Acronym	Senser
Project	**ADS-B Sentence Server**
Doctype	**Requirements**
Author	**Hans-Gerhard Gross, Kai Warendorf**
Contact	hggross@hs-esslingen.de; Kai.Warendorf@hs-esslingen.de
Client	Esslingen University
Contact	Faculty of Information Technology
Version	1.0
Date	August 23, 2014

Contents

Chapter 1

Project Drivers

1.1 Purpose of the Project

1.1.1 Vision Statement

This project aims at developing a server that provides ADSB-sentences locally in a Java application.

1.1.2 Project Outcomes

The Java application fetches ADSB-sentences from an external source.

The Java application creates a sentence object for each sentence obtained.

The Java application prints a string representation of each sentence onto the screen.

1.1.3 Learning Objectives

After having completed this project, as student, you can ...

- develop and integrate Java classes.
- develop and integrate Java interfaces.
- perform simple String operations in Java.
- handle Date objects in Java.
- output Strings on the screen in Java.

1.2 Stakeholders

1.2.1 Project Team

Various members and roles.

1.2.2 Product Users

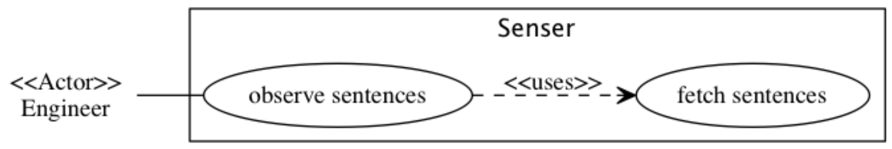
Local Flight Control Engineer, User. Priority: **Key User**.

Chapter 2

Functional Requirements

2.1 Data Model and Data Dictionary

2.1.1 Use Case Diagram



2.2 Senser Functional Requirements

Model [senser] ucd :: Senser

Senser.F.10 Observe ADSB-Sentences **essential**

Model [senser] uc :: Engineer → observe sentences

Feature In order to get an overview of the local flight traffic, as I flight control engineer, I want to be able to observe each incoming ADSB-sentence, with

- Time of sentence arrival
- Originator of the sentence
- Content of the sentence, separated into payload and parity.

Senser.F.20 Fetch Raw Sentences essential

Model [senser] uc :: observe sentences≪uses≫ → fetch sentences

Feature In order to provide ADSB-sentences locally, the system shall fetch the sentences from the following web service:

`http://flugmon-it.hs-esslingen.de/subscribe/ads.sentence`

Feature In order to integrate seamlessly with other OS operations, the web service address shall be provided as input parameter upon application start.

Chapter 3

Non-Functional Requirements

3.1 Look and Feel Requirements

Senser.NF.10 Text Output per ADSB-sentence essential

Feature The system shall display each ADSB-sentence received in the following form (example):

```
Time:          Weekday, DD.MM.YYYY, hrs:min:sec.usec
Dfca:          8D
Originator:    4692CA
Payload:       584720707A0996
Parity:        49890A
```

3.2 Implementation-Specific Requirements

3.2.1 Process

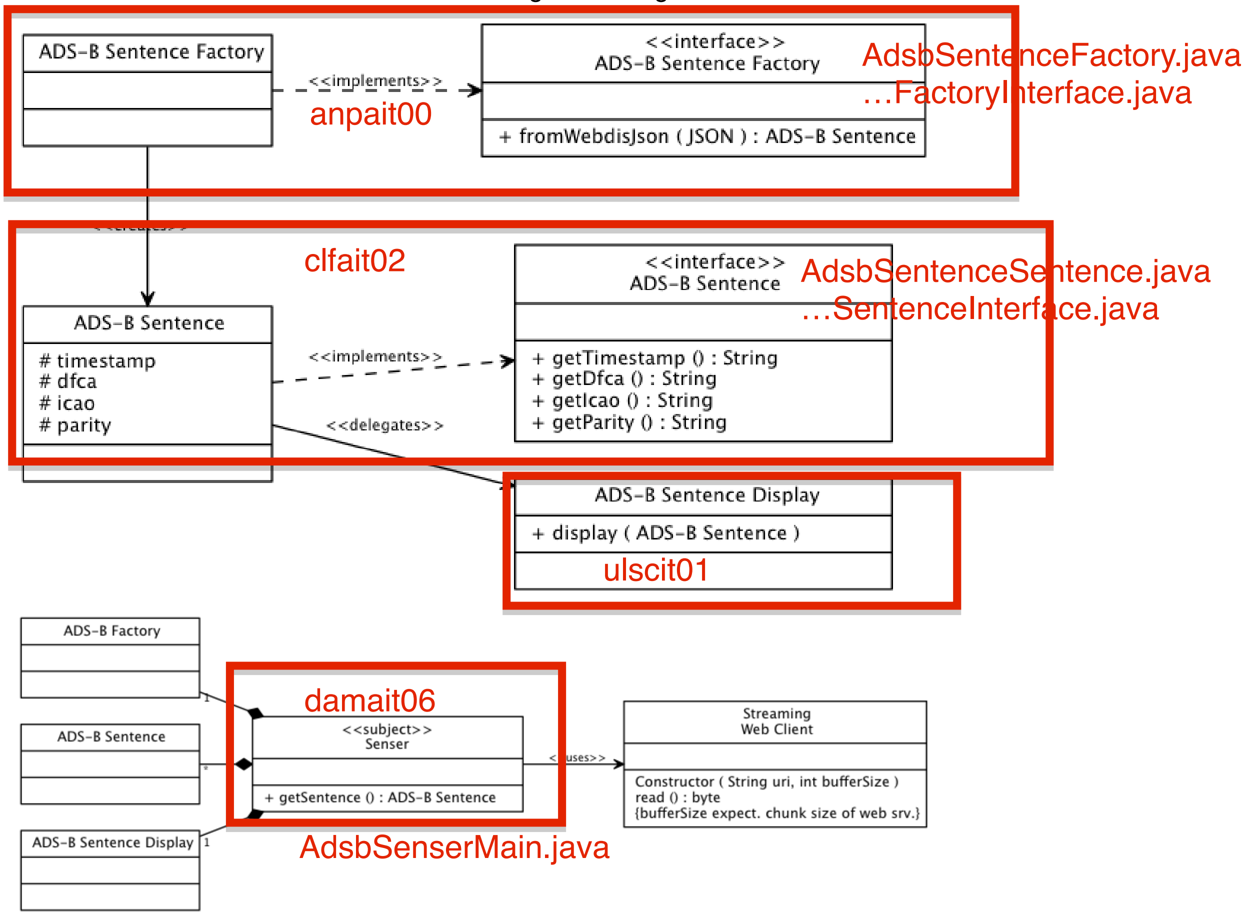
Senser.NF.50 Test Driven Development essential

