Project ref. no.	IST-2000-25426
Project acronym	**VICO**
Project full title	**V**irtual **I**ntelligent **Co**-Driver

Security (distribution level)	Project internal
Contractual date of delivery	June 2001 (M 03)
Actual date of delivery	June 29th 2001
Deliverable number	D4
Deliverable name	System Specification
Type	Report
Status & version	Final 1.0
Number of pages	28
WP contributing to the deliverable	WP 3
WP / Task responsible	Robert Bosch GmbH (Bosch)
Other contributors	DaimlerChrysler AG (DCAG) Istituto Trentino di Cultura (ITC-irst) University of Southern Denmark (SDU) Phonetic Topographics (PT)
Author(s)	Frank Steffens, Bosch Petra Geutner, Bosch
EC Project Officer	Domenico Perrotta
Keywords	functional and requirements specification, user requirements, system architecture, module specifications and interfaces, scenarios
Abstract (for dissemination)	This report describes the functionality and the resulting requirements for the VICO system. It specifies the overall system architecture and the functional requirements. High-level specifications for all system modules are provided as well as a scenario for a typical VICO application.

Table of Contents

1 Introduction

1.1 Motivation

Currently commercially available driver information systems that can be operated by voice are restricted to the control of audio devices like radio, cassette and CD-player, and a telephone. Some of the systems available on the market also allow climate control, notebook functions or some rigid control commands for the navigation application. However none of the systems integrated in cars do allow destination entry by voice (see also Deliverable D3 "Report on Market Situation, Technological Trends and User Expectations").
Usage of speech as input mode is considered to have severe advantages over the traditional tactile interfaces. User tests have shown that the driver is less distracted from his primary driving task, thus resulting in a higher degree of safety. Speech input interfaces also offer a much higher degree of comfort and convenience than input by keys. Especially concerning the navigation task, input of a destination by tactile interface cannot be performed while driving, rather the destination for route guidance usually has to be entered before the car is in motion. Being able to enter the destination by speech would constitute an immense improvement to the usability and user-friendliness of future generations of navigation or driver information systems.

It is announced that the future generation of in-car navigation systems will offer destination entry by speech at least through spelling a city or street name during the course of 2002. Although this is a large improvement compared to currently available systems, these early systems will not allow the entry of a city or street by full words, but are capable of a letter-by-letter entry only. Also the control of the navigation application, if done by speech, will be restricted to the usage of a predefined and fixed set of command words and key phrases that have to be remembered by the user (following the examples of speech operations that are available on the market at the moment). Whereas speech input even if done by "command & control" only, offers several advantages over the traditional interfaces, the ultimate goal must be to allow the driver to use normal everyday language to enter his/her requests into a navigation or driver information system.

To this account extent and quality of currently available speech recognition engines will have to be improved: both by covering large vocabularies and being able to recognize spontaneous speech input with all implicit difficulties, and by developing recognition engines that are robust against the environmental noises typical for the automotive environment.
Goal of the VICO project is the development of a demonstrator system that is capable to meet the demands mentioned above. To this end, VICO, a virtual intelligent co-driver, providing a conversational speech interface to allow natural interaction between humans and digital devices and services will be developed.

1.2 User Requirements

Within the project first Wizard-of-Oz (WoZ) experiments[1] have been conducted at Robert Bosch GmbH. The experiments were conducted to find out in which way users would communicate with a system like VICO when being allowed to use the kind of speech input they prefer, including normal everyday language. Beside the possibility to enter queries freely

[1] A more detailed description of these experiments will be found in Deliverable D7 "Evaluation Report from Simulated Environment Experiments".

and without any restrictions concerning vocabulary or sentence structure, also the expectations of the users were inquired.

As a result of these experiments it is clear that communication with a driver information system by speech is seen as something comfortable and simple, not disturbing from the driving task. Speech is clearly preferred over the usage of traditional tactile interfaces. Also, normal everyday language is preferred over command language. The dialogues should be kept easy to understand, not overloading the users of the system by presenting too much information at once. In summary the following user requirements have to be considered when designing a system within the application scenario of VICO:

- Speech input has to be foreseen
- Input of spontaneous phrases is preferred over command language
- Usage of a keyword instead of a push-to-talk (PTT) button to start speech input is desirable
- Design of easy-to-understand dialogues, not being overloaded with too much information at once

When being asked about the functionality that should be controllable by speech input, the following devices and services were mentioned:

- Traffic Information
- Navigation
- Telephone
- Hotel Reservation/Tourist Information
- Radio
- Cassette and CD-player
- News Reading

2 Functional and Requirements Specification

Considering the user requirements presented above, the desired functionality of a virtual intelligent co-driver has been determined. As navigation and tourist information including a booking assistant are reckoned to constitute the most difficult functionalities to be realized within a speech dialogue it was decided to concentrate on these topics.
In the following a general description of the functionality of the planned demonstrator will be given.

2.1 Functionality

VICO, the virtual intelligent co-driver, will act as a conversational agent between the user of a driver information system and the applications, including devices and services, that are offered by such a system. For the realization of the emerging demonstrator it is planned to offer navigation and tourist information, and a booking assistant for the reservation of hotels and restaurants. This includes a talking electronic car manual and the availability of car status information by voice. In addition, news reader services will be integrated. In detail the final demonstrator will cover the following functionalities:

- navigation:
  - state-of-the-art route planning and negotiation
  - disambiguation of destinations
- tourism:
  - travel: travel information (hotels, restaurants, parking information etc.)
  - tourist services
    - travel guide (e.g. for special points of interest like churches, wineries etc.)
    - customized sightseeing tours (according to the preferences of the respective user)
    - booking assistant (hotel, restaurant, airline, etc.)
- navigation in countries with different languages
  - non-native context (some knowledge of the foreign language)
  - foreign context (little or no knowledge of the foreign language and its pronunciation rules)
- car manual
- car status information
- news reader services

Considering the various tasks VICO will be able to complete, different kinds of dialogues can be distinguished:

1. **The user knows the precise destination** (e.g. an address or a point of interest):
   In this case the dialogue focuses on the different ways of specifying an address (in particular in the case of ambiguous names) such as:
   - city, street, crossing street
   - city, name or point of interest
   - city, postal code, area code, etc.
   - city, additional geographic information ("*Neustadt an der Weinstraße*", "*Stratford upon Avon*", etc.)
2. **The user knows what he/she wants** (e.g. a hotel of a certain category or a restaurant of a certain nationality):
   In this case VICO makes suggestions based on the current route, the user profile and

information retrieved dynamically from appropriate services (e.g. the internet). Typical examples are:

- the next parking garage or gas station
- a restaurant or hotel

After having entered e.g. the request for a place to have dinner, the virtual co-driver will initiate a negotiation dialogue to further specify the user's request . This is either done by the user giving further specifications ("*An Italian restaurant*", *"I don't care")* or taking information from the user profile (e.g. the user has eaten in an Italian restaurant the evening before, so that VICO proposes a different type of restaurant). Based on the acquired information, the system proposes some possible alternatives and the driver will choose one. If desired the system will also make a reservation at the chosen restaurant. This type of query often requires dynamic information, such as vacancies in hotels or e.g. a list of open gas stations on the weekend.

3. **The user asks for the services of a travel guide:**
   In this mode, the user demands a tailor-made sightseeing tour organized by the system. Again a short negotiation dialogue is initiated to find out the special preferences of the user. In parallel, the virtual co-driver utilizes the user profile to customize the tour to the needs of the driver. For this application VICO has the complete information about the entire route and then guides the user to the navigation destination. The optimal route is calculated and then directions are given to the user. The user feedback is also used to update the user profile continuously. As an example information to the next destination can be linked to information given before. The user furthermore can request more details on some topics or modify the tour.

4. **The user inquires about cockpit instruments and devices** (e.g. a light is blinking):
   In this situation the user asks VICO about the functionality the blinking light in front of him is indicating. The talking car manual will then provide him with the necessary information, without forcing the user to stop and look up the relevant information in the printed version of the car manual.

5. **The user asks about up-to-date news:**
   Instead of listening to conventional radio, the driver might be interested in the news of the day. VICO will provide him/her with reading after having asked if the user might be particularly interested in a special topic like politics, sports or finance.

These five different dialogue situations result in different requirements for the speech recognition module. In mode 1 for example, it is expected that an experienced co-operative user will speak in a fairly controlled way (E.g.: "*London, Hyde Park Five"*). In this case the main challenge for the speech recognition is the huge perplexity due to the large number of cities and streets. On the other hand, in the other scenarios the number of vocabulary items is significantly smaller. The queries, however, are more complex and are likely to contain many more spontaneous effects.

Considering the description presented above the following functional requirements both from user perspective as well as from system perspective can be established.

2.2 User Perspective

2.2.1 *Application*

- driver information system including the following features:
  - navigation: destination entry, route guidance
  - tourism: tourist information, travel planning, personalized sightseeing tour
  - reservation: hotel and restaurant
  - car manual: general car information
  - car status: status information (fuel level, fuel consumption, oil level, etc.)
  - news reader services: search and retrieval of information concerning specific topics

2.2.2 *User Interface*

- user-friendly, comfortable, efficient and safe-to-use interface
- minimize driver distraction
- system operation completely by speech input
- multilingual speech input: German, English, Italian
- natural interaction with each user independently of speaking style, accent, etc. without requiring individual training
- allow spontaneous speech, i.e. everyday language and input of sentences
  ⇨ no predefined keywords or commands
- system output by speech
- optional: acoustical output supported by information on a screen
- start of speech recognition (open microphone):
  – pressing PTT button for starting a dialogue
  – microphone remains open until the task is completed
  – dialogue termination/interruption: microphone is closed after timeout
- optical and acoustical feedback about system state:
  – listening or not listening, i.e. microphone open or closed
  – busy processing the user's input
- ergonomic design and installation of input and output devices (e.g. PTT at steering wheel, screen within visual gaze while driving)
- transparency concerning the system's capabilities, limits and properties

2.2.3 *Speech Recognition*

- accept ill-formed input, i.e. handle spontaneous speech effects like ungrammatical sentence structures, speech containing features as hesitations, pauses, false starts, etc.
- robustness against environmental noise (engine noise, background music, crosstalk, noises from outside the car etc.)

- robustness against unlimited vocabulary, e.g. detect unknown words (out-of-vocabulary (OOV) words)
- dynamic extension of the vocabulary
- reliable recognition and pronunciation of geographic names out of very large lists (e.g. all cities and streets in Germany, England or Italy)
- recognition of non-native speech: cross-lingual pronunciation rules/complementary pronunciation lexicon for the most frequent pronunciation variants of foreign words by non-native speakers
- barge-in during speech output

2.2.4 *Dialogue*

- automatic task identification (user may request services without first defining the topic/task explicitly)
- user- and situation-adaptive dialogue management strategies
- dialogue strategies for clarification of incomplete or ambiguous input
- user profiling (habits, preferences, experience, familiarity with the system, knowledge of foreign languages, etc.)
- adaptation and customization of the system's user interface (dialogue guidance, speech output etc.) to specific user
- situation awareness (take the driving situation into account)

2.2.5 *Information Access*

- offer multilingual conversational information and communication services
- access to dynamic information: extensive, dynamically updated database for all kind of information relevant to the driver
- access to travel-related and tourist information as well as in-car information

2.2.6 *System Design*

- extensibility to new features / devices / services
- easy integration of further languages
- portability to other domains
- integration of all developed components into an in-car demonstrator
- laboratory prototype of the virtual co-driver after the first half of the project duration; all features (even though with reduced functionality) will be integrated

2.2.7 *Evaluation*

- user-level evaluation of intermediate laboratory prototype
- driver-level evaluation of final demonstrator in real driving situations

2.3 System Perspective

For meeting the requirements stated above the system has to be structured into several modules, each covering one of the system specific subtasks: speech recognition (SR), natural language understanding (NLU), dialogue manager (DM), response generation (RG), speech synthesis (SS), database management (Car-Wide-Web, CWW) and device manager (DvM).

The user requirements (user perspective) stated above lead to the module specific requirements in this section.

