INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Prof. Dr. Dr. h.c. M. Broy
Dr. H. Ehler, S. Wagner

Lösungsblatt 9
11. Januar 2007

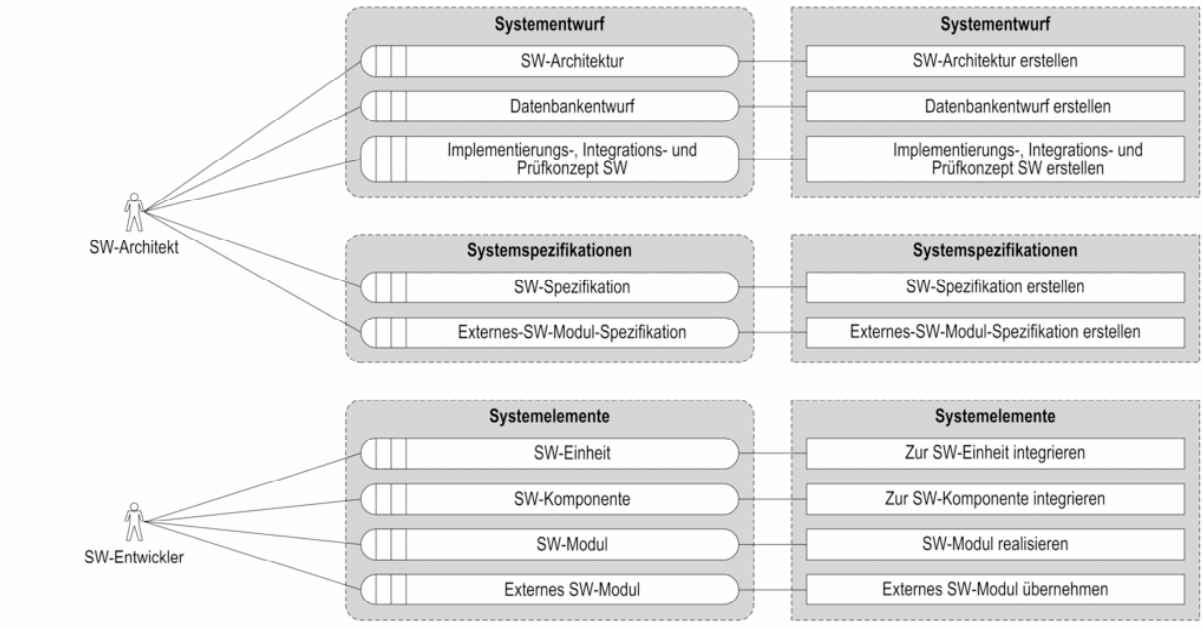
Übungen zu Softwaretechnik

Aufgabe 15 – Systemerstellung / Systemarchitektur nach dem V-Modell XT

Machen Sie sich mit den Vorgehensbausteinen „SW-Entwicklung“ und „Systemerstellung“ und mit deren Aktivitäten und Produkten sowie mit den Abläufen für „Systemarchitektur erstellen“ und „SW-Architektur erstellen“ nach dem V-Modell XT vertraut. Diese können jeweils in der Dokumentation unter http://www.v-modell-xt.de/ gefunden werden. Dies ist eine wesentliche Voraussetzung zum Verständnis der Übung!

