Asynchronität

Jonathan Teige (735692)

Marc Seeger (752872)

Thomas Herkenhoff (738205)

Agenda

- Allgemeines
- Herausforderungen
- Konzepte
- Fazit

ALLGEMEINES

Begriffsdefinition

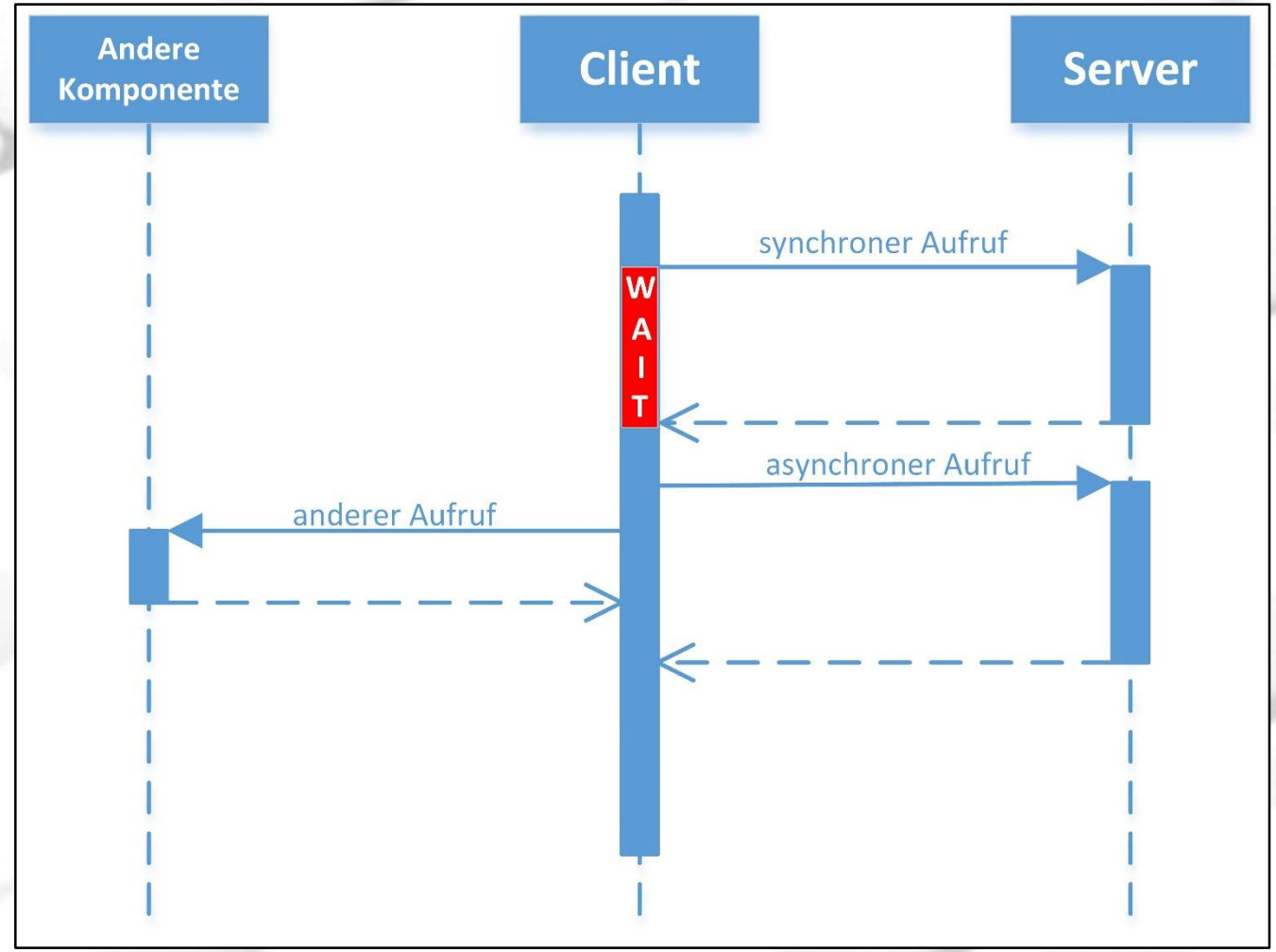
- Asynchron (griechisch) ➔ nicht gleichzeitig
  - „*a*“ ≙ nicht
  - „*syn*“≙ zusammen
  - „*chrónos*“≙ Zeit

Begriffsdefinition

“Asynchroner Nachrichtenverkehr: Hierbei ist der Sender nur bis zur Ablieferung der Nachricht an das Transportsystem blockiert, das heißt der Sender kann weiterarbeiten, ohne auf die Antwort des Empfängers warten zu müssen. Diese Transaktionsform wird auch als unidirektionale, mitteilungsorientierte Transaktion bezeichnet.”

(Hansen / Neumann, Wirtschaftsinformatik I, 8. Auflage 2002, UTB-Verlag, Seite 167)

Nutzen

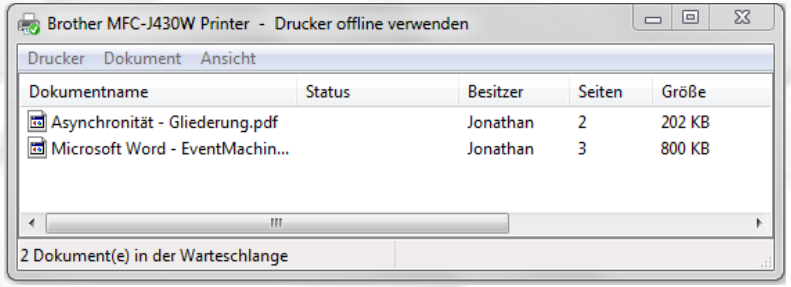

Andere Komponente
Client
Server
synchroner Aufruf
WAIT
asynchroner Aufruf
anderer Aufruf

Vorteile

- Intelligente Nutzung von Ressourcen
- Effiziente Zeitnutzung
- Responsivität
- Dynamik

