



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Verification of the AIXM 5 Procedures Model through cockpit simulation applications

Final Demonstration: Application Implementation and Setup Report

Status: Final Deliverable

Edition: 1.0

Date: 2007-06-30

Part Edition No.	Part Edition Issue Date	Part Author	Reason for Change
1.0	June 30, 2007	Christian Grothe	Initial Version

Table of Contents

1	INTRODUCTION	5
2	PROTOTYPE APPLICATION ARCHITECTURE	5
3	AIXM PROCESSOR.....	6
3.1	Generation of the Core Java Framework	6
3.2	Generation of AIXM Elements	7
3.2.1	Classes	7
3.2.2	Datatypes and Attributes	7
3.2.3	Associations.....	8
3.3	Model changes for the implementation	8
4	PROCEDURE XNOTAM PROCESSOR.....	9
4.1	User Interface.....	9
4.2	Display Controls.....	10
5	NAVIGATION DISPLAY AND PRIMARY FLIGHT DISPLAY	11
5.1	Approach Channel Visualisation.....	11
5.2	Digital NOTAM Extensions	11
5.3	PxP-Display Interaction	11
6	DEMONSTRATION SCENARIOS.....	12
6.1	Procedure Transition Unavailable.....	13
6.2	Minimum Altitude Raised	13
6.3	Redefined RNAV Transition with Different Fixes	14

Table of Figures

Figure 1: /O module class hierarchy for the DesignatedPoint feature	6
Figure 2: Implementation components in the Component View	7
Figure 3: Java classes generated from classes in the dataype package	8
Figure 4: Associations generated as member attributes	8
Figure 5: Some changed classes and associations from the AIXM CM.....	9
Figure 6: Procedure xNOTAM Processor user interface – initial state	10
Figure 7: Procedure xNOTAM Processor user interface – temporary change in database ...	10
Figure 8: ND and PFD with approach channel(s).....	11
Figure 9: TUD's cockpit simulator facility during the demonstration.....	12
Figure 10: ND with TRA and approach channel of unavailable procedure	13
Figure 11: ND with approach channels for original procedure and procedure with temporarily raised minimum altitude	14
Figure 12: ND with TRA and two approach channels, original procedure and temporarily redefined with differen fixes.....	14

1 Introduction

The “AIXM 5 Procedure Model Study” performed by TUD under contract with EUROCONTROL aims at the investigation, verification, implementation, and demonstration of the AIXM 5 Procedure Model using a cockpit simulation environment, in which procedure data will be stored and updated according to the Aeronautical Information Exchange Model (AIXM) version 5.

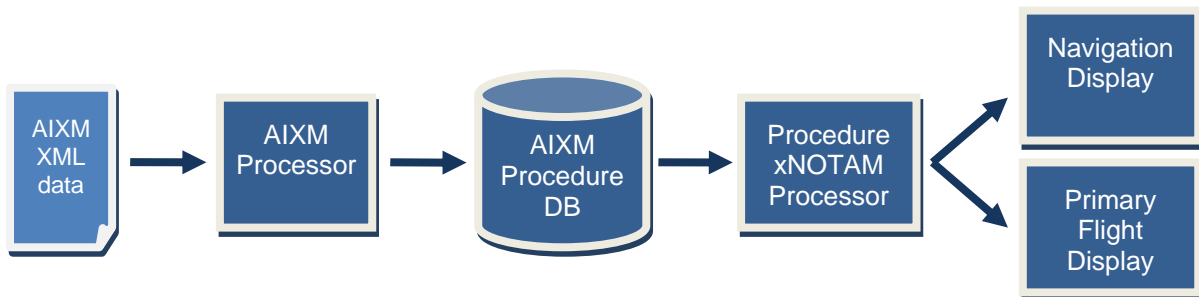
The applicability and usability of this data model as part of AIXM 5, Release Candidate 1, and the temporality model therein (with the “xNOTAM” concept) are investigated and a relational database is implemented based on the Conceptual Model of AIXM 5 RC1.

The study results are demonstrated by designing and implementing a prototype application in TUD’s cockpit simulator environment that uses the Procedure data from the implemented database. This application is being presented in a demonstration to the members of the AIXM 5 Key Concepts Focus Group in a real-world scenario in the cockpit simulator.

This document summarises the design and implementation process of this prototype application and the set-up and the chosen scenarios of the demonstration.