2.3.1 *System Specific Requirements*

Some requirements concern the whole system, including decisions about platform, control and communication mechanisms:

- platform: Windows NT/2000
- portability to other platforms
- allow distributed system design
- system response within reasonable time period
- extensibility
- modularization
- allow stand-alone usage of every module for test purposes
- addressable at any time, independent of system state or load
- central log information

2.3.2 *Speech Recognition (SR)*

- SR engines for German, English and Italian
- uniform interface between the DM and the different SR engines (G, E, I)
- task identification: several SR engines each covering one specific task running in parallel
- dynamic change between different vocabularies
- dynamic update of dictionaries and language models (LMs)
- dynamic update of class lists (dictionaries and language models)
- different recognition modes:
  – isolated and connected words
  – natural continuous speech
  – spelling (including partial spelling)
- speaker-dependent recognizer for recognition of personal name tags
- unsupervised online speaker adaptation
- word spotting techniques
- training of garbage models
- task-specific language models
- barge-in: implementation of echo-cancellation

- output: word hypothesis graphs and first best sentence

2.3.3 *Natural Language Understanding (NLU)*

- semantic parsing based on predictive finite-state like machines, case-frame grammar analysis
- robustness against ill-formed input (incomplete sentences, ungrammatical sentence structures)
- use of linguistic knowledge about the languages German, English, Italian
- use of domain-specific knowledge
- dynamic update of grammars and lexica
- adaptation of grammar to car- and travel-related tasks
- time-critical processing step: ensure provision of result after certain time period
- output: semantic frames

2.3.4 *Dialogue Manager (DM)*

- VICO's "brain": decide on the system's reaction according to the user's input and to all information that is available
- ensure high degree of interaction, e.g. multi-level mixed-initiative strategies
- provide dialogue strategies for the following situations:
  – the information given by the user is missing, incomplete, ambiguous or erroneous (repair, clarification)
  – the user needs completion or clarification of information provided by the system
  – the user modifies his/her request as a function of the information returned by the system (negotiation)
  – the user accepts the solution proposed by the system (acceptance)
- use and maintain dialogue-related knowledge sources: domain handler, dialogue history, task manager and user model
- collect all relevant information about the preferences, habits and experience of a specific user to create user profile, e.g.
  – age
  – lifestyle, e.g. favorite restaurants and hotel chains
  – geographical profile, i.e. information about previously or frequently requested route descriptions and locations
  – expertise in usage of in-car devices and services
  – level of expertise in a foreign language
- customize and adapt dialogue strategies according to the user profile
- adapt dialogue strategies according to the situation (situation awareness)
- consider multicultural aspects of all languages
- obtain necessary information by consulting the CWW
- handle events that concern user interaction: PTT, speech events (barge-in, garbage, etc.), incoming driving instructions

2.3.5 *Response Generation (RG)*

- generate natural language response as textual output (string) based on dialogue information
- generate semantic/meta information for graphical output on the display
- implicit and explicit feedback depending on dialogue history and user profile
- pre-processing of texts retrieved from the information database to a form suitable for speech synthesis

2.3.6 *Speech Synthesis (SS)*

- synthesis of acoustical speech signal from textual input
- integration of text-to-speech (TTS) systems for the different languages German, English and Italian
- provision of common interface for all languages
- high intelligibility of the spoken output
- high naturalness of synthesized speech
- exception dictionary for words not included in the standard dictionary
- switching of parameters depending on the capabilities of the used TTS systems:
  – language
  – speed
  – pitch
  – speaker

2.3.7 *Car Wide Web (CWW)*

- interfaces for access to all kind of relevant data, both static and dynamic
- access independent of used languages
- XML as data exchange format for information requests and retrieval by the DM [4][5]:
  – flexible, versatile and extensible access to information
  – unified access to data from different static and dynamic information sources
- information filtering according to queries provided by the DM
- methods for obtaining information from the internet
  – first prototype: simulated internet (CD), access mechanisms similar to those necessary to retrieve information from the real internet
  – final demonstrator: real internet connection (if provision of a mobile connection requires little integration effort)
- data provided in all 3 languages: German, English and Italian
- database content:
  – geographic database
  – tourist database
  – car manual
  – dynamic information gathered from the internet, e.g. news articles
- structure of tourist database: s. appendix

2.3.8 *Device Manager (DvM)*

- control of peripheral hardware devices connected to the system: PTT button, display, etc.
- generation of internal events in consequence of user actions (e.g. PTT pressed)
- optional: display text and graphics on attached display
- interface for simulating the car including the generation of car events, e.g. low fuel, low oil level

2.3.9 *System Manager (SM)*

- control and monitor availability of all other modules
- (de-)initialize all modules
- collect status information from modules
- restart crashed modules

3 System Architecture

3.1 Overview

The following diagram illustrates all modules identified through the requirements above. It also gives an overview over the interaction of these modules and attached devices.

