Praxissemesterbericht

zu

Fuzzing mit American Fuzzy Lop

bei der Firma

Airbus DS Optronics GmbH

Andreas Eisele

23. Februar 2017, Aalen

Inhaltsverzeichnis

1 Praxisstelle und Klausel

Airbus Defence & Space Optronics GmbH
Abteilung: Softwareentwicklung
Team: ASSESS
Adresse:

Hinweis zur Weitergabe bzw. Verwendung von ....

2 Eidesstattliche Erklärung

Hiermit erkläre ich, **Andreas Eisele** dass ich die vorliegenden Angaben in diesem Bericht zu meinen Tätigkeiten im Praxissemester bei Airbus DS Optronics GmbH wahrheitsgetreu und selbständig verfasst habe. Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war. (Soweit mir bekannt!)

Ort, Datum Unterschrift (Student)

Bestätigung der inhaltlichen Richtigkeit: Der vorliegende Bericht wurde durch den zuständigen Betreuer **Michael Adam** korrekturgelesen und auf inhaltliche Korrektheit geprüft.

Ort, Datum Unterschrift (Betreuer)

3 Kurzfassung

Im Rahmen dieser Arbeit wird folgendes vorgestellt .....

4 Motivation und Ziel der Arbeit

1 Motivation

2 Ziel der Arbeit

5 Vorgehen

6 Grundlagen

1 Fuzzing

Fuzzing auch als "Robustheits - Prüfung"bekannt, ist eine Methode um Software auf Stabilität und Schwachstellen zu testen.

Im Vergleich zu Unit-Tests, bietet Fuzzing ein weitaus größeres Spektrum an generierten Zufallsdaten, welche auf die Zielschnittstellen ausgeführt werden. Zudem werden die Daten nicht abhängig von dem verantwortlichen Entwickler bestimmt, sondern stellen tatslächliche Zufallsdaten oder Mutationen dar. Das grundlegende Prinzip von Fuzzing ist damit das beschießen der Zielsoftware mit generierten Zufallsdaten und so zu überprüfen, ob sich das Programm unerwartet verhält und dadurch eine Schwachstelle afgetreten ist.

Als unerwartetes Verhalten ist in der Regel das hängen oder abstürzen einer Software gemeint. Diese "bugs"haben als Ursache oft Speicherleks, welche häufig in den Sprachen C und C++ auftreten, da diese dort nicht bereits automatisiert abgefangen werden. Die Schwachstellen, welche durch so einen Bug auftreten können, sind oft auch Ziel von Kriminellen die insbesondere bei sensibler Software wie Kommunikationsprogrammen hier schaden anrichten können.

Das erste mal wurde der Begriff Fuzzing und diese Methode zu testen von Barton Miller und seinen Studenten an der University of Wisconsin-Madison 1989 entwickelt. Viele Jahre konnte sich Fuzzing für den breiten Gebrauch nicht etablieren. Aufgrund von sehr aufwendigen Anpassungen die notwendig waren um die damaligen Fuzzer-Tools auf die entsprechende Zielsoftware zuzuschneiden waren diese für die Testphase schlicht zu kostenintensiv.

In den letzten Jahren hat durch die Entwicklung effizienterer Tools das Thema Fuzzing eine deutlich höhere Verbreitung erfahren. Besonders einer dieser Fuzzing-Programme wird im Rahmen dieser Arbeit behandelt.

2 American Fuzzy Lop

AFL wurde 2014 von Michal Zalewski entwickelt. Im Vergleich zu anderen verfügbaren Fuzzing-Tools unterscheidet sich American Fuzzy Lop maßgeblich, durch dessen neuen Ansatz mit einem genetischen instrumenten-geführten Algorithmus. Die Resultate durch den Einsatz dieses Tools haben sich signifikant verbessert und so zu dessen Bekanntheit geführt. Dieses Fuzzer-Programm ist in der Lage sowohl Black-Box als auch White-Box Tests durchzuführen und ist als freie Open-Source-Software unter der Apache Version 2 Lizenz verfügbar.

