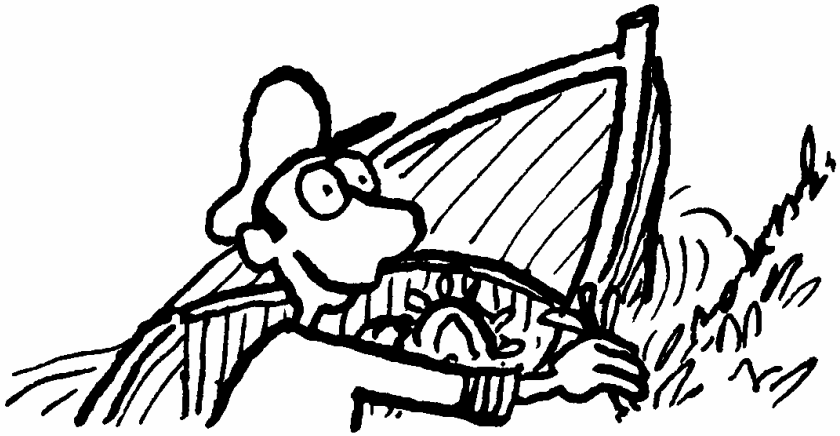
Hochseetauglich mit Open-Source-Tools

Last-/Performancetests mit Open Source

Agenda

- Seemannsgarn – IT Projekte und Performance
- Basiswissen für Last-/Performance-Tests
  - Wohin, womit, mit wem? – Performance Erfolgskriterien
  - Back- und Steuerbord – gemeinsame Sprache
  - Wohin geht die Reise? – Last- und Performance-Testpläne
  - Seemannsregeln an Bord – Gute Testscripts
  - Von Sextanten und Seekarten – Kriterien für Toolauswahl
- „Offene Quellen“ auf hoher See
  - Vorstellung
  - Vergleich
- Fazit / Zusammenfassung
- Weiterführende Links und Quellen

Seemannsgarn –
IT Projekte und Performance



Seemannsgarn – IT Projekte und Performance

Sind oder waren Sie bereits in IT-Projekten, die

- Last-/Performance-Probleme hatten?
- aufgrund von Last-/Performance-Problemen gescheitert sind?

Hatten Sie in Ihren Projekten noch nie Last-/Performance-Probleme?

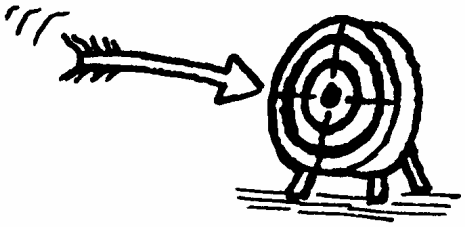
Wer arbeitet in einem Last-/Performance-Team mit > 1 Person?

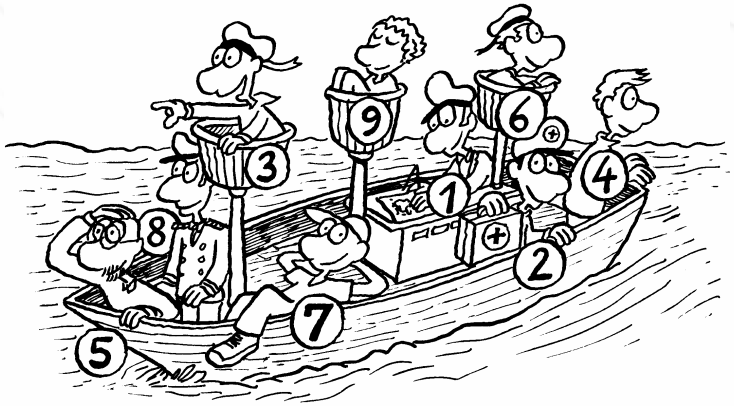
Seemannsgarn – IT Projekte und Performance

Mögliche Ursachen für Performance-Probleme:

- Kein Verantwortlicher für Last-/Performance-Tests
- Keine Dokumentation über Last- und Performance-Verhalten
- Keine Reaktion auf verändertes Last- und Performance-Verhalten
- Last-/Performance-Tests existieren nicht
- Ungenügende Test-Hardware und Software, z.B.
  - Kein isoliertes Netzwerk
  - Ungenügende Netzwerk-Bandbreite
  - „Schwachbrüstige“ Rechner als Lasttreiber
- …

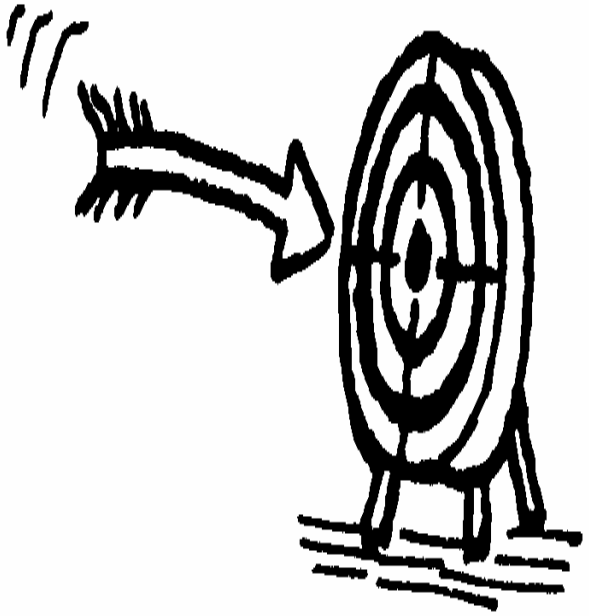
Wohin, womit, mit wem? – Performance Erfolgskriterien





Wohin, womit, mit wem? – Performance Erfolgskriterien

- Wohin? Bewußtsein und Ziele
  - Einheitliche Sprache, Definitionen, „Wording“
  - Klare Zieldefinition für alle Beteiligten, z.B. über Testpläne, Testspezifikationen, etc.

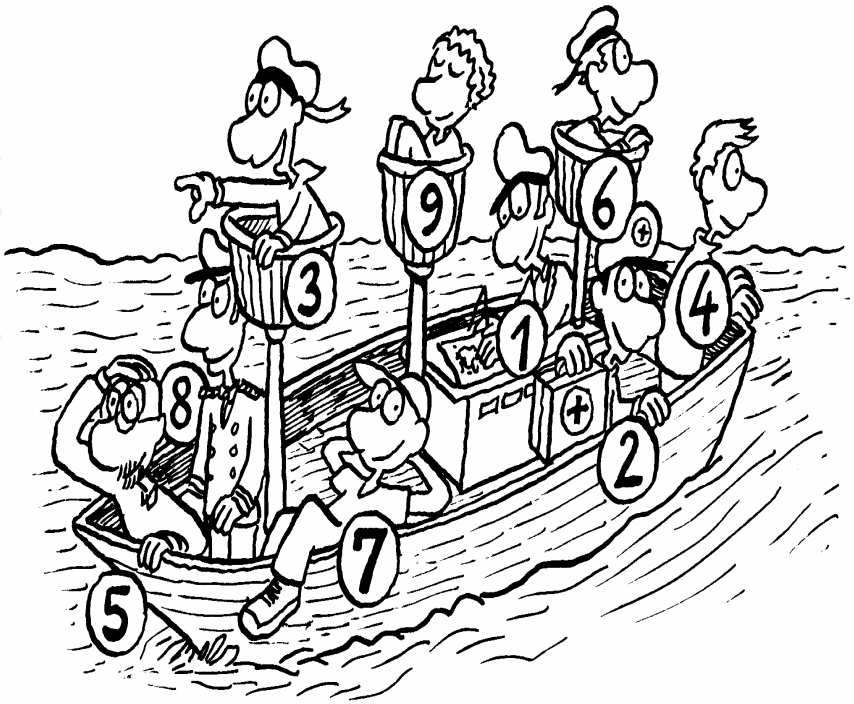


Wohin, womit, mit wem? – Performance Erfolgskriterien

- Womit? Bereitschaft zu investieren in
  - Hardware
  - Software
  - Weiterbildung
  - Manpower
  - Beratung

Wohin, womit, mit wem? – Performance Erfolgskriterien

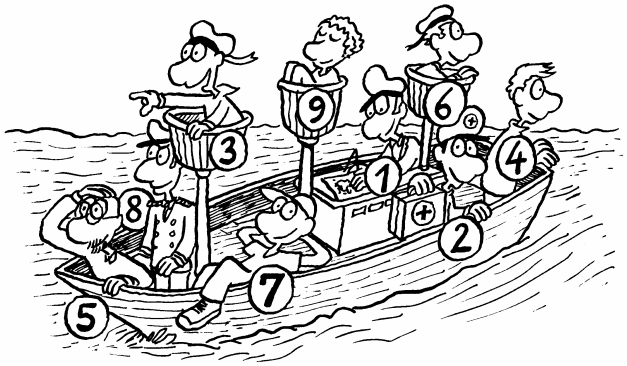
- Mit wem? Team und Kommunikation
  - Offenlegung der Beteiligten mit deren „Pflichten und Rechten“
  - Klärung der
    - Verantwortlichkeiten
    - Schnittstellen
    - Notwendigen Skills



Wohin, womit, mit wem? – Performance Erfolgskriterien

Performance Team Skills

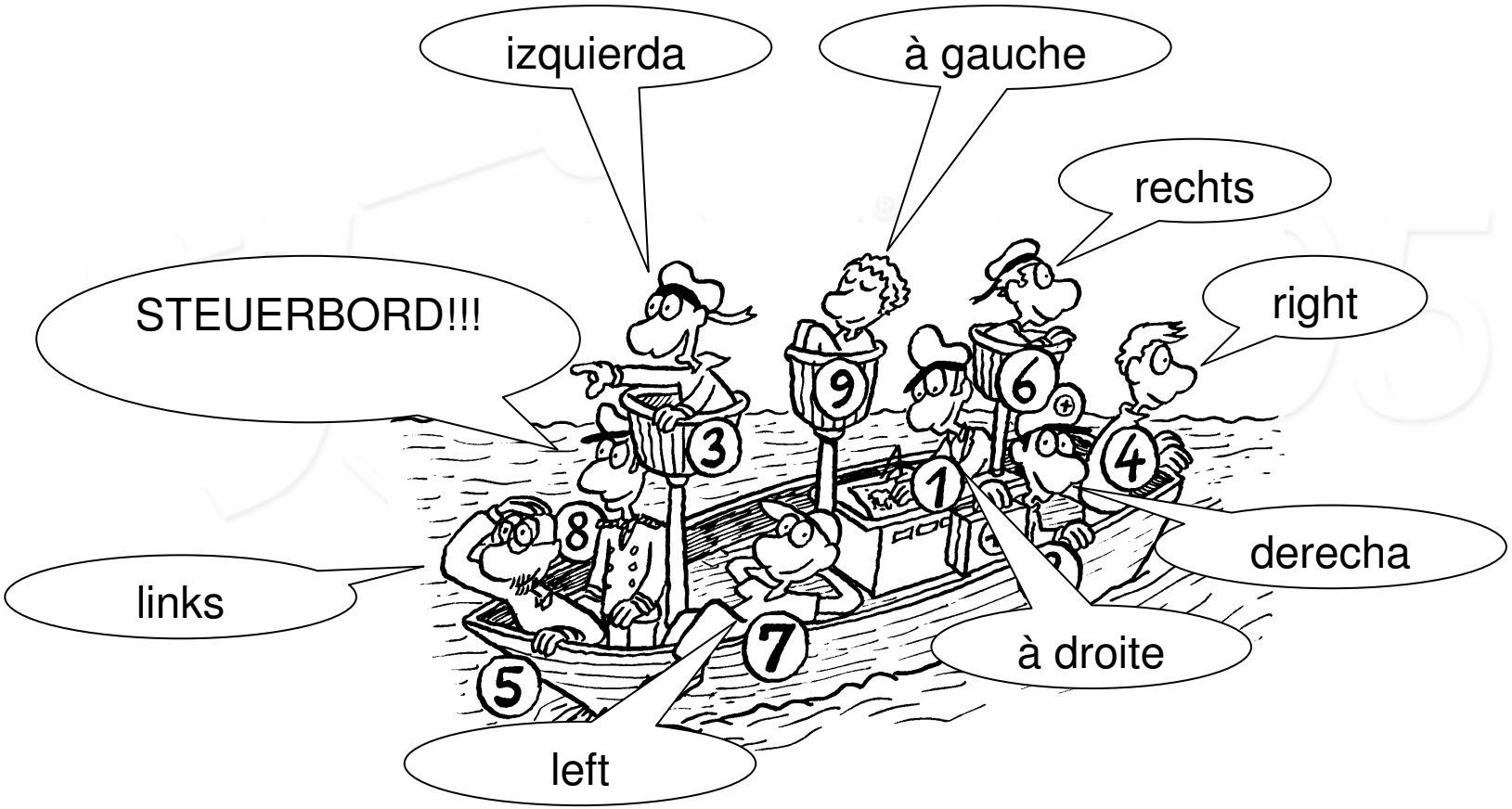
- Anforderungen
- Architektur
- Programmierung
- Testtheorie
- Testwerkzeuge
- Kommunikative Fähigkeiten