Anwendungsbeispiele

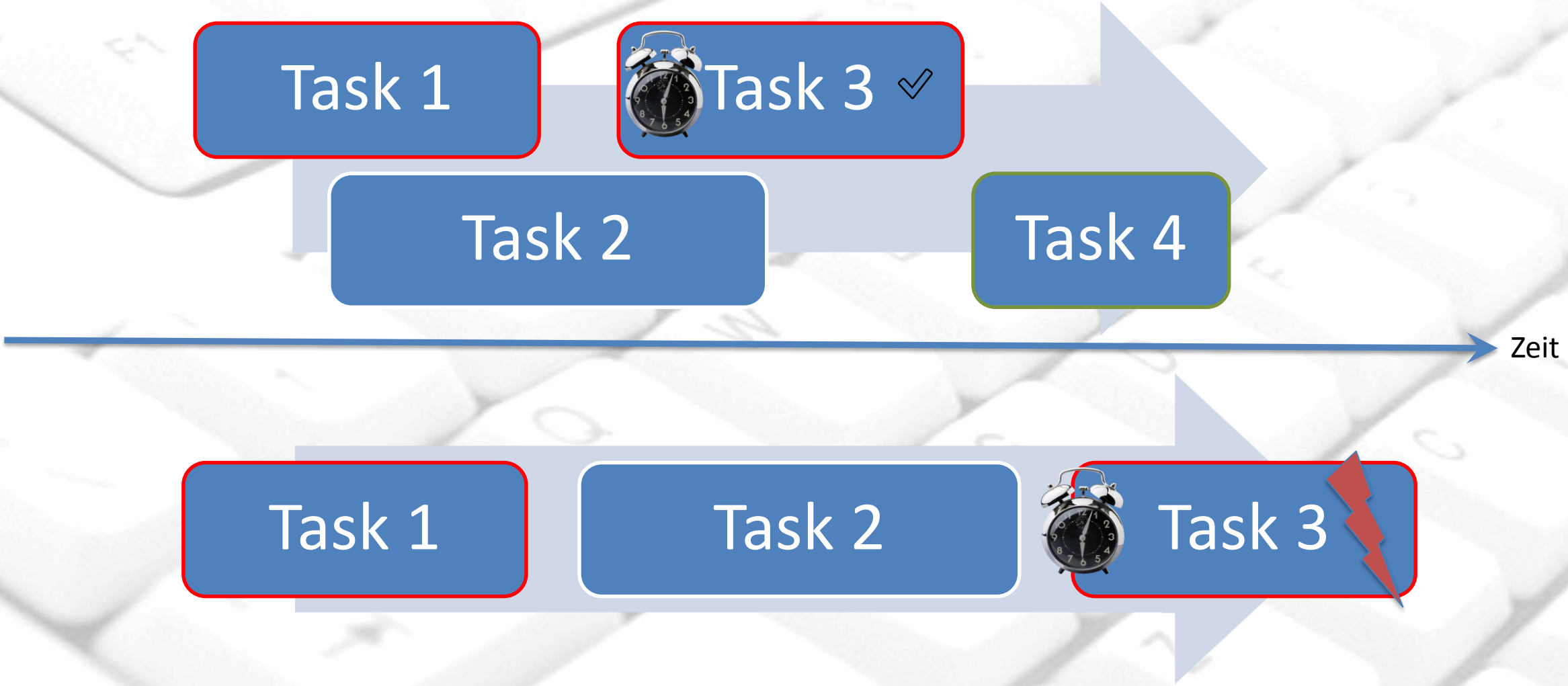
- Echtzeit Suchvorschläge
- Druckerwarteschlange
- Komplexe Dienstanfragen
- Online Registrierung



Thank you for your registration. In the next few minutes you should receive an email with the registration details. To activate your account, please log in at least once within the next 7 days. Otherwise the account will be removed again.

HERAUSFORDERUNGEN

Abhängigkeiten



Exceptions in Hintergrundtasks

- Umgang mit Exception
  - Behandlung beim Client
  - Behandlung auf dem Server
- Kein Blockieren des Servers

Unbeendete Hintergundtasks bei Sessionending

- Relevanz nur bei Rückgabe
- Entscheidung
  - Verwerfen
  - Persistieren
  - Rücktransport

Callback-Handling

- Verfahren nach Abarbeitung der Anfrage
  - Serverseitig
  - Clientseitig
- Unterscheidung verschiedener Status

Übermittlung von Methodenergebnissen

- Möglichkeit von Responses
- „finally“-Objekt

Clientseitige Programmierung

- Browsereinstellungen
- Entscheidung
  - Ausschluss
  - Verminderter Funktionsumfang
- Selektion clientseitiger Tätigkeiten

KONZEPTE

EventMachine

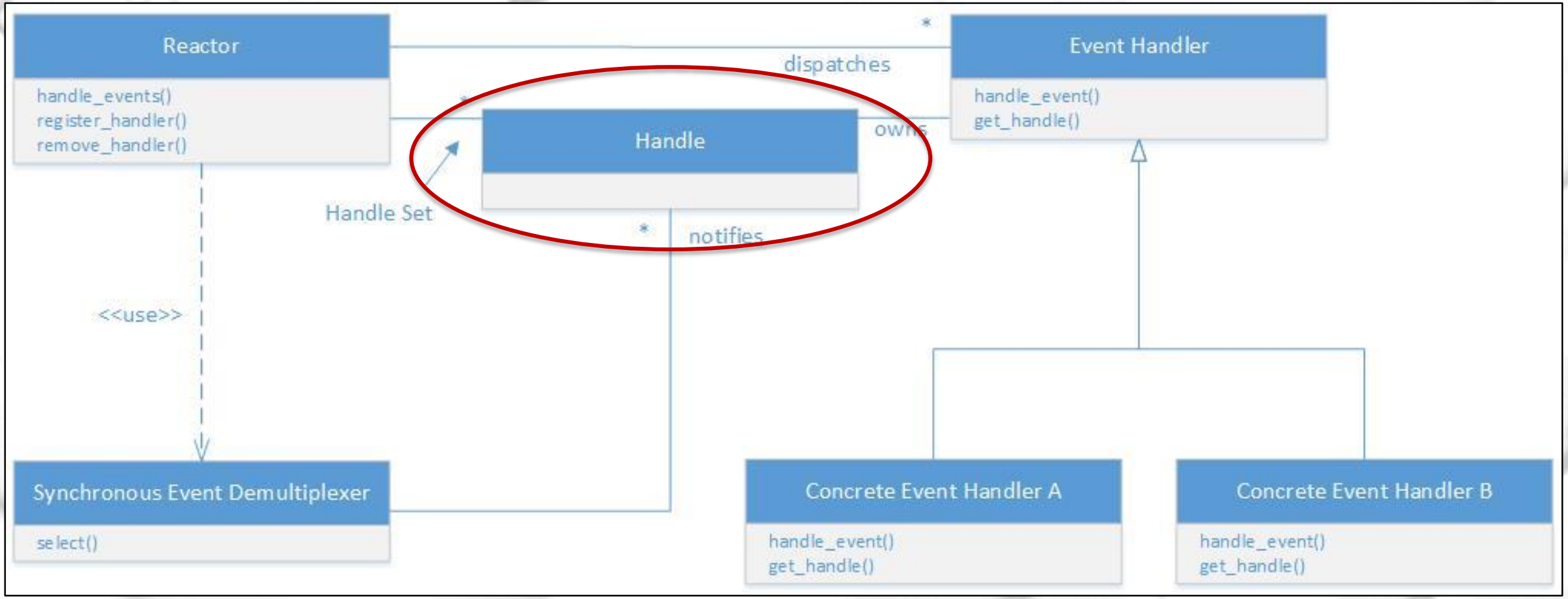
- Gem für die Programmiersprache Ruby
- Ereignisgestützte Anwendungen
- Eliminierung von Komplexität
- Basis Reactor Design Pattern