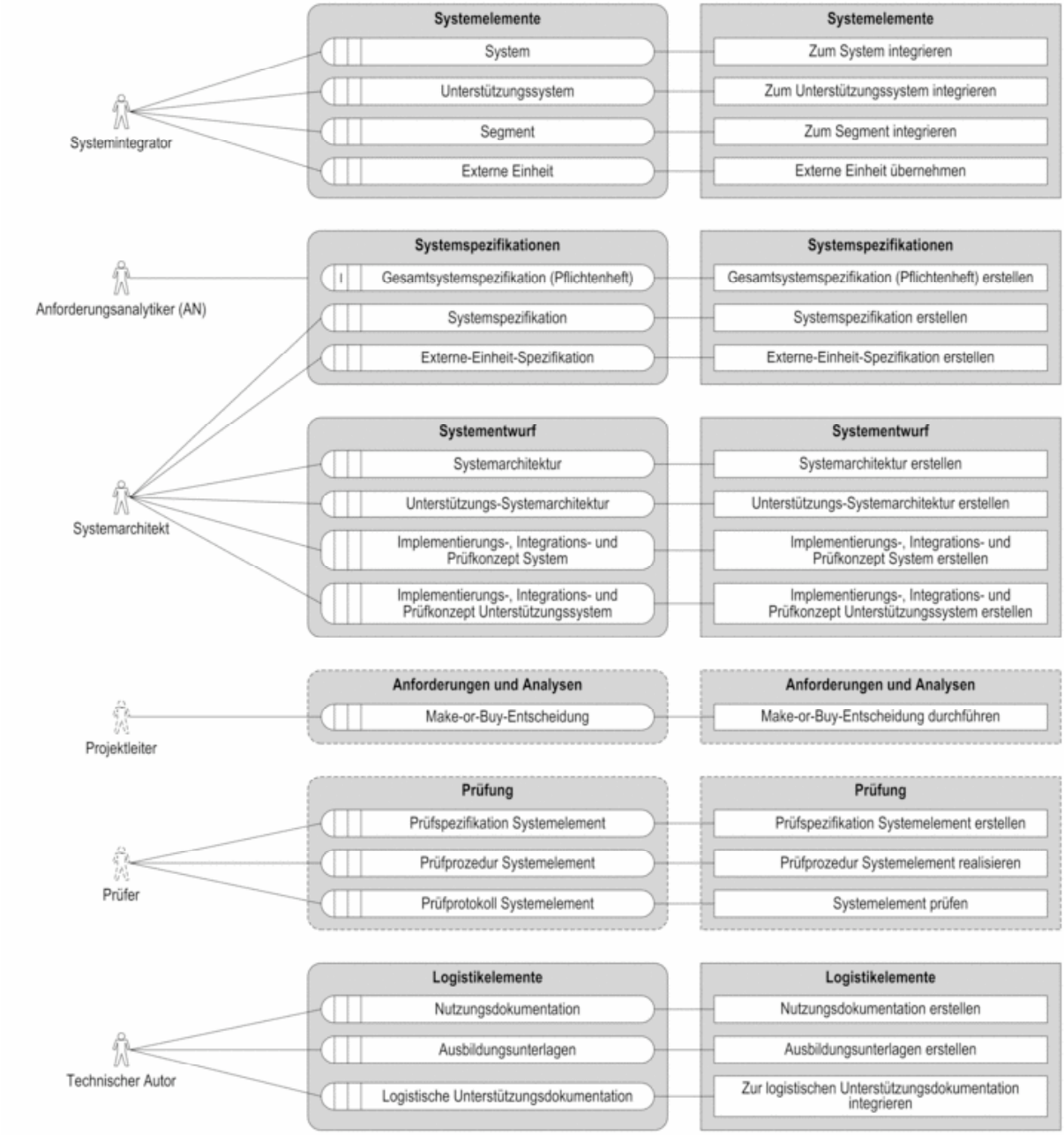
Vorgehensbaustein „SW-Entwicklung“

siehe: http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/html/8b19f694ca5306.html