Performance Team Schnittstellen

- Management
- Marketing
- Architekten
- Entwickler
- Datenbank Administratoren
- System Administratoren
- Netzwerk Administratoren
- System Test, Qualitätsabteilung

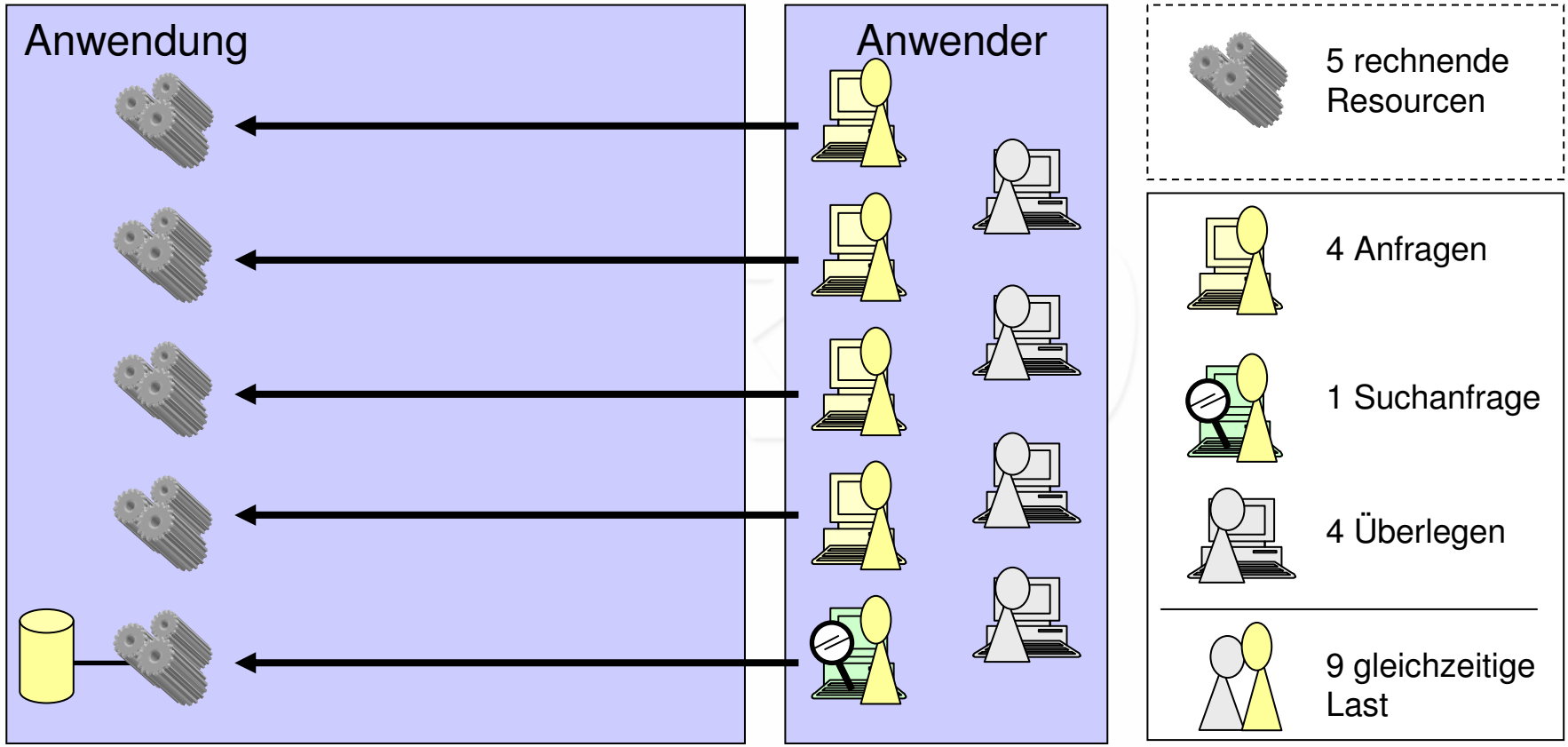
Back- und Steuerbord – gemeinsame Sprache



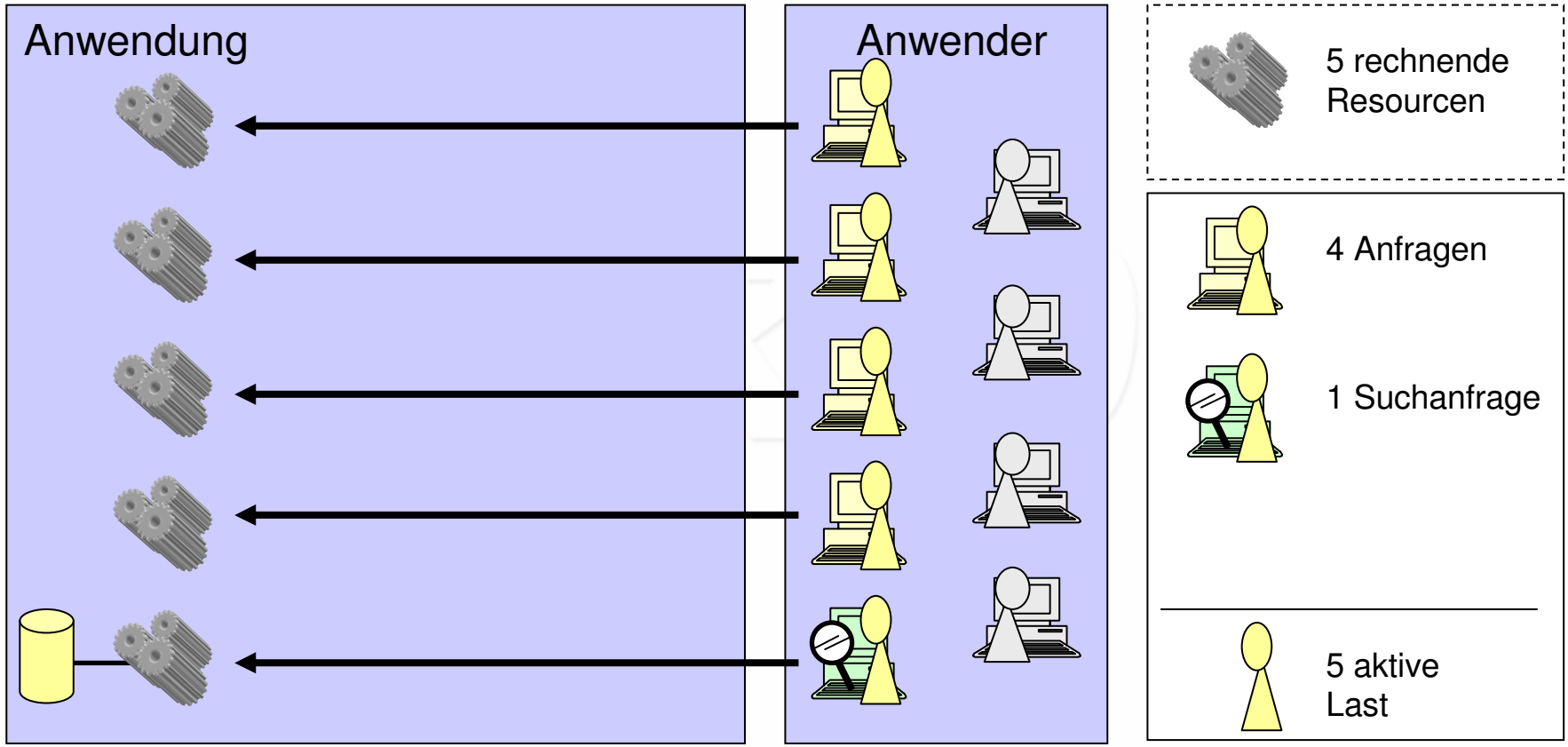
Back- und Steuerbord – gemeinsame Sprache

- Last
  - Gleichzeitige Last
  - Aktive Last
  - Spitzenlast
- Antwortzeiten
- Durchsatz und Durchsatzkurven
- Hits,Transactions,Pages & Users