Quelle: http://www.sweo.de/

Reactor Pattern

Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

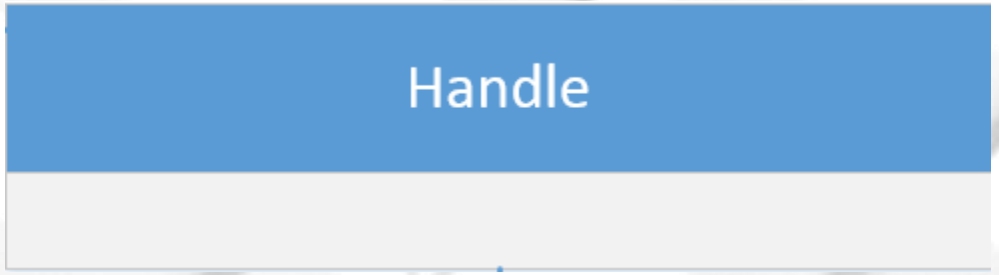
Reactor Pattern



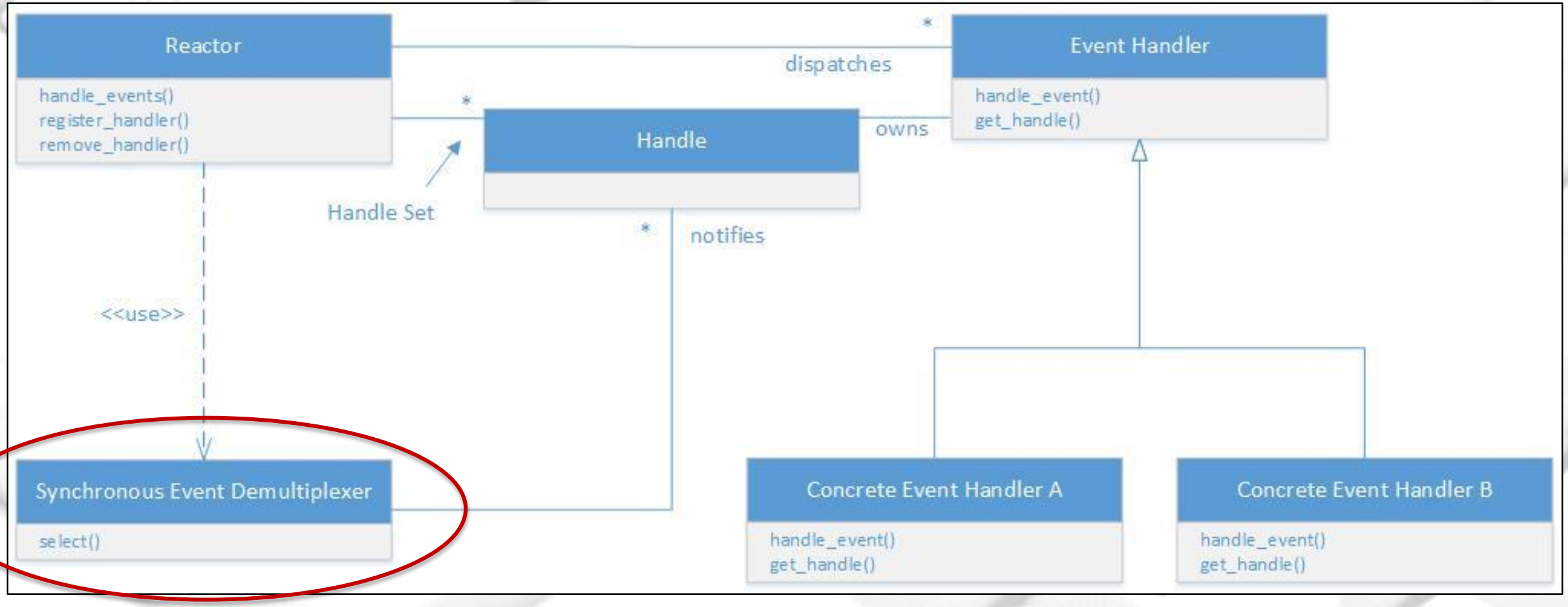
Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

Reactor Pattern: Handle

- Identifizieren von Ereignisquellen
- Pufferung von Anzeigeereignissen
- Anstoßen von Operationen



