UNIVERSITY OF HERTFORDSHIRE

School of Information Sciences

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Project Report

An investigation of the suitability of the blackboard architecture as the basis for planning mobile robot movements

R.Wittwer

April 1996

Chapter 1 — Introduction

1.1 Project motivation **4**

1.2 Aims and objectives **4**
1.2.1 Core objectives 4

1.3 The report structure **5**

Chapter 2 — Background

2.1 Blackboard **6**
2.1.1 Which problems does a blackboard solve? 6

2.2 Mobile robots **7**
2.2.1 The technical abilities of the focused mobile robot 7
2.2.2 The area in which the robot moves 7
2.2.3 The tasks of the focused mobile robot 8
2.2.4 Possible extensions of the focused mobile robot 8
2.2.5 Unsolved problems in the mobile robots world 8

2.3 The interface between blackboard and mobile robot **9**

2.4 Safe and unsafe Areas **10**

2.5 Coarse grained or fine grained **10**

Chapter 3 — The data structure of the blackboard

3.1 Meta plan **12**
3.1.1 Problem definition 12
3.1.2 Problem solving model 12
3.1.3 Policy 12
3.1.4 Evaluation criteria 13

3.2 Executor **13**
3.2.1 Priorities 13
3.2.2 Focus 13
3.2.3 Schedule 13

3.3 Plan abstraction **14**

3.4 Knowledge base **14**

3.5 Plan **15**

3.6 Example **15**
3.6.1 Meta plan (problem solving model) 15
3.6.2 Meta plan (policies) 16
3.6.3 Executor (priorities) 17
3.6.4 Executor (focus) 17
3.6.5 Plan-abstraction (intentions) 18
3.6.6 Where is the robot? 19

3.6.7 Where does the robot want to go? (Part 1) 20
3.6.8 How can the robot leave the room? 21
3.6.9 Where does the robot want to go? (Part 2) 22
3.6.10 Which of the given places can be reached faster? 23
3.6.11 Where does the robot want to go? (Part 3) 24
3.6.12 Map information 24
3.6.13 Find the fastest route 25
3.6.14 Change the focus 26
3.6.15 Move one meter in a straight direction 27
3.6.16 From plan-abstraction and knowledge base to the plan 27
3.6.17 Turn left 90° 28
3.6.18 Plan procedure 28
3.6.19 Move 50 cm 28
3.6.20 Plan procedure 28

3.7 Summary 29

Chapter 4 — Specialist

4.1 Administration 31
4.1.1 The administration specialists for moving 31
4.1.2 Stop the Exploration if has been found the Goal? 34
4.1.3 The administration specialists for exploring 35
4.1.4 When shall the robot stop the exploration? 36

4.2 Top management 36

4.3 Middle management 37
4.3.1 Middle management specialists for moving 37
4.3.2 Middle management specialists for exploring 37

4.4 Director 38
4.4.1 Director specialists for moving 38
4.4.2 Director specialists for exploring 39

4.5 Policy analyst 41
4.5.1 Policy analyst specialists for moving 41
4.5.2 Policy analyst specialists for exploring 41

4.6 Introduction to the planning process 42
4.6.1 One or several approaches 42
4.6.2 Value of information 43

4.7 The planning process itself 44
4.7.1 Strategy, subgoals and how to explore 44
4.7.2 Which is the starting point? 44
4.7.3 Which kind of route should the robot use? 44
4.7.4 Where does the robot want to go? 44

4.8 Summary 45

Chapter 5 — Knowledge representation

5.1 The hierarchical part **46**
5.1.1 Location of items 47
5.1.2 Movement facilities of items 48
5.2 Unsafe areas **49**
5.3 The direction **50**
5.4 The three-dimensional description **53**
5.5 Convert the information into a two dimensional map **54**
5.6 Convert fine grained map into coarse grained map **55**
5.7 The procedural representation **55**
5.8 Inexact data **56**
5.9 Unknown object **57**
5.10 Summary **57**

Chapter 6 — Conclusion

6.1 Evaluation of the objectives **58**
6.2 Evaluation of project management **58**
6.3 Discussion of further work **60**
6.4 Final conclusion **60**

Chapter 7 — References 61

1. Introduction

1.1 Project motivation

This final project is part of a research project at the Fachhochschule Hamburg which is called ICS (short for **I**ntegrating **C**ognitive **S**ystems).

The aim of ICS is to investigate theoretical and practical methods to combine software and hardware components into one system which can solve a wider range of problems than standard systems. The objectives of ICS are

- *Investigation of mechanisms to integrate different concepts and, if possible, software modules into one system, especially mechanisms from the area of Artificial Intelligence*
- *Construction of a modular, orthogonal collection of software modules to support experiments in the areas of Software Engineering and Artificial Intelligence*

[Luck, Owsnicki-Klewe, 1995]

The area of this final project was completely new to me when I started. I had no previous knowledge or experience apart from the lessons of artificial intelligence from the Fachhochschule Hamburg. In these lessons we were told about planner in general. The blackboard architecture was new to me. Even the mobile robot world was a new area except from the 'blocks world'. The 'blocks world' can be relevant when the robot has to deliver something.

1.2 Aims and objectives

The idea of the final project is to research whether the blackboard control architecture is suitable for a planner which focuses on a mobile robot's world.

The word *planner* may have many interpretation. In the context of the blackboard control architecture it involves the whole blackboard and its planning task

The blackboard control architecture was generalised from the Hearsay II speech understanding system. It tries to imitate human behaviour.

The purpose of the blackboard is to offer problems to specialists, to collect the different solutions from the specialists and to choose the best one. Specialists are a lot of small programs which read the problems from the blackboard and give either a solution, the next step or an idea how to solve the problem. Specialists write their problem-solving idea on the blackboard and the blackboard control architecture chooses the best of all ideas.

1.2.1 Core objectives

The first task was to understand the actual properties of mobile robots. The next task was to understand the problems for any artificial intelligence task in the domain of mobile robots.

Further more I had to understand the blackboard control architecture and investigate if it is possible to use it in the mobile robots world.

I worked out my **own examples**, described a practicable **information level structure** and **knowledge representation**, a suitable **data structure** and **design different specialists**.

1.3 The report structure

This report will start with a chapter that provide background information. The background chapter will give an overview about the benefits of using a blackboard and introduce the problems which can be solved with this architecture (2.1 Blackboard). I will then give an idea of the focused mobile robot's (2.2 Mobile robot) and the interface between the blackboard and the mobile robot, although this is not a technical view but a logical one. The last parts of the background chapter will introduce the terms 'Safe and unsafe areas' (2.4) and 'Coarse grained and fine grained' (2.5).

The topic of the third chapter will be the logical structure of the blackboard. This chapter is divided into two parts. The first part will give an introduction in the logical structure of the blackboard (3.1 to 3.5). The second part will present an example of my own that shows how a mobile robot planner, using a blackboard, works (3.6 Example).

In the fourth chapter I will present specialists (4 Specialists).

The last main chapter will present a suitable knowledge representation (5 Knowledge representation).

2. Background

2.1 Blackboard

Humans planning behaviour is opportunistic. In contrast to the most artificial intelligence planning models, humans use a lot of different planning strategies. Planning strategies are for example top-down, bottom-up, backward-chaining or island extension. At each state of the planning process humans choose between various opportunities for plan development.

The blackboard (blackboard control architecture) tries to imitate human behaviour. The idea is to offer a problem on the blackboard, to collect all approaches how to solve the problem and to choose the best.

The best plan strategy could be one of those described above. But it also could be a neuronal net or an other sophisticated method.

Because we do not have to choose the method now, the blackboard is so powerful. The blackboard is only the communication tool. We can build the application including the blackboard now. If artificial intelligence scientist finds new methods of solving particular problems we can implement and integrate it later. We only have to build the communication interface to the blackboard.

2.1.1 Which problems does a blackboard solve?

The first time that the blackboard idea appeared was in the Hearsay-II speech-understanding system [Erman, Lee D.; Hayes-Roth, Frederick; Lesser, Victor R.; Reddy, Ray D., 1980]. From this application it was generalised.