Vorgehensbaustein „Systemerstellung“

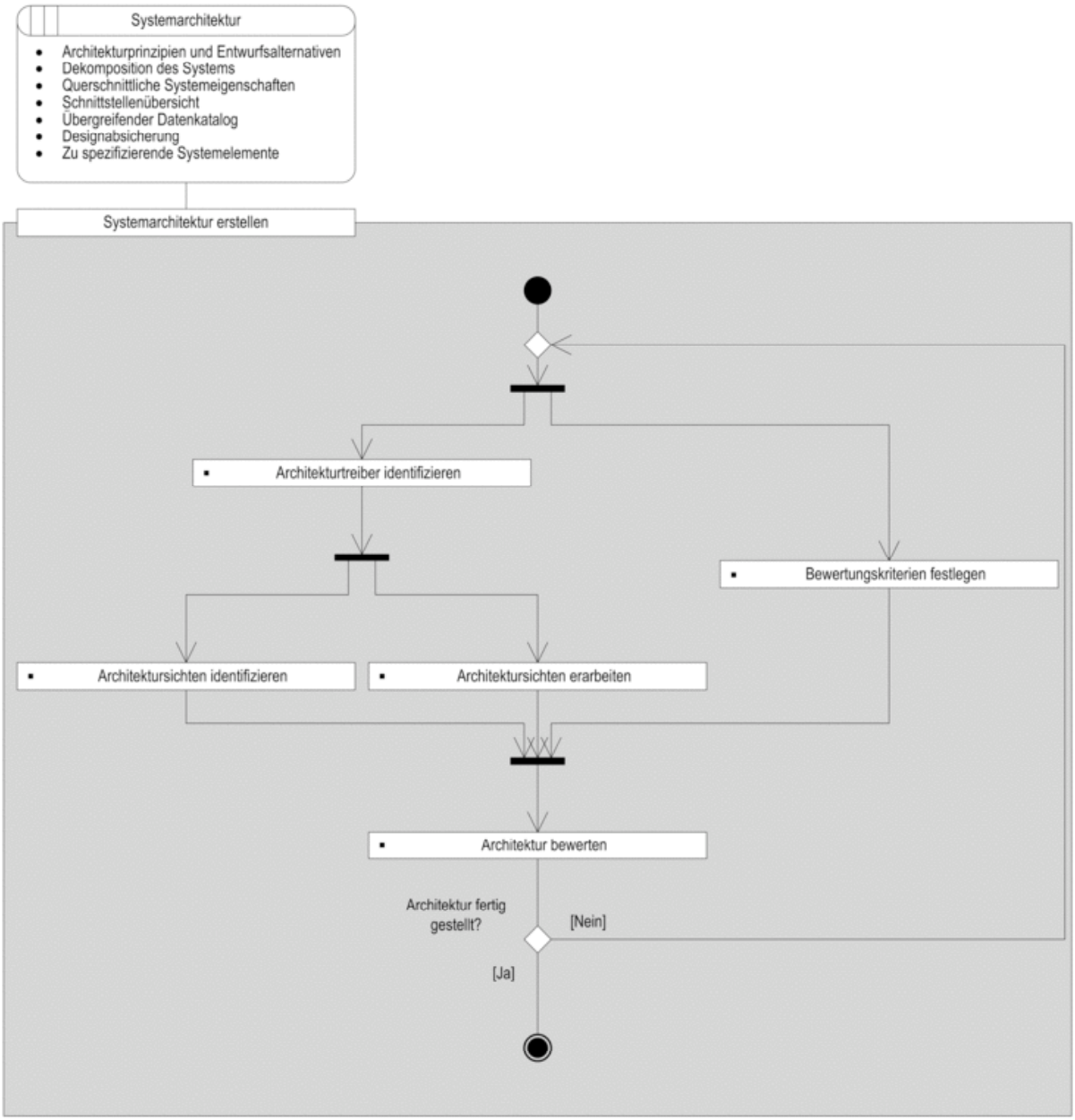
siehe: http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/html/180e3f685107411.html



Abläufe „Systemarchitektur erstellen“ und „SW-Architektur erstellen“

siehe: http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/html/1f9bf6946e4db6.html

siehe: http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/html/14d4af843e9f6d0.html



Aufgabe 16 – Architektur Information Broker

Die Weltstadt Hintercity will ein elektronisches City Information System (CIS) in einem WAN (Wide Area Network) einrichten. Verschiedene Rechner (Server) sollen den potentiellen Besuchern so Informationsdienste bereitstellen zu Themen wie Veranstaltungen, Hotels, Restaurants, Stadtpläne, Nahverkehr und andere mehr. Touristen sollen über Computerterminals (Clients) mit einem Web-Browser Zugriff auf diese Dienste haben. Der Browser dient nur zum Abruf und zur Darstellung auf dem Bildschirm dieser Informationen. Die Information selbst ist verteilt und wird nicht in den Terminals gehalten.

