Prüfungsfragen Betriebssysteme 10/99, 2000

BP I, OODS, Hofmann Oktober 1999

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1,3
- Sehr angenehmer Stil. Weniger Detailwissen, eher Zusammenhänge, Verständnis wichtig.
- Lässt einen vieles selbst vortragen, unterbricht dann nur für Zwischenfragen.

Fragen

- Objektorientierte Kozepte und Nebenläufigkeit:
  Wie würden Sie gegenseitigen Ausschluss in Java implementieren?
- Dann konkreter: Algorithmus von Lampson (RPC)
  Habe Skizze aus dem Skript hingemalt und die einzelnen Teile anhand eines Beispielaufrufs erläutert. Wie implementiert man das nun in Java? (da hatte ich zuwenig Ahnung, mein grösster Minuspunkt)
- Gegenseitiger Ausschluss in verteilten Systemen:
  Einfachste Lösung? (zentr. Koord.) Angenommen Sie wollen den nicht vorgeben? → Wahlalgorithmus. Habe Echo-Election Algorithmus erklärt. Verbesserungen? → Adaptionsalgorithmus (ebenfalls erklärt).
- verteilte Terminierung:
  Hat mich eine Rede halten lassen. Habe Terminierungsproblem erläutert, asynchrones, synchrones + Atommodel... Wann ist jeweils System terminiert? Dann Einfachzählverfahren und warum es nicht funktioniert. → Doppelzählverfahren.

BP1/OODS, Hofmann April 2000

Bemerkungen zu Prüfung und Prüfer

- Angenehme Prüfungsatmosphäre, teilweise eher Gespräch als Prüfung. Nur Fragen zu RPC/RMI und CORBA, dafür sehr ausführlich und viele Transferfragen.

Fragen

- RPC Aufrufsemantiken und wie man sie implementiert; warum gibt es diese Semantiken nicht bei lokalem Aufruf?

  Bei at-least once sind wir eine Zeitlang abgeschweift zu SUN-RPC und der SUN-NFS-Implementierung (zustandslose Server → idempotente Aufrufe → a-l-o reicht aus; das war zwar alles eigentlich kein Vorlesungsstoff, aber er hat mich erzählen lassen und war scheinbar ganz zufrieden)
- Die Orphan-Eliminations-Mechanismen von vorne bis hinten runtergebetet
- Was für Probleme kann es bei RPC noch geben?
  Mit ein bisschen Hilfestellung (wie ist das denn bei heterogenen verteilten Systemen?) auf die Konvertierung von Datenformaten gekommen. Weiter erzählt über Stubs, Marshalling, XDR, Exakte Beschreibung von Parameterformaten (hinter einem char* in C kann alles Mögliche stecken)
- Wie sieht das denn bei CORBA aus: braucht man da auch XDR?
  Nicht innerhalb eines ORB: CORBA legt nur Language Mappings fest; das Marshalling kann der ORB nach Belieben erledigen. Bei Datenaustausch zwischen mehreren ORBs muss man ein allgemeines Datenformat festlegen (IIOP).
- Wenn CORBA nur die Language Mappings festlegt, was passiert dann, wenn verschiedene Compiler auf der gleichen Plattform unterschiedliche Darstellungsformate verwenden?
  Da war ich etwas unsicher; erstmal erwähnt, dass es normalerweise für jede Plattform

und Sprache feste Konventionen gibt (Byte-Ordering, mehrdimensionale Arrays in Row/Column-Order usw.); ansonsten müsste eine ORB-Implementierung wohl mehrere Konvertierungsbibliotheken für die entsprechenden Compiler bereitstellen oder die Verwendung eines bestimmten Compilers vorschreiben(?)

- Wie sieht das denn bei einfachen Nachrichtenmechanismen mit der Konvertierung von Datenformaten aus?
  Findet i.a. nicht statt bzw. ist Sache des Anwendungsprogrammierers

- Das wäre aber doch eigentlich nicht schlecht. Könnte man das nicht machen?
  Man kann maschinenunabhängige Datenformate natürlich auch ohne RPC benutzen. Z.b. bei Java-Streams/Serialisierung.

