Yet Another Encrypted Messenger

Florian Amstutz

02. April 2012
Semesterarbeit an der Zürcher Hochschule für Angewandte Wissenschaften

Inhaltsverzeichnis

1 Management Summary

Mit dem zunehmenden Aufkommen von Attacken und gezieltem Abhorchen von Echtzeitkommunikation via E-Mail oder Instant Messaging steigt der Bedarf an eine sichere und einfache Übertragungsart von Nachrichten oder Daten.

Als Nutzer eines Kommunikationskanals über das öffentliche Internet will ich die Möglichkeit haben meine privaten Daten verschlüsselt und sicher an einen oder mehrere Empfänger übertragen zu können. Ich will dabei eine einfach zu bedienende Applikation zur Verfügung haben um meine geheimen Daten übertragen zu können und so potentiellen Mithörern keine Klartextinformationen zur Verfügung zu stellen.

Diese Applikation soll im Rahmen der Semesterarbeit im dritten Studienjahr an der ZHAW entwickelt werden. Dabei wird der Schwerpunkt der Arbeit auf der methodischen Vorgehensweise der Softwareentwicklung und weniger auf der Implementierung der kryptografischen Algorithmen.

2 Softwareentwicklungsprozess

Software lässt sich nach einer Vielzahl von Prozessen entwickeln. Von iterativen Vorgehen wie Scrum über komplexe Modelle wie RUP hin zu klassischen, linearen Vorgehen wie dem Wasserfallmodell oder dem V-Modell. Zur Entwicklung der Semesterarbeit wurde

3 Anforderungen

Die Anforderungen an die Applikation werden in Use-Case-Diagrammen modellhaft dargestellt und als Use-Case-Spezifikationen ausformuliert. Auf eine natürlichsprachige Dokumentation der Anforderungen wird verzichtet, da die Anforderungen aufgrund der Use-Case-Diagrammen verständlich genug sind und alle zusätzlich zu den Diagrammen zu beachtenden Punkte in den Use-Case-Spezifikationen enthalten sind.

3.1 Systemkontext

Der Systemkontext ist der Teil der Umgebung eines Systems, der für die Definitino und das Verständnis der Anfoderungen des betrachteten Systems relevant ist (nach [1]).

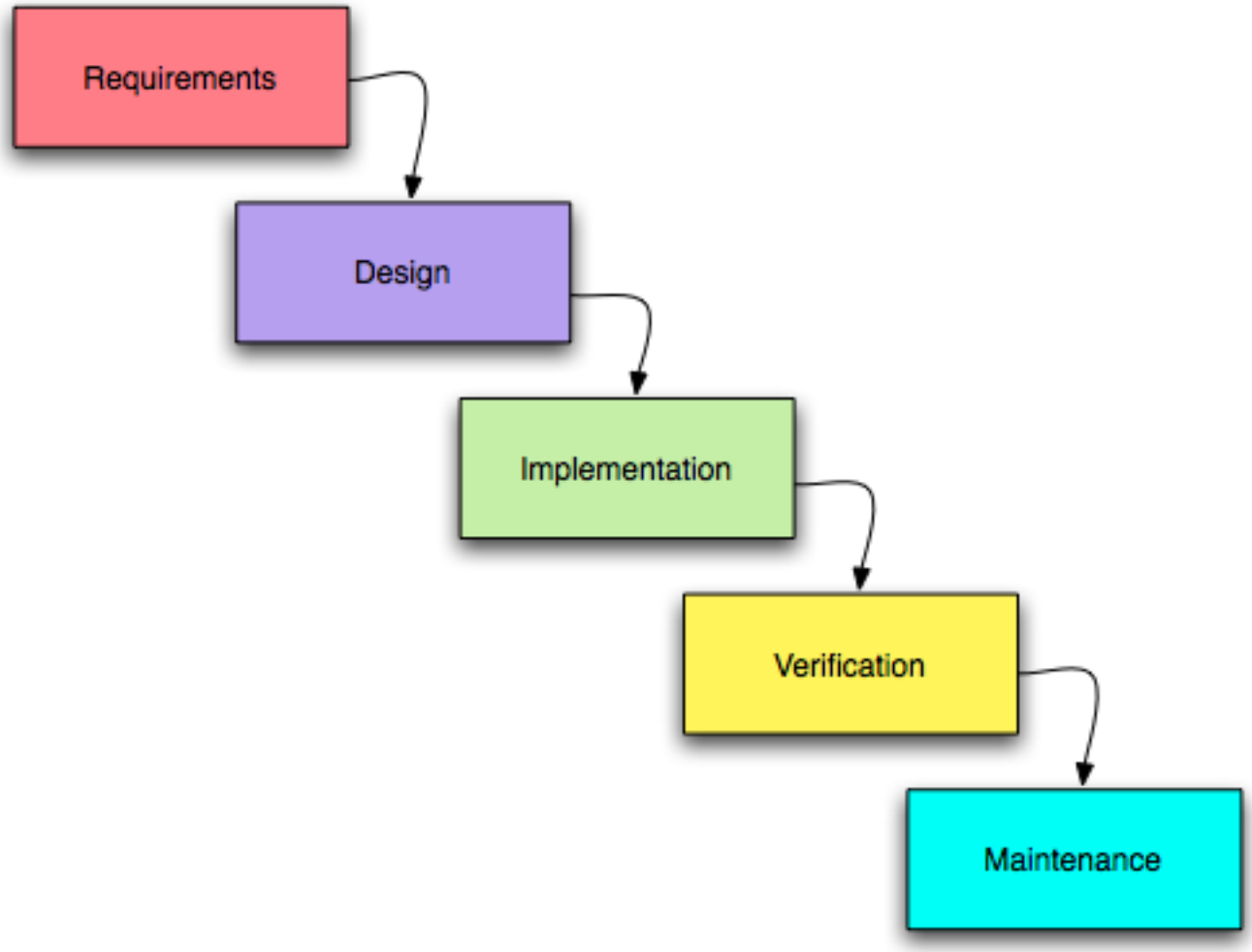
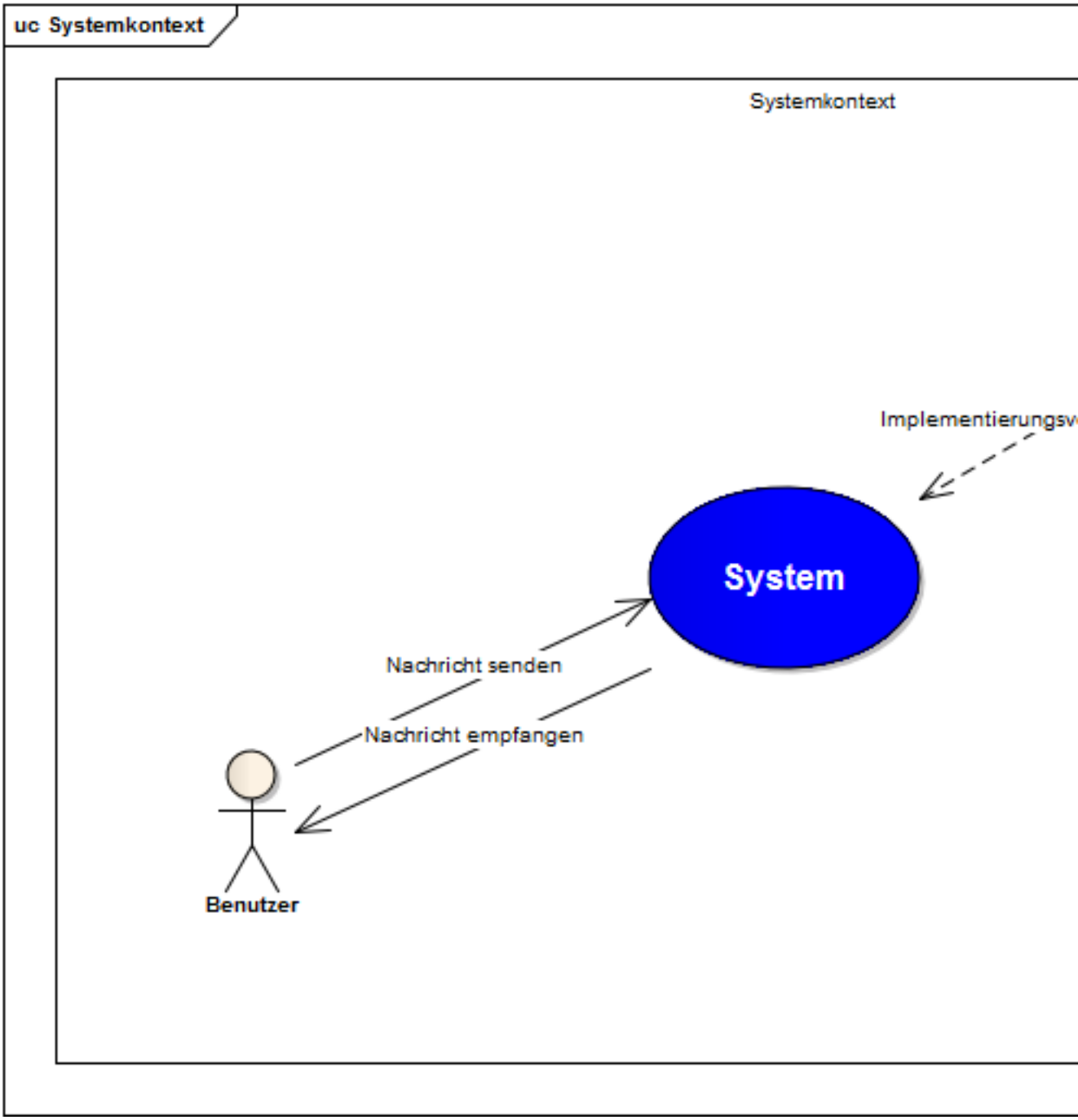