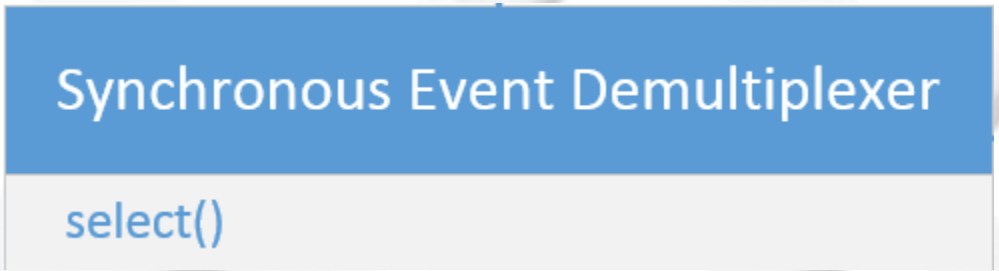
Reactor Pattern



Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

Reactor Pattern: Sychnronous Event Demultiplexer

- Warten auf Anzeigeereignisse auf Handle-Set
- Blockieren bis Ereignis eintritt



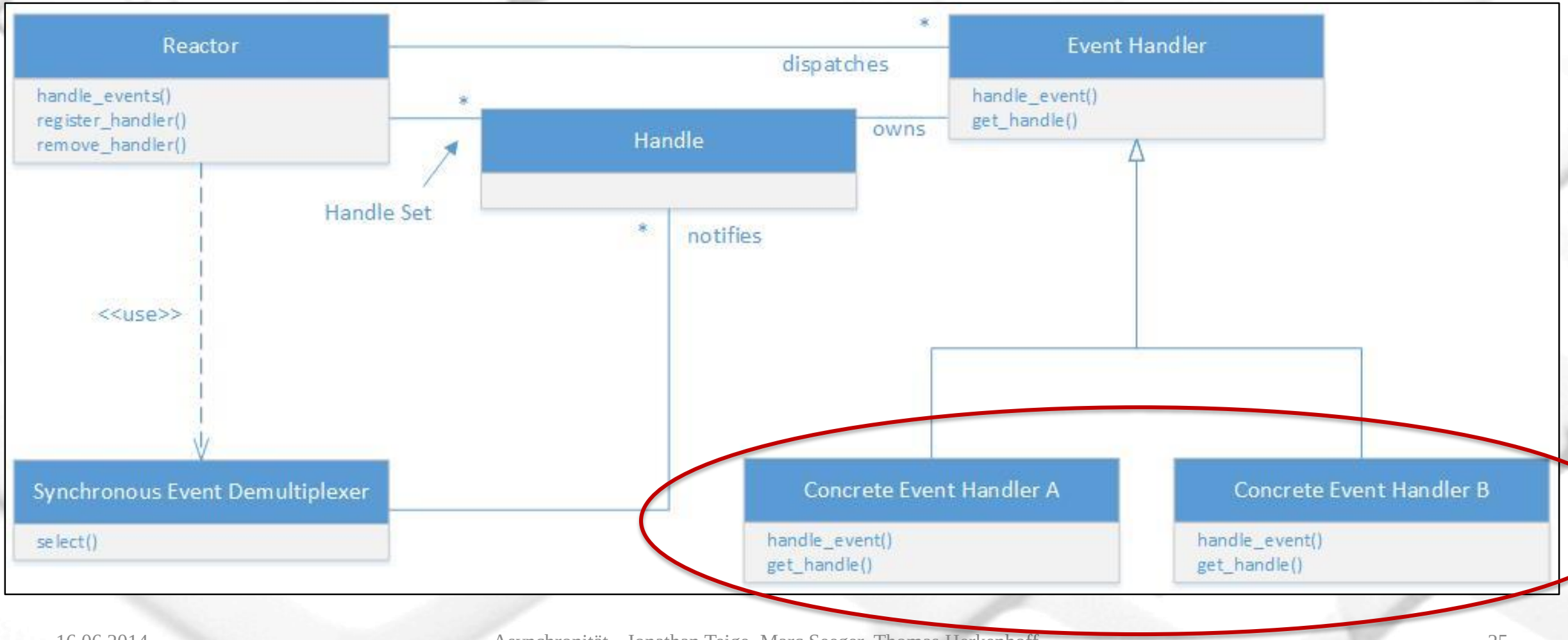
Reactor Pattern

Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

Reactor Pattern: Event Handler

- Spezifiziert die Hook-Methoden

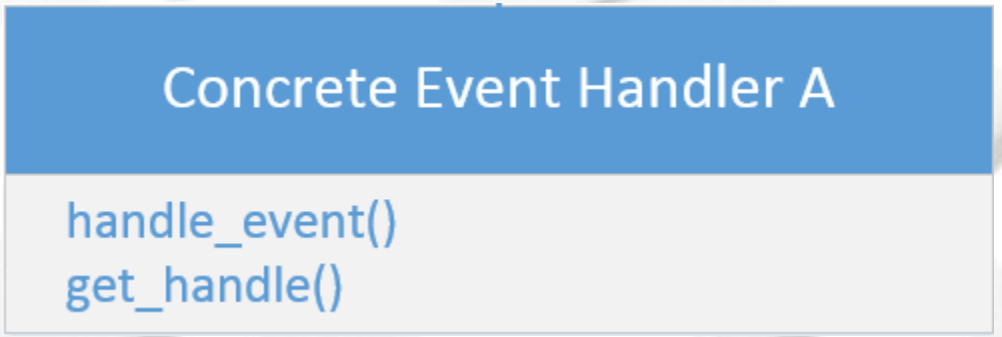
Reactor Pattern



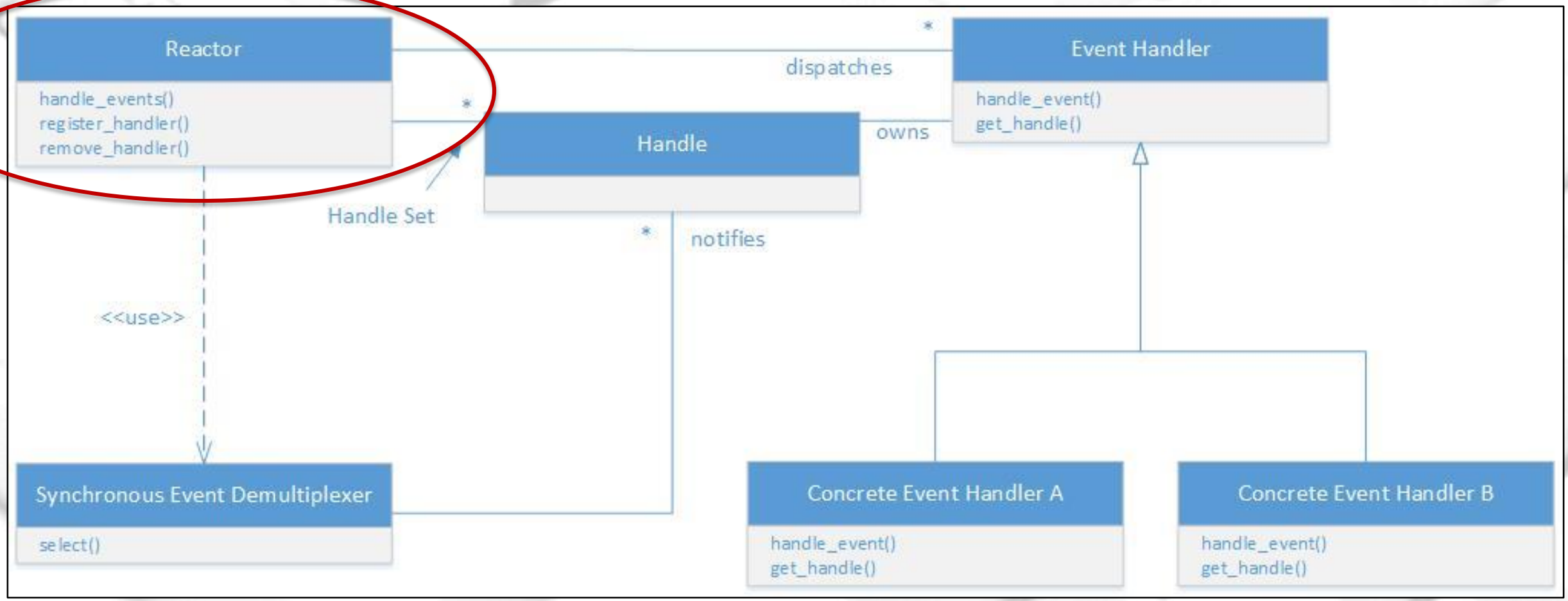
Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

Reactor Pattern: Concrete Event Handler

- Spezialisiert den Event Handler
- Implementiert anwendungsspezifischen Dienst
- Verbindung zum Handle



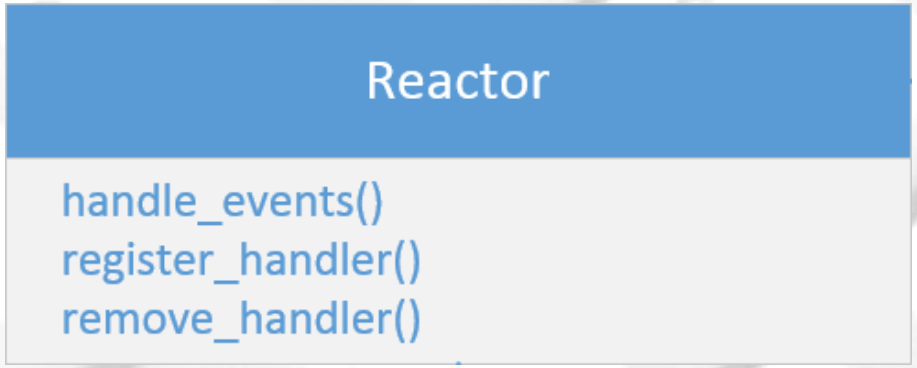
Reactor Pattern



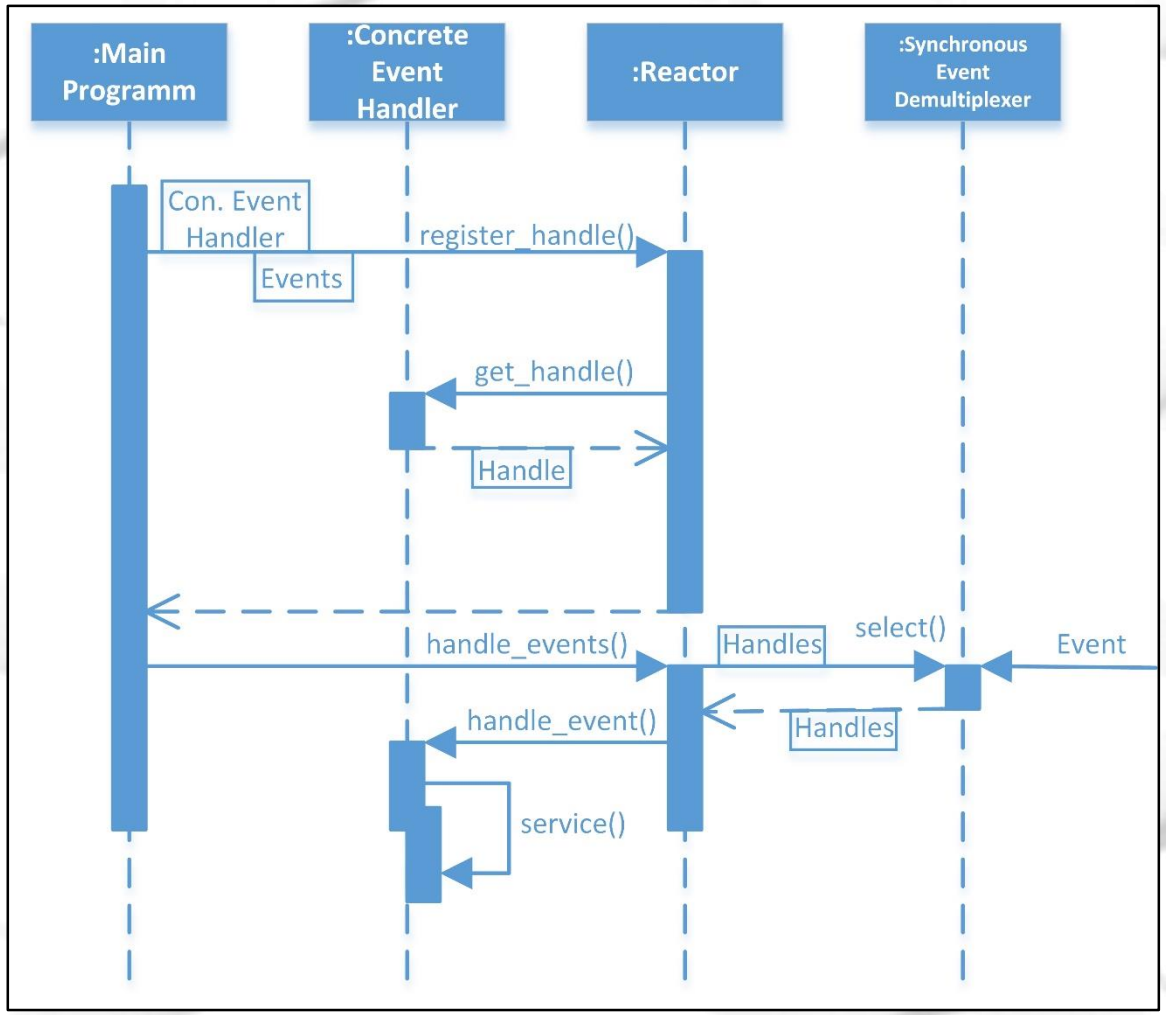
Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

