmen aufgrund von blockierten Ports nicht möglich.

#### Funktionsumfang

Die Standardauswertung zeigt eine Statistik mit folgenden Daten:

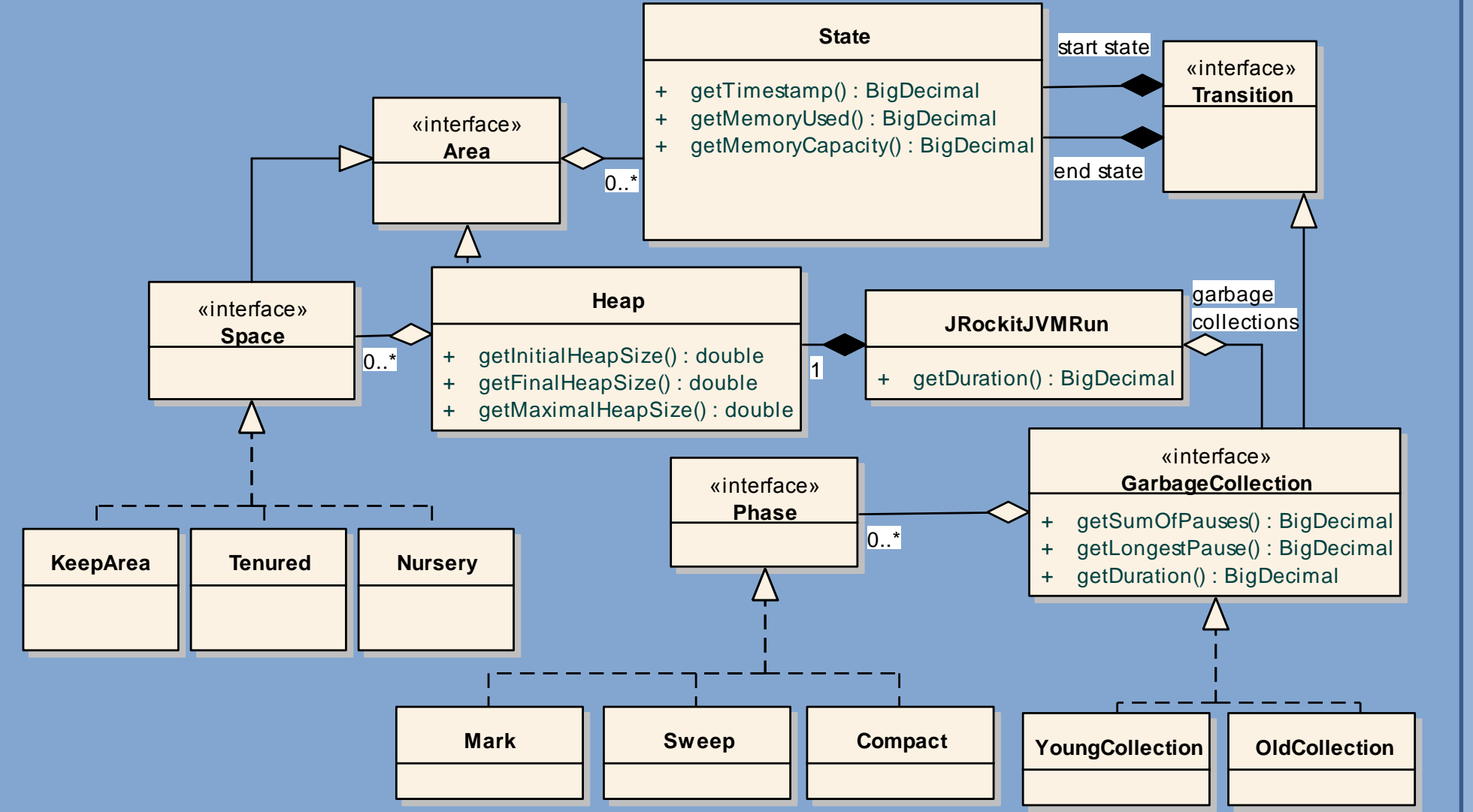
- Garbage Collection Aktivität
- Initiale, durchschnittliche und maximale Kapazität des Heaps, der Young Collection (Nursery) und Old Collection.
- Generelle Informationen wie Dauer der Messung, verwendete Zeit in der Garbage Collection (relativ, absolut)
- Zusätzlich gibt es ein Diagramm zur Auswertung des benutzten Speichers auf dem Heap und eines über die Dauer der einzelnen Garbage Collections.

#### Parser

Das Parsen passiert in zwei Schritten: als erstes wandelt der Lexer die Rohdaten in einzelne Tokens um, anschliessend speichert der Syntactic Analyzer die Werte im Domänenmodell.

#### Domänenmodell

Die gelesenen Daten werden anschliessend in Form dieses Domänenmodells gespeichert. Es entspricht der abstrahierten Struktur des JRockit Garbage Collectors.



#### Implementation

Eine Logdatei wird durch den Import in der Ansicht *Logdateien* sichtbar. Mit einem Doppelklick auf diese Datei oder ein Profil wird die Logdatei eingelese, geparkt, analysiert und im entsprechenden Analysefenster geöffnet.

- Die Ansicht *Logdateien* zeigt alle vom Benutzer importierten Garbage Collection Logs.
- Die Ansicht *Profile* zeigt die vom Benutzer erstellten Profile, diese können an die Bedürfnisse der Analyse angepasst werden.
- Die Hilfe zeigt relevante Themen basierend auf einem Index. Es gibt auch Hilfetemen die an eine bestimmte Aktion oder ein Fenster gebunden sind (kontextsensitiv).
- Die geöffnete Standardauswertung zeigt drei verschiedene Tabs mit der Zusammenfassung der Logdatei, der Heap Analyse und des Diagramms Dauer Garbage Collection.
- Für die Garbage Collection Analyse wurde eine eigene Perspektive (darin wird beispielsweise die Anordnung der Fenster gespeichert) definiert. Der Benutzer muss seine anderen Perspektiven nicht verändern.
- Hier kann der Benutzer das sich darunter befindende Chart anpassen.
- Ein benutzerdefiniertes Chart, welches durch den Benutzer erstellt und angepasst wurde.

#### Was das Tool nicht leistet

- Die Software kann zum jetzigen Zeitpunkt nur Logdateien der JRockit Virtual Machine Release 28 verarbeiten.
- Der Einsatz der Analysesoftware für das Tuning der Garbage Collection auf der JRockit Virtual Machine macht aktuell dann Sinn, wenn die Auswertung mittels JRockit Mission Control oder anderen Werkzeugen aufgrund von Einschränkungen (z.B. Firewall) nicht möglich ist.