Folgende weiteren Anforderungen sollen erfüllt werden:

- *Clients können Dienste anderer Server in Anspruch nehmen ohne Kenntnisse über deren räumliche Verteilung und andere system- und implementierungsspezifische Details.*
- *Das CIS kann sich laufend ändern und wachsen, neue Dienste können hinzukommen, andere sich verändern oder aufgelöst werden. Die Clients speichern hierüber keine Informationen.*
- *Server, die einzelne Dienste zur Verfügung stellen, können im laufenden Betrieb ergänzt, entfernt oder ausgetauscht werden.*
- *Aus Kosten- und Flexibilitätsgründen soll kein eigenes Intranet-System installiert werden sondern das Internet genutzt werden.*

Ihre Firma, die ähnliche Aufträge von mehreren Kunden hat, will ein Architektur-Pattern für solche Informationssysteme erstellen. Dabei soll eine Broker-Komponente verwendet werden,

- *bei der sich Informationsserver an- bzw. abmelden und ihre Dienste zur Verfügung stellen,*
- *über die Clients die Dienste der Server mittels Anforderungen (Requests) in Anspruch nehmen können,*
- *die Server lokalisieren und Anforderungen, Abfrageergebnisse und Fehlermeldungen weiterleiten.*

*Legen Sie die **Struktur** eines solchen Systems fest:*

a) Beschreiben Sie die Komponenten mit deren Aufgaben und Schnittstellen zu anderen Komponenten.

b) Geben Sie diese Struktur und die Methoden der einzelnen Komponenten in Form eines Klassendiagrammes an.

*Beschreiben Sie das **Verhalten** eines solchen Systems in folgenden typischen Anwendungsfällen mittels Message Sequence Charts (MSC) oder Sequenzdiagrammen (UML):*

c) Ein Server meldet sich beim Broker an.

d) Ein Terminal schickt ein Request an den Broker.

e) Zwei über eine Bridge-Komponente getrennte Broker interagieren.

Die Aufgabenstellung und der Lösungsvorschlag sind im Wesentlichen entnommen aus: Buschmann Frank, Meunier Regine, Rohnert Hans, Sommerlad Peter, Stal Michael: Pattern-Oriented Software Architecture – A System of Patterns. John Wiley & Sons, Chichester, 1996.

Zur Vorbereitung der Aufgabe werden einige Erläuterungen zum Begriff Architektur-Pattern gegeben.

Was ist ein Pattern?
Expertenwissen, d.h. ein Problem-Lösung-Paar

Wozu Patterns?
Patterns für SW-Architektur

- behandeln wiederkehrendes Entwurfsproblem in einer konkreten Situation mit einer Lösung dazu,
- dokumentieren erprobte SW-Architekturen und Entwurfserfahrungen,

- beschreiben Abstraktionen oberhalb des Levels einzelner Klassen oder Komponenten,
- ergeben ein gemeinsames Verständnis und gemeinsamen Sprachgebrauch für Entwurfsprinzipien,
- unterstützen die Konstruktion von SW mit definierten Eigenschaften,
- helfen bei der Konstruktion komplexer und heterogener SW-Architekturen,
- helfen bei der Beherrschung der SW-Komplexität.

Was macht ein Pattern aus?
Ein Pattern ist ein Dokument mit folgender Struktur. Optionale Teile stehen in Klammern:
Pattern
- (Beispiel)
- Name, Idee, Zielsetzung, ...
- Kontext
  - Entwurfssituation für bestimmtes Entwurfsproblem
- Problem
  - Menge von Aufgaben, die in der Situation wiederholt auftreten
- Lösung
  - bewährte Architektur, die die Aufgaben erfüllt
    - statische Struktur mit Komponenten und Beziehungen
    - dynamisches Laufzeitverhalten
  - (Implementierung)
  - (Varianten)
  - (Referenzanwendungen)
  - (...)