Reactor Pattern: Reactor

- Schnittstelle für die Anwendung
- Registrierung von Event Handlern
- Definiert Endlosschleife
- Verteilt Ereignisse an Event Handler
- Aufruf von passender Hook-Methode



Reactor Pattern



Nach: Pattern-orientierte Software-Architektur: Muster für nebenläufige und vernetzte Objekte/ Douglas Schmidt

EventMachine

- Einsatzmöglichkeiten
  - Skalierbare ereignisgesteuerte Server
  - Skalierbare asynchrone Clients
  - Effiziente Netzwerkproxies
  - Netzwerk-Monitioring-Tools

Was ist Ajax?

- Amsterdamsche Football Club Ajax
- Gewinner des UEFA Cup
  - 1991
  - 1992
- Synchron 11 Spieler auf dem Feld

Quelle: Erl, SOA Design Patterns

Was ist Ajax?

- Revolutionäres Reinigungsprodukt!!!!
- Säubert ALLES!!!!
- Garantiert nicht Biologisch abbaubar
- Auch ideal zum Putzen von Festplatten

Quelle: http://www.hausfreund.at/files/pix/400x400/412821.jpg

Ajax: Jesse James Garret

- Jesse James Garret
- Nutzte als erstes den Ausdruck „Ajax“
- Definierte den Term Ajax
- Asynchronous XML And JavaScript

Quelle: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

Ajax

"[…] Ajax incorporates:

- standards-based presentation using XHTML and CSS;
- dynamic display and interaction using the Document Object Model;
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- and JavaScript binding everything together."

(Jesse James Garret; http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/*)*

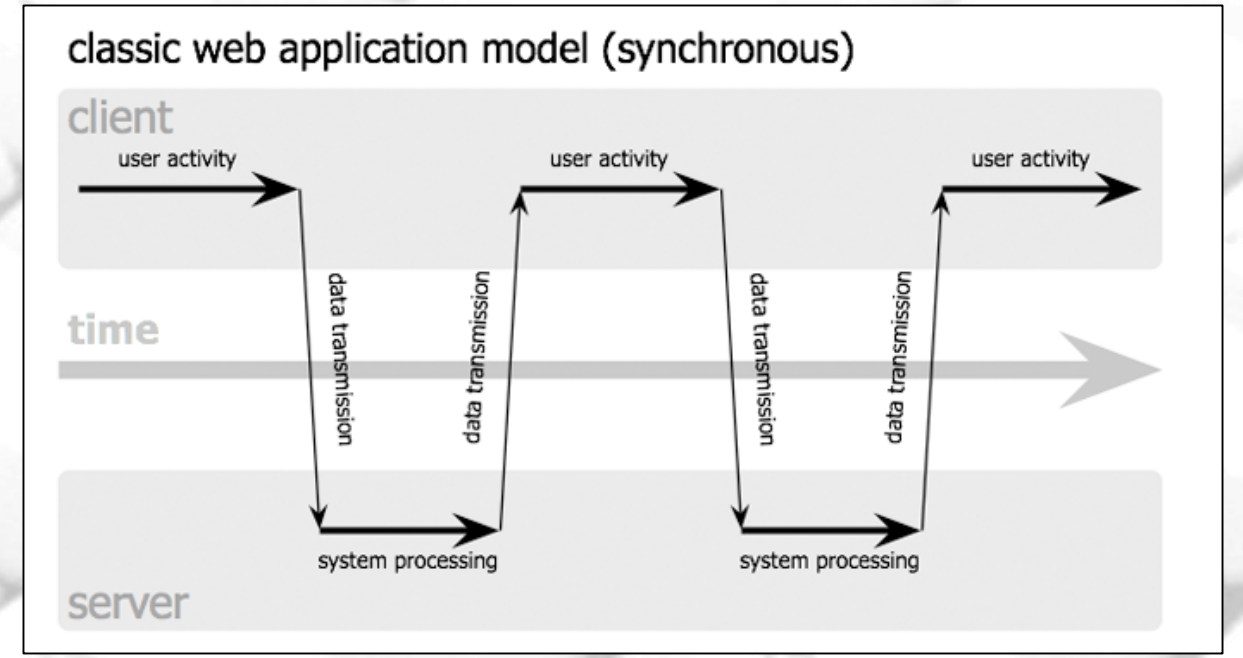
Ajax

“Simpel ausgedrückt: Ajax ist einfach nur ein Ansatz zur Webinteraktion.[...] Die Anfrage erfolgt asynchron. Das bedeutet: Die Ausführung des Codes wird nicht angehalten, um auf die Antwort zu warten, und dann erst fortgesetzt.”

(Zacas / McPeak / Fawcett, Ajax Professionell, 1. Auflage 2006 Redline-Verlag, Seite 23*)*

Ajax: Klassischer Ansatz

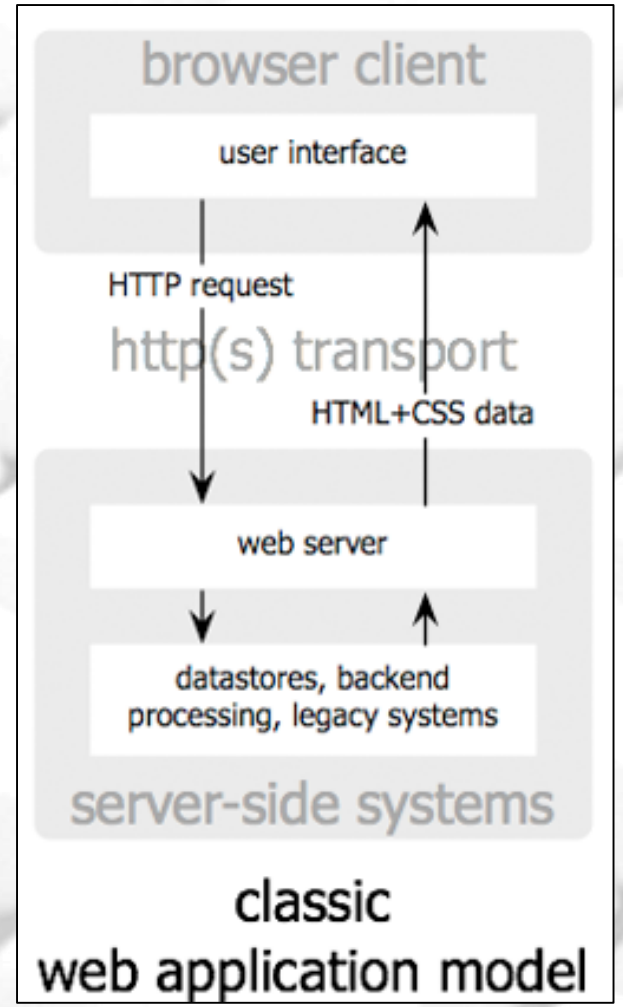
- Interaktion des Clients stößt Prozess auf Server an
- Komplette Webseite wird an Client zurückgesendet
- Rendering der kompletten Webseite im Browser