Abbildung 1: Wasserfallprozess nach [2]

Der Ursprung der Anforderungen des Systems liegt im Systemkontext des geplanten Systems. Aus diesem Grund wird der Systemkontext vor Erhebung und Dokumentierung der Anforderungen festgelegt

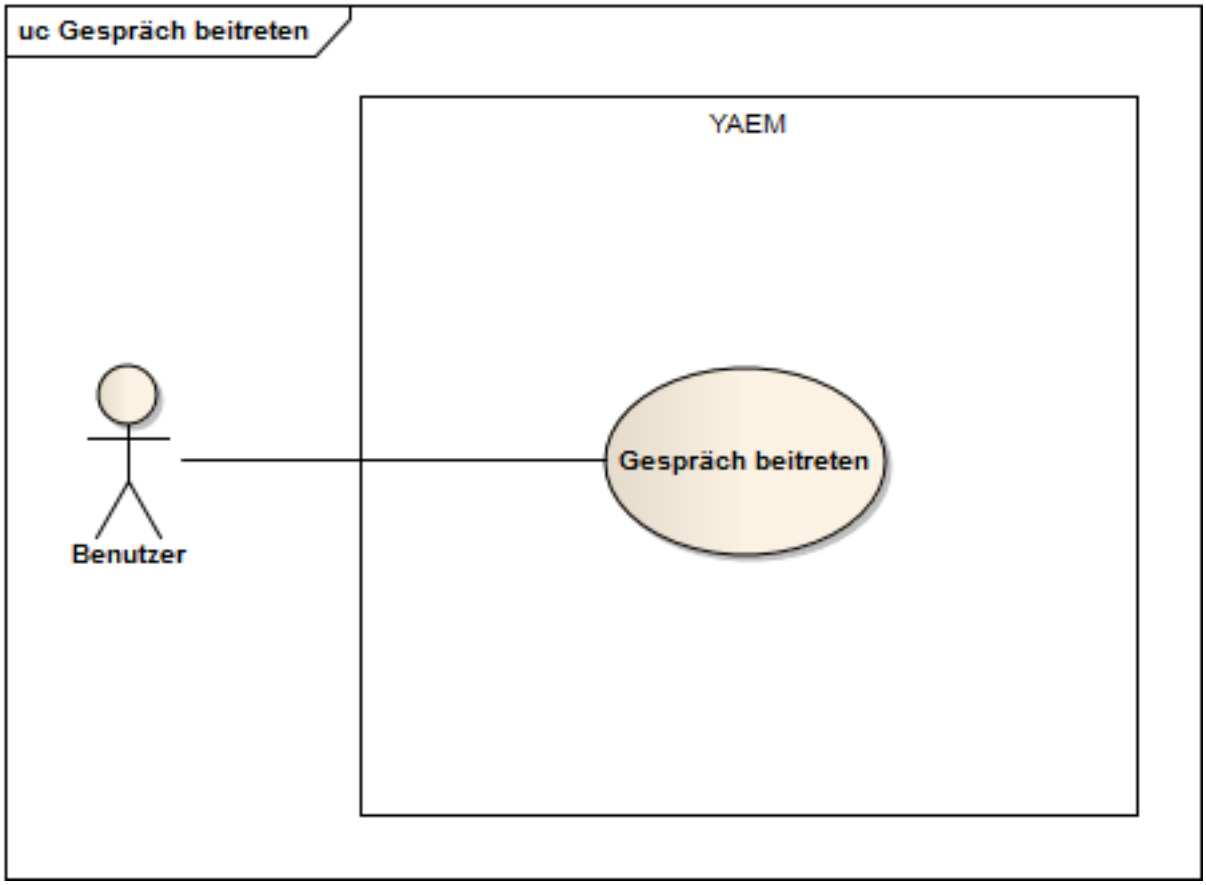


3.2 Use-Case-Spezifikationen

Nach [1] zeigen Use-Case-Diagamme die aus einer externen Nutzungssicht wesentlichen Funktionalitäten des betrachteten Systems sowie spezifische Beziehungen der einzelnen Funktionalitäten untereinander bzw. zu Aspekten in der Umgebung des Systems. Abgesehen vom Namen eines Use-Cases und dessen Beziehungen dokumentieren Use-Case-Diagramme allerdings keinerlei weitere Informationen über die einzelnen Use-Cases, wie z.B. die Systematik der Interaktion eines Use Case mit Akteuren in der Umgebung. Diese Informationen werden unter Verwendung einer geeigneten Schablone zusätzlich zum Use-Case-Diagramm textuell dokumentiert.

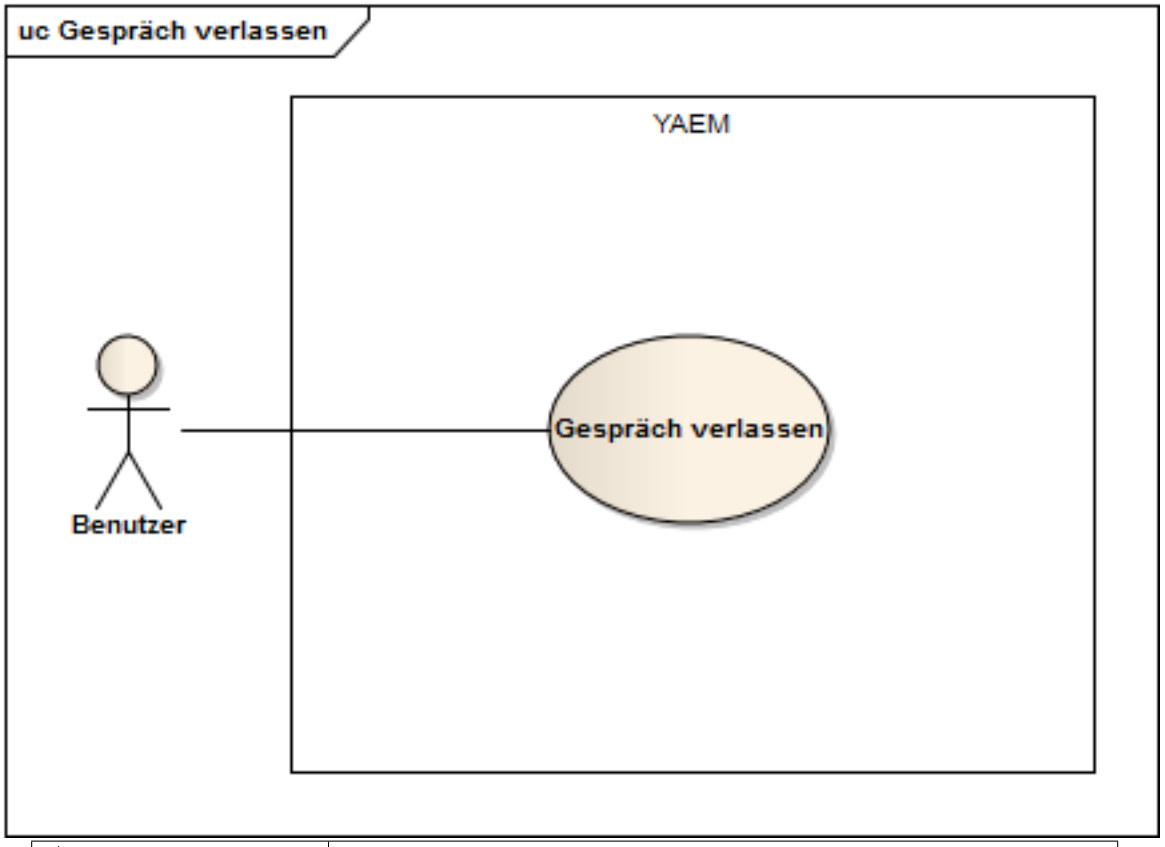
Die verwendete Schablone für die Use-Case-Spezifikationen stammt aus [1] und dient zur zweckmässigen Strukturierung von Typen von Informationen, die einen Use-Case betreffen. Die Abschnitte Autor, Quelle, Verantwortlicher und Qualität werden ausgelassen, da sie für die Semesterarbeit keine Relevant besitzen.