Nowadays existing commercial and free blackboard systems. That means that blackboards are used in various domains. The following examples are from [Hayes-Roth, Barbara, 1993]

- speech understanding
- intensive care monitoring
- semiconductor fabrication
- power plant maintenance
- and ... mobile robot

As can be seen in the list above blackboards are still used in the domain of mobile robots. That means that the idea of this final project is not new.

At the beginning of this final project I thought this idea is new. But during the preparation I found that Stanford University investigated the usability of blackboards for mobile robots [Hayes-Roth, Barbara; Pfleger Karl; Lalanda, Philippe; Morignot, Philippe; Balabanovic, Marko, 1995].

I read a number of papers of the Stanford University. The result has been a lot of theoretical description of the blackboard or a description of the mobile robot task. None of the papers describe the blackboard control architecture together with mobile robots as refined as in this report. The task was to investigate theoretical ideas and concepts. A further task was to describe a concrete application.

2.2 Mobile robots

2.2.1 The technical abilities of the focused mobile robot

At the beginning of this Final Project the idea was to take robots with the technical abilities as described in [Jones and Flynn, 1993] especially the 'Rug Warrior'. This is a very simple and cheap robot. The main components are

- A cheap processor (CPU)
- Memory
- Bus
- 5 Input / Output Ports
- Sensors (light sensors, force sensors, sound sensors, etc.)
- Battery
- Motor
- Wheels

A robot as described above is able to move around and that's it.
The advantages of this robot are that it is very cheap and easy to build. The disadvantages of this robot are the lack of memory and limited computational resources. The disadvantages embody one problem.

In most cases, it is necessary for the programmer to fully understand the bit-level interaction between the processor and the devices it controls. [Jones and Flynn, 1993]

But this is exactly what we do not want to do. We do not want understand the bit-level interaction between the processor and the devices it controls. The purpose of this Final Project is to build several layers of abstraction which will help to improve the behaviour of the robot. But this includes overhead. Therefore the robot needs a more powerful processor and much more memory.

Because this is a theoretical project it is possible to change the technical abilities of the robot very easily. There is no need to consider the costs of the mobile robot.

At this point of the Final Project I will not describe the technical details of the requested robot. Probably there are lot of ways to build it. Nevertheless I will give an idea how it could work. One idea is to have a stationary computer with a fast processor and a lot of memory. This computer is connected with the robot. One example of connection could be a radio connection.
A very interesting question is, which parts of the computation are 'on board' of the robot and which are at the stationary computer. This could be a starting point for further investigation.

2.2.2 The area in which the robot moves

The focused mobile robot is an indoor robot. The robot has not the ability to move up- or down-stairs. That means that the borders of the world are

- Wall
- Steps

- Outside-door

The area of the robot in this final project will be a normal office.

2.2.3 The tasks of the focused mobile robot

The main task of the mobile robot in this Final Project is to navigate and to recognise.

To navigate means that the robot moves and tries to hit against something as little as possible. Of course it is possible to modify the environment with beacons and bar-codes. But our robot should be able to navigate without these utilities.

However, if it hits against a barricade, the task is to find out what kind of barricade it is as fast as possible. That means that the robot has to recognise the barricade. For example if the robot hits against a chair, the task is to find out that it is a chair.
The problems navigation and recognition are very close.

2.2.4 Possible extensions of the focused mobile robot

A robot that can navigate and recognise on its own is very well but mostly useless for humans. Because of that one future task of our robot could be to pick up something and move it to an other place. But how to manage this task and the problems of it will not be described in this Final Project. Sometimes I will use the pick-up task to illustrate an example.

2.2.5 Unsolved problems in the mobile robots world

As described above the task of the mobile robot in this final project is to navigate and to recognise. These tasks are exactly the first two unsolved problems in the mobile robot's world described in [Jones and Flynn, 1993, pp 271-273].

Further described unsolved problems are learning, gnat robots and co-operation.

Learning is one problem that could be solved from the blackboard control architecture and the specialists. Every task, every solution and every result of this solution (represented as one of the next tasks) is written on the blackboard. Learning specialists only have to read the communication on the blackboard. After a while learning specialists will recognise if a robot makes the same mistake all the time. For example if the robot wants to move straight and because of inexact mechanics it turns a little to the left all the time, learning specialist will find out that the robot is never at that position where expected. Next time if the robot wants to move straight the learning specialist would give the command to turn a little to the right to correct the mechanical error.

To build gnat robots means to build smaller robots. This is a hardware problem and can not be solved from the blackboard.

One robot can be very powerful. But compared with one robot a swarm of robots might be much more powerful. A swarm of robots could move heavy things, could take care of big tasks like cleaning a house or could do other sophisticated things. Supposed that the robots are able to communicate and to co-operate.

Communication is a hardware problem but not a big one. It should be possible to build robots that are able to communicate. The co-operation is a software problem that means a problem of the blackboard.

I think that robots with blackboards are suitable for this task. A blackboard is a tool to co-ordinate a lot of specialists. It should be possible to build a blackboard that co-ordinate a lot of robots. It would be a interesting task to check and refine it.

2.3 The interface between blackboard and mobile robot

In this chapter I will describe the Interface between the blackboard and the mobile robot. Interface does not mean the hardware interface. It means that I will describe the architecture between the blackboard and the mobile robot. The idea of this architecture is from [Hayes-Roth, Barbara; et al, 1995].

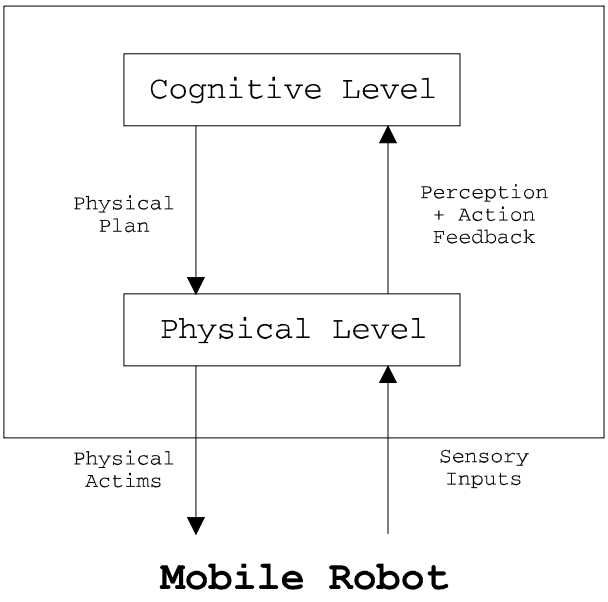

Figure 2.1

The architecture currently has two layers, or levels to control concurrent physical and cognitive behaviours. The physical level manages perception in the external environment. The cognitive level has the task to offer more abstract reasoning activities such as planning and problem solving. Information between the two levels flow bi-directional. The result of each level can influence the behaviours of the other level.

Each level of the architecture has isomorphic internal structure. The structure of the cognitive level is a blackboard. A much simpler implementation of the same basic model is used for the physical level.

The focus of this final project will be the cognitive level.

Planning is the first stage of a two stage problem solving process. The second stage is to control the success of the current movement. It embodies monitoring and guiding the execution of the plan. This final project focuses on the planning stage.

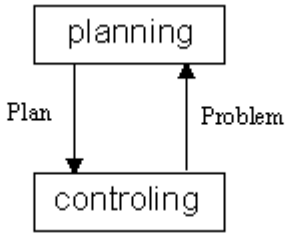

Figure 2.2

2.4 Safe and unsafe Areas

One of my ideas in this final project is to split the room in safe an unsafe areas. In this chapter I will introduce this term.

In our environment exist items that move very often. For example chairs probably will never be at the same place as a day before (if the room is in use).

Imagine now a room with a table and chairs.
Probably the first idea is a table with chairs very close to the table. The reason of this picture is that chairs are usually very close to tables. This is a kind of knowledge. We assume that chairs are very close to tables.

If a mobile robot moves in that room it seems to be a good idea **not** to move in the near of the table. The planner does not know the position of the chairs but it can assume that they will be in the area around the table.