Quelle: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

Ajax: Klassischer Ansatz

- Request vom Browser direkt an Webserver
- Auslieferung kompletter Seite an den Browser
- Darstellung via HTML & CSS



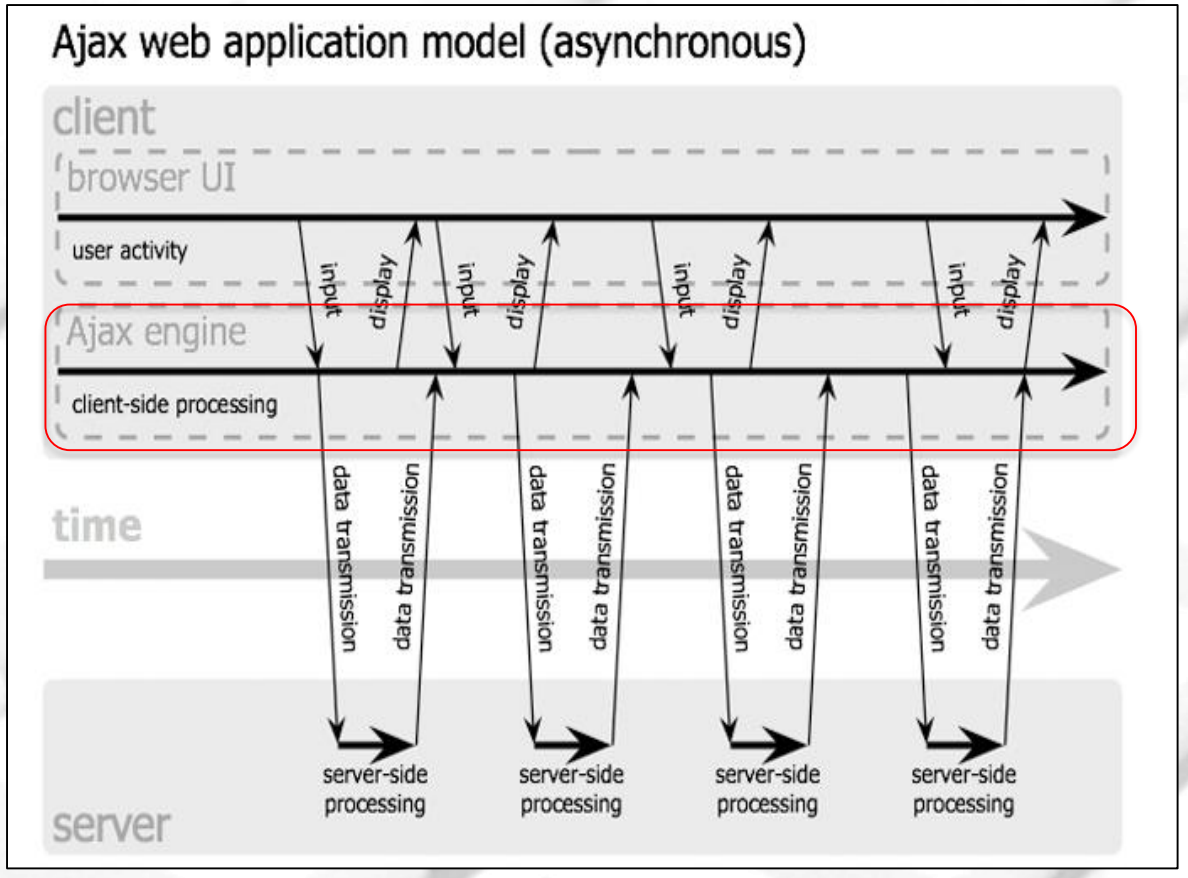
Quelle: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

Ajax: XMLHttpRequest Object

- Wichtigste Eigenschaften
  - Asynchronität
  - open(method, url, async)
  - onreadystatechange(function)
    - readyState
    - status
  - responseText / responseXML

Ajax: Dynamischer Ansatz

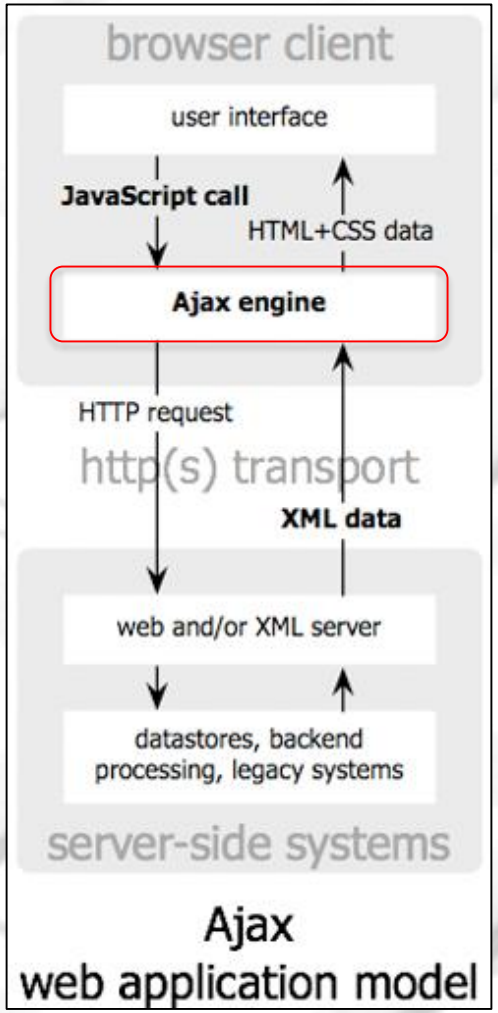
- Interaktion des Clients stößt Prozess auf Server an
- Ergebnis wird an Client zurückgesendet
- Fortwährende Interaktionsmöglichkeit
- Darstellung von Eingabe entkoppelt



Quelle: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

Ajax: Dynamischer Ansatz

- Interaktion wird via Javascript an die Ajax Engine geleitet
- Ajax Engine generiert und verwaltet (asynchron) Anfrage
- Ergebnis der Anfrage wird mittels JavaScript weiter verarbeitet



Quelle: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

Ajax: Synchron

```
600         //Create request object
601         request = new XMLHttpRequest();
602         //Collect parameters
603         var params = "action="+action+"&first="+firstParam;
604         //Set up the request
605         request.open("GET","backend.php?"+params, false);
606         //Send the request
607         request.send();
608         //Update DIV content
609         variableDiv.innerHTML = request.responseText;
```

Ajax: Asynchron

```
600         //Create request object
601         request = new XMLHttpRequest();
602         //Set up the callback
603         request.onreadystatechange = function(){
604             if(request.readyState == 4
605                 && request.status == 200){
606                 //Success - Update DIV content
607                 variableDiv.innerHTML = request.responseText;
608             }
609             else{
610                 //Error
611                 variableDiv.innerHTML = errorMessage;
612             }
613         }
614         //Collect parameters
615         var params = "action="+action+"&first="+firstParam;
616         //Set up the request
617         request.open("GET","backend.php?"+params, true);
618         //Send the request
619         request.send();
```

Ajax

LIVE-DEMO

Eigenes SSL-Zertifikat muss bestätigt werden!