3.2.1 Gespräch beitreten



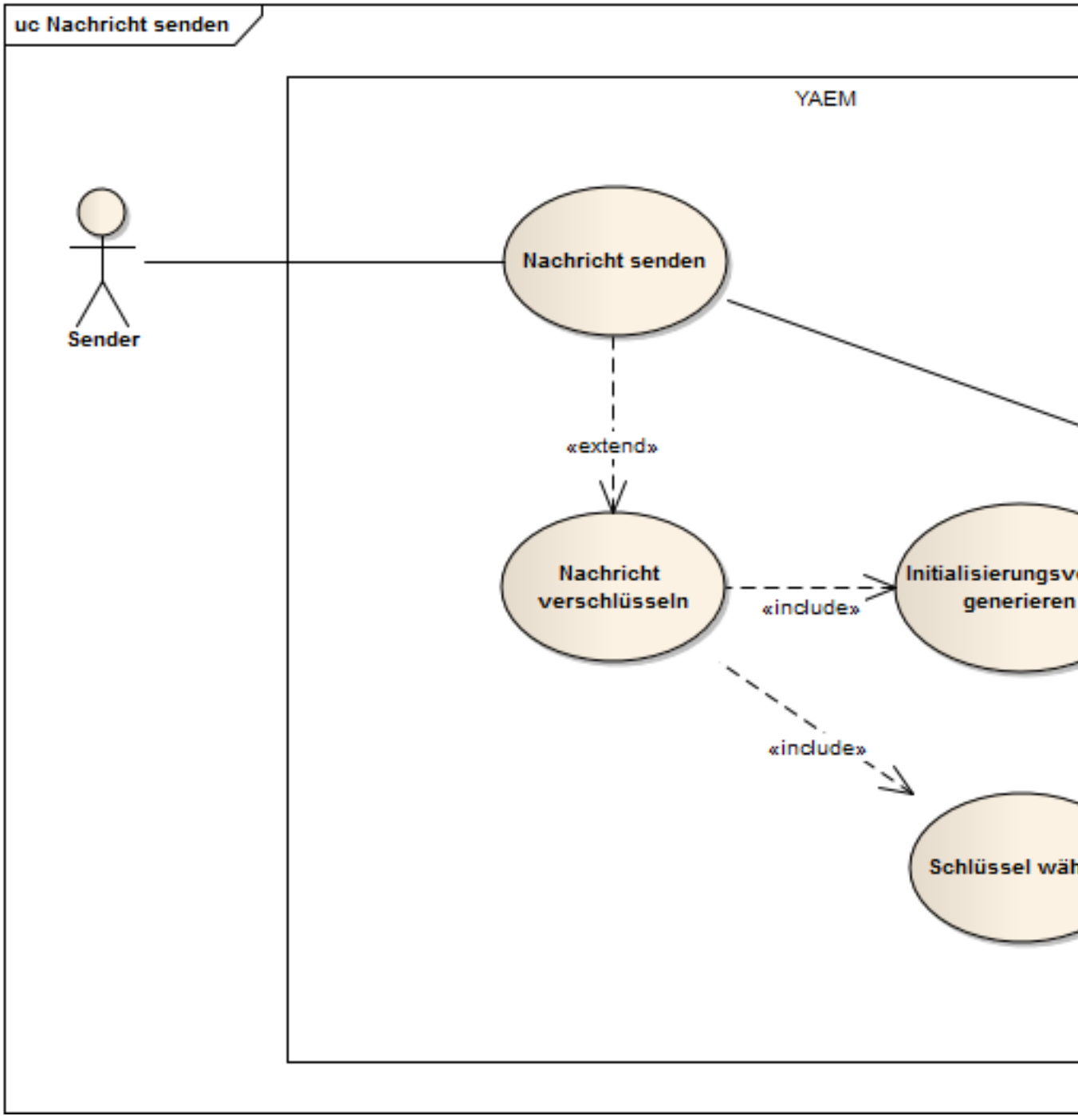
Abschnitt	Inhalt
Bezeichner	UC1
Name	Gespräch beitreten
Priorität	Wichtigkeit für Systemerfolg: hoch Technologisches Risiko: niedrig
Kritikalität	Hoch
Beschreibung	Der Benutzer tritt einem Gespräch bei.
Auslösendes Ereignis	Benutzer möchte einem Gespräch beitreten.
Akteure	Benutzer
Vorbedingung	Der Benutzer ist nicht schon einem Gespräch beigetreten.
Nachbedingung	Der Benutzer kann Nachrichten versenden und Nachrichten anderer Gesprächsteilnehmer empfangen.
Ergebnis	Session-Ticket wird erstellt.
Hauptszenario	1. Der Benutzer wählt einen Benutzernamen. 2. Der Benutzer stellt eine Verbindung zum Server her. 3. Der Server erstellt eine Session-Ticket für den Benutzer und gibt ihm dieses zurück.
Alternativszenarien	2a. Der gewählte Benutzername ist bereits im Gespräch vorhanden. 2a1. Der Benutzer wird aufgefordert einen anderen Benutzernamen auszuwählen.
Ausnahmeszenarien	Auslösendes Ereignis: Der Benutzer kann keine Verbindung zum Server herstellen.

3.2.2 Gespräch verlassen



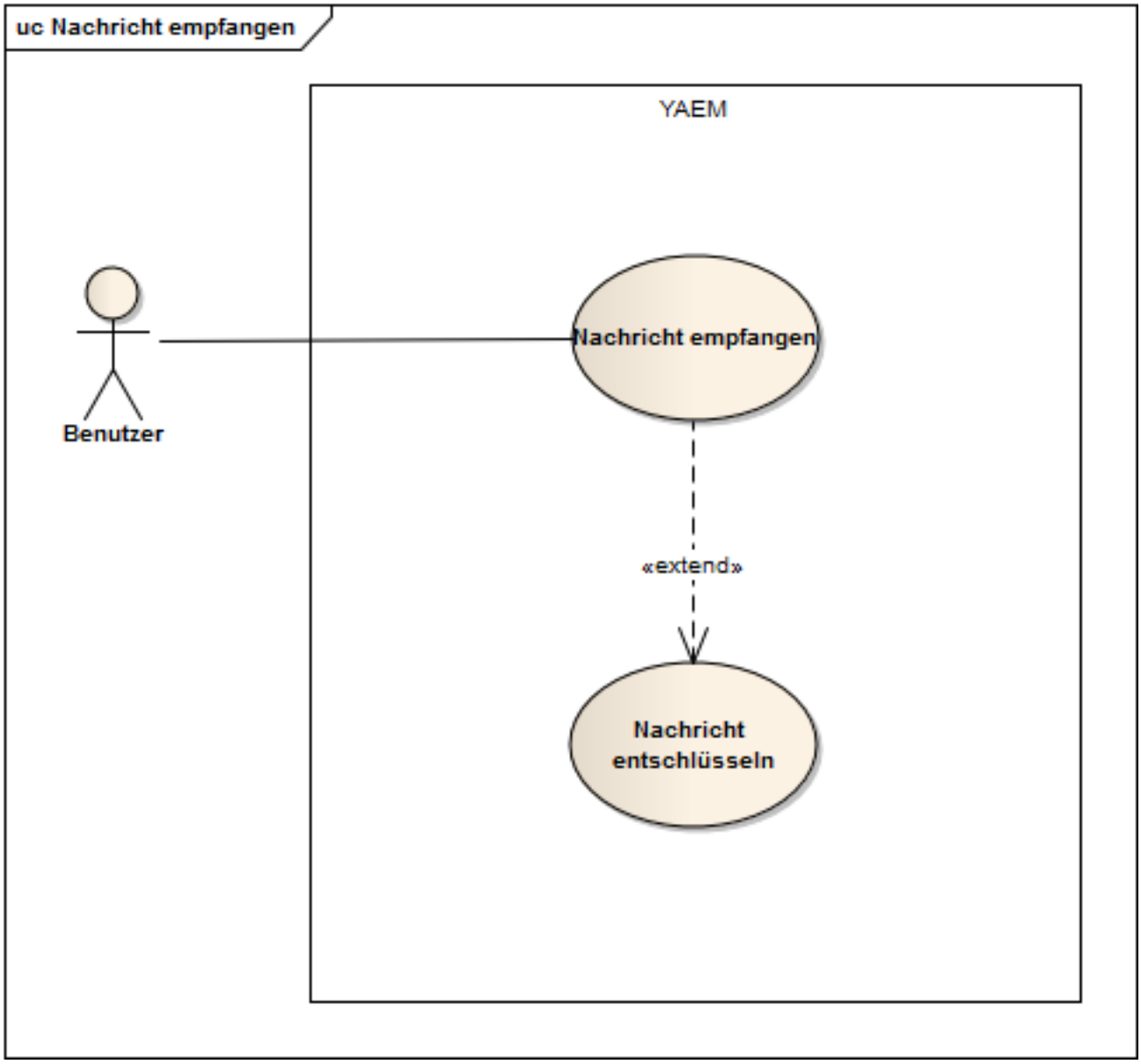
Abschnitt	Inhalt
Bezeichner	UC2
Name	Gespräch verlassen
Priorität	Wichtigkeit für Systemerfolg: hoch Technologisches Risiko: niedrig
Kritikalität	Hoch
Beschreibung	Der Benutzer verlässt ein Gespräch.
Auslösendes Ereignis	Benutzer möchte eine Gespräch verlassen.
Akteure	Benutzer
Vorbedingung	Der Benutzer ist einem Gespräch beigetreten.
Nachbedingung	Der Benutzer kann erneut einem Gespräch beitreten.
Ergebnis	Session-Ticket ist abgelaufen.
Hauptszenario	1. Der Benutzer verlässt das Gespräch. 2. Der Server erklärt das Session-Ticket des Benutzers für abgelaufen und sendet das aktualisierte Ticket dem Benutzer zu.
Alternativszenarien	Keine
Ausnahmeszenarien	Keine