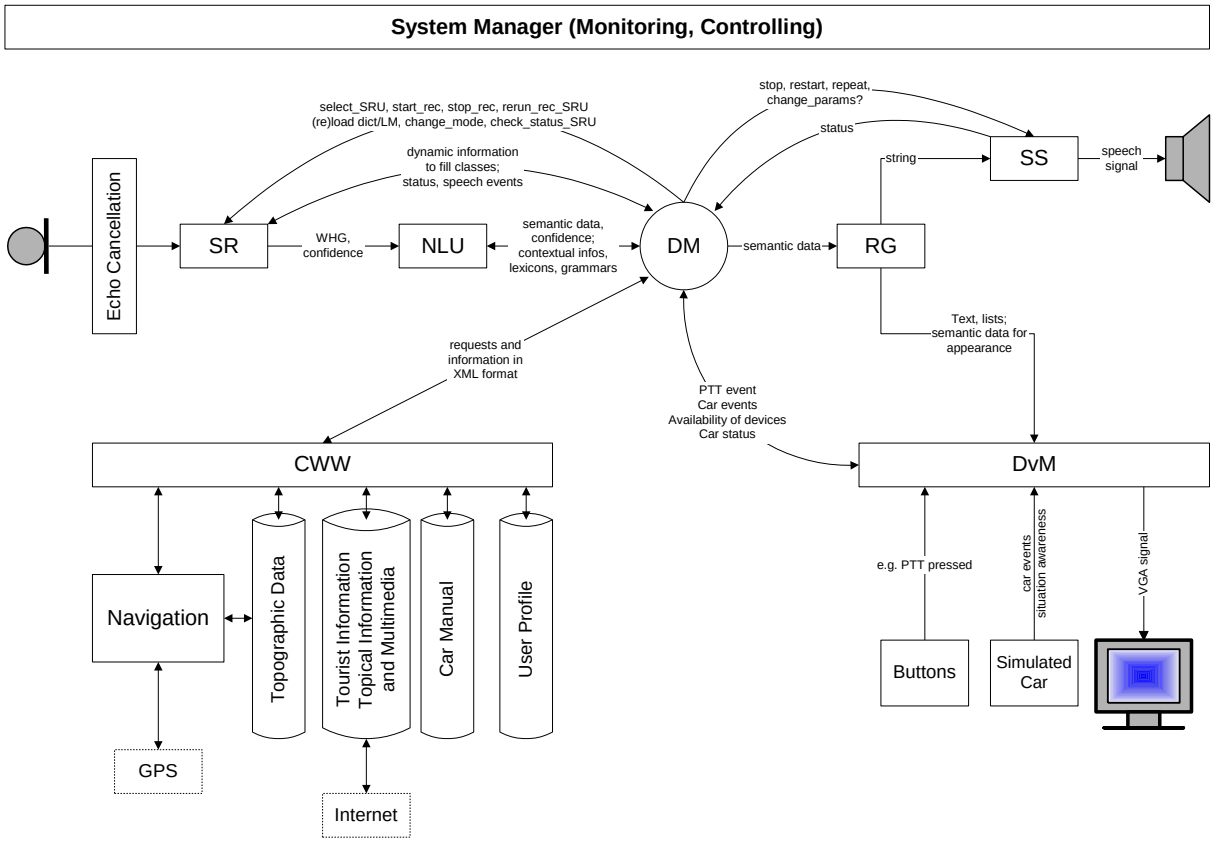

Figure 3.1: System Overview

3.2 View Model of Software Architecture

Views are a model for describing the architecture of software-intensive systems [1][2]. They are abstractions consisting of declarative diagrams in order to specify one aspect of the system to be modeled. Each view is intended to clarify one major issue of system architecture and hence has its own "customers" or stakeholders.

For describing the software architecture of the VICO system at least the following views apply:

- process view, focusing on main concurrent processes and communication between these processes
- physical view, mapping software onto hardware
- scenarios, i.e. instances of general use cases

3.2.1 *Process views*

The following diagram visualizes the main processes of the system running in parallel. Its purpose is to show the main data and control flow.[2]

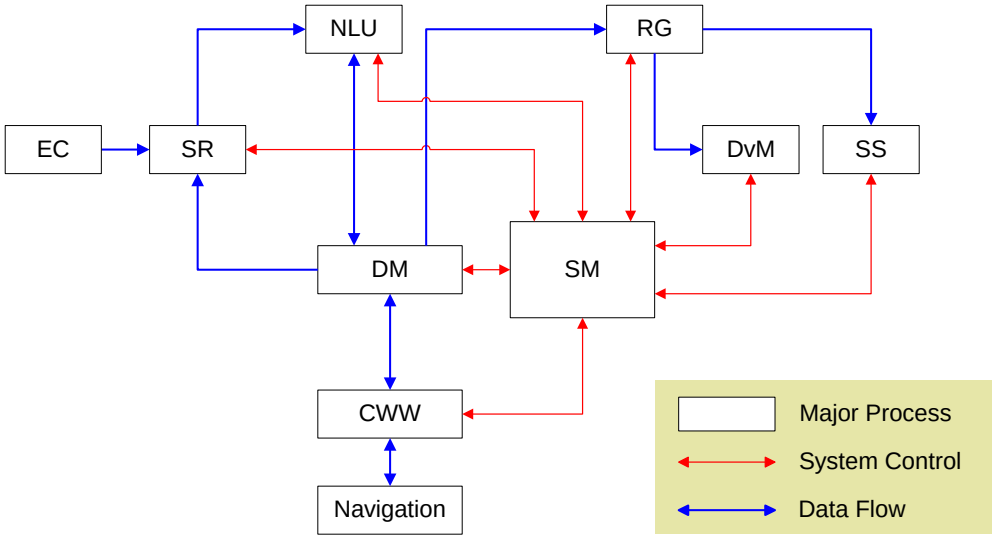

Figure 3.2: Process View

A more detailed view focusing on communication and synchronization aspects is found below. Note that control and monitoring by the SM has been omitted due to clarity:

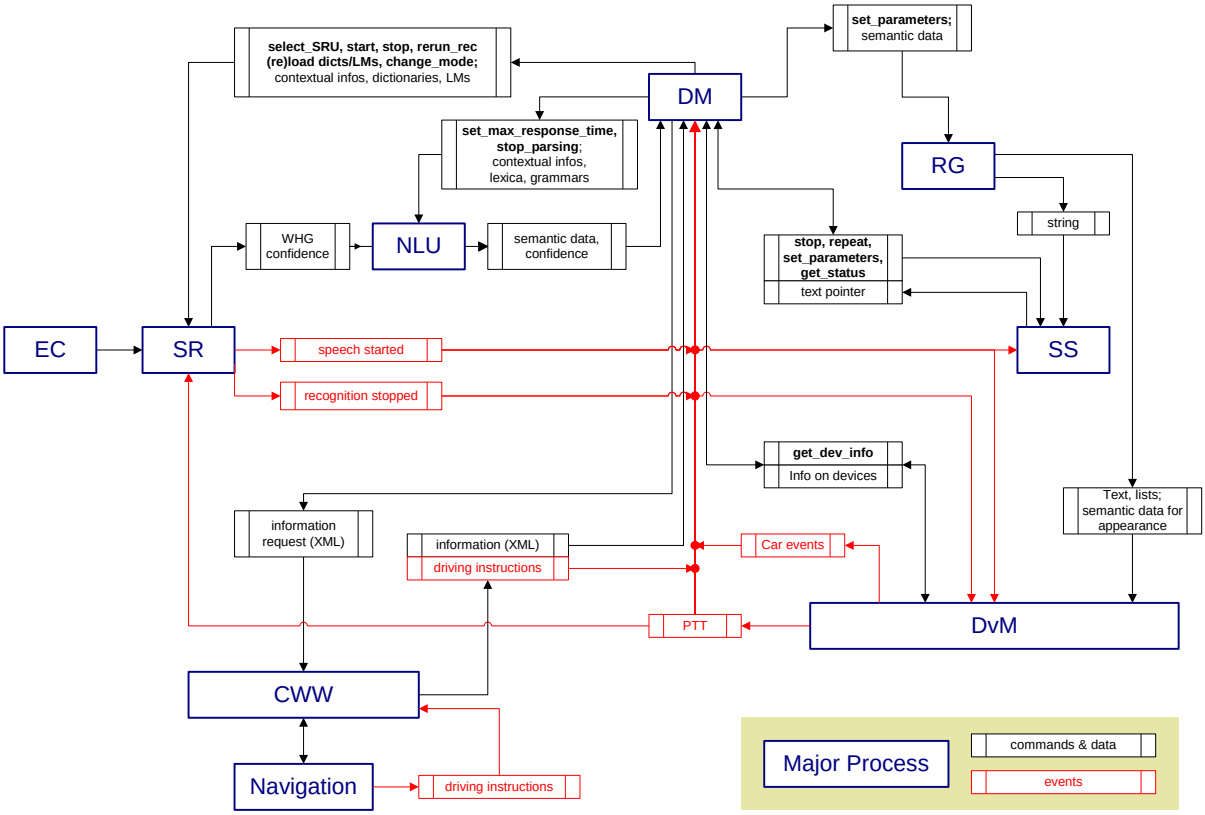

Figure 3.3: Process View and Communication

[2] This data exchange can either be done directly between modules or via a communication manager.

3.2.2 *Physical view*

As so far the usage of only one PC (no distributed architecture) is planned and the view would be rather simple it has been enhanced by the hardware dependencies.

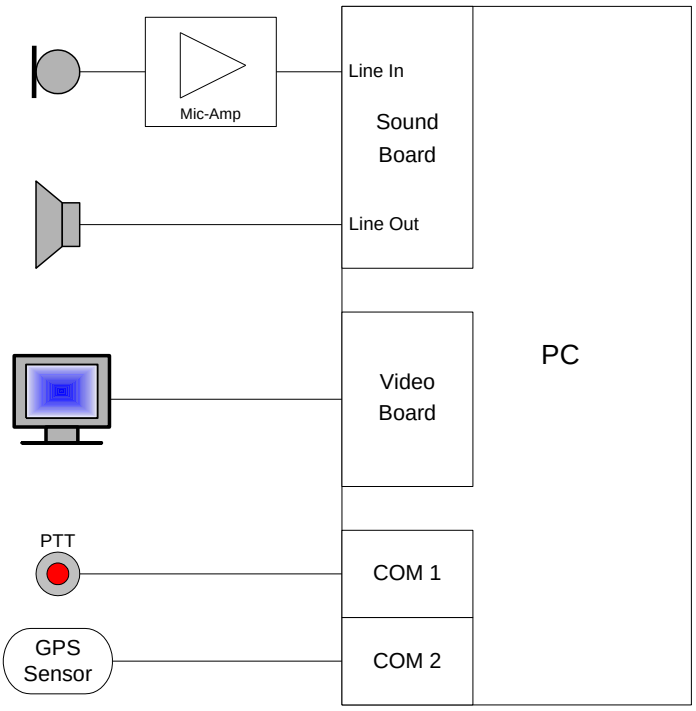

Figure 3.4: Physical View

3.2.3 *Example Scenario: Destination Entry and Hotel Information*

Preconditions and assumptions:

- many SRUs (speech recognition units) available, corresponding dictionaries and LMs (language models) can be loaded during start of the system; contents and subtasks of SRUs have to be decided (e.g. main application tasks, navigation subtask depending on actual position, yes/no-recognizer, see proposal below)
- several SRUs can be selected to be running at the same time, which means one user utterance is being processed by all active SRUs
- DM decides which units to be active for each dialogue step, i.e. recognition process, thus units are selected by DM
- system feedback: appropriate visual and optional acoustical feedback (e.g. green light and beep) will show that the system is listening (open microphone) or not (microphone muted)

Scenario:

Assuming the user is in Munich.

Initial phase:
At least 5 SRUs are initialized with context information (dictionaries, LMs) according to the application tasks and are active at the same time:
1. navigation (city, street names, POIs)
2. tourism: tourist information and sightseeing in general

3. booking assistance: hotel and restaurant reservation
4. car status / car manual
5. newsreader

- DM asks CWW for actual position.
- CWW asks navigation for actual GPS position.
- Either navigation or CWW will determine the current position (city, street) out of the GPS data.
- CWW sends position information to DM.
- DM sends context-based information to SR for updating LMs and dictionaries in specified SRUs depending on car location and user model.
- DM sends context-based information to NLU for updating grammars.

U: *pushes the PTT button*

- DvM receives signal from PTT button
- PTT event is transmitted to DM and SR
- DvM shows green light on display
- SR starts recognition process for all selected SRUs, either working in parallel or sequentially

U: *"I would like to go to Stuttgart."*

- All 5 SRUs send WHG/best sentence including confidence measures to NLU.
- NLU derives semantic representation from the 5 SR results. These semantic frames together with corresponding confidence measures are passed to DM.
- Based on these confidence measures from NLU DM decides which application task to choose.

In the following we assume that the system recognized that the user probably wants to go to Stuttgart. As a reasonable consequence DM intends to specify the destination entry.

- DM sends semantic representation to the RG for prompting the user to specify the destination.
- RG builds a natural language output and sends a corresponding text string to the SS.
- RG sends meta information to the DvM for displaying text (and optional: graphics) on the screen.

VICO: *"Do you already know where in Stuttgart you would like to go?*

- During speech output DM sends task-specific information to SR causing some of the SRUs to load new dictionaries / LMs:
  1. navigation (streets of Stuttgart)
  2. tourism (tourism information related Stuttgart, POIs)
  3. hotel/restaurant reservation (hotels and restaurants of Stuttgart and further information according to both)
- DM sends task-specific information to NLU to adapt grammars.

U: *"I don´t know yet. Are there any sights worth taking a look at?"*

< General processing as above for all following dialogue steps >

In the following we assume that confidence measure of the tourism task was rated highest.