- Aber man kann doch Nachrichtenmechanismen auch auf RPC aufsetzen (+kurzer Monolog von Hofmann wie er das machen würde). Was halten Sie denn davon?
  Im Prinzip schon... Schlechtere Performance durch viele Schichten; besser direkter Zugriff auf die Mechanismen die intern in der RPC-Implementierung verwendet werden.

- Denken Sie, dass der Austausch von großen Datenmengen über viele RPCs weniger effizient ist als Streaming-Protokolle wie TCP?
  Ja, die RPC-Variante muss normalerweise für jedes Paket (=RPC) auf die Antwort des Servers warten (=Stop-and-Wait). Effiziente Kommunikationsprotokolle können dagegen Tricks wie Schiebefenster verwenden und damit Bestätigungen zusammenfassen, trotz ausstehender Quittungen weitersenden oder Pakete außer der Reihenfolge annehmen.

- Ich hatte eigentlich damit gerechnet, dass er jetzt langsam mal das Thema wechselt, aber da war er dann schon fertig...

BP I, OODS
F. Hofmann
April 2000

Bemerkungen zu Prüfung und Prüfer

- Die Fragen waren teilweise etwas unklar und allgemein gestellt, werden aber konkreter, wenn die richtige Antwort nicht kommt. Obwohl ich öfter nachfragen musste, da mir nicht klar war, worauf er hinaus wollte, hat sich das in keiner Weise negativ auf die Note ausgewirkt.

Fragen

- Warum sind objektorientierte Programmiersprachen bei der Realisierung von verteilten Systemen von Vorteil? (Kapselung, Einheit der Verteilung ist das Objekt)
- Implementierung eines verteilten Systems? (RPC); kurze Erklärung (Stubs, Kommunikation, Marshaling, Aufrufsemantiken)
- Ausführliche Erläuterung der verschiedenen Aufrufsemantiken, jeweils mit Vor- und Nachteilen, Orphans
- Wie kann man Orphans bekämpfen? (Extermination, Probleme bei mehreren Ausfällen, Expiration, Reincarnation)
- Was würden Sie verwenden? (Extermination, abgesichert durch Expiration mit großem Timeout)
- Warum nicht Expiration allein? (Probleme, wenn Timeout zu kurz/zu lang)
- Warum nicht Extermination allein? (Probleme, wenn mehrere Rechner Extermination gleichzeitig starten)
- Alternative zum Ansatz „zentralisiertes Objekt“ bzw. Möglichkeit, das Konzept des Objekts als Einheit der Verteilung etwas aufzuweichen? (fragmentierte Objekte); Erklärung, Vorteile (Cachen oft benötigter Daten, effiziente Kommunikation)
- Wie entwickle ich verteilte Systeme in C? Was ist die Einheit der Verteilung

bei nicht-objektorientierten Programmiersprachen? (Prozesse)

- Wie sind damit verteilte Systeme realisiert (nicht RPC!)? (Kommunikation über Sockets, TCP/IP oder UDP/IP)
- Ausführungsmodelle erklären: Atommodell, synchrones Modell, asynchrones Modell
- Warum wird synchrones Modell verwendet, obwohl es doch eher realitätsfern ist? (wenn nur die verteilten Berechnungen und deren Ergebnisse interessieren, nicht der Nachrichtenverkehr)
- Was wird in diesem Modell nicht berücksichtigt? (Verfälschung der Nachrichtenreihenfolge)
- Zustandssicherung: warum können Probleme auftreten? (keine globale Sicht möglich; Kausalität darf nicht verletzt werden, keine Nachrichten aus der Zukunft (empfangen aber nicht gesendet) erlaubt)
- Koo/Toueg-Algorithmus, versuchsweise und permanente Sicherungen erklären; an wen schicke ich „make tentative"-Nachrichten? (nur an die Prozessoren, von denen ich seit meiner letzten Zustandssicherung Nachrichten bekommen habe, damit diese das Senden dieser Nachrichten aufzeichnen)
- Gibt es einen einfacheren Algorithmus? (Chandy/Lamport), Erklärung

OODS
Kleinöder, Beisitzer: Golm
März 2000

Bemerkungen zu Prüfung und Prüfer

- Scheinkolloquium
- Ergebnis: 1,0
- Er stellt häufig sehr allgemeine Fragen, so daß man die Chance hat, über das, was man schwerpunktmäßig gelernt hat zu reden. Es ist also gut, wenn man zu einem gegebenen Sichwort einiges frei erzählen kann. (Überblick ist gefragt!)
- Wenn ihr eine Frage falsch oder gar nicht verstanden habt, sagt ihm das gleich! Versucht ihm zu erklären, warum die Frage mißverständlich war!