2 Prototype Application Architecture

The prototype application developed for the purpose of demonstrating the use of AIXM modelled Instrument Approach Procedure (IAP) data in a cockpit display consists of the following components:



- The **AIXM Procedure Database** is a relational database with a schema designed to store AIXM modelled IAP data. More information about the design and implementation of the DB can be found in the document “AIXM Procedure Database Implementation Report”, submitted to EUROCONTROL as a contractual deliverable in the frame of this study.
- The **AIXM Processor** reads AIXM 5 XML encoded messages and stores the information in the DB. The high-level architecture and the implementation of the Java class framework for this component are described in chapter 3.
- The **Procedure xNOTAM Processor** features a user interface (UI) for controlling the interaction with the displays. Its design and functionality is described in chapter 4.
- The **Navigation Display** and **Primary Flight Display** are former developments of TUD. They feature approach channel visualisation. In the frame of this prototype application implementation, their functionality was extended to allow additional digital NOTAM functions. Those are described in chapter 5.

3 AIXM Processor

The required functionality of the AIXM Processor was defined as:

- Reading of AIXM 5 XML encoded files (messages)
- Writing the data into the AIXM Procedure Database

The high-level design objective for the AIXM Processor was the abstraction from a specific data format for the AIXM Procedure data (XML / relational DB) in order to achieve a high level of extensibility and scalability of the software. Only the underlying AIXM Conceptual Model (CM) should be the basis for the definition of data structures.

This objective resulted in the following design decisions:

- The core of the software is an object-oriented class framework, which was generated from the AIXM 5 RC1 Conceptual Model. It serves exclusively as an internal, temporary container for the data. Hence, the classes in this framework do not contribute to the processing logic of the software, i.e. they expose no business logic methods. The input and output (I/O) modules transform data into this framework after reading and from this framework before writing.
- A generic I/O framework was designed based on abstract, format-independent classes defining basic I/O functionality and concrete classes implementing the methods for a specific data format. The concept of type polymorphism is used to decide at run-time, which data formats are used as input and output formats. Further, a variation of the “factory method pattern” was used to create the concrete objects at run-time. As an example, the class hierarchy of the I/O module for the DesignatedPoint is depicted in figure 1.

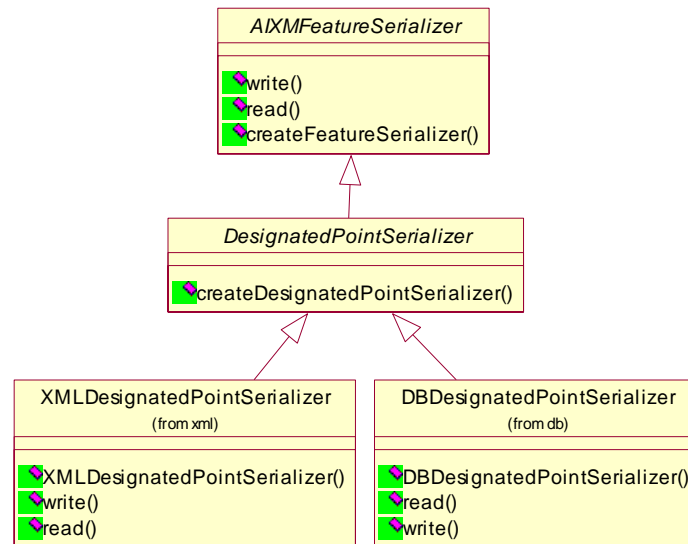


Figure 1: I/O module class hierarchy for the DesignatedPoint feature

3.1 Generation of the Core Java Framework

The internal framework for the AIXM Reader was generated from the Rational Rose UML Conceptual Model using the Rose Java Add-In 8.2.

This approach required to add to the Rose model for each Java class to be generated an “implementation component” in the “Component View” in the intended package structure. Each component realises a class from the Logical View in a programming language, i.e. Java in this case. See Figure 2.