3.2.3 Nachricht senden



Abschnitt	Inhalt
Bezeichner	UC3
Name	Nachricht senden
Priorität	Wichtigkeit für Systemerfolg: hoch Technologisches Risiko: mittel
Kritikalität	Hoch
Beschreibung	Der Benutzer versendet eine Nachricht.
Auslösendes Ereignis	Benutzer möchte eine Nachricht senden.
Akteure	Benutzer
Vorbedingung	Der Benutzer ist im Gespräch angemeldet und besitzt eine gültiges Session-Ticket.
Nachbedingung	Der Benutzer kann erneut eine Nachricht versenden und Nachrichten anderer Gesprächsteilnehmer empfangen.
Ergebnis	Die Empfänger haben die versendete Nachricht empfangen.
Hauptszenario	1. Der Benutzer erfasst die zu versenden Nachricht 2. Der Benutzer wählt einen Kryptoalgorithmus aus. 3. Der Benutzer generiert einen Initalisierungsvektor. 4. Der Initialisierungsvektor wird an alle Empfänger gesendet. 5. Der Benutzer wählt einen Schlüssel. 6. Der Schlüssel wird an alle Empfänger gesendet. 7. Der Benutzer verschickt die (verschlüsselte) Nachricht.
Alternativszenarien	2a. Der Benutzer wählt keinen Kryptoalgorithmus aus. 2a1. Der Benutzer versendet die Nachricht unverschlüselt. 3a. Der Benutzer hat bereits einen Intialisierungsvektor erstellt oder einen Initalisierungsvektor von einem anderen Teilnehmer des Gesprächs erhalten und generiert keinen neuen Initialisierungsvektor. 4a. Der Benutzer hat bereits einen Schlüssel erstellt oder einen Schlüssel von einem anderen Teilnehmer des Gesprächs erhalten und wählt keinen neuen Schlüssel.
Ausnahmeszenarien	Auslösendes Ereignis: Der Benutzer kann keine Verbindung zum Server herstellen.

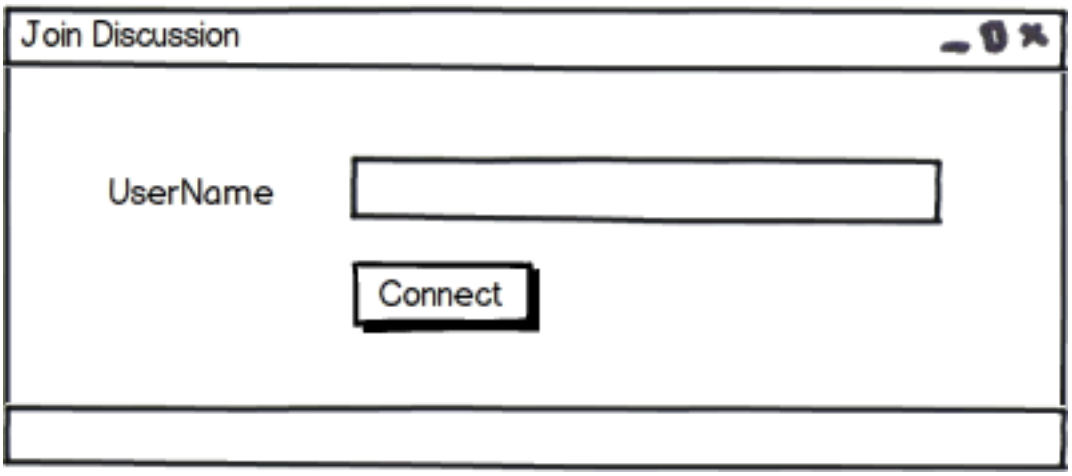
3.2.4 Nachricht empfangen



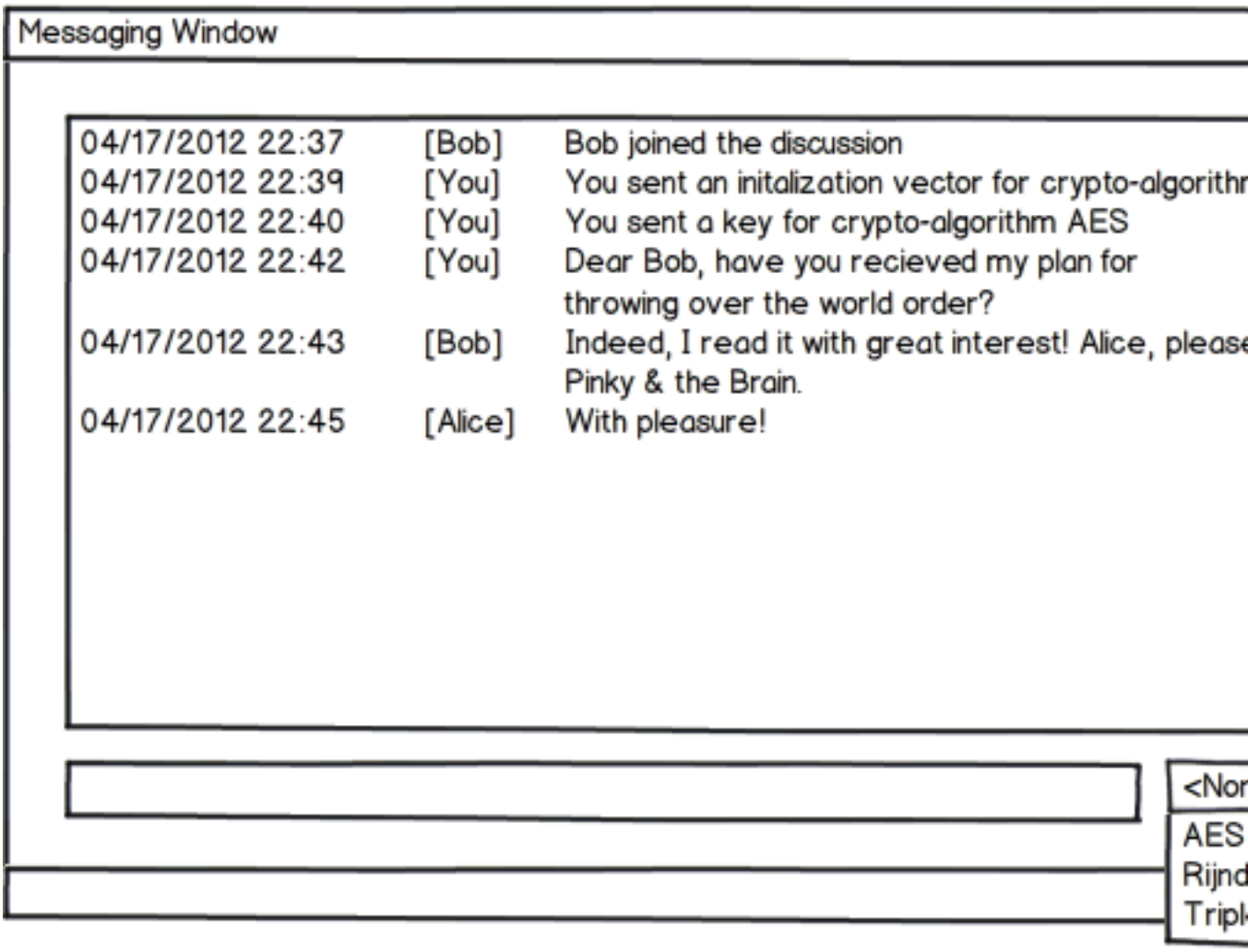
Abschnitt	Inhalt
Bezeichner	UC4
Name	Nachricht empfangen
Priorität	Wichtigkeit für Systemerfolg: hoch Technologisches Risiko: mittel
Kritikalität	Hoch
Beschreibung	Der Benutzer empfängt eine Nachricht.
Auslösendes Ereignis	Ein anderer Teilnehmer des Gesprächs versendet eine Nachricht.
Akteure	Benutzer
Vorbedingung	Der Benutzer ist im Gespräch angemeldet und besitzt eine gültiges Session-Ticket. Ein Teilnehmer des Gesprächs versendet eine Nachricht.
Nachbedingung	Der Benutzer kann Nachrichten versenden und Nachrichten anderer Gesprächsteilnehmer empfangen.
Ergebnis	Die Nachricht wird dem Benutzer angezeigt.
Hauptszenario	1. Der Benutzer empfängt die Nachricht und prüft ob diese verschlüsselt ist. 2. Der Benutzer verwendet den Initialisierungsvektor und Schlüssel zum entschlüsseln der Nachricht. 3. Die entschlüsselte Nachricht wird angezeigt.
Alternativszenarien	1a. Ist die Nachricht nicht verschlüsselt, wird sie direkt angezeigt.
Ausnahmeszenarien	Ist kein Initalisierungsvektor, Schlüssel oder Implementierung des verwendeten Kryptoalgorithmus vorhanden, so wird der unlesbare Geheimtext angezeigt.