The area around the table is unsafe because the possibility to hit against something (a chair or someone sitting on the chair) is very high. In other areas of the room the possibility to hit against something is very low. This areas will be called safe.

2.5 Coarse grained or fine grained

Another term used in this report is coarse grained and fine grained.

Imagine a map of a room that consists of a lot of frames. Every frame describe an area in the room. The smaller the frames are, the more frames are used to describe the room, the more fine grained is the map. The bigger the frames are, the bigger is the described area of every frame, the more coarse grained is the map.

3. The data structure of the blackboard

The aim of this chapter is to describe the data structure of the blackboard. This chapter has two parts. The first part is a more general description of the information levels. It is the essence of [Hayes-Roth, B., Hayes-Roth, F., 1979a and Hayes-Roth B., et al, 1979b]
The second part of this chapter is an example of my own to illustrate how the different conceptual planes work together.
The example focuses the mobile robot world as described in chapter 2.2 **Mobile robot**.

Obviously the blackboard has to be organised. If the blackboard is not organised, specialists would write their messages on the blackboard and every specialist would read every message, if it is important for it or not.

Specialists are a lot of small programs which read a special problem from the blackboard and give a solution, the next step or an idea how to solve the problem (see also Chapter **4 Specialist**). Specialists write their problem-solving idea on the blackboard and the best of all ideas will be chosen.

On the blackboard problems will be split in different categories. Information and requests have to be structured on the blackboard. Therefore the blackboard is partitioned into five conceptual planes: meta-plan, executive, plan-abstraction, knowledge-base and plan. Each plane embodies different categories of decisions and contains several levels of abstraction.

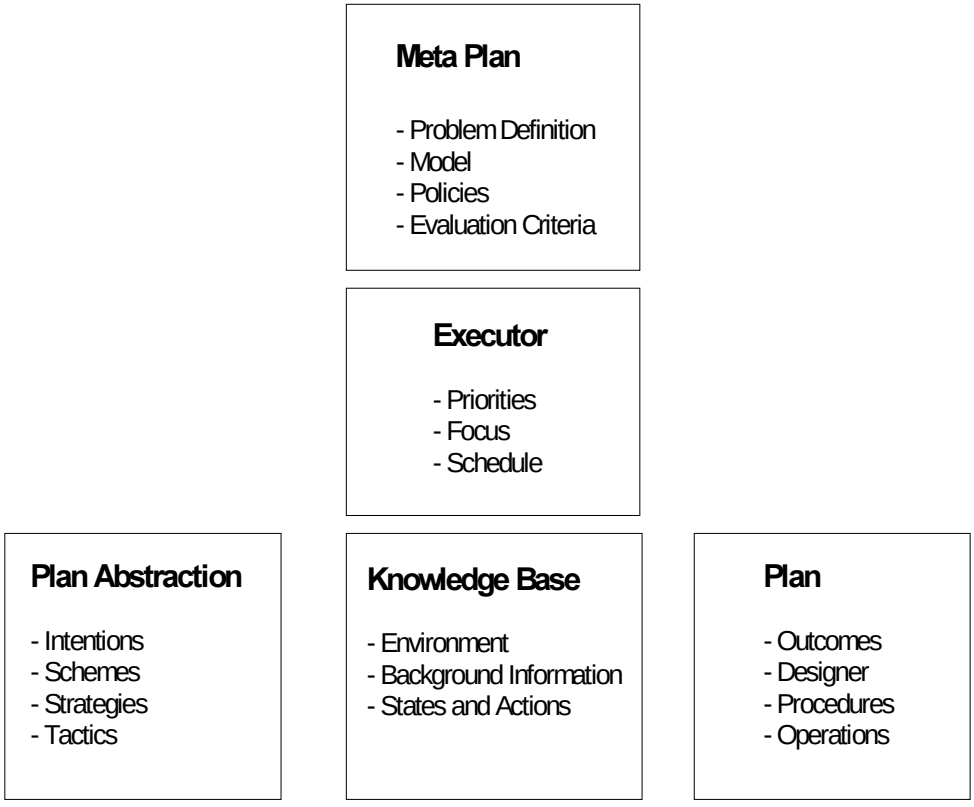

Figure 3.1

Note: The name of the levels of the knowledge-base depends on the problem domain.

3.1 Meta plan

Meta-plan and the decisions in this plane indicate what the planner intends to do during the planning process. The meta-plan plane has four levels*: problem definition, problem solving model (in Figure 3.1 model), policies and evaluation criteria*. These planes are not strongly hierarchical.

3.1.1 Problem definition

The problem-definition contains the planners own formulation of the task. This includes

- description of the goal
- available resource
- possible actions
- constrains

The description of a goal can be something like *leave the room, move to chair_1* or *explore the room*. To refine the description could be phrases like *as fast as possible* or *.... beside explore the environment*.

3.1.2 Problem solving model

The problem-solving-model is a convenient model of the problem definition.

The problem solving model indicates how the planner intends to represent the problem symbolically and generate potential solutions. [Hayes-Roth, B., et al; 1979b]

To find the problem solving model for a given problem means to classify it. For example, the problem *leave the room by the shortest route* is an instance of the travelling salesman problem. If the problem definition changes to *leave the room and beside explore the room* the problem solving model should be the opposite of the travelling salesman problem. In this case the task of the planner is to find the longest route travelling on the same path at least once but twice or more times as little as possible.

In computer science there exists a lot of general problem solving models such as ‘divide and conquer’ or a model to solve the 'travelling salesman problem'.

The problem solving model is essential for the further planning process. It has influence on executive decisions because some problems are for example ‘bottom up’ and other problems are ‘top down’.

3.1.3 Policy

The planner’s policy is less abstract than the problem solving model. It specifies global constrains and desirable features for the developing plan. Often the source of the planners policy is the problem solving model. The travelling salesman problem for example implies a route efficiency policy (shortest route). Another policy in the exploring room example might be to find the longest route travelling on every path at least once but twice or more times as little as possible (longest route).

3.1.4 Evaluation criteria

The lowest level of the meta-plan are the evaluation-criteria. In this level are fixed how the planner intends to evaluate prospective plans. For example, if there is a lot of change in the room it could be a good idea to move very carefully. A little bit more abstractly it means that it is a good idea to calculate what might go wrong.

3.2 Executor

The executive plane (Executor) and the specialists that belong to the executive plane, determine features of the planning process. For example, in this level will be fixed whether the abstract planning specialists or the concrete planning specialists will work at first.
The executive level is absolute hierarchically; every level refines the level above.

3.2.1 Priorities

The priorities level is the most abstract level of the executive plane.

... priority decisions establish principles for allocating cognitive resources during the entire planning process. [Hayes-Roth, B., et al; 1979b]

For example in the mobile robot's world, with the task to move some items, it might be a good idea to define the order of the movements before working out how to put the items on the robot.

3.2.2 Focus

In the focus level will be refined the decisions made in the priorities level and translate it in the terminology the blackboard. The focus contains the current level of planning.
For example in a top down approach the focus will be on the plan-abstraction intentions and plan outcomes at first (see Chapter **3.3 Plan abstraction** and **3.5 Plan**). After that the focus will be on the lower levels.
That means that the focus will change during the planning process.

3.2.3 Schedule

The less abstract level in the executive plane is the schedule level. In this level appears the specialist, which makes a contribution to the plan.

A problem is, if more than one specialist can make a contribution to the plan. This decision can not be taken by the blackboard. Though the blackboard has its own scheduler this task is part of the planning process, not of the communication process. The blackboard is only a communication tool.
This decision should be taken by own specialists. In Chapter **4.6.1 One or several approaches** I will present a solution of this problem.

3.3 Plan abstraction

In the plan-abstraction attributes of potential plans can be characterised. In this plane the kind of action will be described without specifying the action itself. The plan-abstraction is very close to the goal but refines it.
The plan-abstraction plane contains four levels. The highest level is the most abstract level and every level below refines the level above.

The four levels of the plan-abstraction, the knowledge-base and the plan plane have the same grade of abstraction. That means that the specialist reads the intentions of the plan abstraction plane and uses the environment information of the knowledge base to reach the outcome in the plan plane (see Figure 3.1).