Erreichbar unter:
https://funbox.no-ip.info:444/WEB/Index.html
Aber nur für kurze Zeit!

Ajax: Vor- und Nachteile

Synchron	Asynchron
Ausführung des Skripts: Warte auf Server	Ausführung des Skripts: Läuft weiter
[blockiert bis Rückmeldung]	Callback wird angesprochen
Suboptimale Nutzererfahrung	Nutzer kann während Abfrage neue Operationen auslösen

Node.js

- Serverseitige Implementierung
- Basiert auf “Google V8”
- u.a. Inspiriert von EventMachine
- Asynchrone Bearbeitung von Anfragen

Node.js

- Wenig Arbeitsspeicherkonsum
- Nur 1 Thread / Logischer Prozessor
  - Vergleich: Apache 1 Prozess / Connection
  - Prefork
- Handhabung extrem vieler Verbindungen

Node.js

- Nicht auf Blocking I/O warten
  - Direkte Zugriffe aufs Dateisystem
  - Datenbanken
  - Blockierte Ressourcen
- Mit anderen Anfragen fortfahren

Node.js

- Einfach (JavaScript)
- Performant (kein Warten)
- Ressourcensparend (kein Forking)
- OpenSource

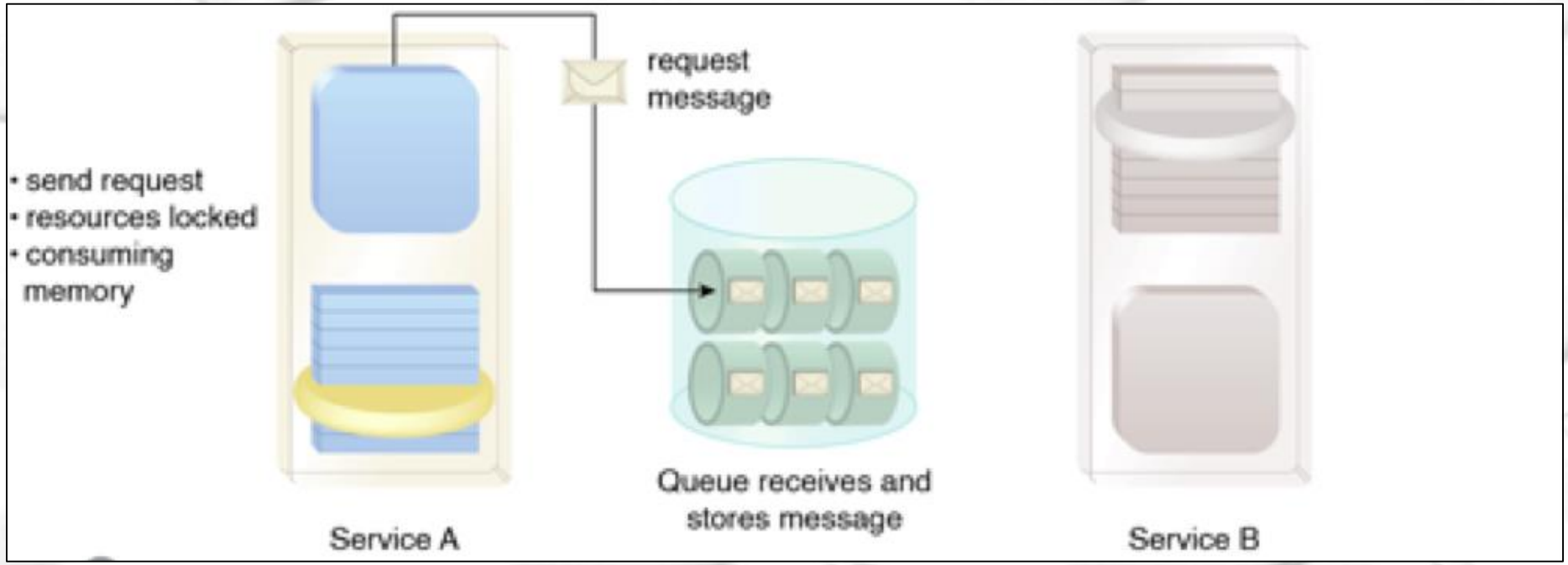
Message Queues

- Design Pattern
- Häufig: Serverseitige Implementierung
- Queue empfängt Nachrichten
- Weitergabe an Service
- Antwort per Response Queue

Message Queues: Eigenschaften

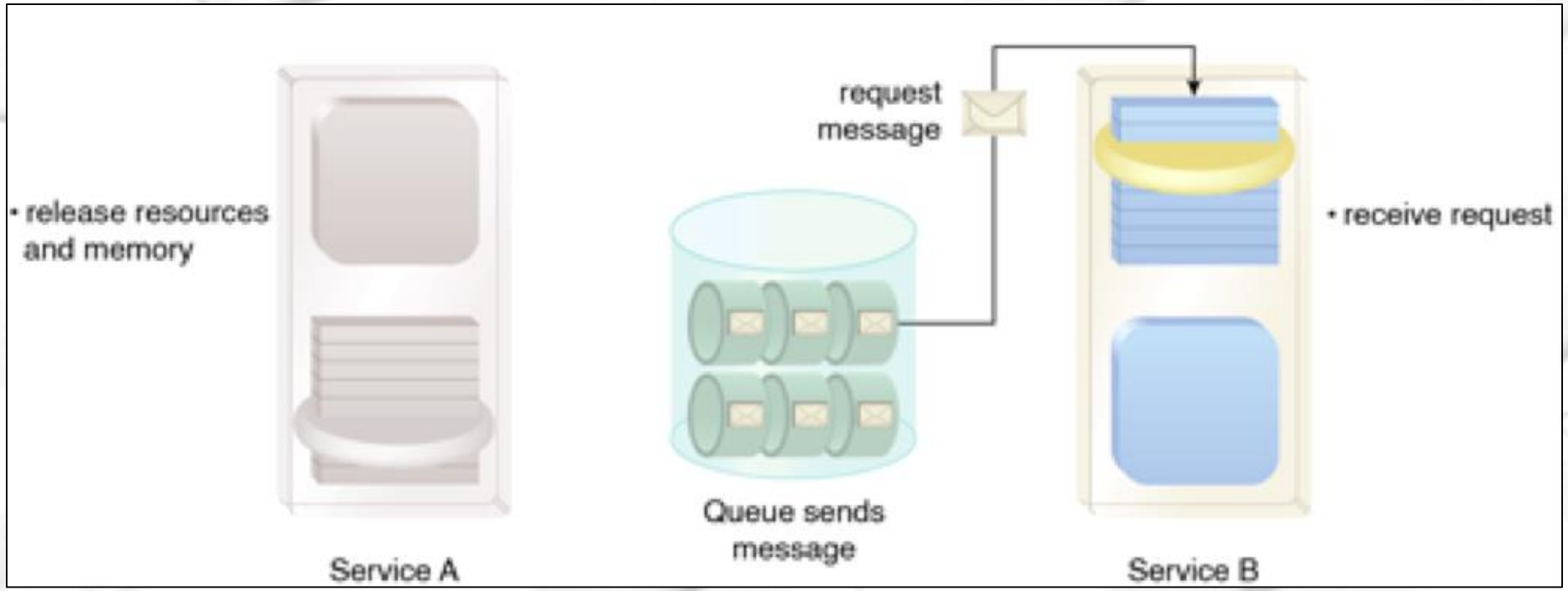
- Übernahme der Zustellung
- Empfang auch nach Reaktivierung
- Anfragen einreihen
- Persistent
- Transaktionell