Fragen

- Basiskonzepte der Objektorientierung
  Antwort: Klassen, Objekte, Abstraktion, Modularisierung, Hierarchie, Polymorphismus, Typen
  Kommentar: wichtig!
- Was ist eigentlich der Unterschied zwischen einer Klasse und einem Typ? Ist das nicht dasselbe?
  Antwort: Nein, z.B. C++ und private Vererbung – Java, Subklassen, Interfaces
- Kommunikationsmethoden (oder eine ähnliche Formulierung)
  Antwort: synchron, asynchron, Message-Passing, Datagram-Message, Rendezvous-Model, Request-Reply, RPC
- irgendetwas über RPC/RMI
  Kommentar: Ich habe diese Frage nicht recht verstanden und habe dann irgendetwas über Sinn und Zweck von RPC erzählt. (Sagte er vielleicht doch RMI?)
- RPC-Aufrufsemantiken und deren Implementierung
  Antwort: at-least-once, at-most-once, exactly-once
- Corba, Hauptbestandteile
  Antwort: ORB, Object Adaptor, Stubs, Skeletons, Interface Repository, Implementation Repostitory, DSI, DII
  Kommentar: Ich glaube, man hätte die Frage problemlos auch anders beantworten können.
- Was ist eigentlich der Unterschied zwischen den Aufrufen bei CORBA und bei JavaRMI?
  Kommentar: Diese Frage hat mich etwas durcheinandergebracht, denn das Prinzip ist bei beiden dasselbe, z.B. existiert ein Naming-Service (RMI-Registry).
- Was hat es mit Software-Components auf sich? Nehmen wir doch JavaBeans als Beispiel!

Antwort: leicht kombinierbar (Builder-Tools), Namenskonventionen, zu JavaBeans: Properties mit Zugriffsmethoden (get/set), Adaptors, Events (source-listener-pattern),...

- Wozu braucht man denn diese Events?
  Kommentar: Ich bin auf keine wirklich sinnvolle Antwort gekommen. Er wollte hören: Eine Komponente muß ihre Listener (zur Compilezeit) nicht kennen. Sie können sich dynamisch registrieren. Ich habe ihm dann erklärt, warum ich das nicht gesagt habe...

- Das war's! Ich hoffe, ich habe nichts vergessen. Viel Glück bei der Prüfung!

Betriebsprogrammierung I, AKBP/Embedded Systems Prof. Hofmann Oktober 2000

Bemerkungen zu Prüfung und Prüfer

- Schwerpunktfach

- Detailwissen war eher unnötig; wichtig ist, die Algorithmen gut verstanden zu haben (Was macht man? / Warum funktioniert das? / Ähnlichkeiten zwischen Algos). Besonders gut gefällt ihm, wenn man Verbindungen zwischen den Algorithmen erkennt (zB. Terminierung und Checkpointing).

- Hat bei seinen Fragen immer ganz bestimmte Antworten im Kopf und versucht den Prüfling möglichst in diese Richtung zu lenken. Allzuweit ab-/ausschweifen sollte man nicht.

- Hat keinen einzigen Beweis (gibt ja sehr viele in BP I) gefragt, hat mich sogar abgewürgt, als ich von selber auf den Beweis zu Optimalität von EDF kam → keine Angst wegen der vielen langen, schweren, unverständlichen Beweise aus BP I, die kommen nicht dran!

- Lesen: Singhal "Advanced Concepts in OS" für BP I, Mattern "Verteilte Basisalgorithmen" für die späteren Kapitel - es sei denn, man hat alle Algos in der Vorlesung verstanden. Die übrige Sekundärliteratur (im Ordner in GI) ist weniger hilfreich. Für ES reichen die Vorlesungs-Folien, Skript aus BP II zur Vertiefung empfehlenswert.