3.3 Mockups

3.3.1 Connect Window



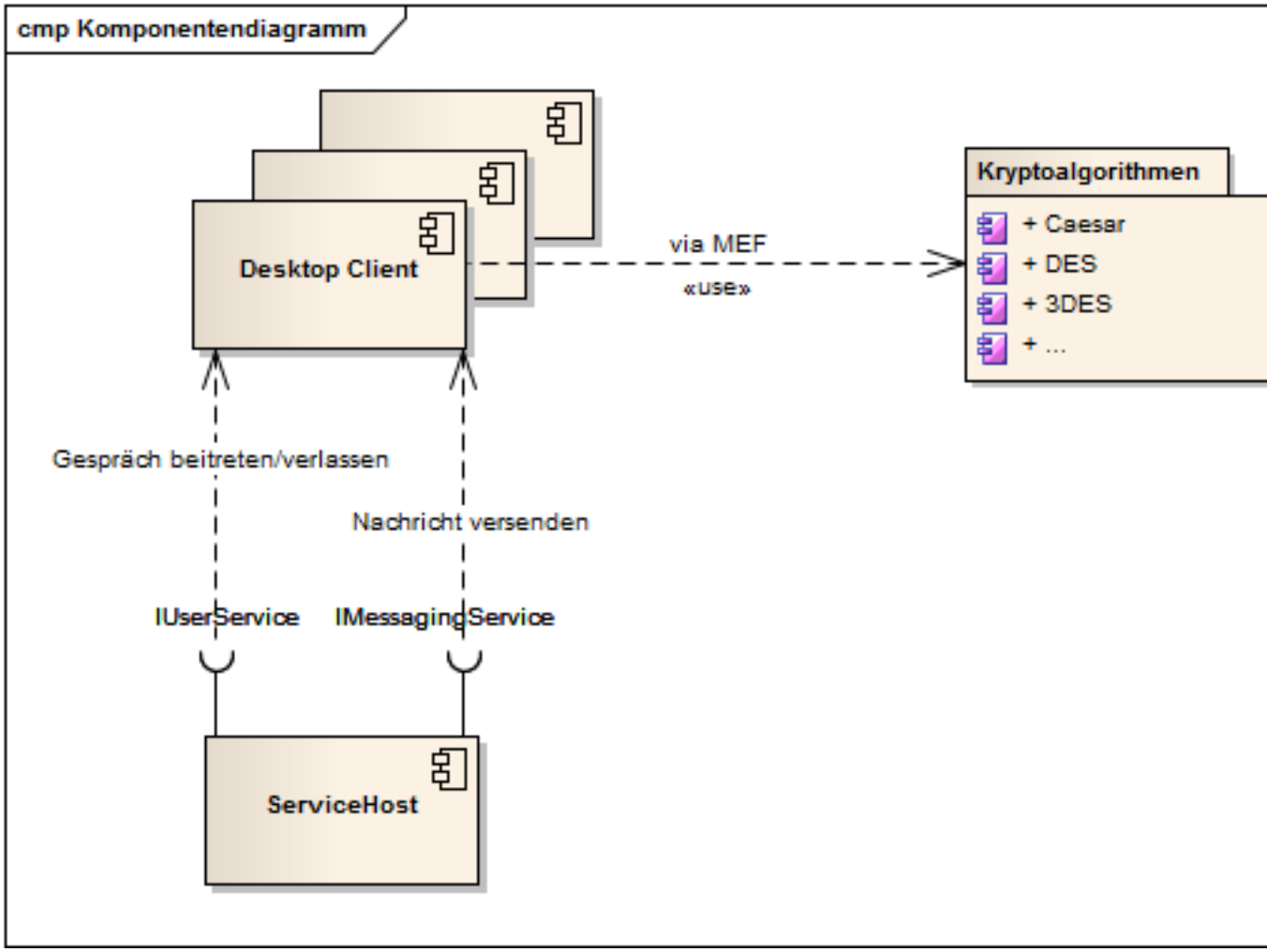
3.3.2 Messaging Window



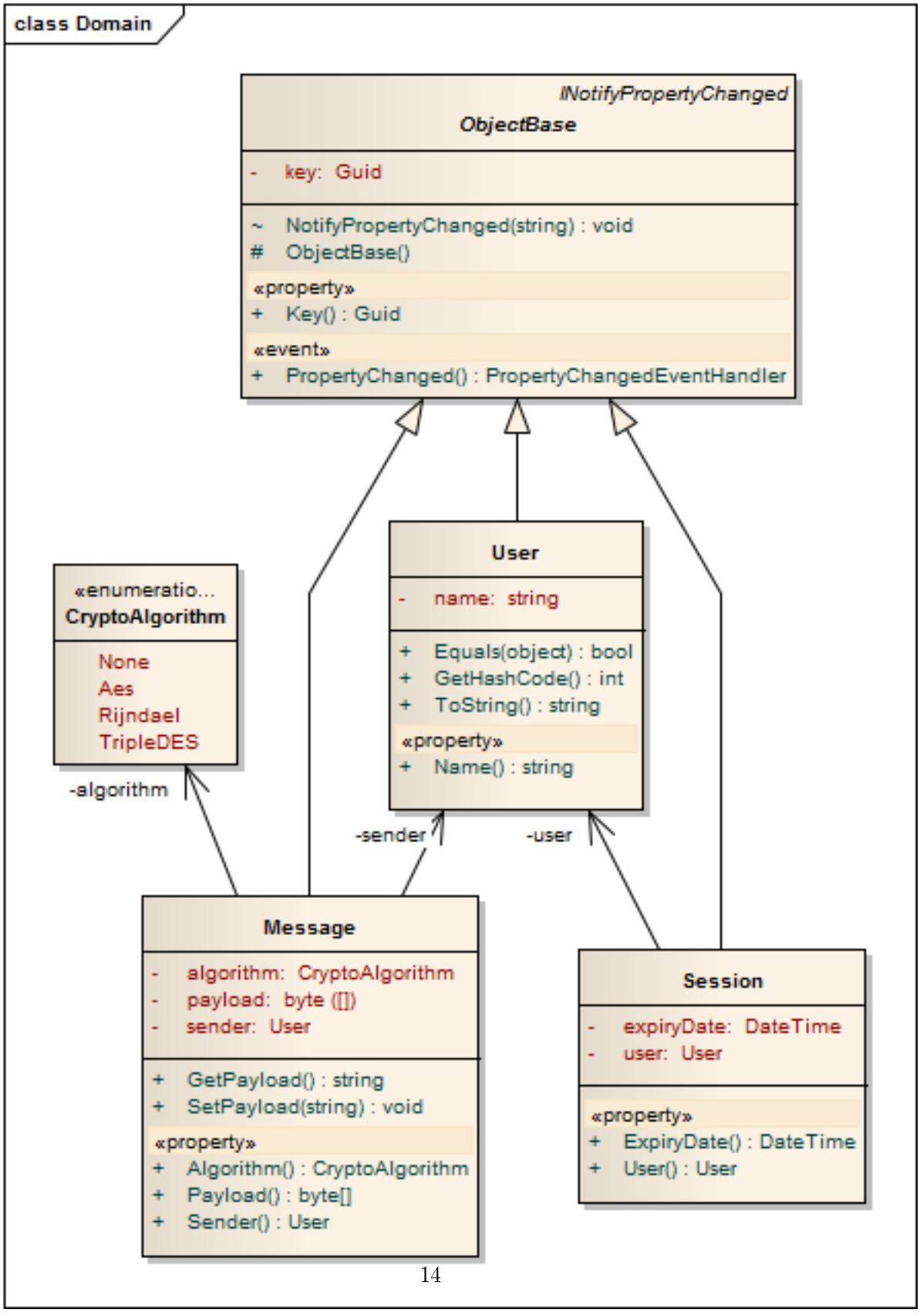
4 Konzept

4.1 Bausteinsicht

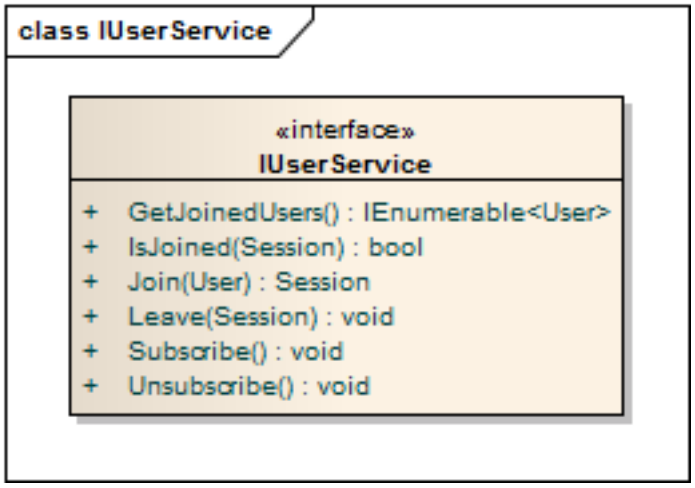
4.1.1 Komponentendiagramm



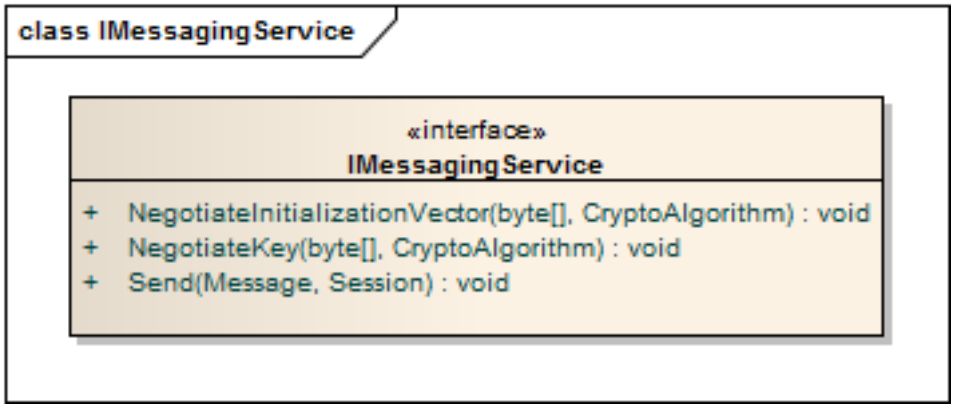
4.1.2 Domänenmodell



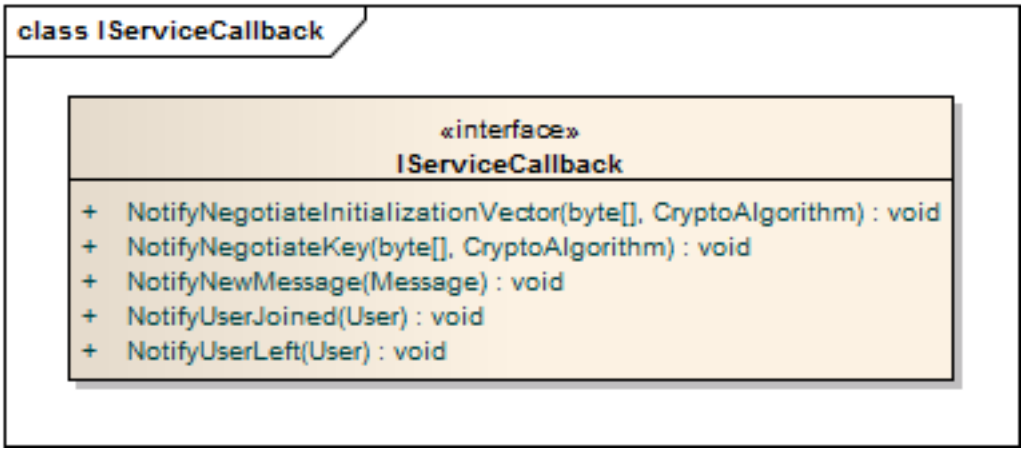
4.1.3 Service Contracts



IUserService

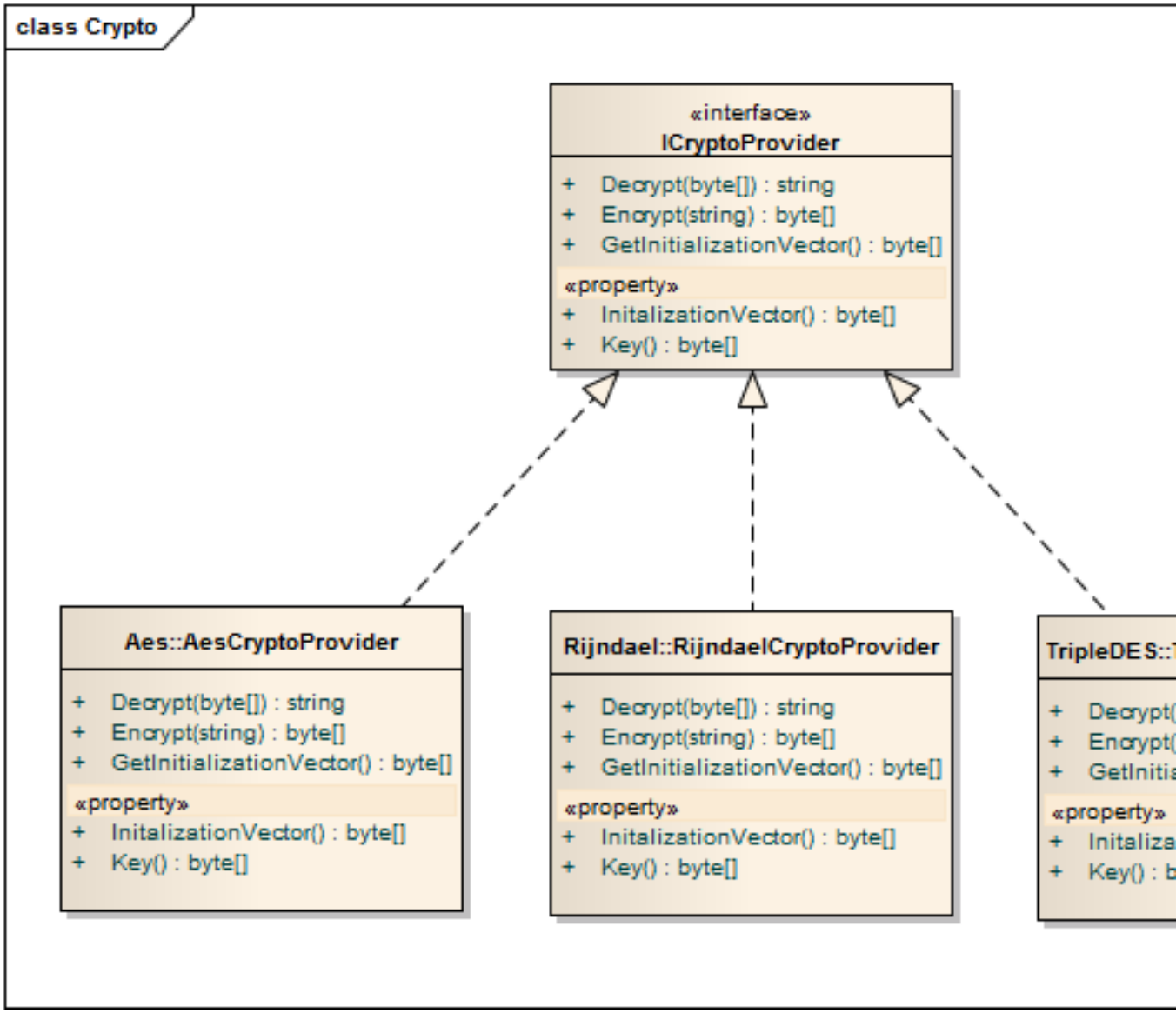


IMessagingService

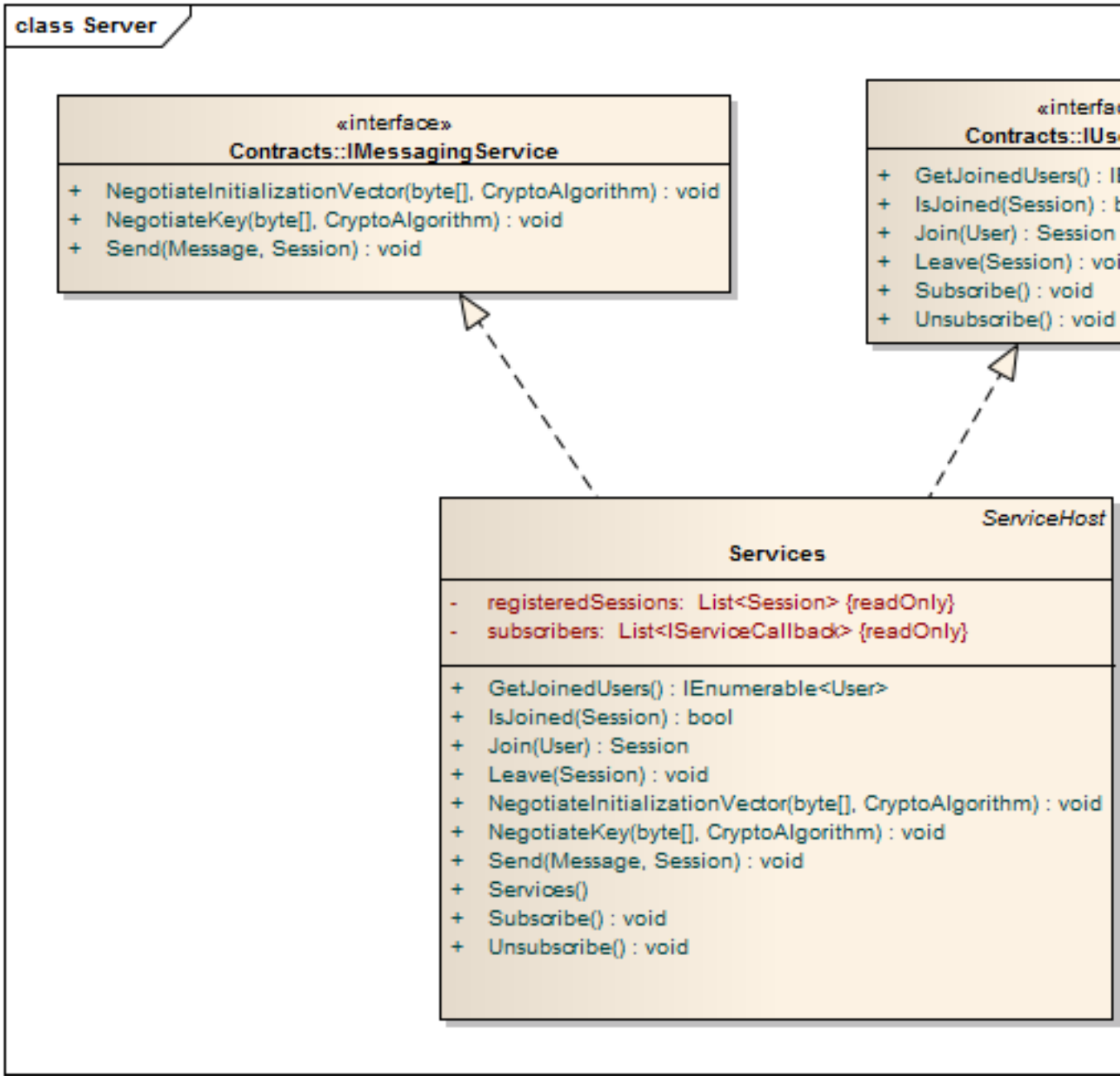