VICO: *"Do you want some information about touristic sights in Stuttgart?"*

< Dynamic update of dictionaries, LMs, grammars, etc. according to available information >

- During speech output DM sends a request to the CWW for retrieving more detailed information about sights in Stuttgart, if not already loaded.
- In addition to all active SRUs DM selects specialized SRU on acceptance and rejection (yes/no recognizer) as a closed question has been asked.

U: *"Yes, please."*

VICO: *"In Stuttgart, there is ... "*

U: *"Uh, ..., also do you have any recommendations on where to stay?"*

- NLU detects a request for overnight stay.

VICO: *"Would you like to stay at a hotel tonight?"*

- DM sends request to CWW for retrieving more detailed information about the hotels in Stuttgart.

U: *"No, I just want some general information on hotels."*

VICO: *"Do you have any preferences concerning price, location, rating, services of the hotel you are looking for?"*

U: *"Ah, you know, I usually stay at a Holiday Inn. Is there one in Stuttgart?"*

VICO: *"There are three hotels in Stuttgart belonging to the hotel chain Holiday Inn."*

4 High-Level Specifications of all System Modules

In the following, module specifications of all modules that have been identified above are given. As no decision has been fixed yet in which way the modules will communicate with each other (both concerning control and data exchange) the interfaces between the modules are described for direct communication even though the real communication will be conducted over a CORBA architecture [6][7][8] or over a separate communication manager module.[3]

4.1 System Manager

<table>
<tr><td colspan="2">Component Name (acronym)</td></tr>
<tr><td colspan="2">System Manager (SM)</td></tr>
<tr><td colspan="2">Description</td></tr>
<tr><td colspan="2">Monitor, start and stop system modules</td></tr>
<tr><td>Component Dependencies</td><td>Hardware/Software Dependencies</td></tr>
<tr><td>all available system modules</td><td>Windows 98/NT/2000</td></tr>
<tr><td colspan="2">Functional Description</td></tr>
<tr><td colspan="2">The System Manager monitors all existing system modules by requesting status information from them regularly. The SM starts (initializes) and stops (de-initializes) all system modules and is able to restart (reinitialize) them if he gets no response or receives an unexpected termination event.</td></tr>
<tr><td colspan="2">Interfaces for Data and Information Exchange</td></tr>
<tr><td>XXX ⇔ SM</td><td>Input from all other modules:
• status (exception code, processing progress)
Output to all other modules: control commands
• initialize
• de-initialize
• get_status</td></tr>
<tr><td>DvM ⇔ SM</td><td>Input from DvM:
• status (exception code) of module
• status of connected peripheral devices
Output to DvM: control commands
• initialize
• de-initialize
• get_status</td></tr>
<tr><td colspan="2">Communication Method: CORBA</td></tr>
<tr><td colspan="2">Data Interface (External Data Resources)</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Programming Language</td></tr>
<tr><td colspan="2">C++</td></tr>
</table>

[3] However, as this does not concern the content that has to be transferred, but only the way information is transmitted, the content description does not vary and will still be valid no matter which alternative will be realized.

4.2 Speech Recognizer

<table>
<tr><td colspan="2">Component Name (acronym)</td></tr>
<tr><td colspan="2">Speech Recognizer (SR)</td></tr>
<tr><td colspan="2">Description</td></tr>
<tr><td colspan="2">Recognition of spoken utterances delivering textual representations</td></tr>
<tr><td>Component Dependencies</td><td>Hardware/Software Dependencies</td></tr>
<tr><td>Dialogue Manager (DM), Natural Language Understanding (NLU), System Manager (SM)</td><td>Windows NT/98
sound device: Soundblaster compatible (e.g. 128 PCI)</td></tr>
<tr><td colspan="2">Functional Description</td></tr>
<tr><td colspan="2">The SR can be used in different modes:
• isolated word recognition
• connected word recognition
• recognition of continuous speech
• speaker-dependent mode (e.g. user defined names)
The system is in general speaker-independent with some implicitly integrated online adaptation methods.
The results of the recognizer are transmitted to NLU in form of a WHG (word hypothesis graph) and a best sentence.</td></tr>
<tr><td colspan="2">Interfaces for Data and Information Exchange</td></tr>
<tr><td>SR ⇔ DM</td><td><u>Input from DM:</u>
• control commands
- select_SRU (units)
- start_rec
- stop_rec
- stop_discard
- rerun_rec (unit, utterance)
- (re)load_dictionaries
- (re)load_LMs
- change_mode (spelling, continuous words , isolated words)
• contextual information, dictionaries, LMs
<u>Output to DM:</u>
• speech events
- speech started
- recognition stopped</td></tr>
<tr><td>SR ⇔ NLU</td><td><u>Input from NLU:</u>
• none
<u>Output to NLU:</u>
• WHG (word hypothesis graph)
• best sentence
• confidence measures</td></tr>
<tr><td>SR ⇔ SM</td><td><u>Input from SM</u>:
• initialize
• de-initialize
• check_status
<u>Output to SM</u>:
• status (exception code, processing progress)</td></tr>
</table>

Communication Method: CORBA
Data Interface (External Data Resources)
• SR-internal interfaces for wave-I/O and data (i.e. lexicon, language model, HMM-parameters)
Programming Language
C, C++

4.3 Natural Language Understanding

Component Name (acronym)	
Natural Language Understanding (NLU)	
Description	
Converts a text string to some kind of semantic representation.	
Component Dependencies	Hardware/Software Dependencies
Dialogue Manager (DM), Speech Recognition (SR), System Manager (SM)	Windows 98/NT/2000
Functional Description	
The NLU module processes a WHG/best sentence and extracts from it the semantics of a user utterance.	
Interfaces for Data and Information Exchange	
SR ⇔ NLU	Input from SR: • recognition result, i.e. - WHG, - best sentence, - confidence measures Output to SR: • none
DM ⇔NLU	Input from DM: • control commands - set_max_response_time (time) - stop_parsing • context information on which lexicons and grammars to use Output to DM: • a semantic representation of the meaning of a user utterance • confidence measures from SR and NLU
SM ⇔NLU	Input from SM: • initialize • de-initialize • check_status Output to SM: • status (exception code, processing progress)
Communication Method: CORBA	
Data Interface (External Data Resources)	
Programming Language	
C++, Prolog, or Java.	