Back- und Steuerbord – Gleichzeitige Last

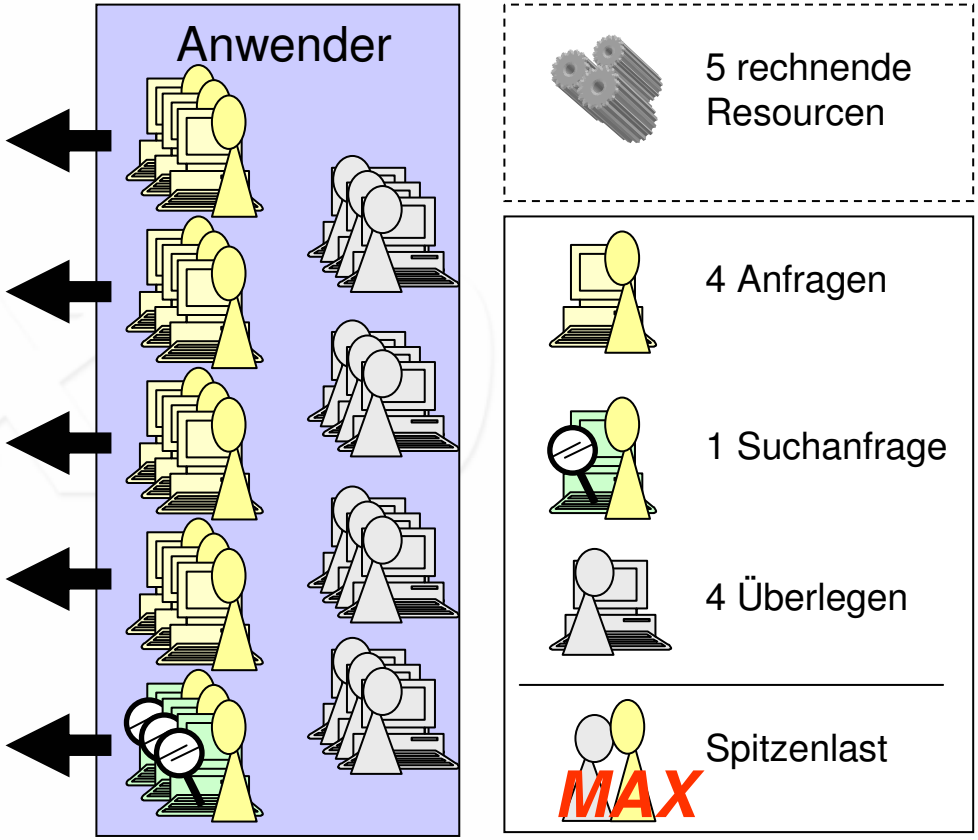


Back- und Steuerbord – Aktive Last

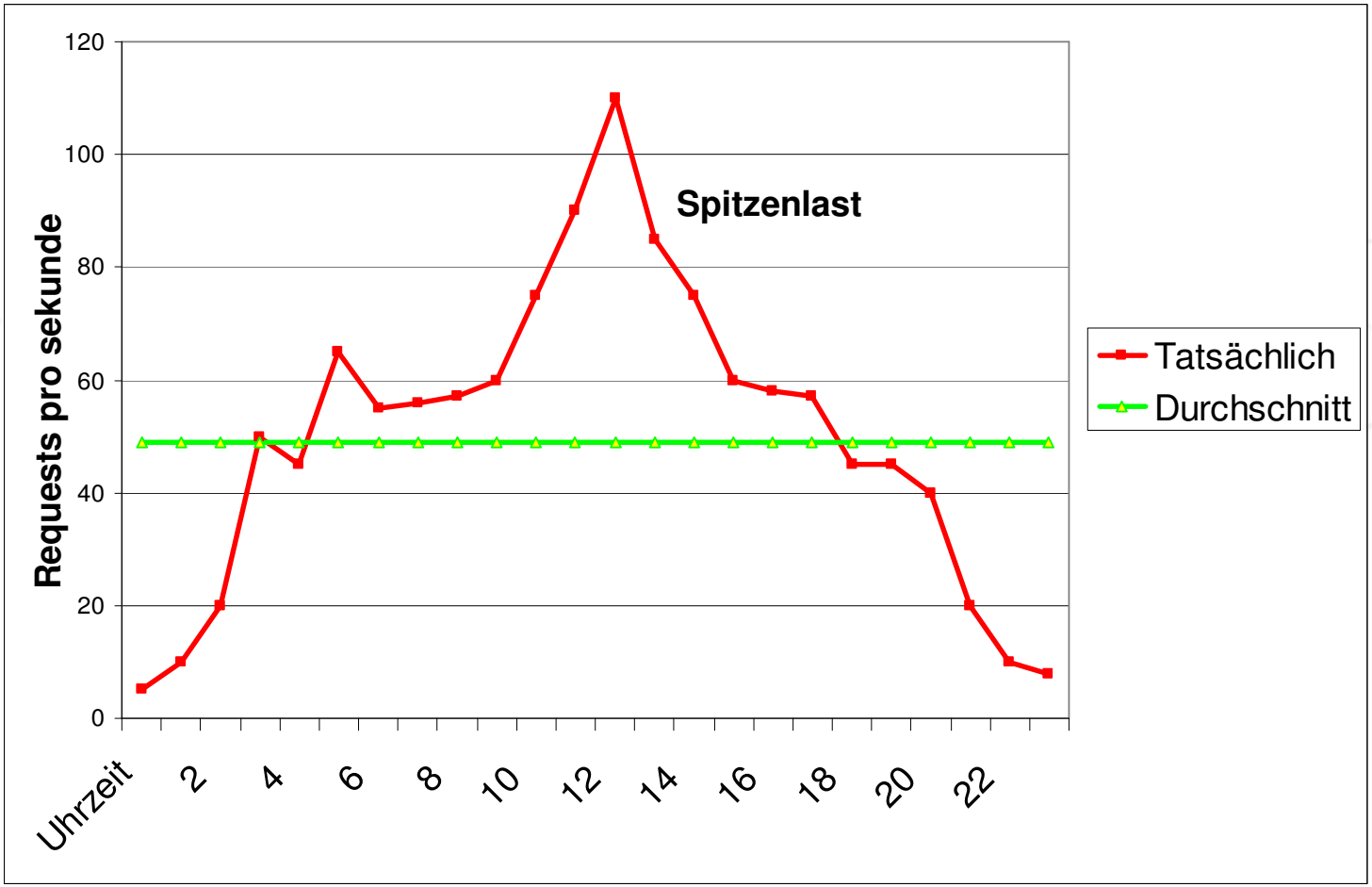


Back- und Steuerbord – Spitzenlast

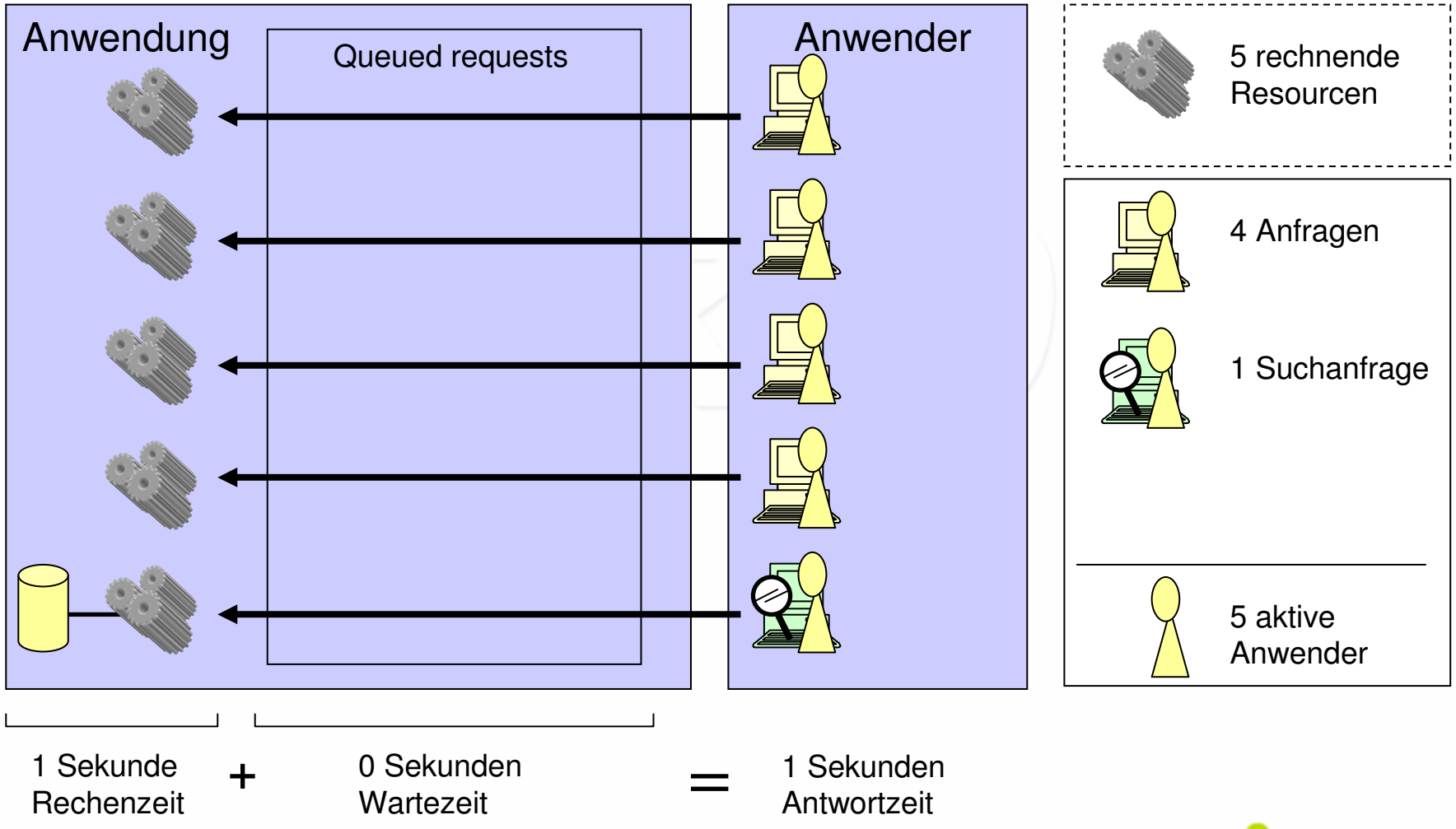
- Maximale gleichzeitige Last
- Starke Abhängigkeit von betrachteten Zeitraum und Anwendung
  - Welche Minute der Stunde?
  - Welche Stunde des Tags?
  - Welchen Tag des Jahres?
- Starke Abweichung von durchschnittlicher Last (Amazon, Pizzadienste, etc.)