IServiceCallback

4.1.4 Kryptoalgorithmen

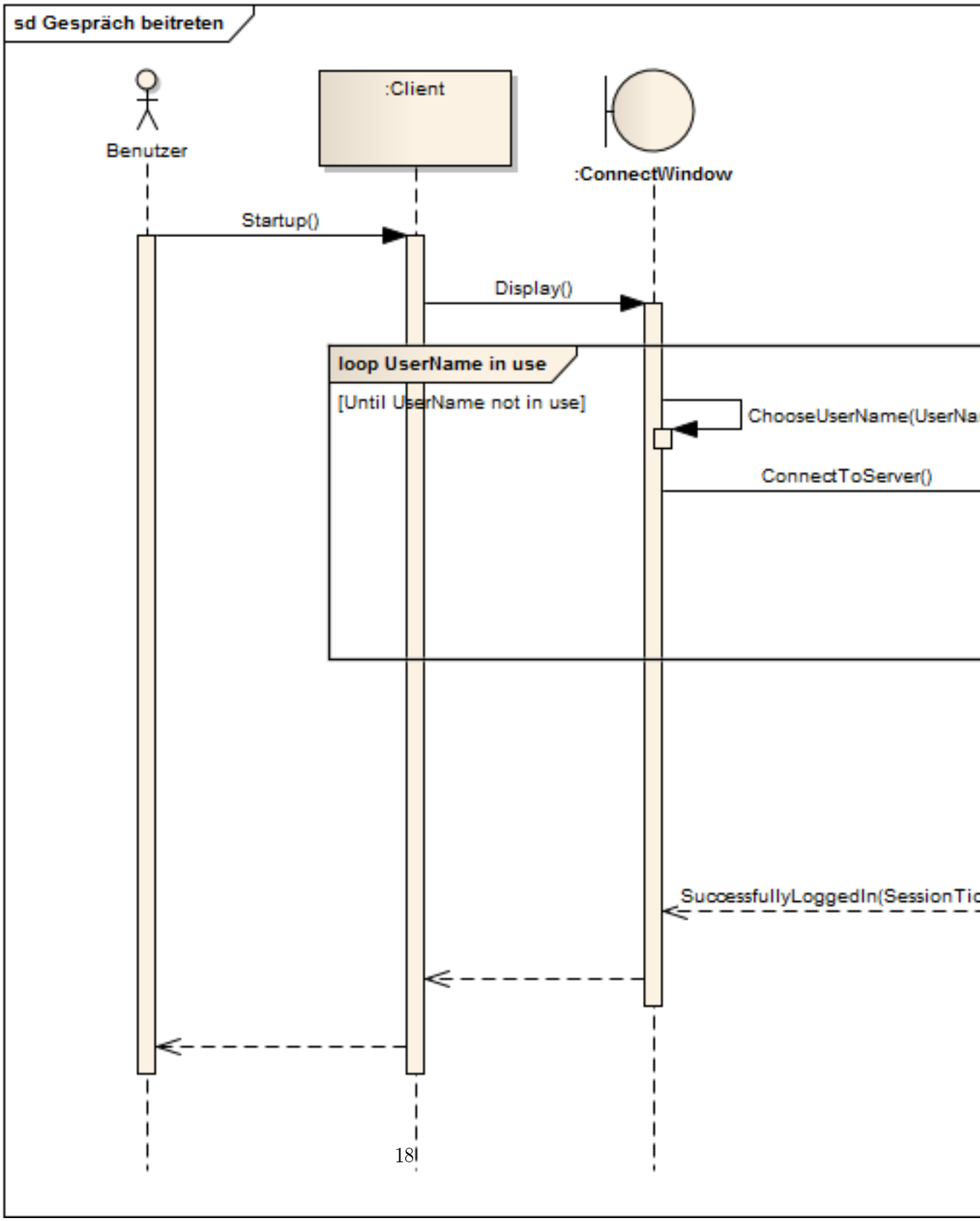


4.1.5 Server

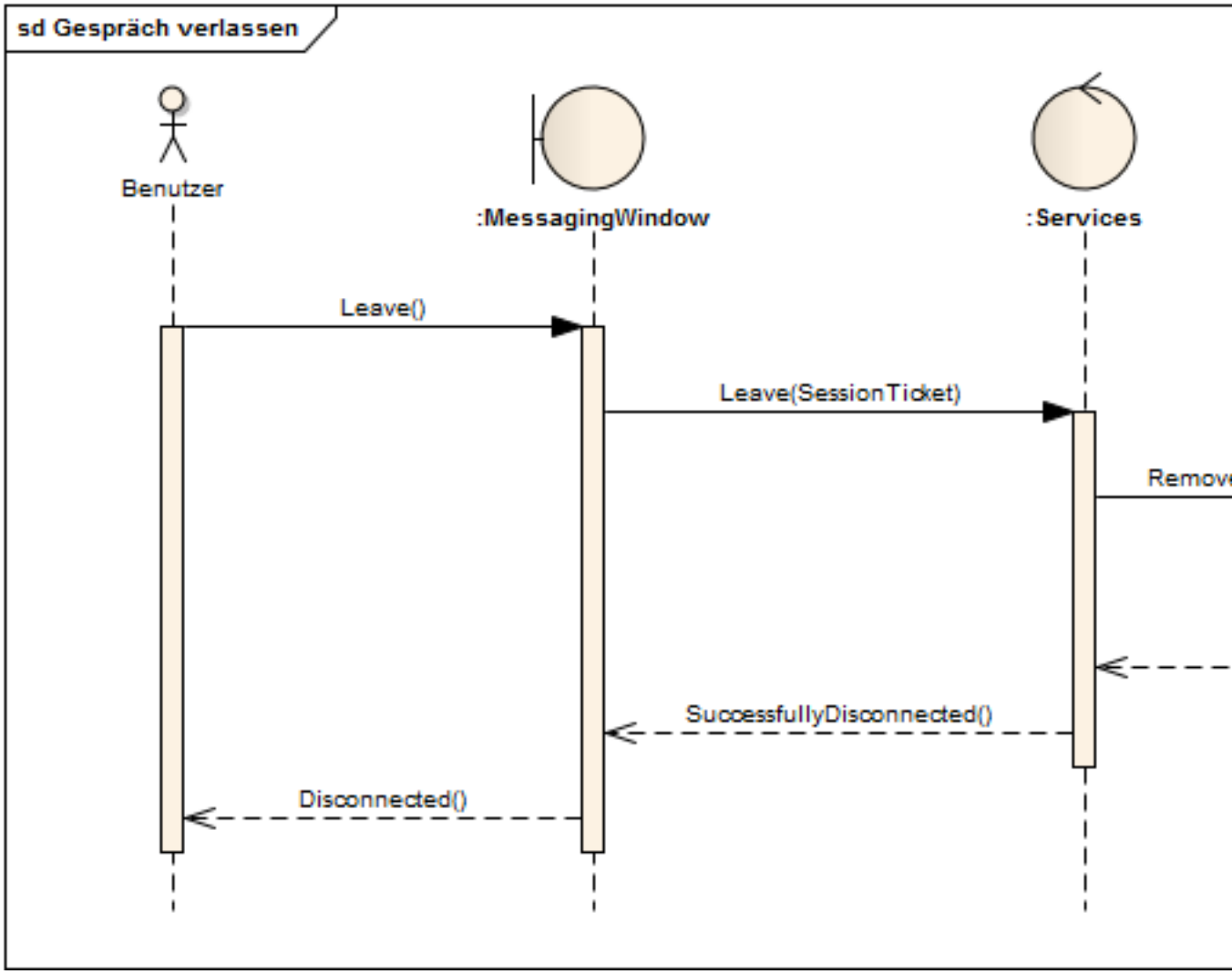

class Server
«interface»
Contracts::IMessagingService
+ NegotiateInitializationVector(byte[], CryptoAlgorithm) : void
+ NegotiateKey(byte[], CryptoAlgorithm) : void
+ Send(Message, Session) : void
«interfa
Contracts::IUs
+ GetJoinedUsers() : I
+ IsJoined(Session) : b
+ Join(User) : Session
+ Leave(Session) : voi
+ Subscribe() : void
+ Unsubscribe() : void
ServiceHost
Services
- registeredSessions: List<Session> {readOnly}
- subscribers: List<IServiceCallback> {readOnly}
+ GetJoinedUsers() : IEnumerable<User>
+ IsJoined(Session) : bool
+ Join(User) : Session
+ Leave(Session) : void
+ NegotiateInitializationVector(byte[], CryptoAlgorithm) : void
+ NegotiateKey(byte[], CryptoAlgorithm) : void
+ Send(Message, Session) : void
+ Services()
+ Subscribe() : void
+ Unsubscribe() : void