- In meiner Prüfung kam fast nur ES dran, und da war Hauptteil Scheduling. Es kam keine Frage zur Übung selbst, somit muß man nur 2SWS lernen! → sowas wie ES sollte in keinem Prüfungsplan fehlen! ;-)

Fragen

- Embedded Systems
  - Allgemein (Was ist ein ES? / Kennzeichen / Deadlines / Hard, Soft RT / usw., quasi Einführung von ES-Vorlesung runterbeten)
  - Threads (Warum Aktivitätsträger in ES / Was sind Threads / Vorteil zu Prozeß)
  - Scheduling (Kategorien: statisch - dynamisch, implizit - explizit / Erklären Cyclic (auch Vorgang zur Findung von Plan), Suchen, RMS, EDF, PEDF / darüber ging die Hälfte der Prüfung!)
  - Vorraussetzung für Planung (Schleifen: max. Durchlaufzahl bekannt / Ablaufdauer einer Funktion gewinnen durch Compiler, Simulation / Bei Einsatz von Caches, Pipelines kein Determinismus → werden in ES nicht eingesetzt)
- Betriebsprogrammierung I
  - Terminierung (Problem erklären / Was ist schlecht am Zählverfahren / Unterschiedliche Verfahren erklären (2 Wellen, Zeitzonen, ...) / Verwandtheit mit Checkpoint- und Wahlproblem)
  - Checkpoints (Warum? / Problem erklären / Algorithmus von Lamport erklären, Vorraussetzungen, Probleme / Algorithmus von Toueg, Vorteile (keine Nachricht → kein Checkpoint), gegenüber Lamport (resilient) / Vergleich mit einem von Hofmann ausgedachtem Algorithmus, der in 1. Welle Kommunikation stoppt, in 2. Welle Checkpoints veranlaßt → funktioniert, ist ähnlich Lamport und Terminierungsalgorithmen, Nachteil geg. Toueg: Unnötige Checkpoints)

BP I, AKBP/ES
Hofmann, Beisitzer: Bellosa
Oktober 2000

Bemerkungen zu Prüfung und Prüfer

- Sehr angenehme Atmosphäre; man sollte den Prüfer ruhig ausreden lassen, bevor man die Frage beantwortet, dann kann man aber so viel erzählen, bis er einen unterbricht. Geht die Antwort in die falsche Richtung, gibt er Hilfestellungen.

- Auch wenn die Fragen relativ direkt sind, fragt er nicht auswendig gelernten Stoff ab, sondern ist zufrieden, wenn man den Stoff verstanden hat.

- Fragen, die über den Stoff hinausgehen und bei denen man nicht auf die Idee kommt, die der Prüfer dabei hat, werden nicht unbedingt negativ gewertet.

- Da Hofmann seine Fragen sehr abwechslungsreich aussucht, *sollte man auf alle Fälle einen sehr guten Überblick über den gesamten Stoff haben.*

- Teilweise ist für das Verständnis des Stoffs Sekundärlektüre erforderlich, zu empfehlen ist vor allem:

  - Singhal, M; Shivaratri, N. G.: Advanced Concepts in Operating Systems. McGraw Hill, 1994.
  - Mattern, F.: Verteilte Basisalgorithmen. Springer Verlag, IFB 226, 1989.

- Note: 1.0

Fragen

AKBP/ES

- Scheduling in Echtzeitsystemen: (kurz EDF und PEDF erklärt) Kann man Rechenzeit von Tasks vorhersagen? Nein, wegen Cache (compulsory misses), TLB, IRQs.

- Prozesswechsel: Geht das schell? Bei Prozessen nicht → Threads, wieso?

BP I

- Sicherungspunkte: lokale → globale; kann man nach dem Wiederanlauf feststellen, zu welchem Zeitpunkt der globale Sicherungspunkt gemacht wurde? Nein, da der aufgezeichnete Zustand nie wirklich existiert haben muss.