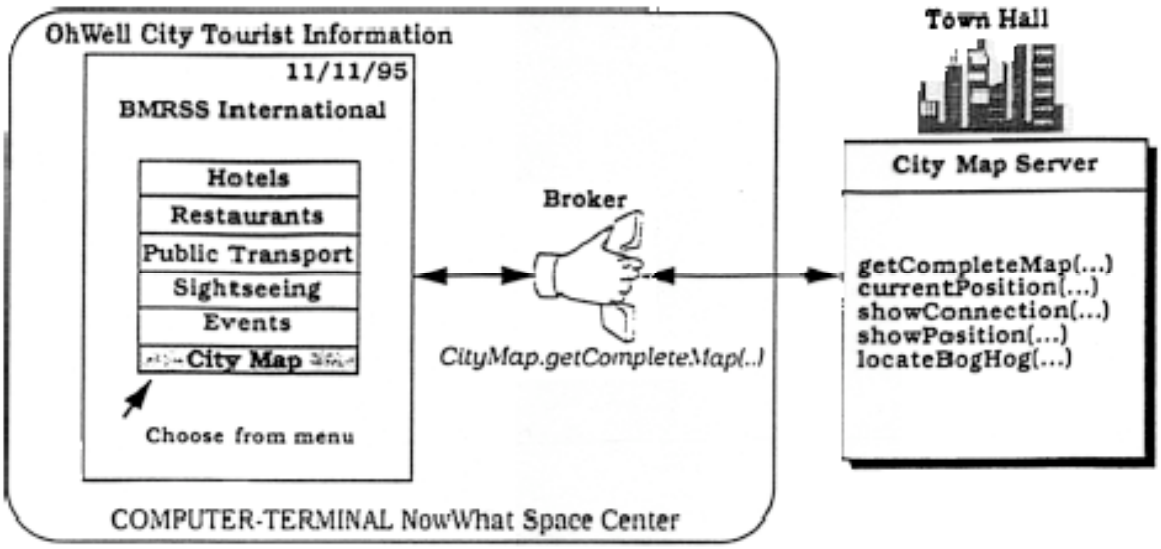
Wie unterscheiden sich Architektur-Pattern und Design-Pattern?

Architektur-Pattern: Pattern für (Architektur-)Entwurf eines großen Systems.

Design-Pattern: Pattern für Entwurf von Subsystemen oder Komponenten.

*Legen Sie die **Struktur** eines solchen Systems fest:*

a) Beschreiben Sie die Komponenten mit deren Aufgaben und Schnittstellen zu anderen Komponenten.



Class Client	Collaborators • Client-side Proxy • Broker
Responsibility • Implements user functionality. • Sends requests to servers through a client-side proxy.	

Class Server	Collaborators • Server-side Proxy • Broker
Responsibility • Implements services. • Registers itself with the local broker. • Sends responses and exceptions back to the client through a server-side proxy.	

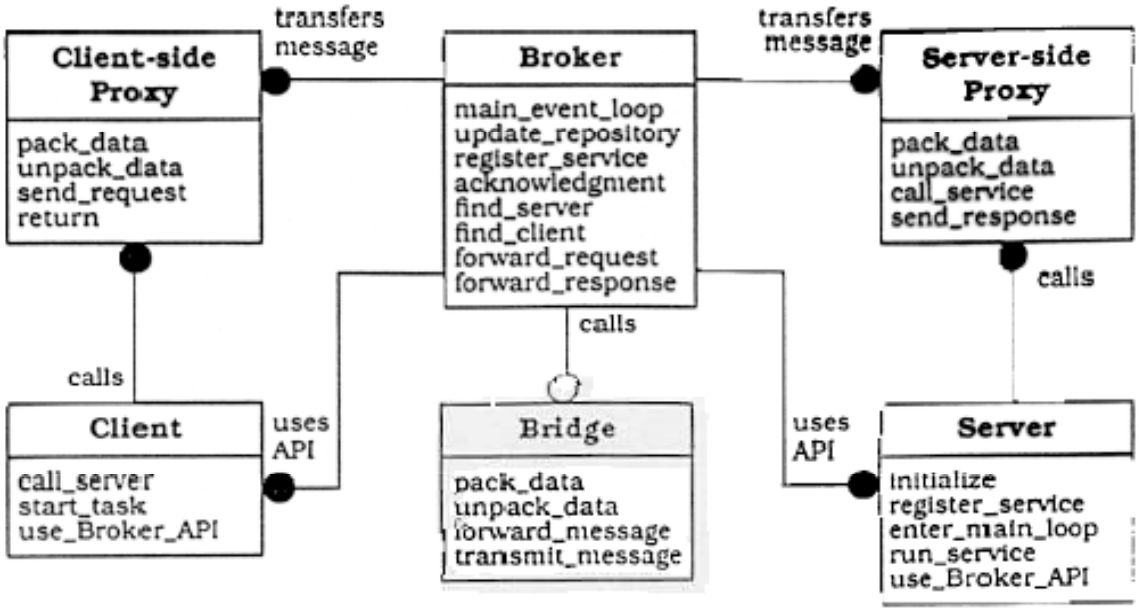
Class Client-side Proxy	Collaborators • Client • Broker
Responsibility • Encapsulates system-specific functionality. • Mediates between the client and the broker.	