The intention level is the most abstract level. If a specialist can not solve a problem direct, it will refine (split) the problem. This refined problems will appear in the scheme level. Very often these problems are very general or they are background that it seems strange to say that they are less abstract. For example the intention ’park the robot’ contains the schemes ’Where can the robot park?’ and ’Which of the given parking positions can be reached at first?’.

The strategies level describe how the robot has to move. For example 'turn left 90°'. The tactic level will be used if problems with the strategy appear. For example if the robot is not able to turn left 90° the problem will appear in the tactic level and the solution could be 'turn right 270°'.

In chapter **3.6 Example** I will present an example which demonstrates how to work with the plan abstraction.

3.4 Knowledge base

The knowledge base plane also contains four level of abstraction. Each level corresponds to the same level of the plan abstraction and the plan itself.

The knowledge base plane is not the knowledge base itself. The task of the knowledge base plane is to ensure the communication between the specialists that know the structure of the knowledge base (and write on the knowledge base plane) and the planning specialists that need the knowledge from the knowledge base.

The knowledge base itself must not have the same structure and indeed it has not (see Chapter 5 **Knowledge representation**).

In chapter **3.6 Example** I will present an example which demonstrates how to work with the knowledge base plane.

3.5 Plan

The plan plane contains four corresponding levels to the plan abstraction plane and the knowledge base plane. As in the plan abstraction plane and the knowledge base plane the highest level is the most abstract level.

In chapter **3.6 Example** I will present an example which demonstrates how to work with the plan plane.

3.6 Example

The example is very easy. However, it demonstrates how the levels and planes work together. The initial state of the information levels is shown in Figure 3.2. Each step represents the invocation and execution of a specialist.
The problem definition is given as *'leave the room as fast as possible'*. In the knowledge base exists a map of the room.

In the text below I will underline the names of the levels to make it more readable.

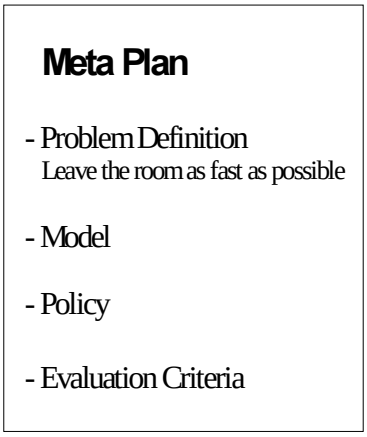

Figure 3.2

3.6.1 Meta plan (problem solving model)

In the beginning we have a problem definition, a map and the knowledge bases. The problem definition includes two facts:

- leave the room
- be as fast as possible

The specialist who executes the first step should recognise that the phrase '*leave the room*' indicates a 'route finding problem'.

More interesting is the phrase '*as fast as possible*'. The fastest route must not be the shortest. Sometimes a longer route is faster than a shorter route because of the possibility to drive faster. The shorter route could be slower because of a lot of corners and barricades.

The decision of the best problem solving model depends on the information in the map. If the map includes information about how fast the robot can drive on different routes, the best problem solving model is a model to solve a ‘time limited scheduling problem’.

The model to solve a ‘time limited scheduling problem’ and the model to solve a ‘travelling salesman problem’ can be similar. If the 'time limited scheduling problem' joins the 'route finding problem' the difference between the models is that the ‘travelling salesman problem’ focuses on the length of the route, and the ‘time limited scheduling problem’ focuses on the time that the robot needs to move a route. If the 'time limited scheduling problem' joins, for example, the 'exploring problem' it is not similar to the ‘travelling salesman problem’.

In this example we have a 'time limited scheduling problem' and a 'route finding problem'. If the information about how long the robot needs to travel from one point to another point is not in the map, the specialist should choose the model of the ‘travelling salesman problem’ as the best problem solving model. In this case we assume the shortest route is the fastest route. This problem solving model should be the best approximation on the real problem.

In this example the map includes the time information and the difference between each point.

The chosen problem solving models are ‘route finding problem model’ and ‘time limited scheduling problem model’. When the first specialist has made its choice, there exists a problem definition and a problem solving model (Figure 3.3).

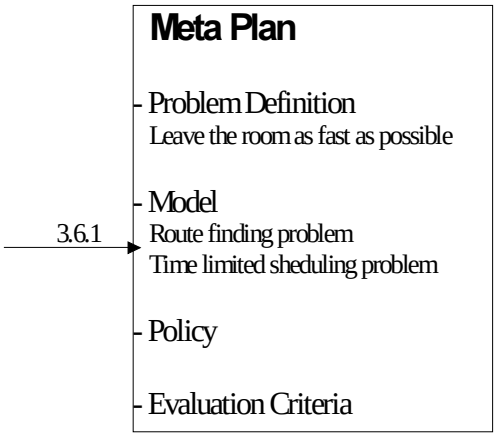

Figure 3.3

3.6.2 Meta plan (policies)

The next step is to find a good policy. In this particular case this job is very easy.
The model 'time limited scheduling problem' embodies that the policy has to be fast. The model 'route finding problem' embodies that the policy must be a route policy (Figure 3.4).

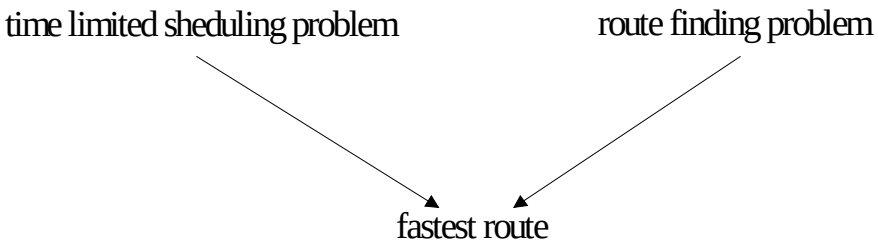

Figure 3.4

3.6.3 Executor (priorities)

Now we have a problem definition, a problem solving model and a policy. This point is very interesting because it is possible that now different specialists offer different possibilities. One possibility could be to work out the evaluation criteria. Another possibility could be to work out the executor priorities.

Of course the first step could have involved more than one specialist, too. But to find a problem solving model in the first step was obvious.

In this particular situation the next step is not obvious. The next specialist can be either a specialist that offers meta-plan evaluation criteria or a specialist that offers executor priorities. The specialist with the highest rate wins. In our example the specialist that offers to work out the executor priorities wins.

This specialist knows that the best possibility to solve a ‘route finding problem’ is first to decide ‘which route’ and at second to decide ‘how to move on this route’. This approach is top down. To decide the route is the more abstract part of the approach; 'how to move on this route' refines it.

3.6.4 Executor (focus)

As described in Chapter **3.2.2 Focus** the focus of the planner will be refined at this level using the levels of abstraction with which the planner intends to work at first.

The executor priorities motivate a decision to focus on the plan-abstraction intentions and the plan outcomes at first. As described above (chapter 3.6.3 Executor (priorities)) the chosen approach is top down. As can be seen in Figure 3.1 the chosen levels are the most abstract levels.

This is the first time the blackboard terminology is used. In all steps above the specialists refine the given problem in a common way. At this point appears the first translation of how the blackboard planner intends to solve the problem.

The current state of the blackboard is shown in Figure 3.5

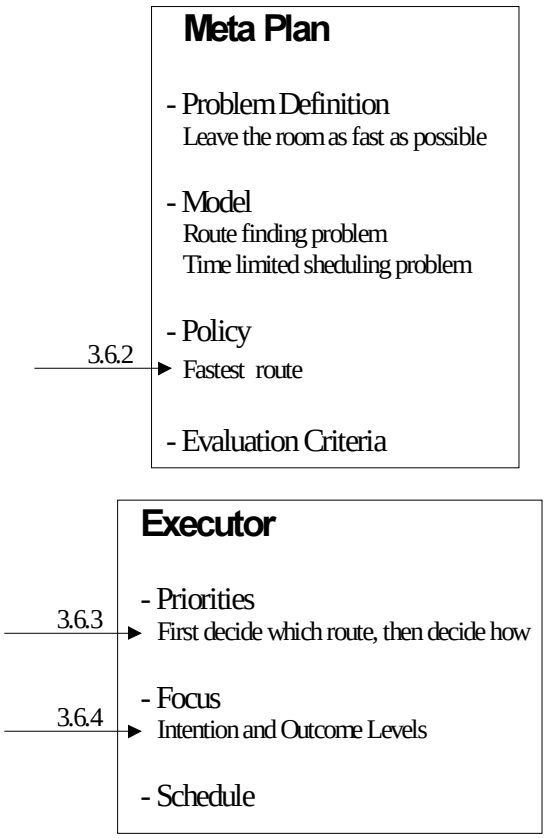

Figure 3.5

3.6.5 Plan-abstraction (intentions)

All steps that the specialists have done until now have had the goal of figuring out how to plan and how the planner should behave. The planner knows now which is the best problem solving model and on which levels it should have a focus at first. In this moment starts the planning process itself.

