Verteiltes Programmieren mit Space Based Computing Middleware

Vorlesungsübung (VU 185.226)

WS 2014 Beispielausgabe

Martin Planer
TU Wien, Institut für Computersprachen
A-1040 Wien, Argentinierstrasse 8 / 4.Stock
E-Mail: sbc-tutor@complang.tuwien.ac.at
Web: www.complang.tuwien.ac.at/eva

Abgabegespräche

- **Allgemeines**
  - Verpflichtend für alle Gruppen
  - Anmeldung für einen Termin (Timeslot) in TUWEL
  - *ACHTUNG*: Teilnahme am ersten Abgabegespräch → Zeugnis wird auf jeden Fall ausgestellt
- **Abgabegespräch 1**
  - Mi, 10.12.2014 9:00 – 13:00 (Besprechungsraum Galerie) [1]
  - Do, 11.12.2014 14:00 – 18:00 (Besprechungsraum Galerie)
  - Ausgabe von Beispiel 2
- **Abgabegespräch 2**
  - Mo, 12.01.2015 14:00 – 18:00 (Besprechungsraum Galerie)
  - Di, 13.01.2015 9:00 – 13:00 (Besprechungsraum Galerie)

[1] Besprechungsraum Galerie: Argentinierstrasse 8 → Türe gleich nach dem Eingang rechts → 1. Stock

Beispiele - Überblick

➔Teil 1: Feuerwerksfabrik

- Effiziente Lösung eines Koordinationsproblems
- 2 Implementierungen*:* ***space-basiert*** und ***alternativ***

➔Teil 2: Feuerwerksfabrik-Erweiterung

- Mehr Info @ Abgabegespräch 1

➔Teil 3: Präsentation

- Technologieauswahl mit Begründung
- Präsentation der Lösung
- LOCs, Aufwand, Fazit

Feuerwerksfabrik (i)

- Simulation einer Fabrik für Feuerwerksraketen
- 4 verschiedene Akteure
  - **Lieferanten/-innen**
    - Anlieferung der einzelnen Bestandteile (Holzstäbe, Gehäuse mit Zünder, Treib-/Effektladungen)
  - **Produzenten/-innen**
    - Fertigung der Feuerwerksraketen aus Komponenten
  - **Qualitätskontrolleure/-innen**
    - Kontrolle und Qualitätssicherung
  - **Logistiker/-innen**
    - Verpackung und Auslieferung

Feuerwerksfabrik (ii)

- **Anzeige (GUI)**
  - aller vorhandenen Bestandteile
  - aller hergestellten Raketen
    - beteiligte Mitarbeiter
    - verwendete Komponenten

- **Gemeinsamer Space**

- **Auf Konflikte bei synchronem Zugriff achten!**
  - Transaktionen
  - Deadlocks
  - Möglichst hohe Parallelität erlauben

Feuerwerksfabrik (iii)

- **Implementierung in zwei Varianten:**
  - Space-basierte Lösung
  - Alternativ-Lösung (nicht space-basiert)
    - z.B. RMI, JMS, Sockets, Corba, WCF, ...
    - gleiche Funktionalität wie SBC-Lösung
  - Lösungen müssen nicht interoperabel sein

- **Alle Akteure arbeiten als unabhängige Programme**

Abgabe

➔ Deadlines

- Abgabe 1 bis 09.12.2014 23:55 in TUWEL
- Abgabe 2 bis 11.01.2015 23:55 in TUWEL
- Beide Implementierungen + Präsentation + Dokumentation als ZIP hochladen

➔ Programm

- Abgabe des Quellcodes (funktionsfähig, kommentiert)
- README zum Kompilieren, Starten, etc.
- Präsentation des Programmes bei Abgabegespräch

➔ Präsentationsfolien

- Datenstruktur, Koordination, Besonderheiten, ...
- Begründung für die Auswahl der Middleware-Systeme
- Unterschiede zwischen verwendeten Technologien

Abgabe

- ➔ Alle Programmteile müssen implementiert werden
- ➔ Transaktionale Sicherheit bei Space-Operationen
  - ○ Vorsicht bei gleichzeitigem Zugriff mehrerer Peers
  - ○ Dynamisches Kommen und Gehen von Peers
- ➔ Saubere Koordination wichtiger als schönes GUI
  - ○ Polling, Busy Waiting, unnötige Kommunikation vermeiden
  - ○ Möglichst hohe Nebenläufigkeit
- ➔ Beide Gruppenmitglieder sollen sich zu gleichen Teilen an beiden Implementierungen beteiligen!
  - ○ Fragen zu Implementierung/Funktion/Architektur bei Abgabe
- ➔ Weitere Anforderungen siehe Angabe in TUWEL

TUWEL

- **Koordination der Übung über TUWEL**
  - Nachrichten der LVA-Leitung
  - Angabe
  - Unterlagen
  - Forum
  - Abgabe

Downloads

- MozartSpaces
  - www.mozartspaces.org
- ApplicationSpace
  - http://www.xcoordination.org
- GigaSpaces
  - http://www.gigaspaces.com/LatestProductVersion
    (Lite Edition)
- Maven
  - http://maven.apache.org
- Ant
  - http://ant.apache.org

Hilfreiche Dokumentation

- VO Folien
  - TUWEL
- MozartSpaces
  - Tutorial
  - Beispiele
  - http://www.mozartspaces.org
- XcoSpaces .Net Tutorial

Viel Spaß!