Es zeichnet sich insbesondere durch die Instrumentierung während des Kompilierprozesses aus und erreicht dadurch eine sehr hohe Pfadabdeckung des Ziels. Indem Instruktionen in den Assembler-Code geschrieben werden, ist AFL in der Lage auf Veränderungen die im Ziel-Programm ausgelöst werden zu erfassen und darauf zu reagieren. Eine weitere Besonderheit sind die Mutationen auf die Sample-Dateien. Um überhaupt fuzzen zu können muss mindestens eine minimale valide Datei vorliegen, welche von dem zu testenden Porgramm verarbeitet wird. Diese Dateien werden in einer Queue gespeichert und fortlaufend modifiziert oder verworfen. Wenn eine dieser Veränderungen einen neuen Programzustand verursacht wird diese Datei in die Queue mit aufgenommen.

Dieser Prozess wird immer wieder wiederholt, bis alle möglichen Programpfade mit den mutierten Dateien in der Zielsoftware durchlaufen wurden.

2.1 Welche Software kann mit AFL getestet werden?

AFL wurde in erster Linie für Programme entworfen, die Dateien verarbeiten oder über Stdin Eingaben erhalten.

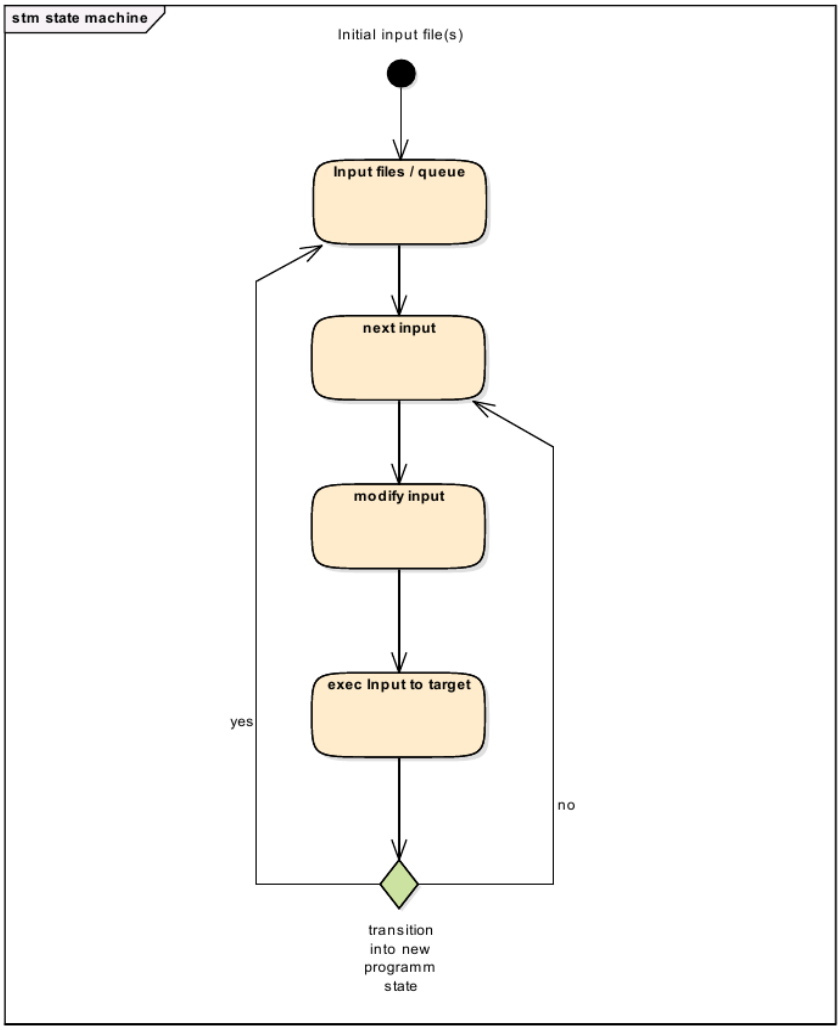

Abbildung 6.1: AFL State-Machine

2.2 Instrumentierung

sjfnl

2.3 Erfassen neuer Programzustände

2.4 Mutation

2.5 Fuzzing Session

2.6 Evaluation

3 mehr Grundlagen .....

7 Zielsoftware (System - Stack)

Der System Stack ist die softwareseitige Lösung für die Kommunikation zwischen verschiedenen Aufnahmegeräten.