The first focus is on plan-abstraction intentions and on plan outcomes. That means that we need a specialist that can figure out a plan abstraction at first.
The meta-plan policy embodies the plan-abstraction. From the policy 'find the fastest route' the specialist extracts three abstract plans.

- Where is the robot?
- Where does the robot want to go?
- What is the fastest route between these points

The first plan-abstraction is to find out where the robot is. If the planner does not know the position of the robot it can not plan the route to the goal position. The next plan-abstraction is to figure out where does the robot want to go. If the goal position is well defined this point will not be a problem. If the goal position is not well defined, the planner has to figure it out as we will se in this example. The last plan-abstraction is to find the fastest route between the start position and the goal position.

The specialist writes the first plan-abstraction intentions on the blackboard. The state of the blackboard is shown in Figure 3.6.

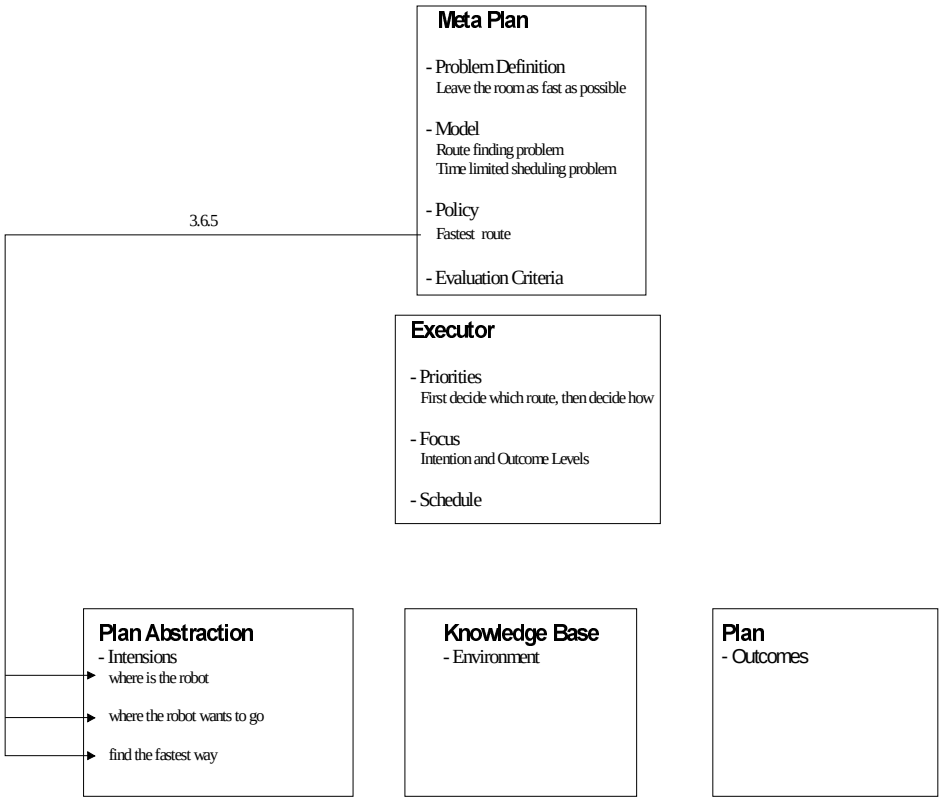

Figure 3.6

3.6.6 Where is the robot?

The next specialist will pick up the first plan abstraction. The first plan abstraction is to figure out the current position of the robot.

At this state exists two possibilities. The first possibility is that the information about the position of the robot is in the knowledge base. In this case the specialist will read this information from the knowledge base and write it on the blackboard. The second possibility is that the information about the position of the mobile robot is not in the knowledge base. One way to solve this problem would be to write the plan 'figure out the position of the robot' on the plan outcomes.

In our example the current position of the robot is stored in the knowledge base. The specialist finds this information in the knowledge base and writes it on the knowledge base plane of the blackboard. The current position of the robot is corner-3.

3.6.7 Where does the robot want to go? (Part 1)

The first plan abstraction is satisfied. The next plan abstraction 'Where does the robot want to go?' appears now. It is more difficult than the first one. The current specialist has a look in the problem definition to find the goal position, and finds the statement 'leave the room'. The position 'leave the room' does not exist in the knowledge base.

Because in the knowledge base the position 'leave the room' does not exist, the current specialist creates a new abstract plan in the scheme level. The plan abstraction to solve this problem is 'How can the robot leave the room?'

This example shows the difference between the intention level and the scheme level. The abstract plans in the intention level are caused direct from the main problem. The abstract plans in the scheme level are caused from sub-tasks. The plans in the scheme level are more general. For instance the question 'How can the robot leave the room?' can be a request from several tasks.

This situation shows that if a specialist can not solve a problem, one suitable way is to split the problem. The task is 'leave the room'. The split tasks are 'How can the robot leave the room' and 'Which is the better place?'. We need the task 'Which is the better place?' only if the first task has more than one answer.

Another general approach is to try to solve this problem at a more abstract level. But this particular problem can not be solved at a more abstract level.

This is one way we assume that humans work. Humans do not work strictly top down or bottom up. Either they try to solve a problem at a more abstract level or they try to refine (split) a problem. But the whole planning process is a mixture of both.