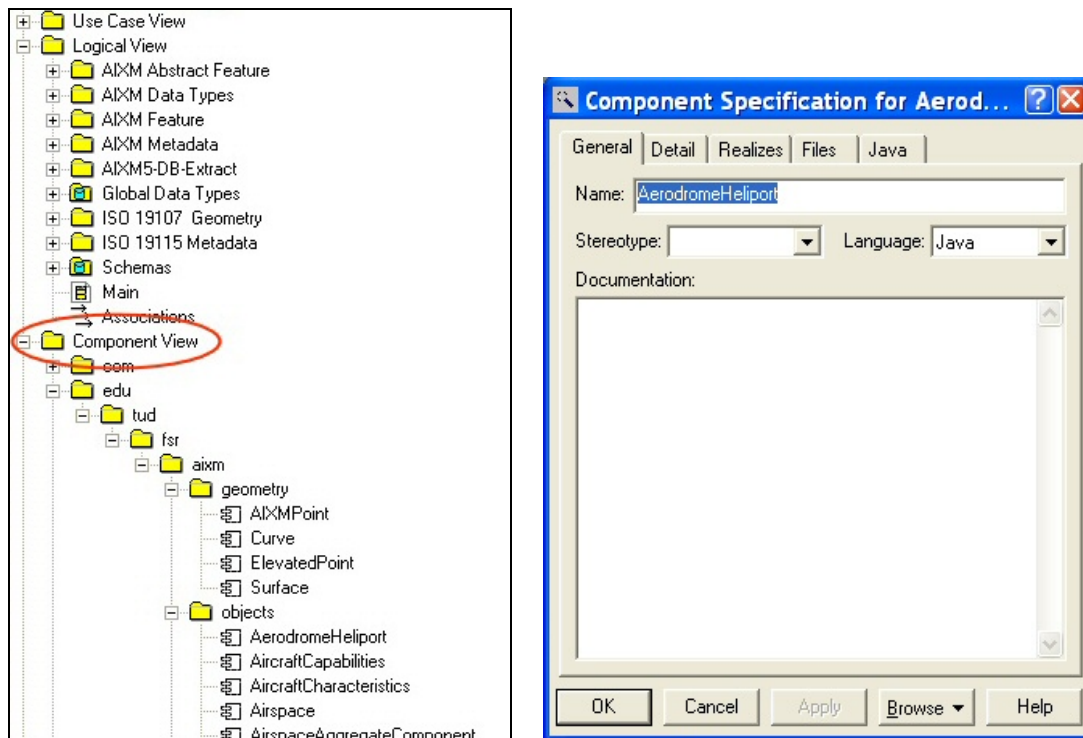


Figure 2: Implementation components in the Component View

The generation of the Java classes (with attributes and methods) from the implementation components can then be done automatically.

While the Java Add-In generates only the code frame, i.e. classes, their attributes, and method headers, it leaves manually created code (inside the methods) alone when a component is re-generated. Since the AIXM CM is a purely static (in the UML sense) data model though, no functionality (i.e. no methods) is foreseen in the individual classes.

3.2 Generation of AIXM Elements

3.2.1 Classes

The class framework including documentation of classes and attributes was generated as intended without encountering bigger problems.

3.2.2 Datatypes and Attributes

The datatype classes though, could not be generated. The specific form, in which the types are modelled in AIXM, could not be understood by the Java Add-In. It generally disregards all class stereotypes and thus does not interpret the <<enumeration>>, <<codelist>>, and <<datatype>> classes correctly. They are generated as every other class. For instance, the member attributes of a <<codetype>> class, which are thought to be enumerated values to be exclusively assigned to an attribute, are generated as member attributes of the Java class. The special “pattern” attribute of most datatype classes could not be interpreted as intended either and was generated as a simple member attribute. Furthermore, the assigned pattern is not a valid Java literal and its generation resulted in a Java class with errors. See Figure 3 for examples.

In the case where attributes of classes from datatype package contain special characters, which are not allowed in Java identifiers, the generation of the class is aborted with an error. This is the case for many of the <<codelist>> and <<enumeration>> classes.