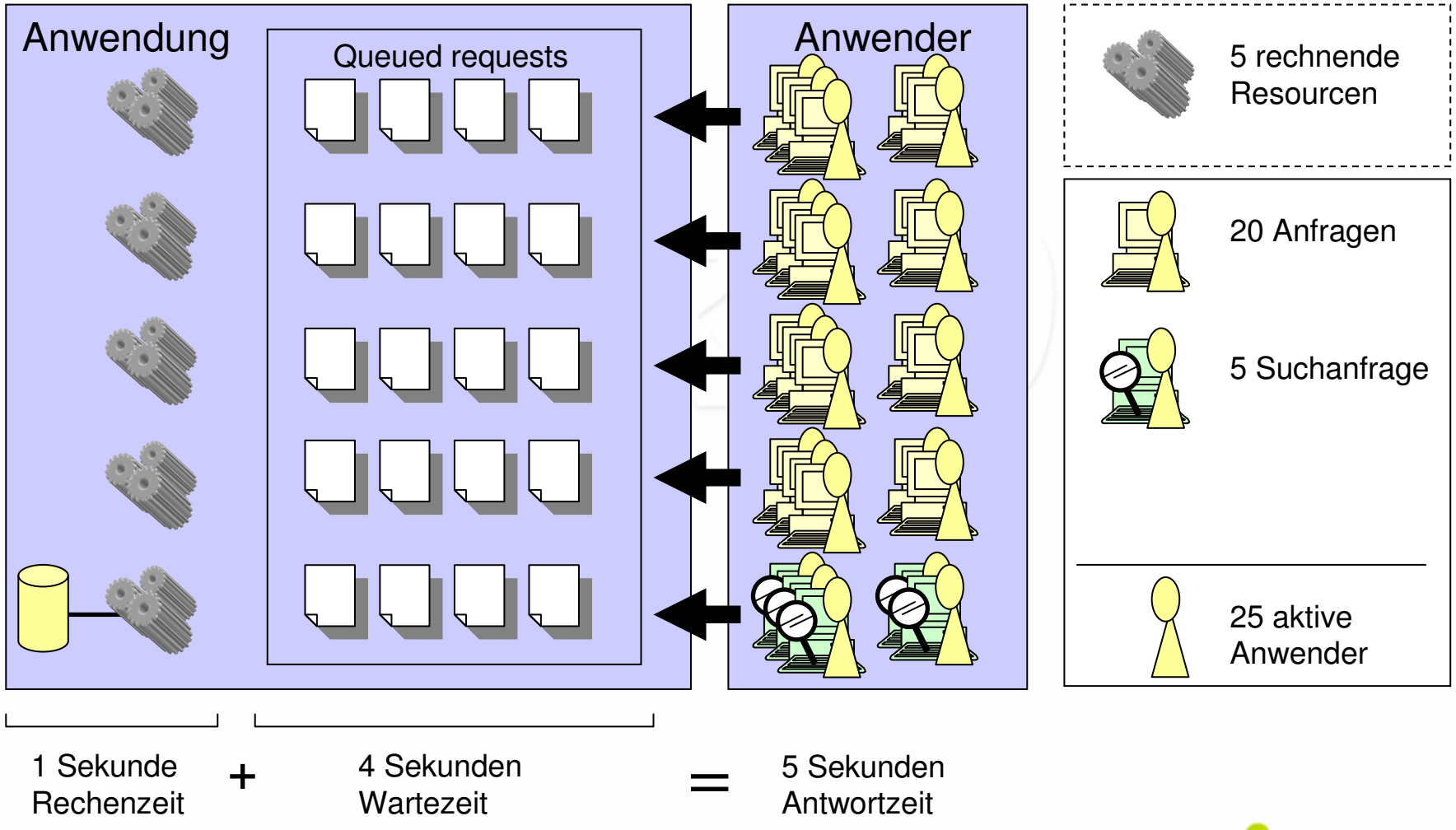
Back- und Steuerbord – Spitzenlast



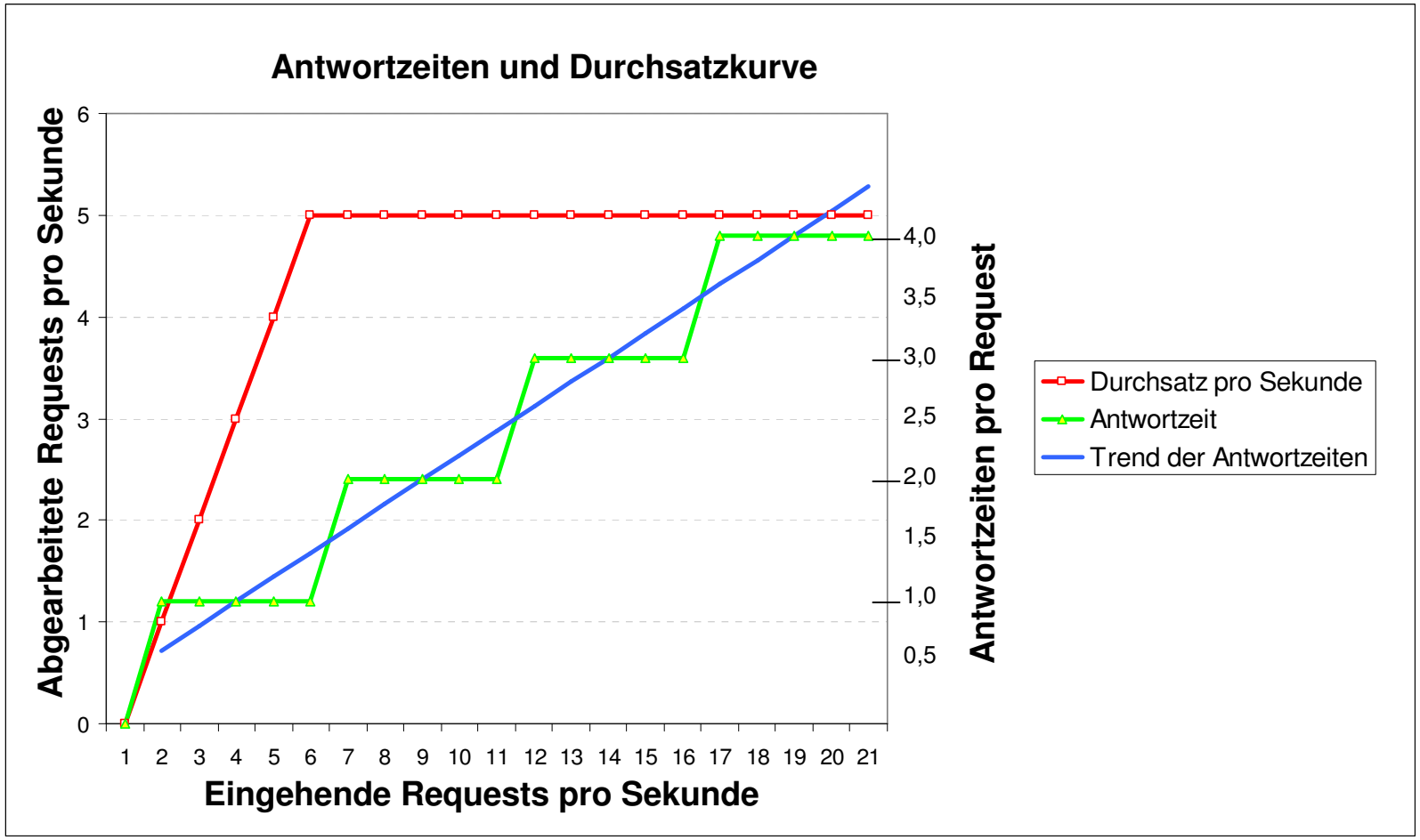
Back- und Steuerbord – Antwortzeiten



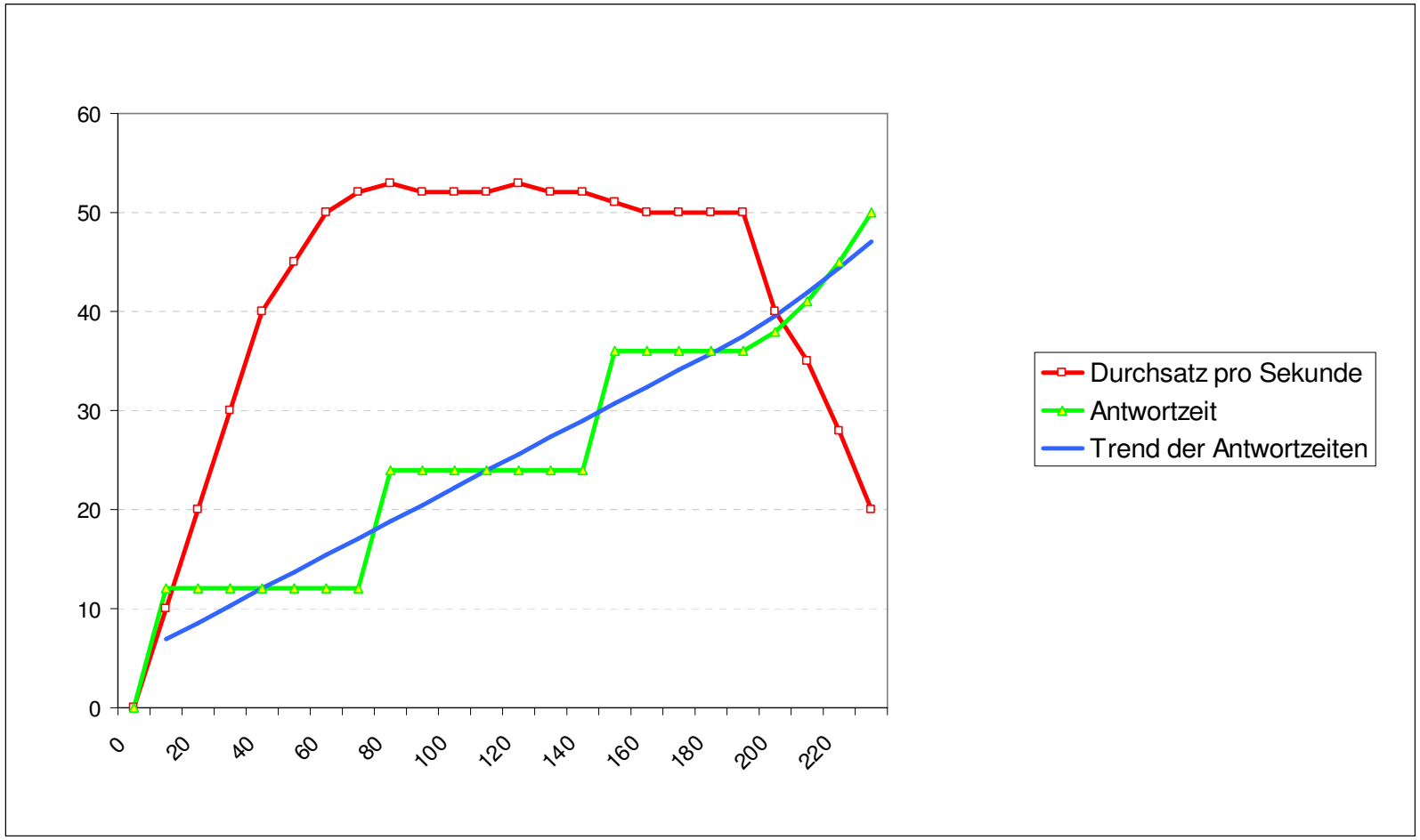
Back- und Steuerbord – Antwortzeiten



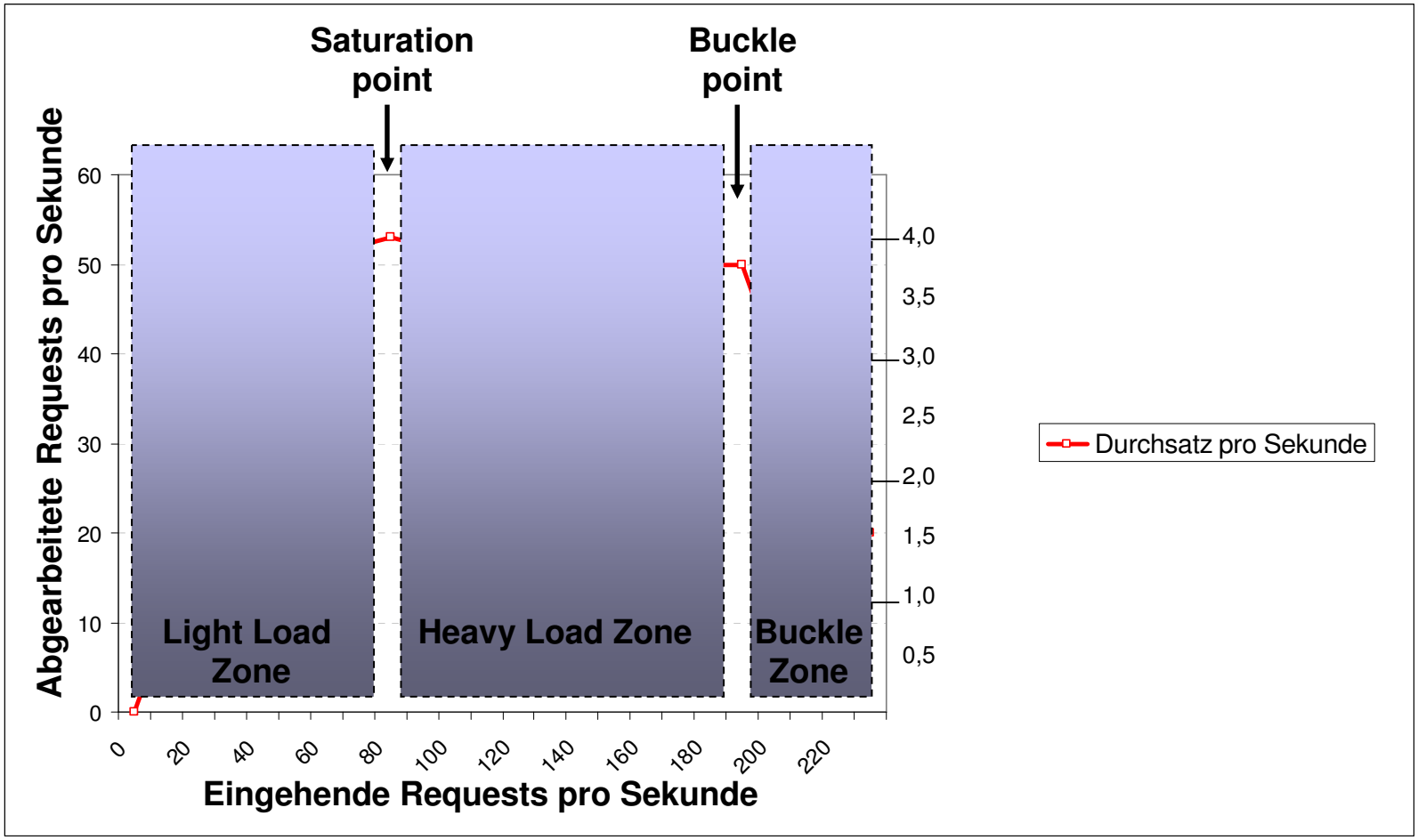
Back- und Steuerbord – Antwortzeiten und Durchsatz



Back- und Steuerbord – Antwortzeiten und Durchsatz



Back- und Steuerbord – Durchsatz



Back- und Steuerbord – Hits, Transactions, Pages & Users

- Hit
  - HTTP request
  - (Entire page)
  - (Entire visit by user)
- Transaction
  - HTTP request and response pair
- Page
  - >= 1 transactions
- User
  - >= 1 page
  - Workflows

Wohin geht die Reise –
Last- und Performance-Testpläne

Wohin geht die Reise – Voraussetzungen und Ziele der Testplanung

- Voraussetzungen
  - Funktionaler Test erfolgreich
  - Reproduzierbare Testumgebung verfügbar
- Testziele
  - Abzudeckende Spitzenlast festlegen
  - Erwartete Antwortzeiten
  - Notwendige Messungen: Was, wann, wie und wie oft?
- Strategien
  - Validierung durch Testen kleiner Teile
  - Einfachst beginnen, Erfahrungen sammeln

Wohin geht die Reise – Spitzenlast bestehender Anwendungen

- Analyse eigener oder ähnlicher Anwendungen
  - HTTP Access logs
    - „Traffic patterns“
    - Anwenderzahl, Sessions
    - …
  - Datenbank logs
    - Connections
    - Transactions
    - …

Wohin geht die Reise – Spitzenlast über Schätzungen

„10.000 Hits pro Tag“

„10.000 Seiten pro Tag“

„10 Millionen Anwender im Jahr“

„100.000 Anwender pro Tag“

…

Wohin geht die Reise – Spitzenlast über Seiten pro Tag

- Annahmen und Erwartungen Variante 1
  - 10.000 Seiten/Tag
  - 30 % Verkehr in Spitzenstunde
  - 7 Requests/Seite
  - Verhältnis dynamischer zu statischer Anteil 2 : 5

Berechnung:

10.000 Seiten/Tag * 30 % Verkehr in Spitzenstunde = 3.000 Seiten/Spitzenstunde
3.000 Seiten/Stunde / (3.600 Sekunden/Stunde) = 0,83 Seiten/Sekunde

3.000 Seiten/Stunde * 7 Requests/Seite = 21.000 HTTP Requests/Stunde
21.000 HTTP Requests/Stunde / (3.600 Sekunden/Stunde) = 5,83 Requests/Sekunde