4.2 Laufzeitsicht

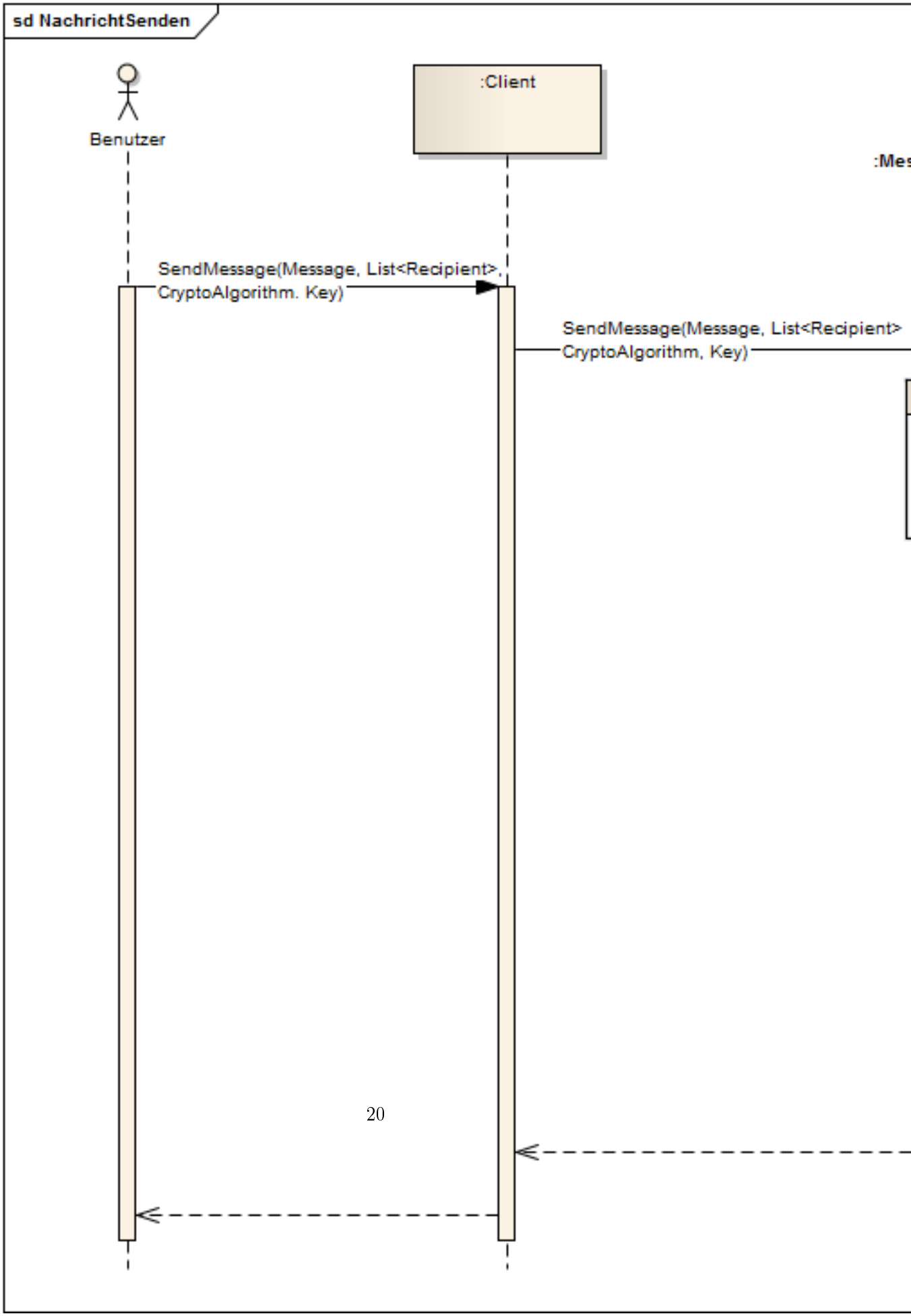
4.2.1 Gespräch beitreten



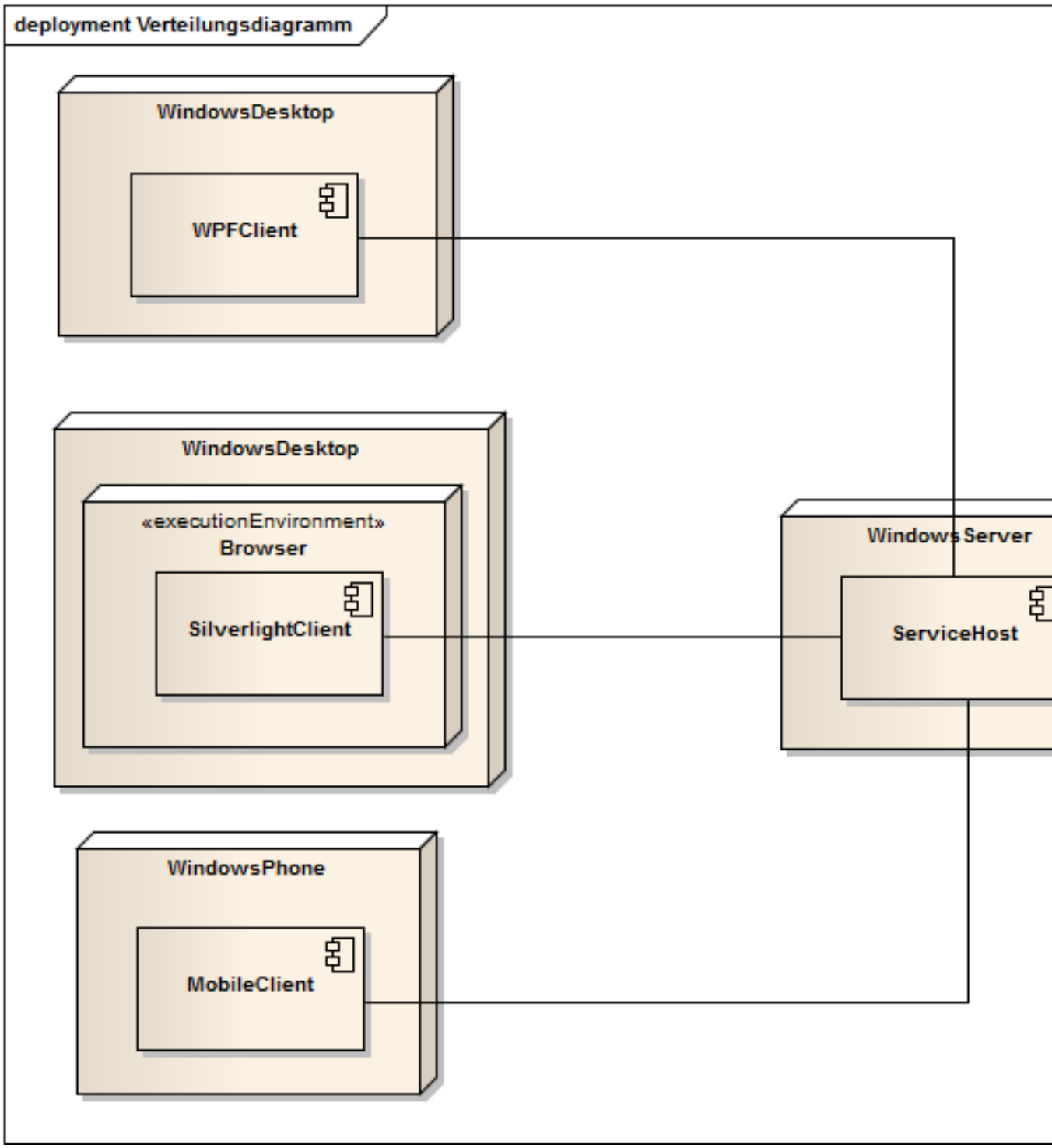
4.2.2 Gespräch verlassen



4.2.3 Nachricht senden



4.3 Verteilungssicht



5 Anhang

6 Akronyme

UC Use Case

YAEM Yet Another Encrypted Messenger

RUP Rational Unified Process

7 Glossar

Nomenclature

Geheimtext Der Geheimtext ist der Text, der durch die Verschlüsselung mittels eines kryptografischen Verfahrens unlesbar gemachte wurde.

Kryptoalgorithmus Ein Kryptoalgorithmus ist im Kontext von YAEM die konkrete Implementierung des Interfaces YAEM.Crypto.ICryptoProvider und bietet die Möglichkeit beliebige Nachrichten zu verschlüsseln beziehungsweise zu entschlüsseln.

RUP Der Rational Unified Process ist ein kommerzielles Vorgehensmodell zur Softwareentwicklung von IBM.

Use Case Ein Use Case (deutsch Anwendungsfall) bündelt alle möglichen Szenarien, die eintreten können, wenn ein Akteur versucht, mit Hilfe des betrachteten Systems ein bestimmtes fachliches Ziel zu erreichen. Er beschreibt, was inhaltlich beim Versuch der Zielerreichung passieren kann, und abstrahiert von konkreten technischen Lösungen. Das Ergebnis des Anwendungsfalls kann ein Erfolg oder Fehlschlag/Abbruch sein.

V-Modell Das V-Modell ist ein Vorgehensmodell in der Softwareentwicklung, bei dem der Softwareentwicklungsprozess in Phasen organisiert wird. Neben diesen Entwicklungsphasen definiert das V-Modell auch das Vorgehen zur Qualitätssicherung (Testen) phasenweise.

8 Bibliographie

Literatur

[1] Klaus Pohl and Chris Rupp. *Basiswissen Requirements Engineering*. dpunkt.verlag, 2011.

[2] Trung Hung VO. Software development process, 07 2007.