The attributes of <<feature>> and <<object>> classes were generated correctly, but since the datatype classes were not, the types assigned to the attributes did not exist. So the resulting Java classes could not be compiled without modification.

```

/**
 * Code indicating the prefix used in support of current approach procedure
 * conventions
 */
public class codeApproachPrefix
{
    /**
     * Approach procedures with segments such as a high altitude teardrop p
     * that are within the high airspace stratum, requested by Military. N
     * and E Minimums only.
     */
    public String HI;

    /**
     * Indicates helicopter approach
     */
    public String COPTER;

    /**
     * Parallel or joining operation with parallel or joining ILS courses
     */
    public String CONVERGING;
}

public class alpha
{
    public string pattern = "[A-Z]*";
}

```

Figure 3: Java classes generated from classes in the dataype package

3.2.3 Associations

By default, a directed association (with navigability on one side only) from a class A to a class B (navigable side) is generated as a member attribute of type B in A. If the association has a role name on the navigable side, this name is used as identifier for the member attribute. If no role name is modelled, it is created automatically in the generation process in the form “the[classname]”, e.g. “theDesignatedPoint”. For a multiplicity upper bound > 1 on the navigable side, the member attribute is generated as an array. The specific type of the association (aggregation, composition, or none) is not interpreted by the Java Add-In. Figure 4 shows an example.

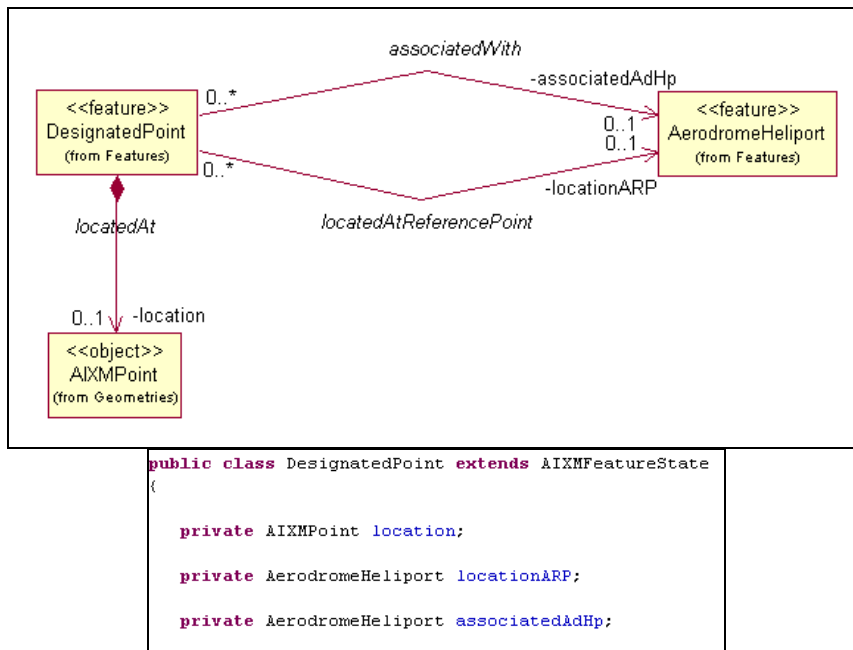


Figure 4: Associations generated as member attributes

3.3 Model changes for the implementation

For the purpose of implementing the AIXM Processor, some changes to the model were necessary in order to be able to use the classes generated by the Rose Java Add-In:

Methods and Visibility: Conforming to Java “best practice”, getter- and setter-methods for the attributes were added to each class. Further, the “visibility” property of all attributes and associations was changed to private.

Attributes: The types of the classes' member attributes was changed to Java base types (String, int, float, ...) to allow the generation of error-free Java classes from the model's <<feature>> and <<object>> classes without generating the classes from the datatype package. In this step, member attributes of a codelist or enum type were changed to "String" by default (except those of type "codeYesNo"; they were changed to "boolean"). For each "val..." type that includes a unit of measure subtype, an additional "uom..." attribute was added to the class.

Associations: Role names were given to the navigable side of every association in order to have meaningful names for the member attributes in the generated classes.

Geometry: Since there is a de-facto standard library available for dealing with geometry in Java, the Java Topology Suite or JTS, the base AIXM geometry classes (which are in fact ISO 19107 classes) were exchanged with JTS classes.

Figure 5 shows examples of all these changes in the DesignatedPoint-AIXMPoint-Point classes and associations.