5,83 Requests/Sekunde * 2/7 = 1,67 Dynamische Requests/Sekunde
5,83 Requests/Sekunde * 5/7 = 4,16 Statische Requests/Sekunde

Wohin geht die Reise – Spitzenlast über Seiten pro Tag

- Annahmen und Erwartungen Variante 2
  - 10.000 Seiten/Tag
  - 8 Stunden Tag
  - Verhältnis von Spitzen zu durchschnittlicher Last 5 : 1
  - 7 Requests/Seite
  - Verhältnis dynamischer zu statischer Anteil 2 : 5

Berechnung:

10.000 Seiten/Tag / (8 Stunden/Tag) = 1.250 Seiten/Stunde
1.250 Seiten/Stunde * 5 = 6.250 Seiten/Stunde
6.250 Seiten/Stunde / (3.600 Sekunden/Stunde) = 1,7 Seiten/Sekunde

6.250 Seiten/Stunde * 7 Requests/Seite = 43.750 HTTP Requests/Stunde
43.750 HTTP Requests/Stunde / (3.600 Sekunden/Stunde) = 12,15 Requests/Sekunde

12,15 Requests/Sekunde * 2/7 = 3,47 Dynamische Requests/Sekunde
12,15 Requests/Sekunde * 5/7 = 8,68 Statische Requests/Sekunde

Wohin geht die Reise – Spitzenlast über Anwender pro Tag

- Annahmen und Erwartungen
  - 100.000 Anwender/Tag
  - 30 % des Verkehrs in Spitzenstunde
  - 10 Minuten durchschnittliche Aufenthaltsdauer
  - 5 Seiten während eines 10 minütigen Aufenthalts
  - 2 dynamische und 5 statische Requests pro Seite

Berechnung:

100.000 Anwender/Tag * 30 % Verkehr in Spitzenstunde = 30.000 Anwender/Spitzenstunde
30.000 Anwender/Stunde / (3.600 Sekunden/Stunde) = 8,3 Neue Anwender/Sekunde

8,3 Neue Anwender/Sekunde * 600 Sekunden = 4.998 Neue Anwender in 10 Min (~ 5.000)
(5.000 Anwender * 5 Seiten / Anwender) / 600 Sekunden = 41,667 Seiten/Sekunde (~ 42)

7 Requests/Seite * 42 Seiten/Sekunde = 294 Requests/Sekunde
2 dynamische Requests/Seite * 42 Seiten/Sekunde = 84 dynamische Requests/Sekunde
5 statische Requests/Seite * 42 Seiten/Sekunde = 210 statische Requests/Sekunde

Wohin geht die Reise – Spitzenlast pro Server

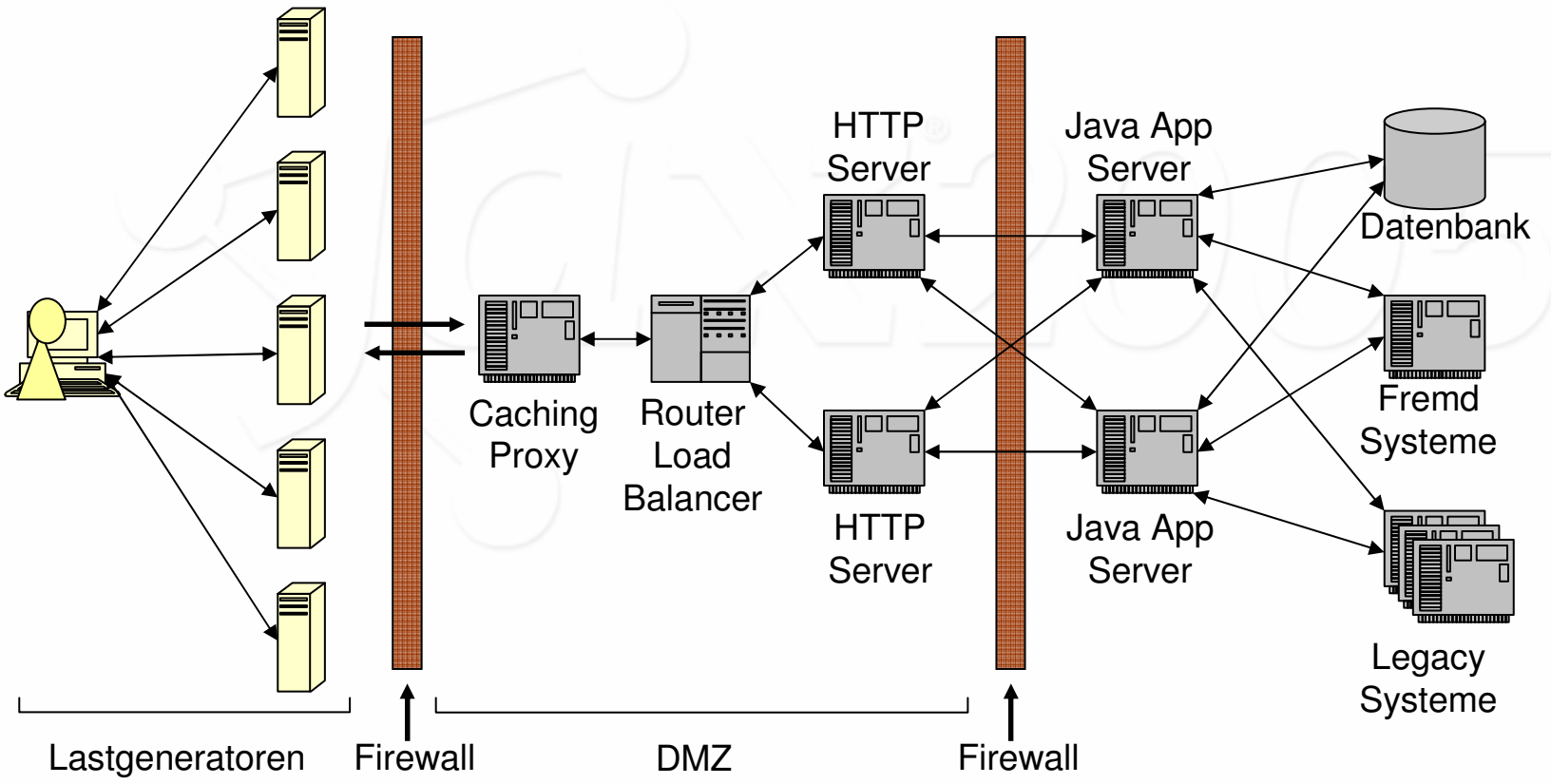
- Annahmen und Erwartungen Variante 3
  - 10 Server
  - 42 Seiten/Sekunde
  - 294 Requests/Sekunde
  - 5.000 Anwender in der Spitzenzeit
  - 50 % Puffer für jeden Server

Berechnung:

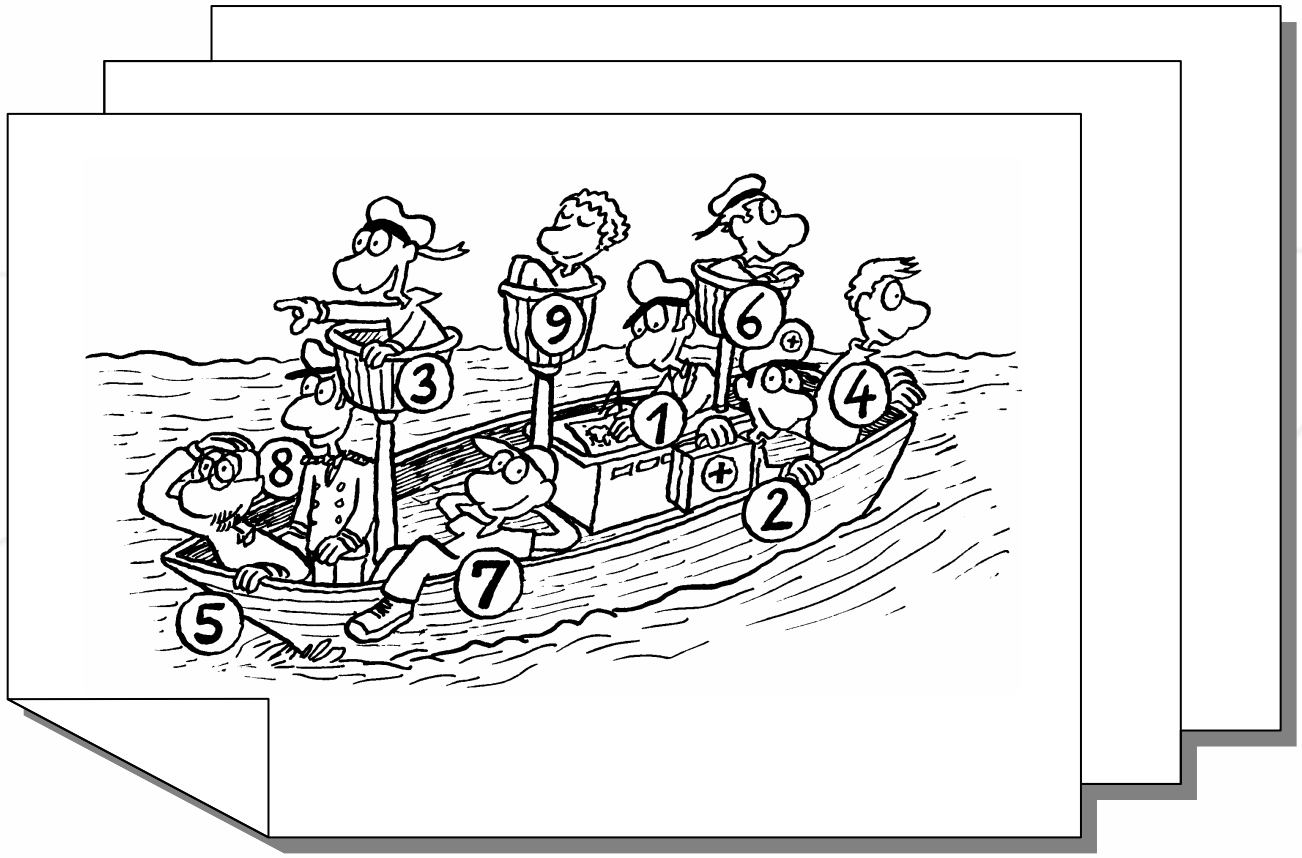
(42 Seiten/Sekunde * 2 Pufferkapazität) / 10 Server = 8,4 Seiten/Sekunde
(294 Seiten/Sekunde * 2 Puffer) / 10 Server = 58,8 Requests/Sekunde

(5000 Anwender in Spitzenzeit * 2 Puffer) / 10 Server = 1.000 Anwender in Spitzenzeit

Wohin geht die Reise – Einfachst beginnen, Erfahrungen sammeln



Seemannsregeln an Bord – Gute Testscripts



Seemannsregeln an Bord – Inhalte von Testscripts

- Session Daten
  - Anwenderdaten (z.B. User Account)
  - Anwendungsdaten (z.B. Session Id, Cookies)
- Timing
  - Denkzeiten zwischen Aufrufen
  - Verhalten abhängig von Antwortzeiten
- Anfragen
  - URLs
  - SOAP Requests