4.4 Dialogue Management

<table>
<tr><td colspan="2">Component Name (acronym)</td></tr>
<tr><td colspan="2">Dialogue Management (DM)</td></tr>
<tr><td colspan="2">Description</td></tr>
<tr><td colspan="2">Manages the dialogue between user and system</td></tr>
<tr><td>Component Dependencies</td><td>Hardware/Software Dependencies</td></tr>
<tr><td>Speech Recognition (SR), Natural Language Understanding (NLU), Response Generation (RG), Device Manager (DvM), Car-Wide-Web (CWW), System Manager (SM)</td><td>Windows 98/NT/2000</td></tr>
<tr><td colspan="2">Functional Description</td></tr>
<tr><td colspan="2">The Dialogue Management component receives semantic input from the Natural Language Understanding component and sends semantic output to the Response Generation component. It sends requests for e.g. position, location, or distance to the Car-Wide-Web and receives this information from the Car-Wide-Web. It sends context information to the SR to load lexicons and language models.</td></tr>
<tr><td colspan="2">Communication Interfaces</td></tr>
<tr><td>SR ⇔ DM</td><td><u>Input from SR</u>:
• speech events
- speech started
- recognition stopped
<u>Output to SR</u>:
• control commands
- select_SRU (units)
- start_rec
- stop_rec
- stop_discard
- rerun_rec (unit, utterance)
- (re)load_dictionaries
- (re)load_LMs
- change_mode (spelling, continuous words , isolated words)
• contextual information, dictionaries, LMs</td></tr>
<tr><td>NLU ⇔ DM</td><td><u>Input from NLU:</u>
• a semantic representation of the meaning of a user utterance.
• confidence measures from SR and NLU
<u>Output to NLU</u>:
• control commands
- set_max_response_time (time)
- stop_parsing
• context information on which lexicons and grammars to use
• confidence measures from SR and NLU
• contextual information from the linguistic history.</td></tr>
<tr><td>CWW ⇔ DM</td><td><u>Input from CWW</u>:
• Response (e.g. on position, location or distance) in XML
• navigation events (including driving instructions)
<u>Output to CWW</u>:
• Request (e.g. for information on position, location or distance) in XML</td></tr>
</table>

RG ⇔ DM	Input from RG: • none Output to RG: • control commands - set_parameters (language, speaking-style) • a semantic representation of an oral response to the user • a semantic representation of the graphics response to be displayed on the screen
SS ⇔ DM	Input from SS: • progress of textual output (speech output active/inactive; pointer on current text passage) Output to SS: • control commands - stop speech output - restart / repeat utterance - set_parameters (language, speed of speech etc.) - get_status
DvM ⇔ DM	Input from DvM: • information on peripheral device(s) (parameters depending on device type) • events - generated events from simulated car - PTT pressed Output to DvM: • get_dev_info (device)
SM ⇔ DM	Input from SM: • initialize • de-initialize • check_status • status information about availability of devices Output to SM: • status (exception code, processing progress)
Communication Method: CORBA	
Data Interface (External Data Resources)	
Programming Language	
C++, Prolog, or Java.	

4.5 Response Generation

<table>
<tr><td colspan="2">Component Name (acronym)</td></tr>
<tr><td colspan="2">Response Generation (RG)</td></tr>
<tr><td colspan="2">Description</td></tr>
<tr><td colspan="2">Formulation of the information that is to be sent to the user.</td></tr>
<tr><td>Component Dependencies</td><td>Hardware/Software Dependencies</td></tr>
<tr><td>Dialogue Manager (DM), Speech Synthesis (SS), Device Manager (DvM), System Manager (SM),</td><td>Windows 98/NT/2000</td></tr>
<tr><td colspan="2">Functional Description</td></tr>
<tr><td colspan="2">The Response Generation component receives a semantic representation of an oral response to the user as well as of the graphics to be sent to the screen. The RG module formulates an appropriate textual version of the oral response, and sends this text string to the TTS module. It also formulates the graphics to be sent to the screen and sends a representation of it to the DvM.</td></tr>
<tr><td colspan="2">Communication Interfaces</td></tr>
<tr><td>DM ⇔ RG</td><td><u>Input from DM</u>:
• control commands
- set_parameters (language, speaking-style)
• a semantic representation of an oral response to the user
• a semantic representation of the graphics response to be displayed on the screen
<u>Output to DM</u>:
• none</td></tr>
<tr><td>SS ⇔ RG</td><td><u>Input to from SS</u>:
• none
<u>Output to SS</u>:
• string containing the system utterance(s) to be synthesized</td></tr>
<tr><td>DvM ⇔ RG</td><td><u>Input from DvM</u>:
• none
<u>Output to DvM</u>:
• representation(s) containing the system output to be displayed</td></tr>
<tr><td>SM ⇔ RG</td><td><u>Input from SM</u>: control commands
• initialize
• de-initialize
• check_status
<u>Output to SM</u>:
• status (exception code, processing progress)</td></tr>
<tr><td colspan="2">Communication Method: CORBA</td></tr>
<tr><td colspan="2">Data Interface (External Data Resources)</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Programming Language</td></tr>
<tr><td colspan="2">C++, Prolog, or Java.</td></tr>
</table>