Class Server-side Proxy	Collaborators • Server • Broker
Responsibility • Calls services within the server. • Encapsulates system-specific functionality. • Mediates between the server and the broker.	

Class Broker	Collaborators • Client • Server • Client-side Proxy • Server-side Proxy • Bridge
Responsibility • (Un-)registers servers. • Offers APIs. • Transfers messages. • Error recovery. • Interoperates with other brokers through bridges. • Locates servers.	

Class Bridge	Collaborators • Broker • Bridge
Responsibility • Encapsulates network-specific functionality. • Mediates between the local broker and the bridge of a remote broker.	

b) Geben Sie diese Struktur und die Methoden der einzelnen Komponenten in Form eines Klassendiagrammes an.

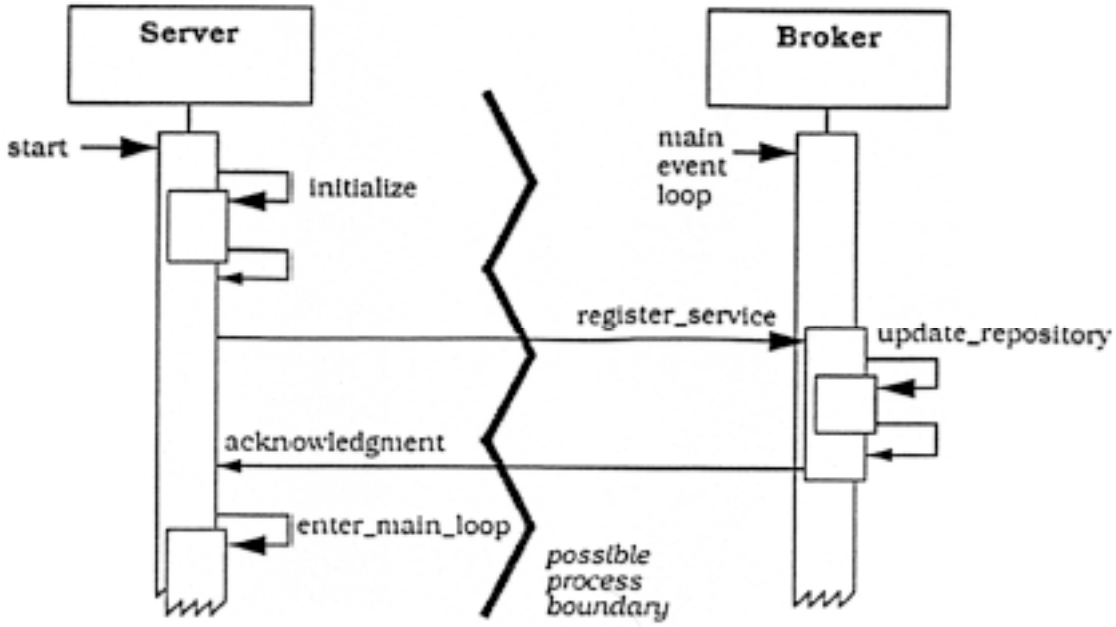


Korrekturen hierzu:

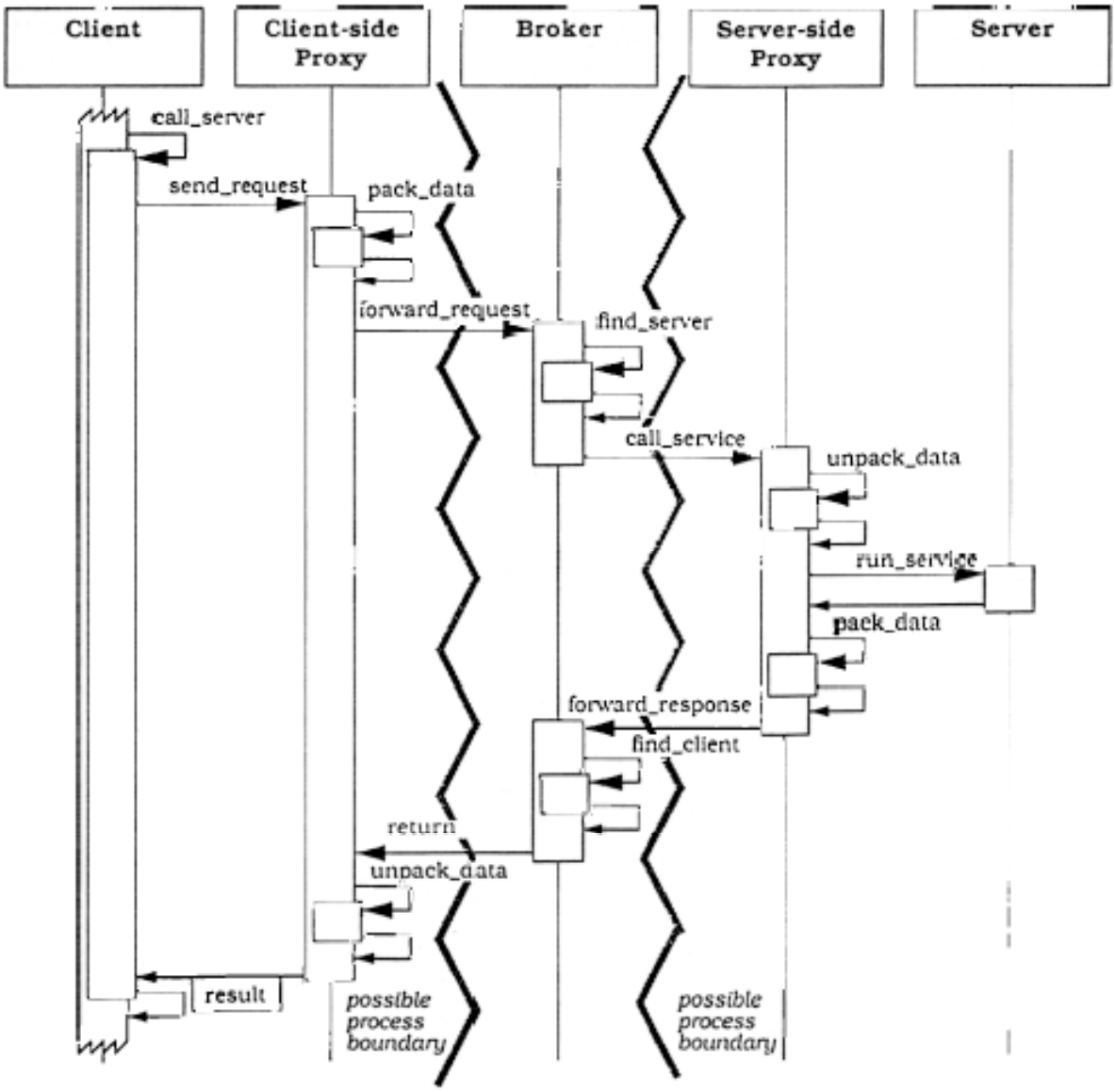
- acknowledgement in **Broker** streichen und in **Server** ergänzen
- find_bridge in **Bridge** ergänzen
- register_service in **Server** streichen
- 1-n Beziehung transmits_message ergänzen zwischen **Bridge** und **Bridge**

*Beschreiben Sie das **Verhalten** eines solchen Systems in folgenden typischen Anwendungsfällen mittels Message Sequence Charts (MSC) oder Sequenzdiagrammen (UML):*

c) Ein Server meldet sich beim Broker an.



d) Ein Terminal schickt ein Request an den Broker.



e) Zwei über eine Bridge-Komponente getrennte Broker interagieren.