Ein Master-Stack kann mit mehreren Slave-Stack‘s kommunizieren. Der Stack ist in mehrere Schichten aufgeteilt die über Schnittstellen miteinander verbunden sind. Im Einsatz ruft ein Device seinen Stack beziehungsweise die oberste Schicht auf um Daten zu übertragen und der Stack wiederum über seine unterste Schicht ein entsprechendes IO_Device um die Daten zu übertragen.

1 Aufteilung des Ziels in Schichten

Wie in dem Abschnitt ..... angesprochen, ist Software die von sich aus nicht schon Dateien oder Stdin-Eingaben verarbeitet daraufhin anzupassen.

Im Falle des System-Stack wurden die einzelnen Schichten für sich oder teils mit darüber- oder darunterligenden Schichten geprüft.

2 Mitschneiden von Programmdaten

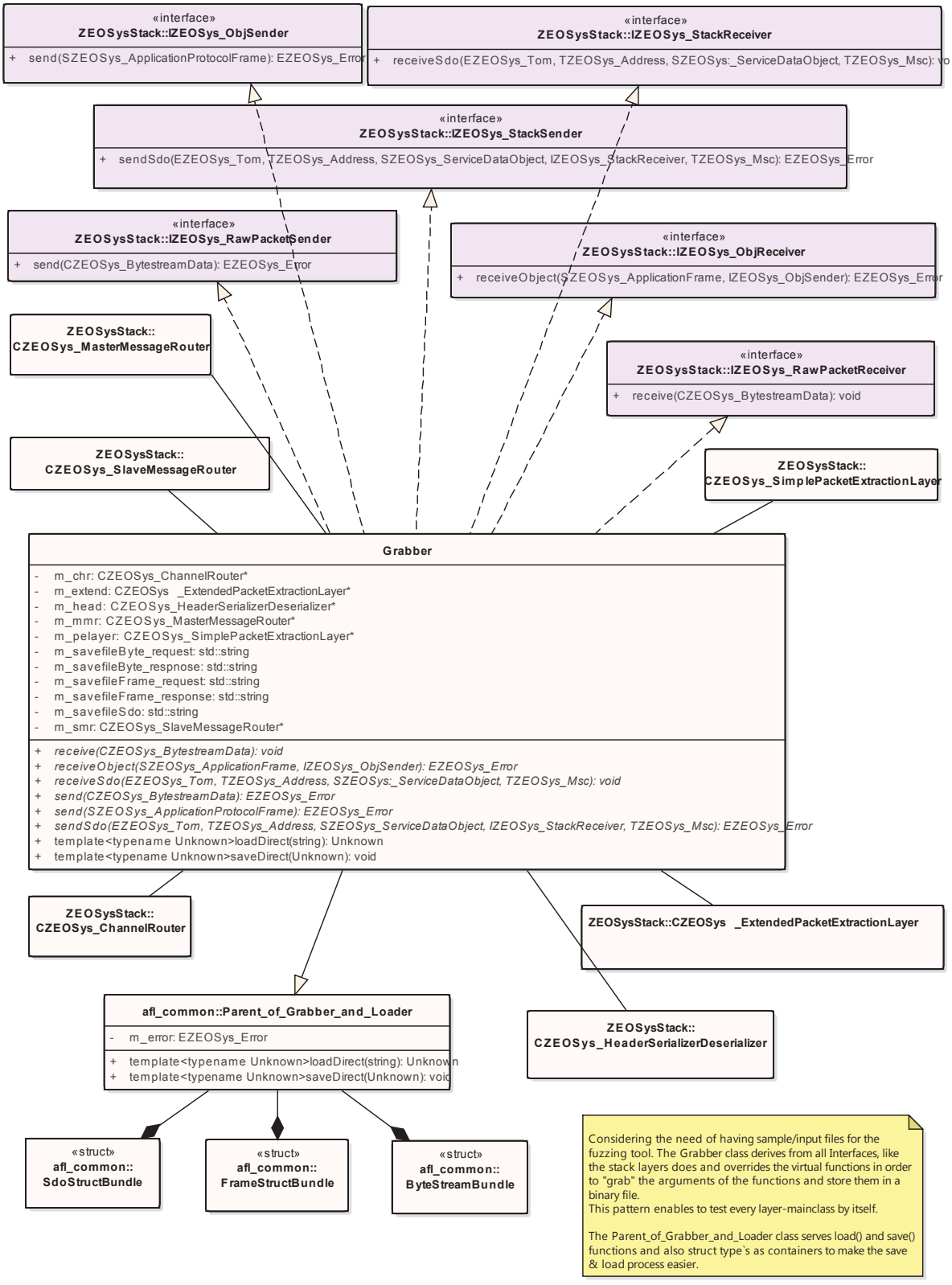

Abbildung 7.1: Grabber Klasse

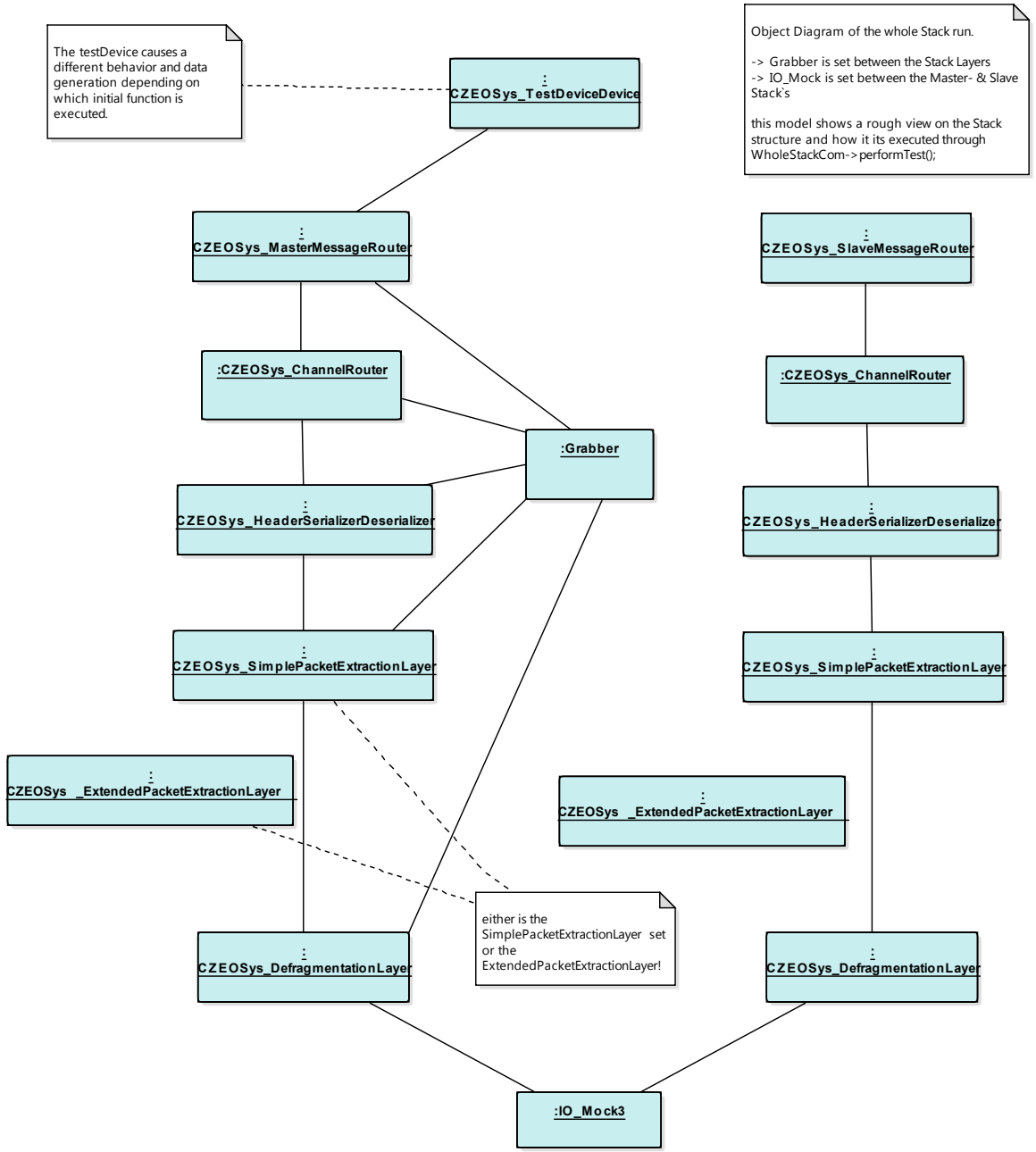

Abbildung 7.2: Einbettung Grabber

3 Testumgebungen anlegen