- Was haben verteilte Terminierung und Deadlock-Erkennung gemeinsam? Habe beides kurz erklärt (Vektormethode, einfacher CMH). Er wollte darauf hinaus, dass beide Zustände – wenn einmal eingenommen – nicht mehr von alleine verlassen werden, stabile Zustände sind.

Security for Distributed Systems and the Internet [SDSI] Object-oriented Concepts for Distributed Systems I [OODS I] Distributed Systems in Computational Engineering [DSCE]
Dr. Kleinöder
April 1999

Comments on the examination and the examiner

- he will speak a lot, sometimes just listen to him and don't know why he speak but not ask question

- you can tell what you know freely. Because sometime i don't know what he want to ask (he just want to hear some keywords) so I go around and talk everything of this topic. If finaly you can say what the specific word and everything you have said is correct then there is no problem.

- it is not like a exam, but like a conversation.

Questions

- I think that he begin with general questions.

- First he spoke about the thing he taught in the lecture: software develop, object, object model, then he ask me: could you please tell me something?

- I start from the phases of software development and compare the procedure oriented method to the object-oriented method. And why we use the object to build software:solve the complexity. and something related.
- then he ask what is a object A: a thing with state, identity and behavior.
- what is type? a thing(?) can take different forms for example? Polymophism? overloading, inclusion, cast, dynamic binding... then he write a superclass and a subclass and ask how dynamic binding works. (use pointer)
- if object in different nodes, how can they work? A: use stub, skeleton, client, server, ORB...
- how do create stub? I don't know what he want to know so I talk about DII, interface repositry and he was not satisfied. After 2-3 minute I realize that he just want to here 'interface definition language'.
- what problem will happen in these cases? A: request/reply message is lost, server/client crash... he just want to know what semantics we use to maintain reliability. So: At-least-once, at most once, exactly once... and explain them. (I did not explain at least once well and after the exam he said that's my weak point)
- What kind of methods we use in Cryptography?

BP I, OODS I
Hofmann
April 1999

Bemerkungen zu Prüfung und Prüfer

- Naja, alles in allem eine atmosphärisch angenehme Prüfung aber leider stand ich da wohl etwas zu sehr auf dem Schlauch (source quench ;-). Außerdem habe ich dummerweise versäumt, die Schlagworte, die mir durch den Kopf gingen, einfach mal auszusprechen, da sie mir dann doch etwas unmotiviert vorkamen, allerdings Hofmann wohl doch darauf gewartet hat.
- ⇒ immer mit Schlagworten parat sein (die man natürlich auch erklären können muß)
- Ergebnis: bestanden

Fragen

- Thema: Wie bringe ich einem OO-Programmierer verteilte Systeme bei
- Welche Vorteile bietet OOP bezüglich verteilter Systeme? - Kapselung, somit auch gewisse Transparenz
- Nachteile? - sind mir ganz spontan nicht so viele eingefallen... wollte wohl etwas Richtung Lücken der Transparenz
- Welche Probleme hat man bei Fernaufrufen, bzw. ganz allgemein beim Nachrichtenaustausch, welches ist am häufigsten? - Verlust, Verdopplung, Reihenfolge.
- Was muß ein OO-Programmierer wissen, wenn ich ihm ein verteiltes System unterschieben will? - Aufrufsemantik, nebenläufige/serielle Abarbeitung
- Was ist IDL, wozu dient sie? - habe ich ausgehend von CORBA erklärt, daher fragte er nochmal nach, ob es die wohl auch woanders gäbe.
- Was ist das Hauptproblem an der Nebenläufigkeit? - Ich vermute, er wollte auf Deadlocks hinaus...
- Warum ist ganz allgmein eine Transparenz der Verteilung nicht vollständig möglich? - Marshalling, Objekte sind evtl. nicht serialisierbar. Hören wollte er eigentlich, daß nur call-by-value funktioniert und Sprachen die nur call-by-reference haben kaum zur Verteilung taugen.
- Welches verallgemeinerbare Prinzip steht hinter dem Algorithmus von Lamport? - Letztlich wollte er auf verteilte Semaphore/Objekte(!) hinaus.
- Welche Probleme habe ich durch die Verteilung? - Konsistenz.