4.6 Speech Synthesis

<table>
<tr><td colspan="2">Component Name (acronym)</td></tr>
<tr><td colspan="2">Speech Synthesis (SS)</td></tr>
<tr><td colspan="2">Description</td></tr>
<tr><td colspan="2">Synthesizes speech signal, that is reproduced by the system loudspeakers</td></tr>
<tr><td>Component Dependencies</td><td>Hardware/Software Dependencies</td></tr>
<tr><td>Response Generation (RG), Dialog Manager (DM), System Manager (SM)</td><td>Windows 98/NT/2000
APIs of commercially available TTS systems
sound device: Soundblaster compatible (e.g. 128PCI)</td></tr>
<tr><td colspan="2">Functional Description</td></tr>
<tr><td colspan="2">The Speech Synthesis component receives textual input from the Response Generation component. It internally uses API functions of commercially available TTS systems to synthesize speech that is reproduced by the system loudspeakers. Speech output can be interrupted which means stopped and restarted by the DM. It provides functions to change parameters like language, speed of speech etc.</td></tr>
<tr><td colspan="2">Interfaces for Data and Information Exchange</td></tr>
<tr><td>RG ⇔ SS</td><td><u>Input from RG</u>: start command with output string
• string (containing the text to be synthesized)
<u>Output to RG</u>:
none</td></tr>
<tr><td>DM ⇔ SS</td><td><u>Input from DM:</u>
• control commands
- stop speech output
- restart / repeat utterance
- set_parameters (language, speed of speech etc.)
- get_status
<u>Output to DM</u>:
• progress of textual output (speech output active/inactive; pointer on current text passage)</td></tr>
<tr><td>SM ⇔ SS</td><td><u>Input from SM:</u> control commands
• initialize
• de-initialize
• check_status
<u>Output to SM</u> (in return of the SM requests):
• status (exception code, processing progress)</td></tr>
<tr><td colspan="2">Communication Method: CORBA</td></tr>
<tr><td colspan="2">Data Interface (External Data Resources)</td></tr>
<tr><td colspan="2">• user dictionary for exceptions
• initialization files for changing parameters (e.g. language, speaker, pitch, speed)</td></tr>
<tr><td colspan="2">Programming Language</td></tr>
<tr><td colspan="2">C++</td></tr>
</table>

4.7 Device Manager

<table>
<tr><td colspan="3">Component Name (acronym)</td></tr>
<tr><td colspan="3">Device Manager (DvM)</td></tr>
<tr><td colspan="3">Description</td></tr>
<tr><td colspan="3">Monitoring peripheral hardware devices, simulating hardware devices, generating events</td></tr>
<tr><td colspan="2">Component Dependencies</td><td>Hardware/Software Dependencies</td></tr>
<tr><td colspan="2">Dialog Manager (DM), Response Generation (RG), System Manager (SM)</td><td>Windows 98/NT/2000
communication port (e.g. serial interface COM)
hardware devices:
• PTT button
• Display</td></tr>
<tr><td colspan="3">Functional Description</td></tr>
<tr><td colspan="3">The Device Manager monitors and controls the peripheral hardware devices connected to the system. It provides status information about these devices and is responsible for generating internal events transferred to the DM in consequence of user actions (e.g. pressing the PTT). The DvM offers a user interface for simulating the car (including the generation of car events, e.g. low fuel, low oil etc.). It is responsible for displaying information received from the RG, too.</td></tr>
<tr><td colspan="3">Interfaces for Data and Information Exchange</td></tr>
<tr><td>DM ⇔ DvM</td><td colspan="2"><u>Input from DM</u>:
• get_dev_info (device)
<u>Output to DM</u>:
• information on peripheral device(s) (parameters depending on device type)
• events
- generated events from simulated car
- PTT pressed</td></tr>
<tr><td>RG ⇔ DvM</td><td colspan="2"><u>Input from RG:</u>
• representation(s) containing the system output to be displayed</td></tr>
<tr><td>SM ⇔ DvM</td><td colspan="2"><u>Input from SM:</u> control commands
• initialize
• de-initialize
• check_status
<u>Output to SM</u> (in return of the SM requests):
• status (exception code) of module and connected peripheral devices (availability)</td></tr>
<tr><td colspan="3">Communication Method: CORBA</td></tr>
<tr><td colspan="3">Data Interface (External Data Resources)</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Programming Language</td></tr>
<tr><td colspan="3">C++</td></tr>
</table>

4.8 Car Wide Web

Component Name (acronym)	
Car Wide Web (CWW)	
Description	
Interface to all the databases and the internet	
Component Dependencies	Hardware/Software Dependencies
Dialog Manager (DM), System Manager (SM)	Windows 98/NT/2000
Functional Description	
The CWW acts as a gateway for the access to the touristic databases, navigation component, Internet, car manual and user profile. It should process requests from the DM and convert them into specific calls to the various APIs provided for the databases. Requests and related responses are in XML format.	
Interfaces for Data and Information Exchange	
DM ⇔ CWW	Input from DM: • Requests in XML (current_position, load_POI, load_touristic_info, load_user_profile, save_user_profile, load_dialog_history, save_dialog_history, route_planning, hotel_reservation, restaurant_reservation, car_manual, news_reading, etc.) Output to DM: • Responses in XML (e.g. on position, location or distance) • navigation events (including driving instructions)
SM ⇔ CWW	Input from SM: control commands • initialize • de-initialize • get_status Output (in return of the SM requests): • status (exception code, processing progress)
Communication Method: CORBA	
Data Interface (External Data Resources)	
• DTD files • Various databases	
Programming Language	
Java	

5 Bibliography

[1] Ph. Kruchten. Architectural Blueprints – The "4+1" View Model of Software Architecture. In *IEEE Software*, 12 (6), pp. 42-50, 1995

[2] http://www.math.uwaterloo.ca/~mctanuan/cs746g/FourPlusOneSlides.html

[3] G. Bannert and M. Weitzel. Objektorientierter Softwareentwurf mit UML. Addison Wesley Longman, 1999

[4] http://www.w3.org/XML/

[5] http://www.w3schools.com/xml/default.asp

[6] http://www.omg.org/

[7] http://www.corba.org/

[8] http://patriot.net/~tvalesky/freecorba.html

Appendix

The following table shows structure and content of the tourist database:

Points of Interest	Ready at APT	Ready TA-DB
Cultural places		
• historical churches	-	-
• museums	-	✓
• castles	-	✓?
Natural places		
• lakes	✓	✓
• mountain peaks/passes	✓	✓
• parks = natural reserves	✓	✓
Accommodation		
• hotels	✓	✓
• campgrounds	✓	✓
Leisure		
• cinemas	-	✓
• discos	-	-
• theatres	-	✓
• casinos	n.a.	n.a.
• amusement parks	n.a.	n.a.
Food		
• restaurants	only I	✓
• wineries	only I	✓
Sport sites		
• public swimming pools	only I	✓
• stadiums/arenas	only I	✓
Sport activities		
• skiing	✓	-
• ice-skating	-	-
Public services		
• airports	-	✓
• banks	-	-
• train stations	-	✓
• parking garages	-	✓
• car repair	-	✓
• post offices	-	✓
• universities	-	✓
• gas stations	-	✓
Health services		
• pharmacies	-	✓
• hospitals	-	✓
• doctors	-	✓