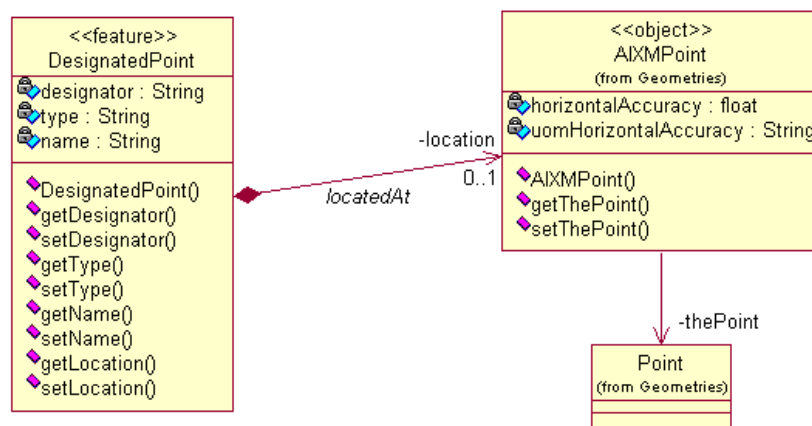


Figure 5: Some changed classes and associations from the AIXM CM

4 Procedure xNOTAM Processor

The Procedure xNOTAM Processor (PxP) component was implemented and integrated in the cockpit simulator facilities at TUD as a prototype application that allows the user (the pilot) to interact with the AIXM Procedure Database and the depiction of approach channels in the Navigation Display and Primary Flight Display.

4.1 User Interface

The PxP features a user interface (UI), which is rendered on a touch pad. By pressing the appropriate input items of the UI on the screen, the pilot can select an approach procedure from the Procedure Database by choosing, first the airport, second the runway, and finally the transition that he wants to fly.

The PxP then retrieves the data of the selected approach transition from the database, and enables an "Accept" Button (Figure 6).

In the case where a digital NOTAM regarding the selected procedure is available in the database as a temporary change, this information is given to the pilot in the "Remarks" field and, when the pilot selects the procedure, the "Temporary Change" box in the bottom part is enabled featuring information items and interaction controls for the changed procedure (Figure 7).

If this temporary change is not currently effective, the time difference until or since the effectivity period is displayed in textual form (as shown in Figure 7 left, above the white box with the digital NOTAM explanatory text).

If the temporary change, on the other hand, is currently effective, the “Temporary Change” box is framed red (Figure 7 right).