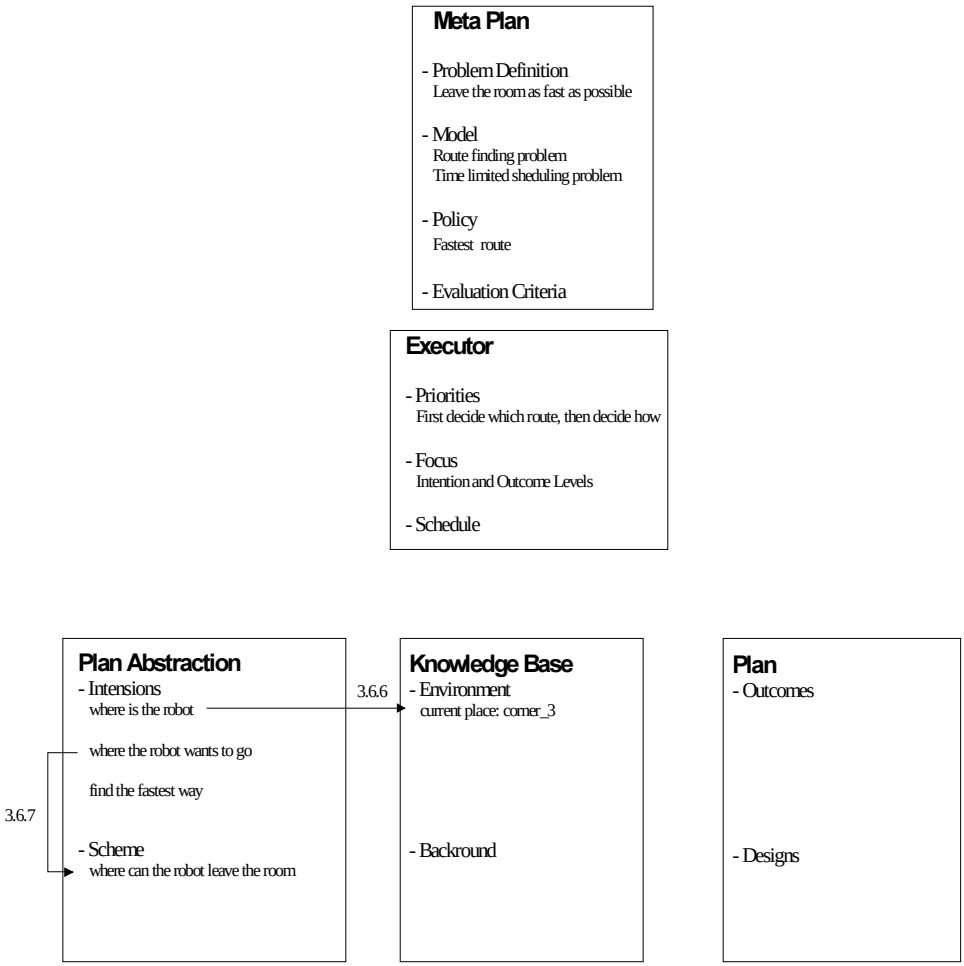

Figure 3.7

3.6.8 How can the robot leave the room?

The plan abstraction 'How can the robot leave the room?' motivates another specialist to search in the knowledge base. Very often attributes characterise positions. Other attributes are, for example, 'parking place' or 'refill the engine of the robot'. In the knowledge base we need this kind of information.

In our example the specialist gets the information that the robot can leave the room at two positions: door-1 and door-2. The specialist writes this information on the knowledge base plane of the blackboard.

If this information about how to leave the room was not in the knowledge base the specialist would create a plan and write it on the blackboard. The plan would be 'Explore the room and find out how to leave the room'.

3.6.9 Where does the robot want to go? (Part 2)

Now the specialists can try again to solve the problem ’where does the robot want to go?’. The situation has changed. Whereas at **3.6.7 Where does the robot want to go? (Part 1)** the problem was that there was no goal state the problem now is that there are two goal states. On the blackboard are the two possible positions, door-1 and door-2, to leave the room.

Because the specialist can not decide between these two possibilities, it creates a new plan-abstraction scheme. The new plan-abstraction scheme is ’Which of the given places can be reached faster?’.

To create this scheme the specialist needs two pieces of information. The first piece of information is obviously that there are two existing goal states. The second piece of information is how the robot should reach the goal state. That means for example that the robot should be as fast as possible or the route should as short as possible. The specialist finds this information at the meta-plan policy.

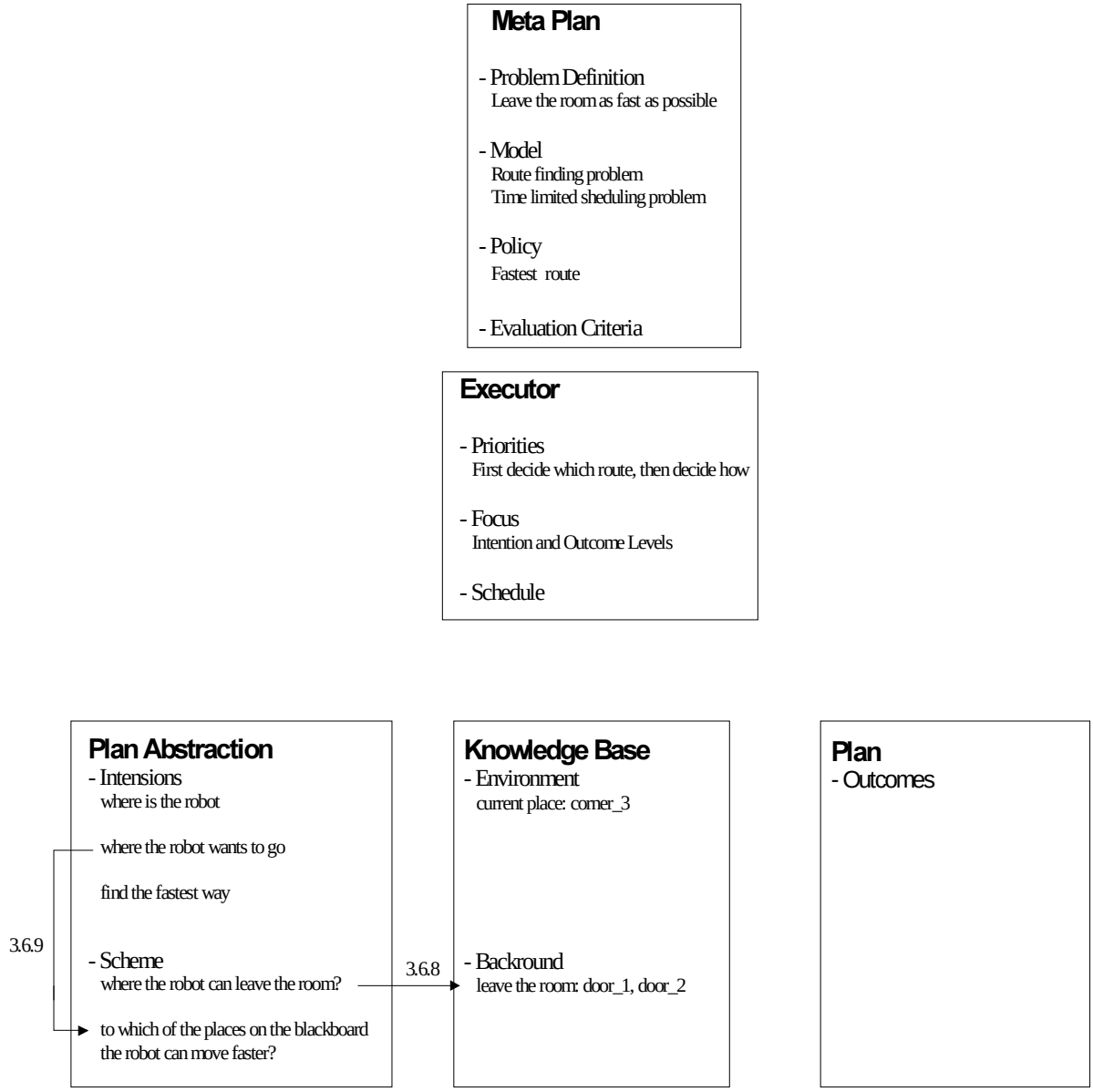

Figure 3.8

3.6.10 Which of the given places can be reached faster?

This task is a real artificial intelligence task. It embodies three parts.

- Find the fastest route to door-1
- Find the fastest route to door-2
- Compare them

To shorten this example and focus on the interesting points I will not refine these three tasks. It is obvious how it works.

In our example the route to door-2 is the fastest route. The specialist writes this result on the blackboard.

3.6.11 Where does the robot want to go? (Part 3)

Now the specialist which tries to solve the problem 'Where does the robot want to go?' begins to work again. All opportunities are so much refined that the specialist can decide easily which is the goal state. It takes the background information from the knowledge base plane and writes it as the goal state on the blackboard.

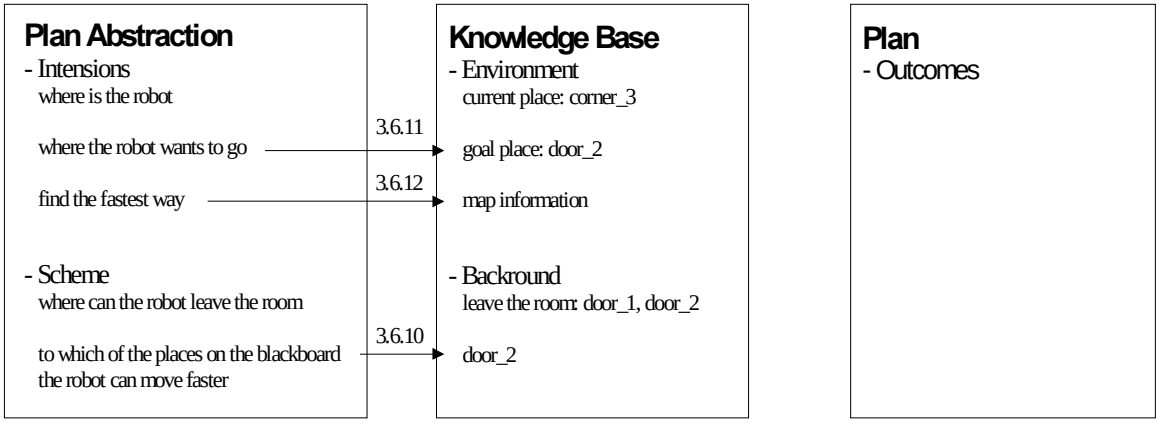

Figure 3.9

3.6.12 Map information

At this moment the current position and the goal position of the robot are on the blackboard. The last piece of information that a planner needs for its planning task is information about the environment; a suitable map.