Seemannsregeln an Bord – Größe und Granularität von Testscripts

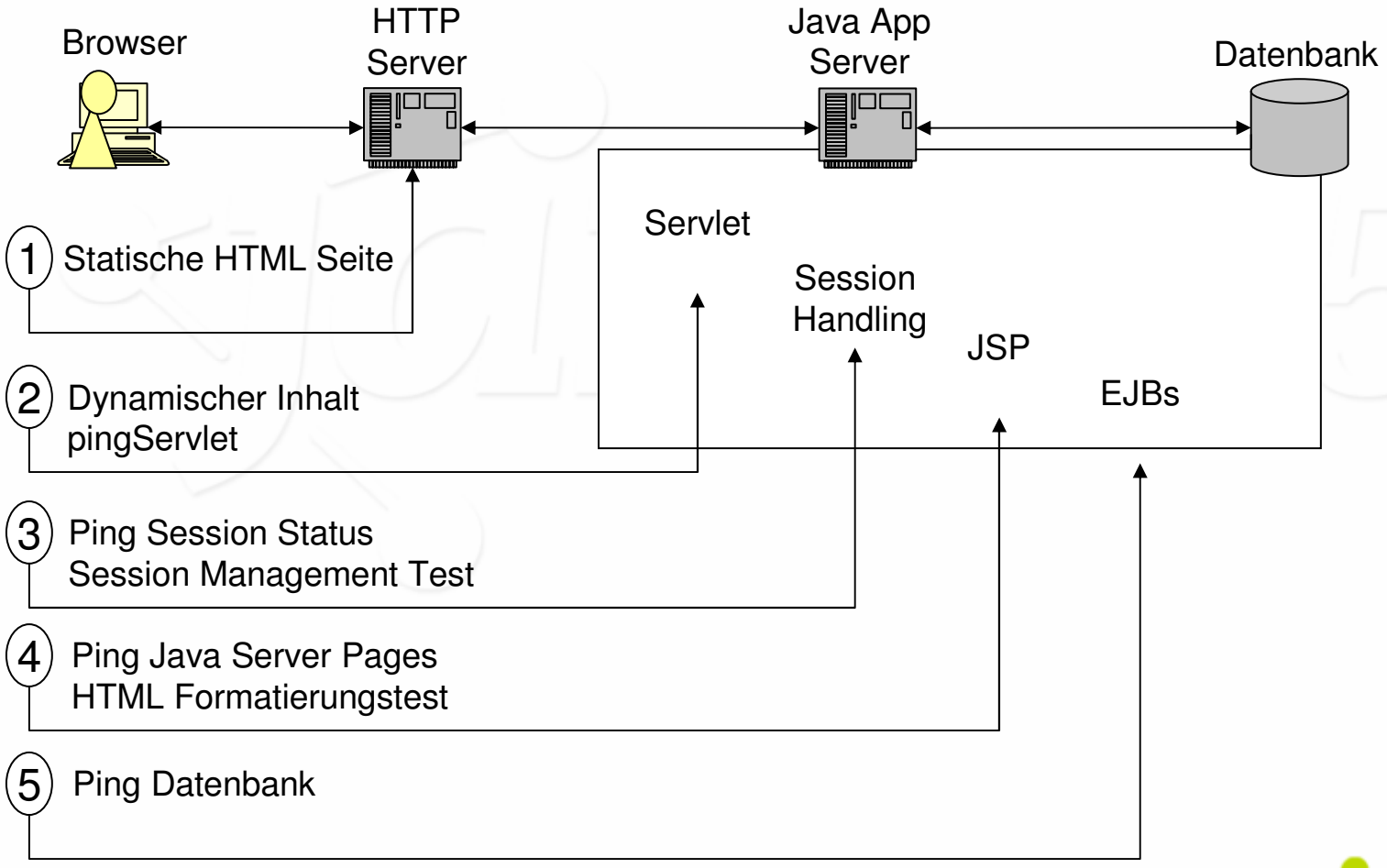
- Gute Testscripts sind
  - So kurz wie möglich, so lang wie nötig
  - Atomar (Prüfe, ob Anwender alle Bestandteile benötigt)
    - Login – Browse – Purchase – Logout
    - Browse – Login – Purchase – Logout
- Verbessert bzw. ermöglicht
  - Wartung
  - Erweiterbarkeit
  - Zusammenstellung neuer Szenarien
  - Veränderte Gewichtungen

Seemannsregeln an Bord – Testszenarien

- Erstellung virtueller User mit
  - Atomaren Scripts
  - Workflow
- Abbildung Last (Szenario) durch Gewichtung virtueller User

Virtueller User	Gewichtung
Browse	77 %
Browse – Purchase	3 %
Search	18 %
Search – Purchase	2 %

Seemannsregeln an Bord – Primitive Testscripts



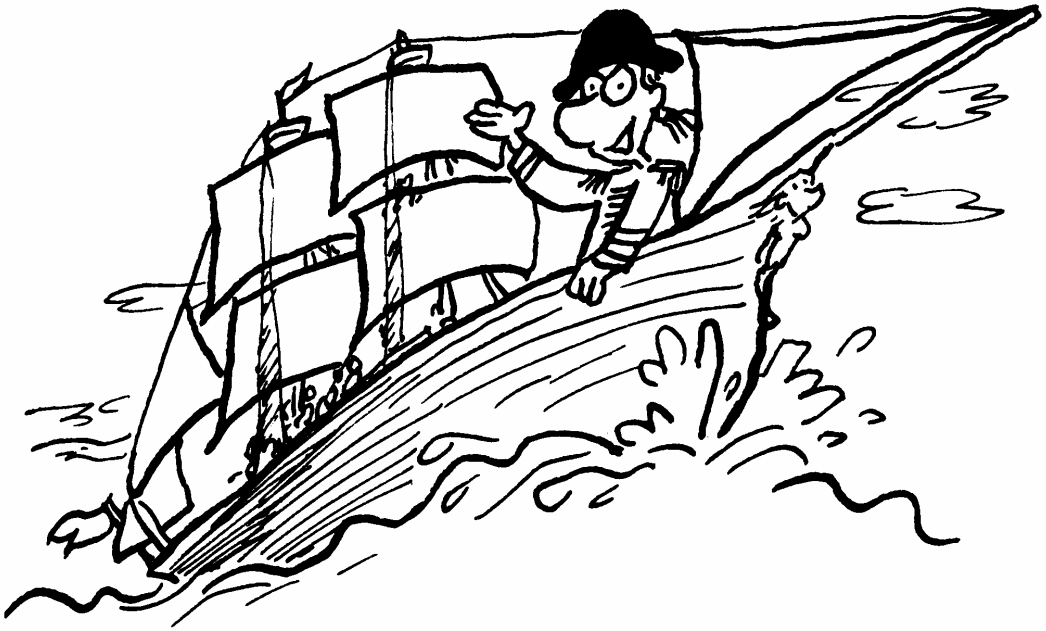
Seemannsregeln an Bord – Dynamisierung von Testscripts

- Was
  - Recording ähnlicher Aktionen
  - Unterschiede der Skripte sind Kandidaten für Dynamisierung
- Womit
  - Auswahl von Werten
    - aus Dateien
    - Parameterlisten
  - Dynamisierung der Linkauswahl
    - Parsing der zurückgegebenen Seite
    - Ermittlung der Links
    - Zufällige Auswahl eines Links

Seemannsregeln an Bord – Vermeidung der häufigsten Fehler

- Fehlerhafte Links
  - Links in falscher Reihenfolge
  - Links zu anderen Sites
- Ausschließliche Konzentration auf triviale oder komplexe Seiten
- Hart kodierte Cookies
- Unpassende Warte- und Denkzeiten
- Ungenügende Parametrierung
- Idealisiertes Benutzerverhalten (z.B. ohne Logout)

Von Sextanten und Seekarten – Kriterien zur Toolauswahl



Von Sextanten und Seekarten – Kriterien zur Toolauswahl

- Erfolgreiche Tests erfordern
  - Simulation der angestrebten Produktionsumgebung
  - Gewinnung von verläßlichen und wiederholbaren Messungen
- Kriterien für Tools
  - User Simulation
  - Script Sprache
  - Steuerung
  - Auswertung
  - Sonstiges

Von Sextanten und Seekarten – Kriterien zur Toolauswahl: User Simulation

Kriterium	Erklärung
Cookies	Ermittlung, Aufnahme und Wiedergabe von HTTP Cookies.
Caching	Emulation des Caching Verhaltens von Web Browsern.
Link Ermittlung	Möglichkeit aus zurückgegebene Seiten Links zu ermitteln und weiter zu verwenden.
Denkzeiten	Fähigkeit Denkzeiten des Anwenders zu simulieren, zentral anzupassen, etc.
IP Spoofing	Emulation des Zugriffs verschiedener IP Adressen auf das zu testende System. Notwendig für den Test von Load Balancern.
WAN/LAN	Emulation des Verhaltens verschiedener Netzwerk Infrastrukturen.
Netzwerk-geschwindigkeiten	Emulation verschiedener vom Anwender genutzter Netzwerkgeschwindigkeiten.

Von Sextanten und Seekarten – Kriterien zur Toolauswahl: Steuerung

Kriterium	Erklärung
Last verteilbar	Fähigkeit die Generierung der Last über viele Lastgeneratoren zu verteilen.
Zentrale Kontrolle	Verteilte Lastgeneratoren sollen zentral gesteuert werden.
Skalierbarkeit	Fähigkeit Anzahl virtueller User zu generieren und der dafür erforderliche Resourcenverbrauch (abh. von Skript-Komplexität).
Fernzugriff	Möglichkeit die Ausführung von Testskripten remote zu steuern, evtl. auch über Firewalls hinweg
Ant TaskDef	Ausführung des Werkzeugs als Ant Task
Kommandozeile	Fähigkeit das Werkzeug über die Kommandozeile zu bedienen
Szenarien	Möglichkeit aus verschiedenen Skripten gewichtete Szenarien zusammenzustellen

Von Sextanten und Seekarten – Kriterien zur Toolauswahl: Script Sprache

Kriterium	Erklärung
Protokolle	Kommunikationsprotokolle, die aufgenommen, verändert und wieder abgespielt werden können.
Aufnahme & Wiedergabe	Möglichkeiten Skripte ablaufen zu lassen und Debugging durchzuführen.
Erweiterbarkeit	Fähigkeit die Funktionalität des Werkzeugs zu erweitern.
Editoren	Schnittstellen, welche das Werkzeug zum Zwecke der Script Editierung unterstützt.
Korrelation	Angebotene Unterstützung bei der Ersetzung von Werten in dynamische Daten für erfolgreiche Wiedergabe.
Parametrierung	Automatische Veränderung dynamischer Daten für eine genauere Emulation richtiger Anwender.
Komplexität, Verbreitung	Verbreitungsgrad und Lernkurve der verwendeten Script Sprache.

Von Sextanten und Seekarten – Kriterien zur Toolauswahl: Auswertung

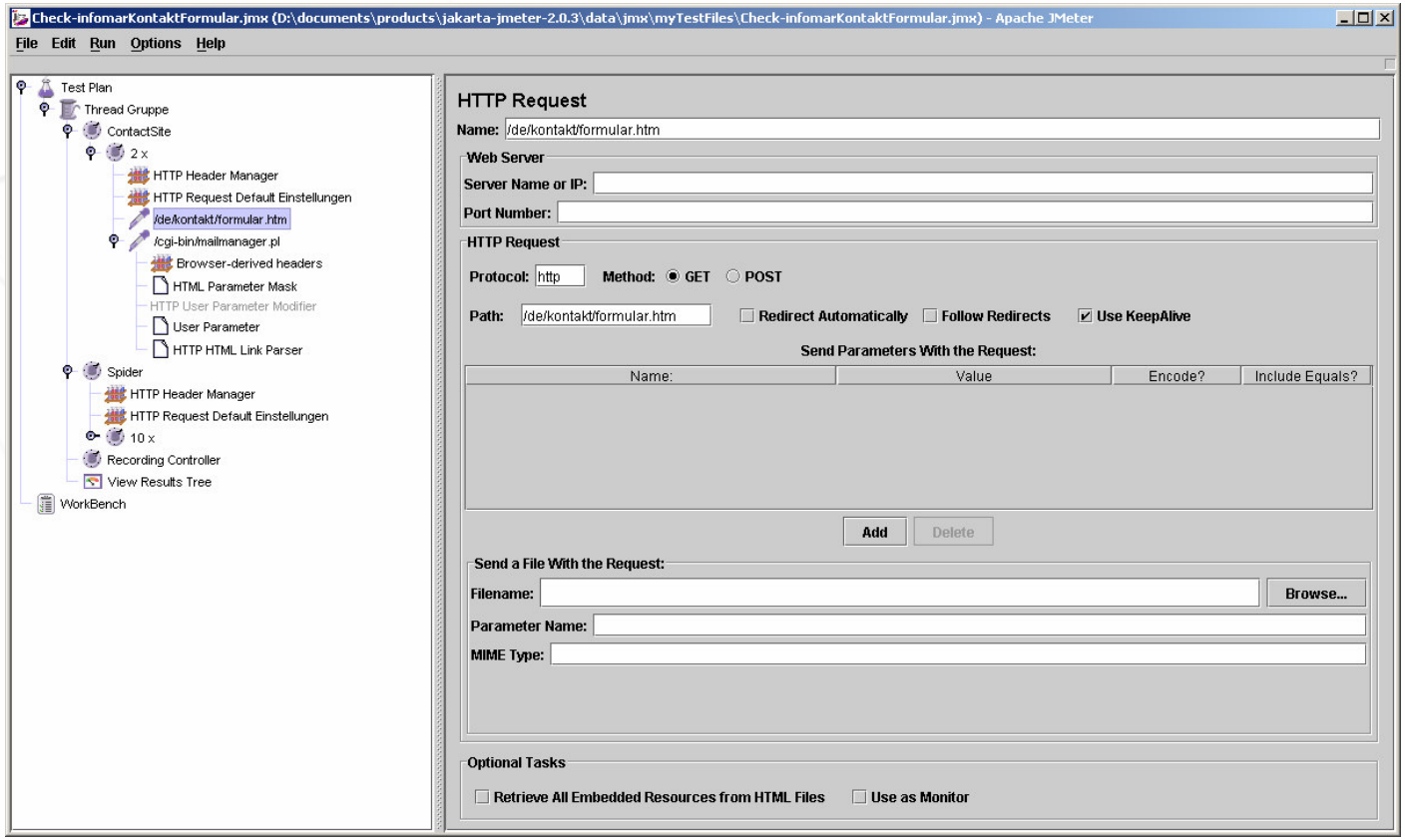
Kriterium	Erklärung
Monitoring	Messung des Resourcenverbrauchs während eines Testlaufs und für Performance Berichte.
Berichte & Analyse	Möglichkeiten Ergebnisse eines Testlaufs zu untersuchen und graphisch dazustellen. Am Besten mit Timer und den beobachteten Resourcen.
Warm-up & Cool-down	Unterscheidung von gemessenen Werten nach Phasen. Möglichkeiten Warm-Up- und Cool-down-Phasen herauszurechnen.
Eingeschwungene Meßwerte	Fähigkeit des Werkzeugs, einen eingeschwungenen Zustand des zu testenden System festzustellen und erst in diesem Zeitraum relevante Messungen durchzuführen.
Überprüfungen	Unterstützung von Checks, ob ein Testlauf in Ordnung war
Konsolidierte Ergebnisse	Zusammenführung von Ergebnissen von verschiedenen Testmaschinen