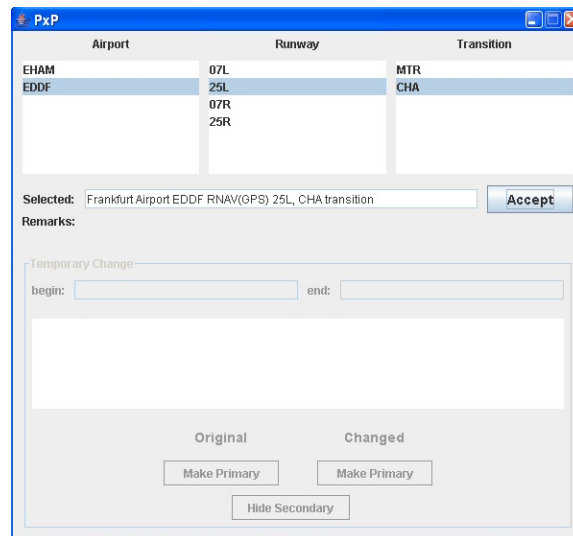


Figure 6: Procedure xNOTAM Processor user interface – initial state

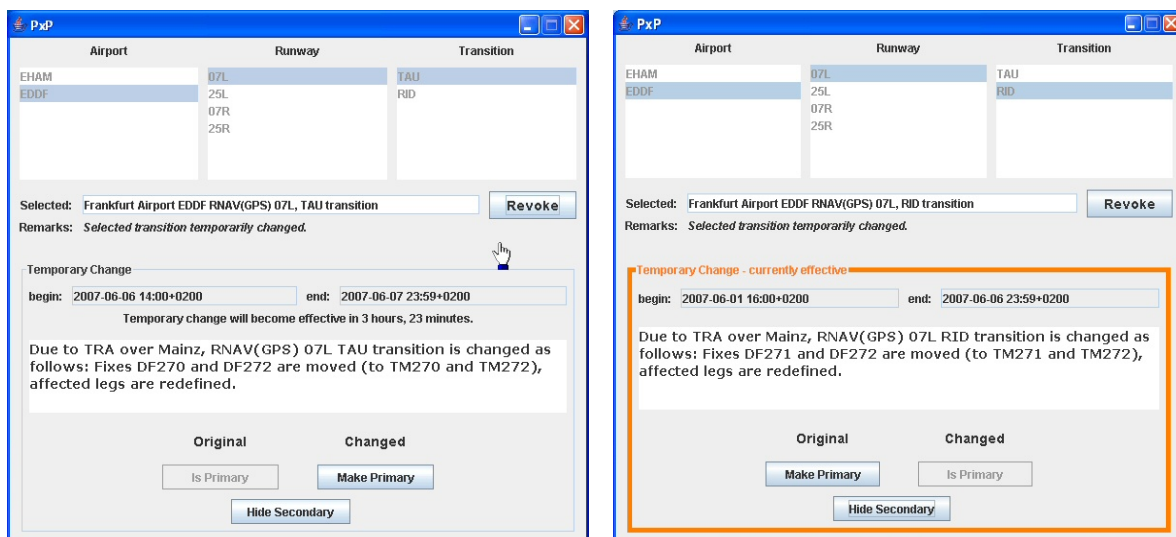


Figure 7: Procedure xNOTAM Processor user interface – temporary change in database

The temporary change data retrieved from the database is applied to the “original” procedure data and yields a different, temporarily changed procedure. These two versions of the procedure (“Original” and “Changed”) can be further worked with in the UI.

4.2 Display Controls

The buttons in the “Temporary Change” box allow the pilot to control the way the two versions of the procedure are rendered in the displays (the Navigation Display and Primary Flight Display, see chapter 5). Using the “Make Primary” button, the pilot can select either version as the “primary” procedure, while implicitly making the other one the secondary procedure. Hitting the button “Hide Secondary” sends a command to the displays to only render the primary procedure. When pressed, its caption changes to “Show Secondary” and pressing it again advises the displays to render both procedures again. How these control commands affect the displays’ rendering is described in the next chapter.

5 Navigation Display and Primary Flight Display

The aeronautical displays that were used as part of this prototype digital NOTAM application are former developments of TUD. The functionality of the display software was extended with features for digital NOTAM functions. The functionality of the displays is not described to the full extent here. Only the functions important for the understanding of the digital NOTAM extensions are described.

5.1 Approach Channel Visualisation

The Primary Flight Display (PFD) and the Navigation Display (ND) both can display an approach channel. The PxP controls whether the channel is displayed and, in the case of the ND, whether both primary and secondary approach is rendered. The PFD renders the primary approach only. The ND renders a 2D plan view and the PFD renders a 3D view from the aircraft's perspective of the approach channel.