I will refine this task a little to show how it works. One specialist reads from the knowledge base and decides which information about the environment is suitable for the planning task. Obviously the planner does not need three-dimensional information about the room. All route planning algorithms are use two-dimensional maps. If there is only three-dimensional information in the knowledge base, it has to be converted (see Chapter **5.5 Convert the information into a two dimensional map**). The next decision that the specialist will make is how grained the information will be. If the map is to coarse grained the result will not exact enough. If the map is too fine grained, the planning algorithm will need too much time to find the fastest route.

However, the specialist has to decide all these things and after it has to write the map information on the knowledge base plane.

3.6.13 Find the fastest route

At this moment the necessary information for the task ’find the fastest route’, the current position, the goal position and the map, are on the blackboard. A specialist with a typical artificial intelligence algorithm, like shortest path algorithm, appears now. Of course the algorithm will use the information of how much time the robot will need on different routes. That means that the algorithm will give the fastest and not the shortest route.

The specialist will find the fastest route and write it on the blackboard as plan outcome and as plan-abstraction strategies.
In the first moment it seems strange that one specialist writes the plan outcome and plan-abstraction strategies. In this case it makes sense. The plan outcome:

- move from corner-3 to door-2

But the specialist figured out more than this information. It figured out the fastest route to do this. This information is the result of the first part of the planning process, and at the same time it is the start information of the second part of the planning process. The plan-abstraction strategies:

- move one meter straight
- turn left 90°
- move 50 cm straight

Figure 3.10 shows the current situation on the whole blackboard.

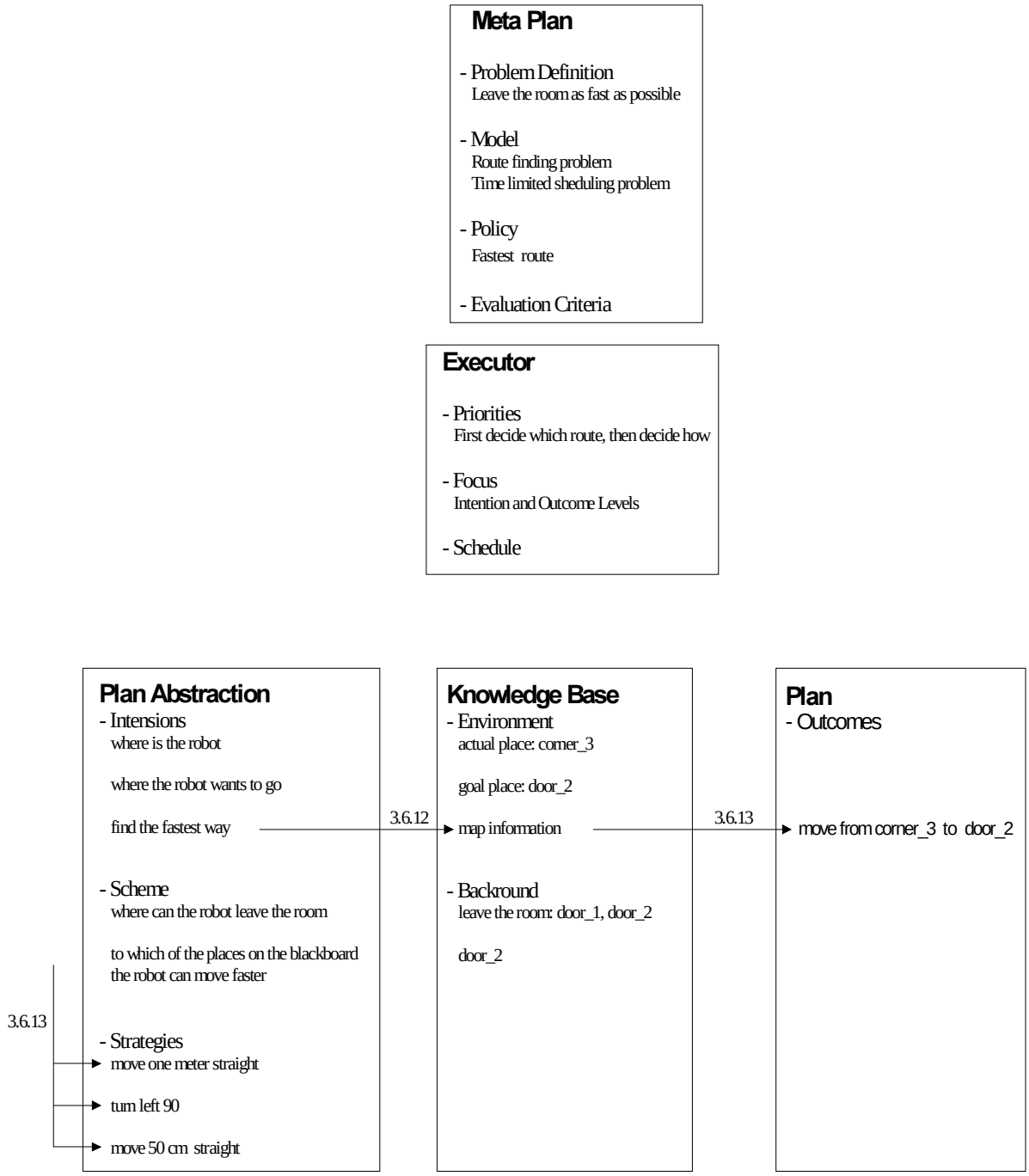

Figure 3.10

3.6.14 Change the focus

The executive priorities embody two parts. In the first part the planner has to figure out the route, in the second part the planner has to figure out how to move.
The first part is finished now. The planner has stored the fastest route in the plan-abstraction strategies. The next task of the planner is to figure out how to move. Therefor the planner changes the focus. While the first part was very abstract the second part is very concrete.

The current levels after changing are plan-abstraction strategies and plan procedures.

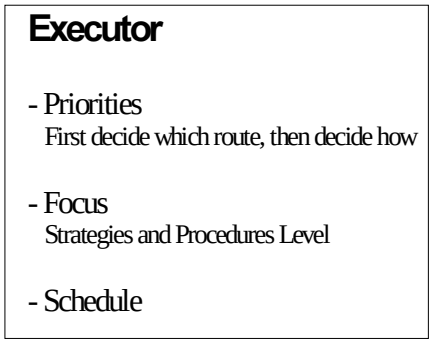

Figure 3.11

3.6.15 Move one meter in a straight direction

The first plan-abstraction strategy is to ’move one meter straight’. Therefore the specialist which will create the plan needs some information about the environment. Important questions appear like:
’Is it possible to move as fast as possible or does the robot have to be careful because of some movable barricades?’. Another question could be: ’Which material is the floor produced from? Is it carpet or linoleum?’
In this example we will only have a look at the first question. As described it is possible to characterise the areas in the room, in safe and unsafe areas. If the robot is in a safe area it can move as fast as possible because the possibility to hit against a barricade is low. If the robot is in an unsafe area it has to be careful.

In this case the specialist finds the information that there is no barricade and the mobile robot can move as fast as possible (no barricade). The specialist writes this information on the blackboard (knowledge base plane).

3.6.16 From plan-abstraction and knowledge base to the plan

With the plan-abstraction strategy ’move one meter straight’ and the knowledge states and action ’no barricade’ it is not difficult to create the plan procedure ’move one meter straight as fast as possible’.