Von Sextanten und Seekarten – Kriterien zur Toolauswahl: Verschiedenes

Kriterium	Erklärung
Kosten Anschaffung	Anschaffungskosten für Werkzeug und Lizenzen, ohne Upgrades und Support.
Kosten pro virtueller Anwender	Zusätzliche Kosten auf Basis der unterstützten virtuellen User. Kosten für HW nicht eingerechnet.
Support & Beratung	Verfügbarkeit und Kosten für Support und Beratung rund um das Werkzeug.
Trainings	Trainingsangebote und deren Kosten für das Werkzeug.
Verfügbarkeit Sourcecode	Möglichkeit auf den Source Code der Anwendung zuzugreifen.
Anforderungen OS	Unterstützte Betriebssysteme und Java Versionen des Werkzeugs.
Anforderungen HW	Hardware-Anforderungen des Werkzeugs.

„Offene Quellen“ auf hoher See

„Offene Quellen“ auf hoher See – Kurzvorstellung JMeter



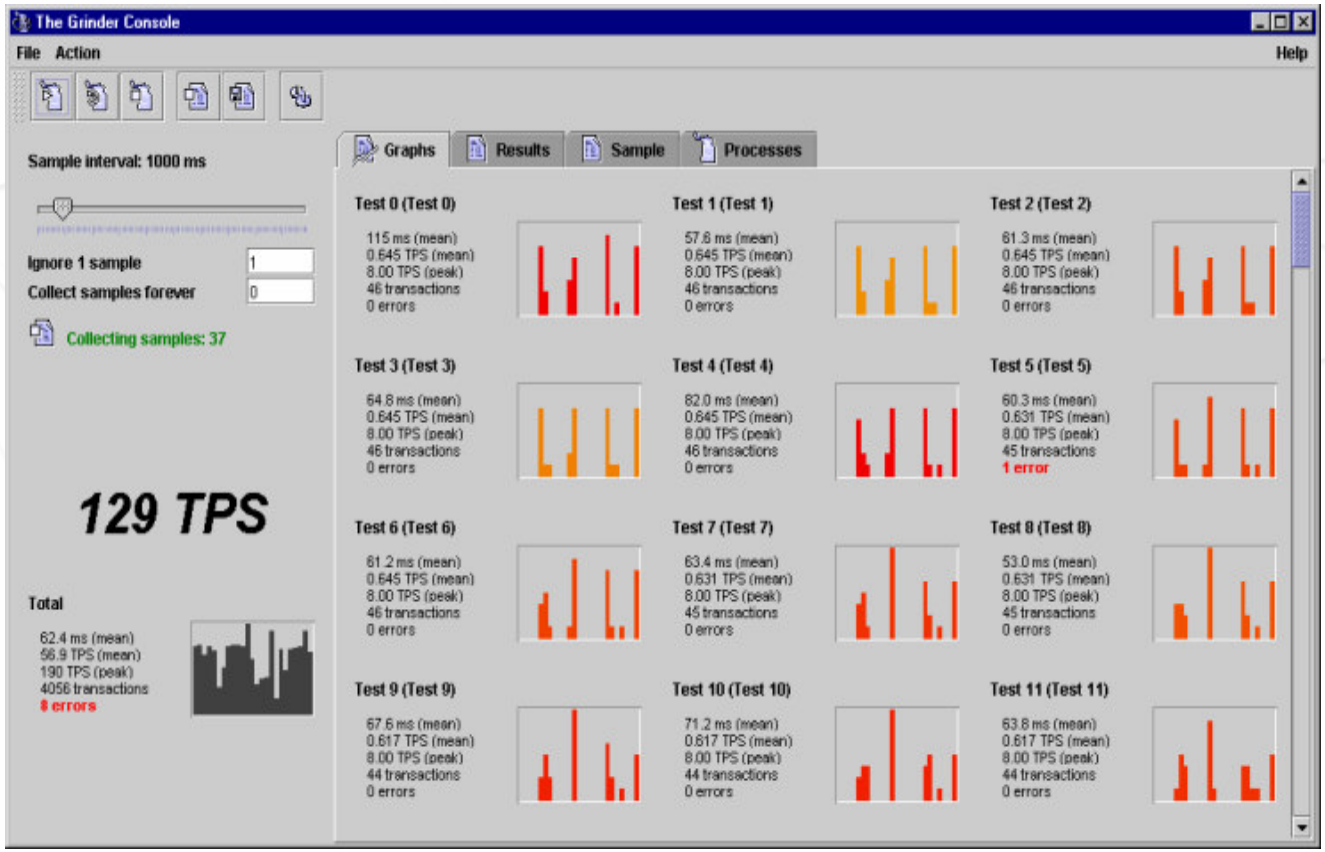
„Offene Quellen“ auf hoher See – Kurzvorstellung JMeter

- Historie
  - Version 1.0 von Stefano Mazzocchi (02/99)
  - Aktuelle Version 2.0.3 von Michael Stover (03/05)
  - Release in halbjährigem Abstand
- Ziele / Zweck
  - Ursprünglich: Test Performance von Apache JServ
  - Simulation von Lastszenarien
  - Messung von Performance
- Open Source
  - unter http://jakarta.apache.org/jmeter/ (Apache License)
  - 729 Klassen, ~ 51 Kilo NCSS

„Offene Quellen“ auf hoher See – Kurzvorstellung JMeter

- Features
  - GUI basierte Erstellung von Testplänen
  - HTTP Proxy Recording
  - Unterstützung vieler Protokolle (HTTP, SOAP, Webserivce, LDAP, FTP, etc.)
  - Verteilte Lastgenerierung
  - Einfache Erweiterbarkeit von JMeter
  - Details siehe Session „(J)Meterweise integrieren“ ☺

„Offene Quellen“ auf hoher See – Kurzvorstellung Grinder



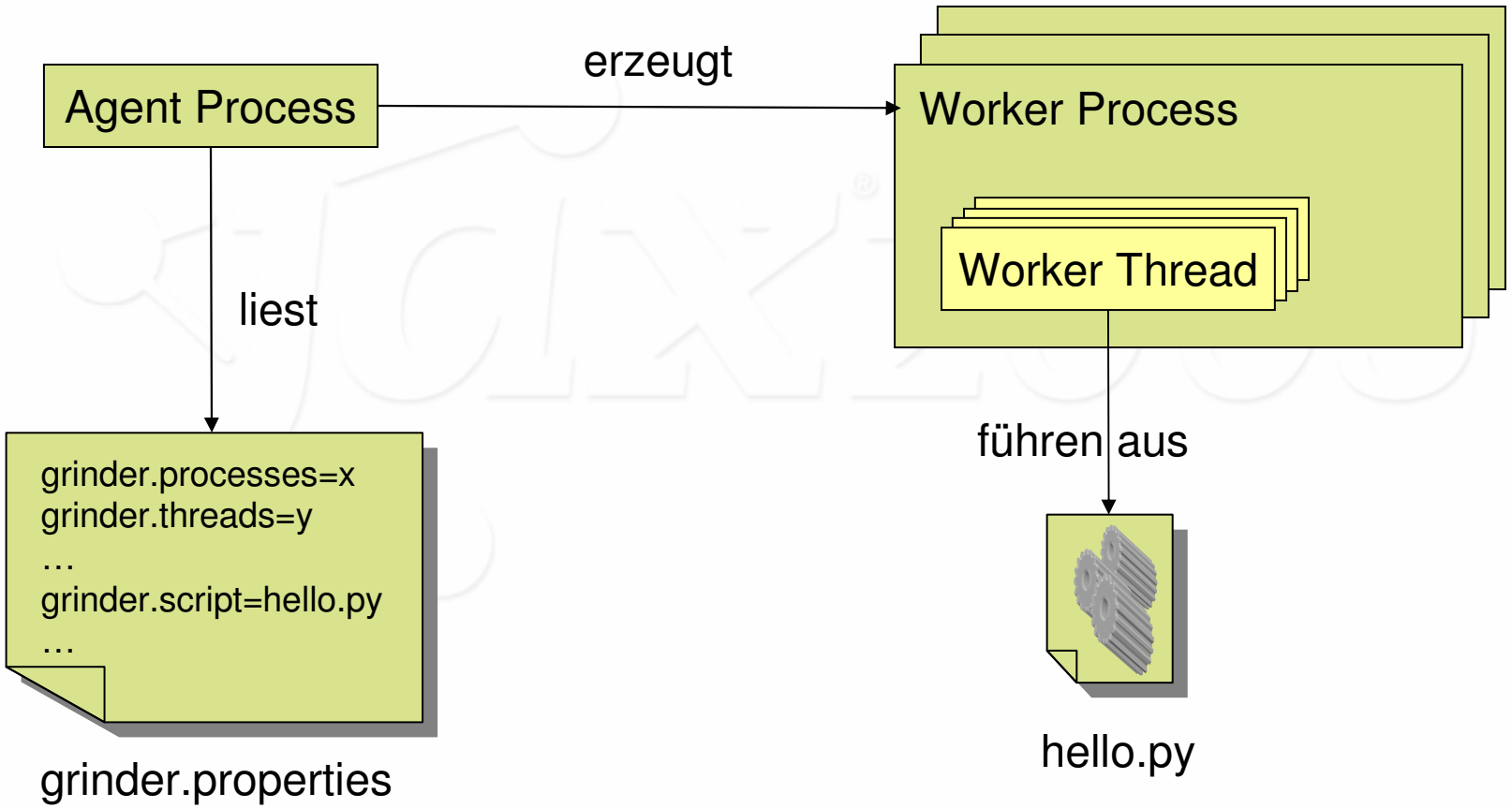
„Offene Quellen“ auf hoher See – Kurzvorstellung Grinder

- Historie
  - Version 1.0 von Paco Gomez (2000)
  - Aktuelle Version 3.0-beta25 von Philip Aston (03/05 mit Jython als Skriptsprache)
  - Releasezyklen im letzten Jahr sehr rege, wegen beta Status
- Ziele / Zweck
  - Ursprünglich: Buchbeigabe zu „Professional Java 2 Enterprise Edition with Bea Web Logic Server“
  - Simulation von Lastszenarien
  - Messung von Performance
- Open Source
  - unter http://grinder.sourcefoge.net/ (BSD License)
  - 311 Klassen, ~ 28 Kilo NCSS