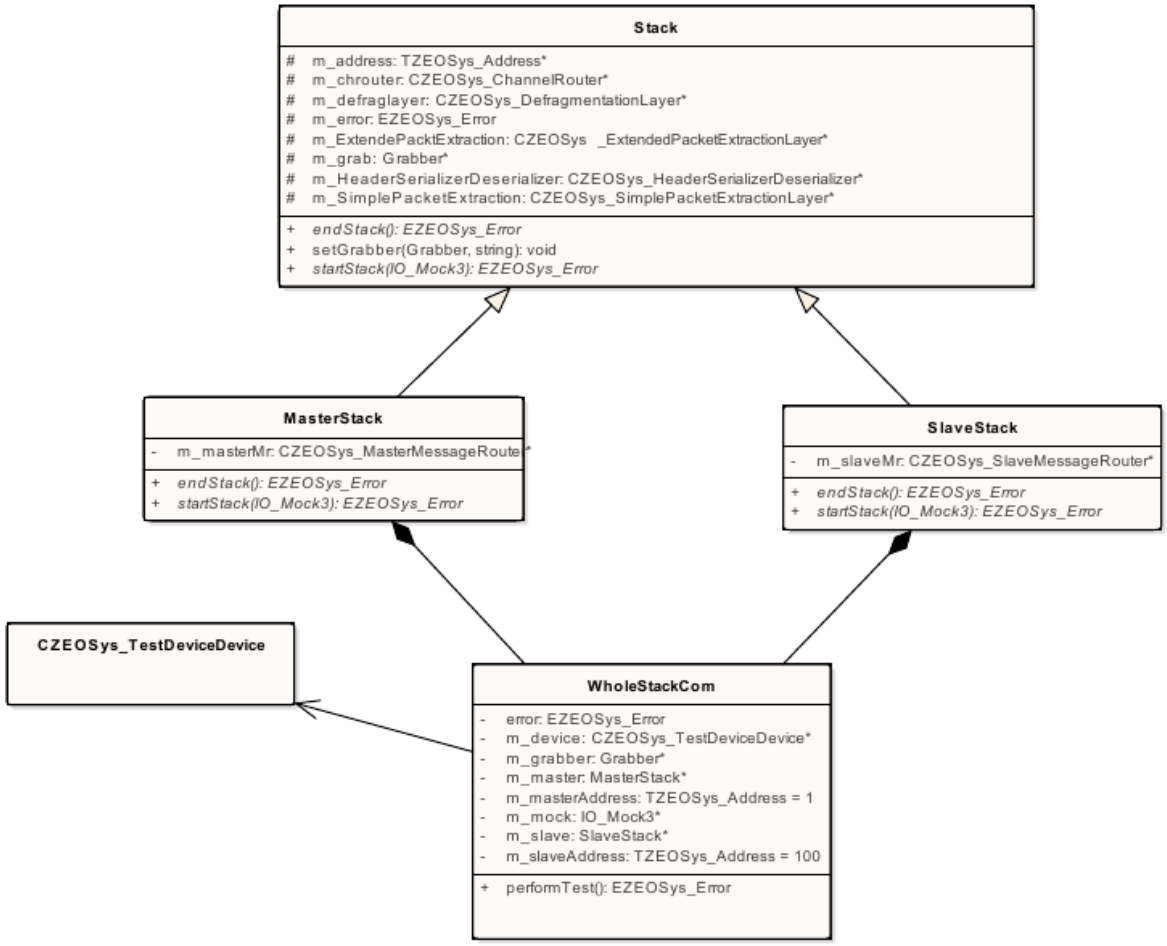

Abbildung 7.3: Stack Aufbau

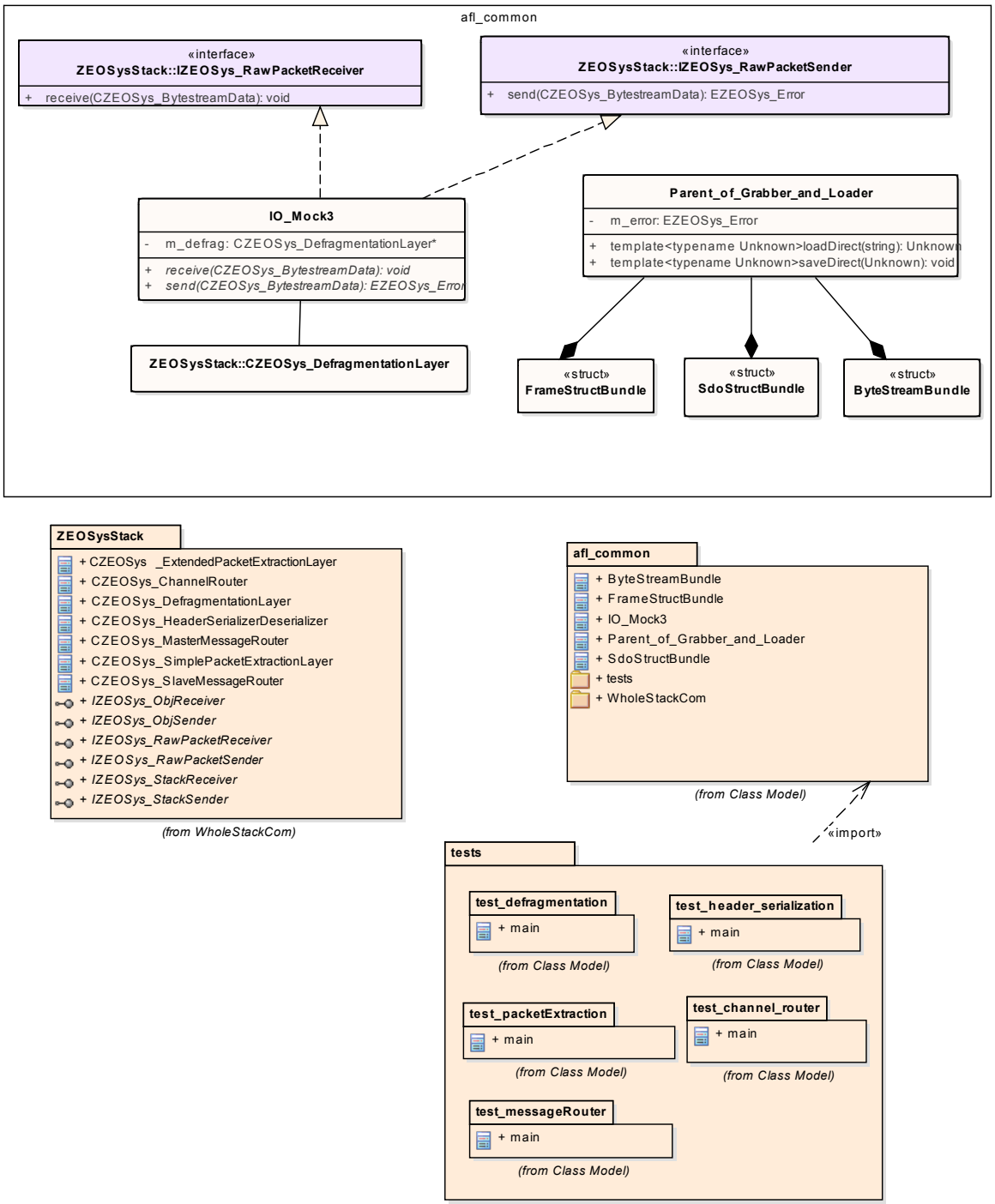

Abbildung 7.4: Paketmodell Testumgebung

3.1 Auswertung der Testdaten

4 Probleme und Lösungen

5 Fazit und Zusammenfassung

6 Quellen

7 Abbildungsverzeichnis

Abbildungsverzeichnis