Message Queues



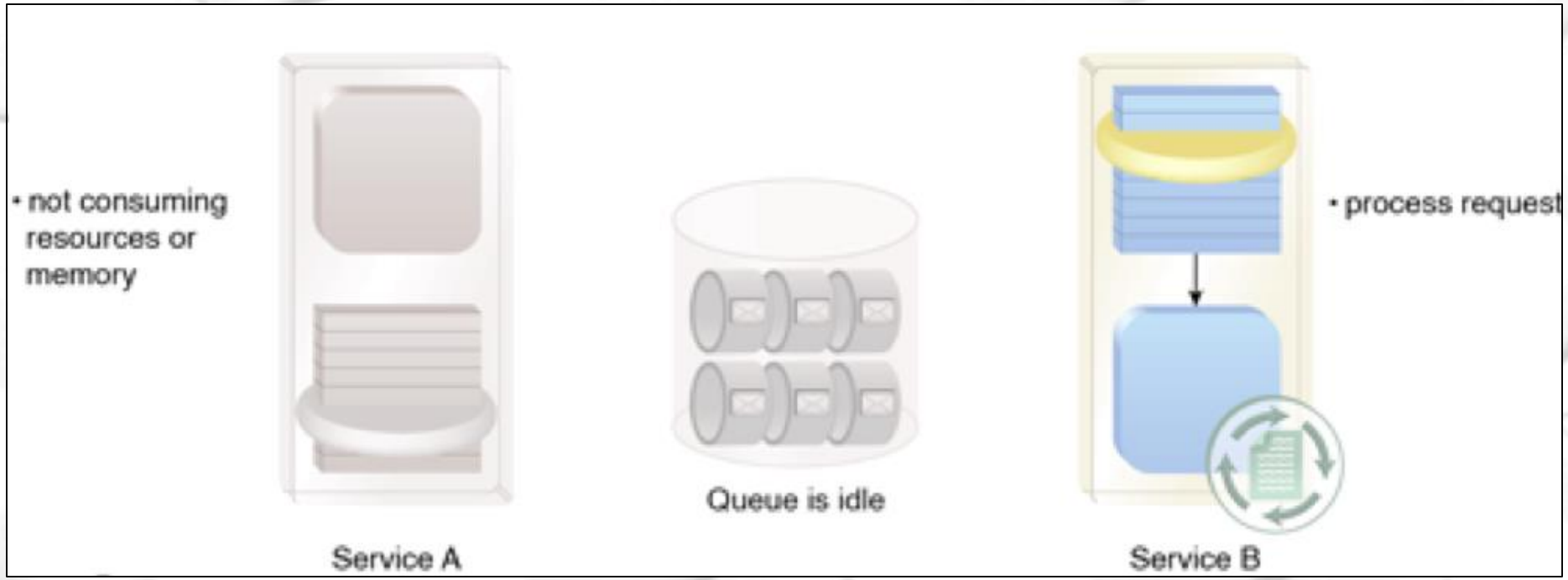
Quelle: Erl, SOA Design Patterns, 1. Auflage 2009, Prentice Hall Verlag

Message Queues



Quelle: Erl, SOA Design Patterns, 1. Auflage 2009, Prentice Hall Verlag

Message Queues

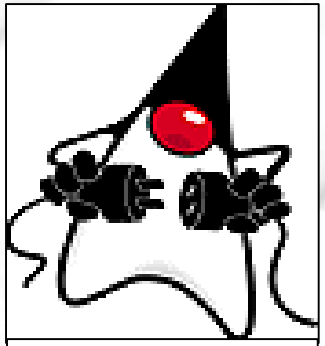


Quelle: Erl, SOA Design Patterns, 1. Auflage 2009, Prentice Hall Verlag

Message Queues: Weitere Möglichkeiten

- Untersuchung vor Bearbeitung
- Speicherung in Objekt
- Implementierung autark von Services möglich
- Kommunikation zwischen Services
- Verwaltung von Background-Jobs

Message Queues: Populäre Implementierungen



Quelle: Wikipedia.de; jaxenter.com; jboss.org; swiftmq.com; mq.java.net

Message Queues: API

- Verschiedene Standard-APIs
  - Java Message Service (JMS)
  - Advanced Message Queuing Protocol (AMQP)
  - Streaming Text Oriented Messaging Protocol (STOMP)

- Frameworkimplementierung

Quelle: Wikipedia.de; amqp.org; stomp.github.io

Message Queues: Herausforderungen

- Erhöhte Komplexität
- Zyklischer Zustellversuch
  - Verbindungsaufbau
  - Blockade zustellbarer Nachrichten

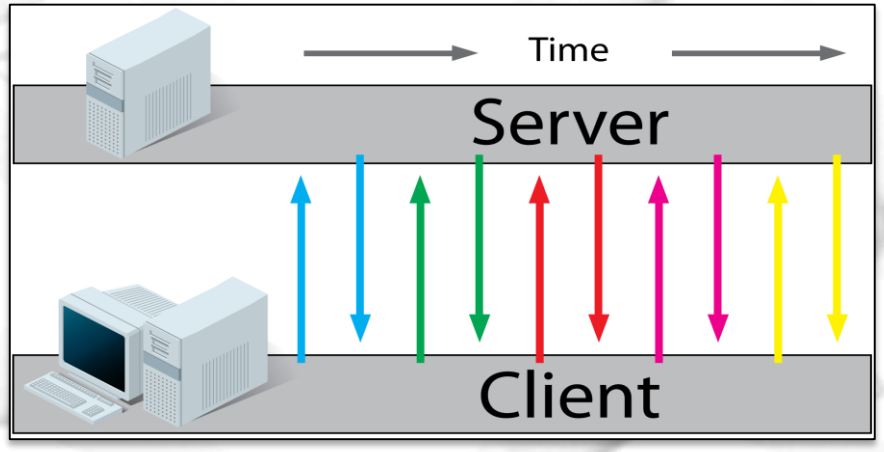
Websockets

- Möglichkeit zur aktiven Rückmeldung
- Dezember 2011 spezifiziert
- RFC 6455
- Anwendungsschicht-Protokoll

Quelle: pusher.com

Websockets: Standard - HTTP

- Client sendet Request
- Server sendet Response
- „Long-Polling“



Quelle: http://i.stack.imgur.com/qlMEU.png

Websockets

- Interpretation als HTTP-Upgrade
- Vollduplex-Verbindung
- EventHandler
- Header: 2 - 14 Bytes

Websockets: Verbindungsaufbau

- Client

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

- Server

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo= Sec-
WebSocket-Protocol: chat
```

FAZIT

Zusammenfassung

- Erhöhung des Nutzenkomforts
- Entlastung der Ressourcen
- Vielfältige Implementierungsmöglichkeiten
- Erhöhte Komplexität
- Designentscheidungen

Empfehlung

- Selektiver Einsatz
  - Bei komplexen Operationen
  - Zur Unterstützung von Echtzeitbedienung
  - Bei nicht zeitkritische Anfragen
  - Zur Skalierung

Vielen Dank für Eure Aufmerksamkeit!