In order to ascertain sufficient testing of the product, the implementation must be carried out following a test-driven development approach.

3.2.2 Architecture

Senser.NF.60 Use of Classes and Interfaces essential

Feature The organization of the system implementation shall reflect the classes and interfaces shown in the following class diagrams:



3.3 Maintainability Requirements

Senser.NF.70 Documentation essential

In order to ascertain high understandability, the source code must be self-explanatory.

Senser.NF.80 Cohesion and Coupling **essential**

In order to support high maintainability, the modules of the system must be realized with high-cohesion and low coupling.

Senser.NF.90 OO Design Principles **essential**

In order to support high maintainability, the other well-known principles of good object-oriented design must also be applied.

Chapter 4

Additional Domain-Specific Information

4.1 JSON Format

The ADS-B sentences provided by the web service have the following (example) format:

```
{"subscribe":["subscribe","ads.sentence",1]}
{"subscribe":["message","ads.sentence","1408776292.1584036!ADS-B*8D3C4895586DF0F922005F59BE84;\r\n"]}
{"subscribe":["message","ads.sentence","1408776292.2016194!ADS-B*8D3C4895586F00F946005F5D067F;\r\n"]}
{"subscribe":["message","ads.sentence","1408776292.6264563!ADS-B*8D3C489599C00438207808E23FA3;\r\n"]}
{"subscribe":["message","ads.sentence","1408776292.6363628!ADS-B*8D3C4895200854B8C3506056AC62;\r\n"]}
{"subscribe":["message","ads.sentence","1408776293.0063913!ADS-B*8F3C6635587BF426EBF51890E6AB;\r\n"]}
{"subscribe":["message","ads.sentence","1408776293.0464215!ADS-B*8D3C4895586F10F98E005F63F468;\r\n"]}
{"subscribe":["message","ads.sentence","1408776293.5064864!ADS-B*8D48417090353418A9F58C0F4EA2;\r\n"]}
{"subscribe":["message","ads.sentence","1408776293.5984044!ADS-B*8F3C6635587BF4273FF51A31EE08;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.0763857!ADS-B*8F3C663599901B3468400E6A48BF;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.0768626!ADS-B*8D3C489599C004382078091DCBAA;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.0965719!ADS-B*8D3C489599C004382078091DCBAA;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.5664067!ADS-B*8D3C4895586F20F9FA0060E91107;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.5763803!ADS-B*8D4841709035241883F5A40E253D;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.6084318!ADS-B*8F3C6635587BE0B30E024D04AB8B;\r\n"]}
{"subscribe":["message","ads.sentence","1408776294.9817092!ADS-B*8F3C663599101A3468400EF8850E;\r\n"]}
...
```

4.2 ADS-B Format

Each valid ADS-B Sentence is comprised of a timestamp inidcating the arrival of the sentence in the ADS-B base station and a raw sentence as HEX-string. The timestamp represents seconds since the ”Epoch” before the comma, and milliseconds after the comma (see below).

```
1380130780.6415110!ADS-B*8D440C9C9037B0689400D388832D;
Timestamp:     1380130780.6415110
Raw sentence:  8D440C9C9037B0689400D388832D
```

The raw sentence is interpreted in the following way:

```
DFCA:          8D
ICAO:          440C9C
Payload:       9037B0689400D3
Parity:        88832D
```