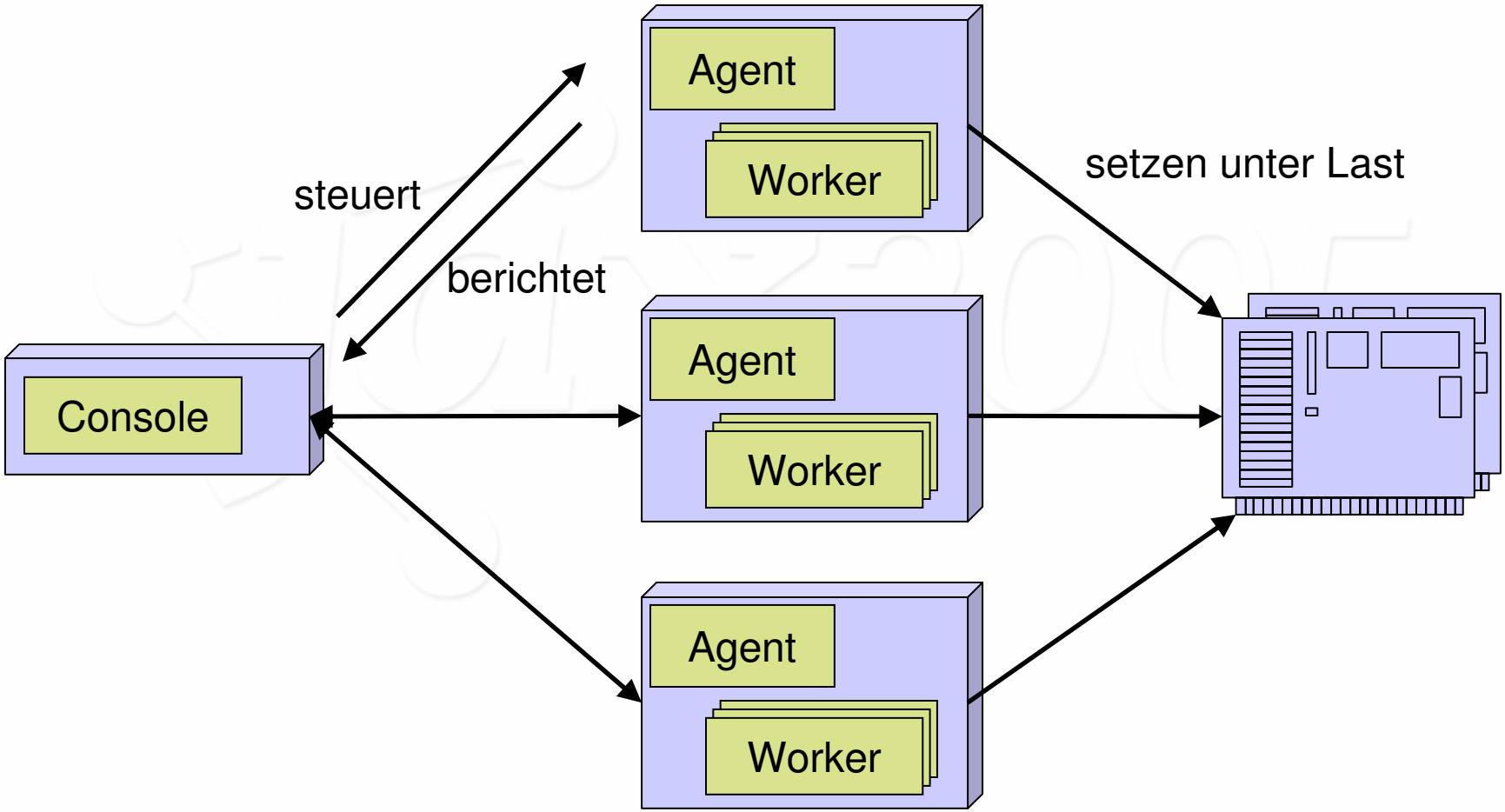
„Offene Quellen“ auf hoher See – Kurzvorstellung Grinder

- Features
  - Controller – Agenten Architektur
    - Controller Console zur Steuerung und Anzeige der Ergebnisse
    - Agenten als Lastgeneratoren
  - Lastgenerierung mit Jython Scripts
  - Beinhaltet erweiterbaren TCPProxy um
    - HTTP(S) Verkehr zu „beobachten“
    - HTTP(S) Requests aufzunehmen

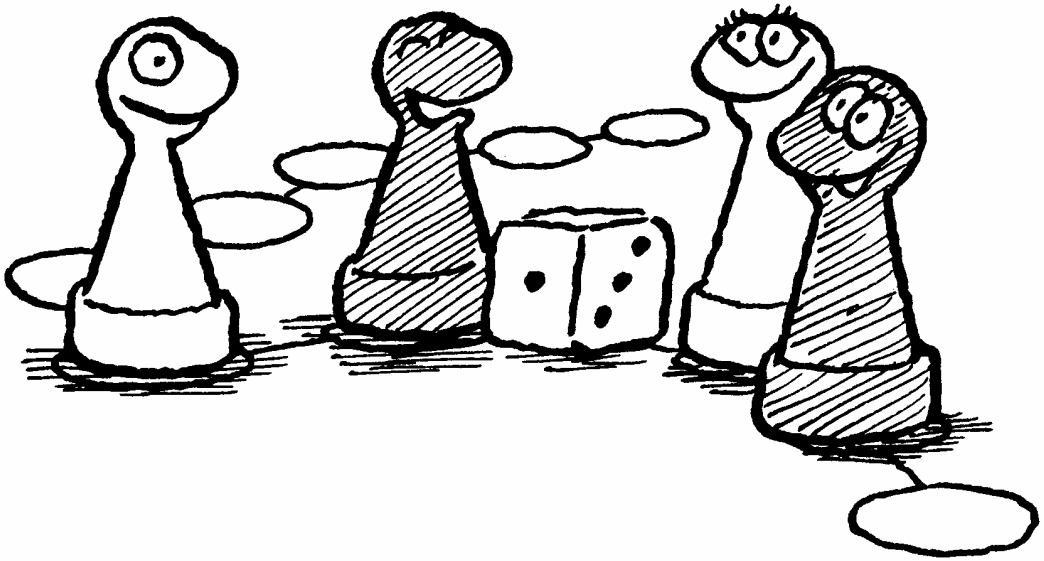
„Offene Quellen“ auf hoher See – Kurzvorstellung Grinder



„Offene Quellen“ auf hoher See – Kurzvorstellung Grinder



„Offene Quellen“ auf hoher See –
JMeter & Grinder im Vergleich



„Offene Quellen“ auf hoher See – Kriterienkategorie User Simulation

Kriterium	JMeter	Grinder
Cookies	Ja, HTTPCookieManager	Ja
Caching	Beim Recording wird Browserverhalten berücksichtigt. Nicht zuschaltbar	
Link Ermittlung	Preprocessor HtmlLinkParser mit HttpRequest Sampler	Nein
Denkzeiten	Verschiedene Timer-Implementierungen, beim Recording nicht berücksichtigt	grinder.sleep, wird auch mit aufgenommen
IP Spoofing	Nein	Nein
WAN/LAN	Nein	Nein
Netzwerk-geschwindigkeiten	Nein	Nein

„Offene Quellen“ auf hoher See – Kriterienkategorie Steuerung

Kriterium	JMeter	Grinder
Last verteilbar	Ja	Ja
Zentrale Kontrolle	Ja	Ja
Skalierbarkeit	*1)	*1)
Fernzugriff	Ja, aber nicht durch Firewalls	Ja, aber nicht durch Firewalls
Ant TaskDef	Ja	Nein
Kommandozeile	Ja	Ja
Szenarien	Nur durch Merge verschiedener Testpläne in einen gemeinsamen Testplan	Nur auf Script Basis, keine weitere Unterstützung

„Offene Quellen“ auf hoher See – Kriterienkategorie Script Sprache (I)

Kriterium	JMeter	Grinder
Protokolle	Builtin: HTTP, SSL, FTP, LDAP, SOAP, Webservices	Builtin: HTTP(S) 1.0 / 1.1
Aufnahme & Wiedergabe	HTTP 1.0 / 1.1 (recordable) Kein Debugging	HTTP(S) 1.0 / 1.1 (recordable) Was Jython bietet ...
Erweiterbarkeit	Plugins für JMeter, Einbindung von Beanshell-Skripts	Leichtgewichtiges Framework für Einsatz von Jython, alle Freiheit bei Skript Implementierung
Editoren	Eigenes GUI, Baum – Detail	Editor mit Syntax-Highlighting für Jython

„Offene Quellen“ auf hoher See – Kriterienkategorie Script Sprache (II)

Kriterium	JMeter	Grinder
Korrelation	Preprocessor HtmlLinkParser	Nein
Parametrierung	Benutzerab- und unabhängige Variablen, Counter, Einlesen von Werten aus Dateien	Nur auf Script Basis, keine weitere Unterstützung
Komplexität, Verbreitung	JMeter Dateiformat trotz xml eher exotisch, Beanshell reicht nicht an die Popularität von Jython heran	Gute Verbreitung von Jython aufgrund der Popularität von Python und Java

„Offene Quellen“ auf hoher See – Kriterienkategorie Auswertung

Kriterium	JMeter	Grinder
Monitoring	Nur für Tomcat 5.0.19 über management web service	Keine
Berichte & Analyse	Jede Menge Visualizer, Aggregate Reports, Result Tree, etc.	Lediglich Ansicht der Grinder Console sowie die Logdateien der Testagenten
Warm-up & Cool-down	RampUp Unterstützung	RampUp WorkerProcesses, grinder.processIncrement (Intervall)
Eingeschwungene Meßwerte	Möglichkeit für Visualizer den Scope zu begrenzen	Ignorieren von Proben möglich
Überprüfungen	Unterstützung von speziellen, erweiterbaren Assertions	Nur auf Script Basis, keine weitere Unterstützung
Konsolidierte Ergebnisse	*2)	*2)

„Offene Quellen“ auf hoher See – Kriterienkategorie Verschiedenes (I)

Kriterium	JMeter	Grinder
Kosten Anschaffung	Keine	Keine
Kosten pro virtueller Anwender	Keine	Keine
Support & Beratung	Kein professioneller Support verfügbar, User manual ok, Dokumentation bzgl. Architektur/ Erweiterungen spärlich.	Kein professioneller Support verfügbar. In Auslieferung und auf Website verfügbare Infos sehr gut.
Trainings	Nicht verfügbar	Nicht verfügbar

„Offene Quellen“ auf hoher See – Kriterienkategorie Verschiedenes (II)

Kriterium	JMeter	Grinder
Verfügbarkeit Sourcecode	Open-Source (apache License) unter http://jakarta.apache.org/jmeter	Open-Source (BSD) unter http://grinder.sourceforge.net/
Anforderungen OS	Ab Java 1.3 (empfohlen wird 1.4)	Ab Java 1.3.1
Anforderungen HW	*1)	*1)

*1) Benchmark steht noch aus

*2) Prüfung steht noch aus

Fazit / Zusammenfassung

- JMeter
  - Ermöglicht grafische Erstellung von Testplänen (+)
  - Gute Grundausstattung an Visualizern, Sampler, Controller, etc. (+)
  - Gute Erweiterbarkeit durch eigene Sampler, Timer, Controller, etc. (+)
  - Kein Testplan übergreifendes Include (-)
  - Kein HTTPS Recording (-)
  - Keine Verwaltung von Testplänen (-)
- Grinder
  - Leichtgewichtiges Framework mit mächtiger Skriptsprache (+)
  - Erweiterbarer TCPProxy unterstützt HTTPS Recording (+)
  - Gute Dokumentation (+)
  - Wenig grafische Auswertungen (-)
  - Viele Features nur auf Skript-Basis, z.B. Timing, Szenarien, … (-)
  - Rudimentäre Skriptverwaltung (-)

Weiterführende Links und Quellen

- Bücher
  - Performance Analysis for Java Web Sites, Stacy Joines, Ruth Willenbourg and Ken Hygh, Addison Wesley, 2003
  - J2EE Performance Testing, Peter Zadrony, Expert Press, 2002
  - J2EE-Entwicklung mit Open-Source-Tools, Martin Backschat / Stefan Edlich, Spektrum Verlag, 2004
- Artikel
  - Overview of load test tools, Buret Julien, Droze Nicholas, 2003
  - Wie man sich (test-)bettet, so liegt man, Sabine Winkler, Ulf Krum, 2003
- Links
  - JMeter Website http://jakarta.apache.org/jmeter
  - Grinder Website http://grinder.sourceforge.net/

Besten Dank

Martin Heider
Infomar software
mh@infomar.de
www.infomar.de