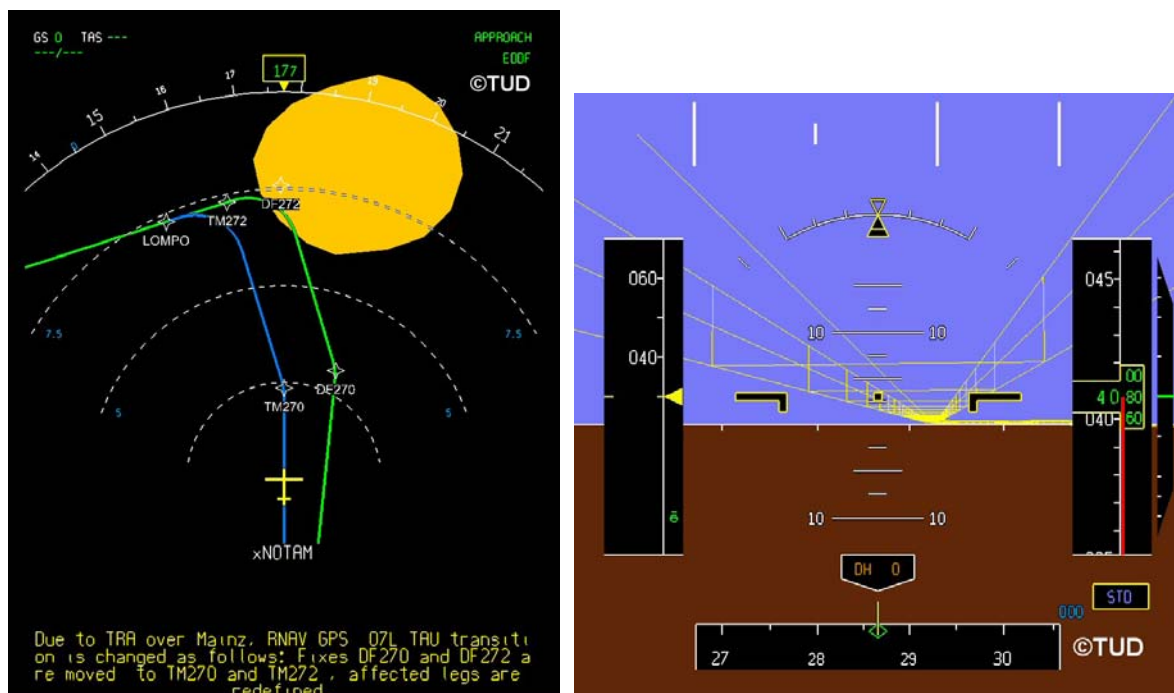


Figure 8: ND and PFD with approach channel(s)

5.2 Digital NOTAM Extensions

For the demonstration of the possibilities enabled by digital NOTAM, the ND's functionality was extended with these functions:

- The approach channel for both primary and secondary approach can be rendered. The primary approach is always depicted in green, the secondary approach in blue.
- An explanatory text for a temporary change can be depicted at the bottom.
- An airspace can be rendered as an orange polygon.

5.3 PxP-Display Interaction

As described in section 4.1 and 4.2, the PxP can send commands to the displays controlling the rendering of the approach channels. When an approach transition is selected in the PxP and accepted, the approach channel data is sent to the displays and the displays render the channels. If a temporary change exists for the selected approach, both versions of the approach are sent to the displays and the ND displays the "original" approach as primary procedure and the "changed" as secondary. This can be switched with the "Make Primary"

button in the PxP and whether or not the secondary procedure is shown at all can be selected with the “Hide Secondary”/“Show Secondary” button.

The explanatory text with the temporary change is rendered in the displays only when the “original” approach is selected as “primary” procedure in the PxP while there is a temporary change, which is currently effective.

If the temporary change to the procedure is due to a temporary restricted airspace, the PxP sends the airspace data (its spatial extent) to the displays and the airspace is visualized as an orange polygon in the ND.

If a procedure transition is selected in the PxP, which is temporarily unavailable (“completely withdrawn”), then the approach channel for this procedure is rendered with red colour in the ND. In this case, no approach channel is rendered in the PFD.

Examples for the display symbology elements are shown with the individual demonstration scenarios described in the next chapters.

6 Demonstration Scenarios

As mentioned in the introduction of this document, the prototype application was implemented for the purpose of demonstrating in a real-world use-case the potential benefits if fully data-modelled procedure data with time-dependent effectivity were available to cockpit applications.

This demonstration was given to the members of the AIXM 5 Key Concepts Focus Group in June 2007 at the cockpit simulator facilities of TUD. Figure 9 shows the set-up of the cockpit simulator during the demonstration. The PxP runs on a side display on the left side of the pilot. To allow a good visibility of the displays for the participants of the demonstration, the ND and PFD for approach channel visualisation run on the co-pilot’s side.



Figure 9: TUD's cockpit simulator facility during the demonstration

For the purpose of the demonstration, three possible real-world scenarios with temporarily changed procedures were chosen and the necessary functionality for these scenarios was implemented in the prototype application as described in the previous chapters. These scenarios are described in the following sections.

The visualisation of the scenarios is given for the ND only. Whether or not the approach channel is rendered in the PFD as well is explained in the text.

6.1 Procedure Transition Unavailable

This scenario is a typical situation for the publication of a conventional NOTAM. A specific transition of a published approach procedure is not available due to a temporary restricted airspace (TRA).

When the affected procedure is selected, the restricted airspace is rendered and the explanatory text with the digital NOTAM is shown in the ND. The approach procedure is depicted as unavailable (in red colour). No approach channel is rendered in the PFD.

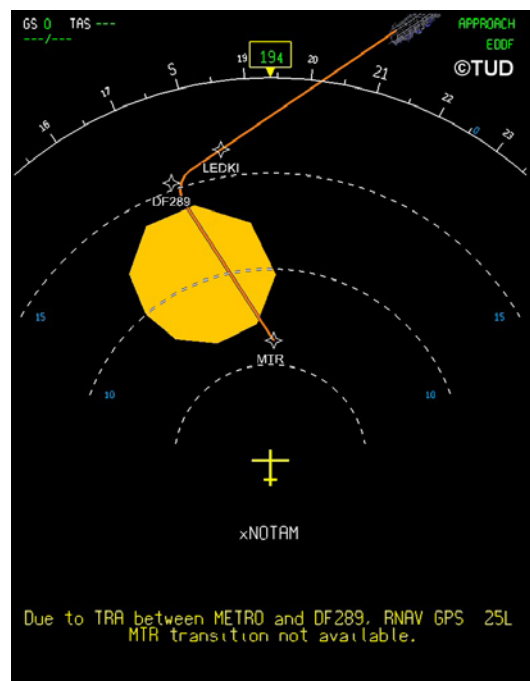


Figure 10: ND with TRA and approach channel of unavailable procedure

6.2 Minimum Altitude Raised

In this scenario, the minimum altitude is raised in a part of an approach due to temporary obstacles in the area. This is also a typical case for the publication of a NOTAM today.

When the pilot selects the affected procedure, both approach channels (for the “original” procedure and for the procedure with temporarily raised minimum altitude) is rendered in the displays. Since both channels are geographically in the same location, they are rendered above each other. Where the approach channel of the changed approach (selected as secondary procedure and thus rendered in blue) is higher than the original approach, the approach channel is rendered in blue colour (Figure 11).

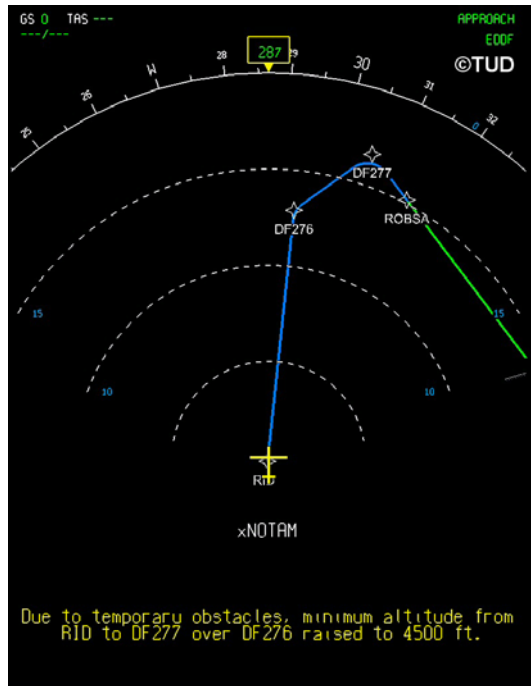


Figure 11: ND with approach channels for original procedure and procedure with temporarily raised minimum altitude

6.3 Redefined RNAV Transition with Different Fixes

In this scenario, again an approach is affected by a temporary restricted airspace. But in this case, a redefinition of the affected procedure is encoded in the digital NOTAM. The redefined procedure uses different fixes to avoid the restricted airspace.

Initially, both versions of the approach, original and changed, are rendered in the ND. The pilot can assign at the PxP's UI the primary procedure status to either approach and hide the secondary procedure.

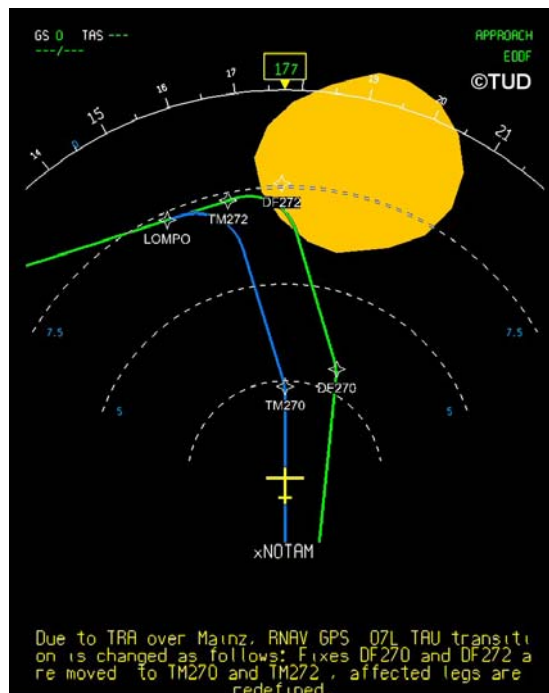


Figure 12: ND with TRA and two approach channels, original procedure and temporarily redefined with different fixes