This is the first time in this example that a concrete step, a real order to move appears. To refine this plan is not task of the cognitive level, that means it is not task of this blackboard. This step will be sent to the physical level. The physical level knows details about the motor and how to transmit the commands to the robot. But if it leaves the cognitive level it leaves this final project.

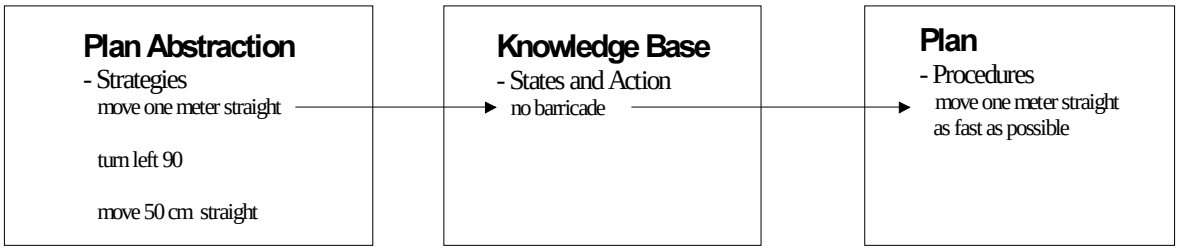

Figure 3.12

3.6.17 Turn left 90°

The next plan-abstraction strategy is to turn left. Therefore the planner needs information about how much space the robot needs to turn left 90° and whether there is enough space or not. Different robots need different space to turn. It depends on how the robot is built.

In this example enough space exists to turn. The specialist finds this information in the knowledge base and writes it on the knowledge base plane.

3.6.18 Plan procedure

It is obvious what happens now. One specialist creates the plan procedure 'turn left 90°' from the plan-abstraction strategy and the knowledge-base states and action.

3.6.19 Move 50 cm

The last plan abstraction strategy is that the robot has to move 50 cm. Because the robot is now very close to the door it has to be very careful. The door can be open, closed or something between this two states. That means that the robot could hit against the door.

The specialist will find this information in the knowledge base. It writes the information on the blackboard.

3.6.20 Plan procedure

The specialist which create the last plan procedure reads the hint that the area where the robot moves now is a very unsafe area. The result is that the specialist creates a plan procedure that takes care about this situation.

Figure 3.13 shows the final state on the blackboard.

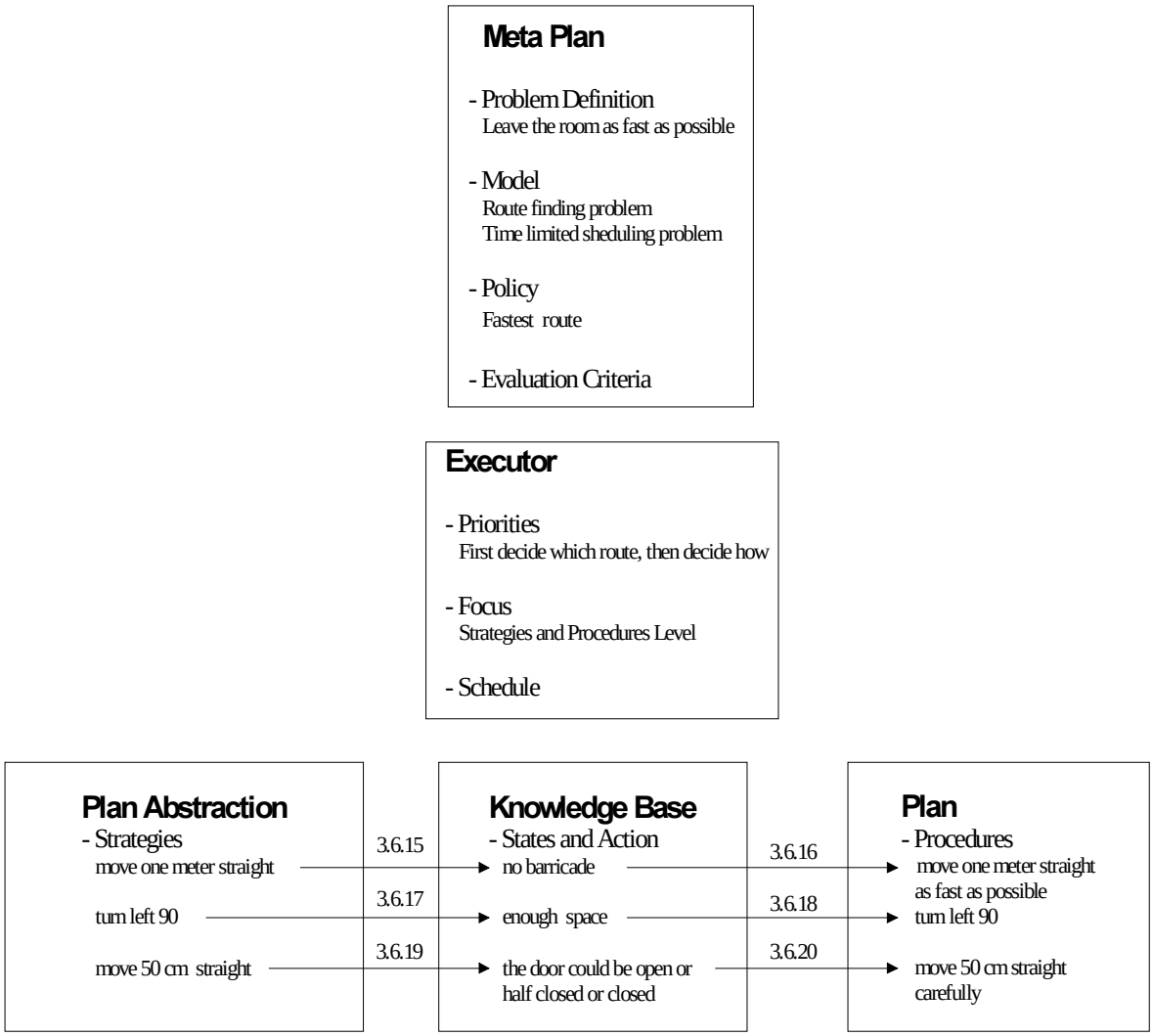

Figure 3.13

3.7 Summary

The example shows how the blackboard works in general and especially in the domain of mobile robots. It seems that the blackboard and the given layers are suitable.

The influence of the knowledge base on the behaviour of the robot is impressive. The fact that the robot in our example sometimes moves fast and sometimes moves slow (carefully) shows that the information in the knowledge base changes the behaviour of the robot.

The reason of sophisticated behaviour are various specialists. The blackboard offers the possibility that different actions can be planed from different specialists. For instance the task to move one meter straight contains other problems than the task to turn left. Even the task to move might change in different areas in the room and requires different specialists.

The given structure of the blackboard offers the possibility to behave different. The conceptual layers and planes order the information and requests in a very efficient way.

4. Specialist

One of the most important objects of the blackboard control architecture are the specialists. Specialists are a lot of small programs which read information from the blackboard and give a solution, the next step, an idea how to solve the problem or only new information.

A specialist is a condition action pair. The condition component describes in which situations or states the specialist can contribute to the plan. The action component describes the behaviour of the specialist.

In this chapter I will describe some specialists of the mobile robot task. I will describe the specialists up to point when the planning process starts very refined (from Administration to Policy Analyst). It is essential to understand this part. The planning process itself embodies so many possibilities that I will describe only a few specialist to give an idea how it works..

In the focused mobile robot world we can split the problems in two main groups. The first group is if the robot has to move from one position to one or more positions. The other group is if the robot has to explore the room. It seems to be a good idea to split the specialists too.

The exploring problem embodies a special difficulty. It is not possible that the planner plans the task and after the planning process the robot will execute this plan. Exploring the environment embodies a toggle between planning and controlling. I will not describe the specialists that are involved in this toggle. Nevertheless, in front of the exploring task the planner has to plan. I will describe the planning process of the exploring task and the involved specialists.

In the chosen information level structure we need different types of specialists. Figure 4.1 is adapted from [Hayes-Roth, Barbara; Hayes-Roth, Frederick, 1979]. It shows the described specialists.

It is very helpful to have a look on Figure 4.1 during the reading of the whole chapter

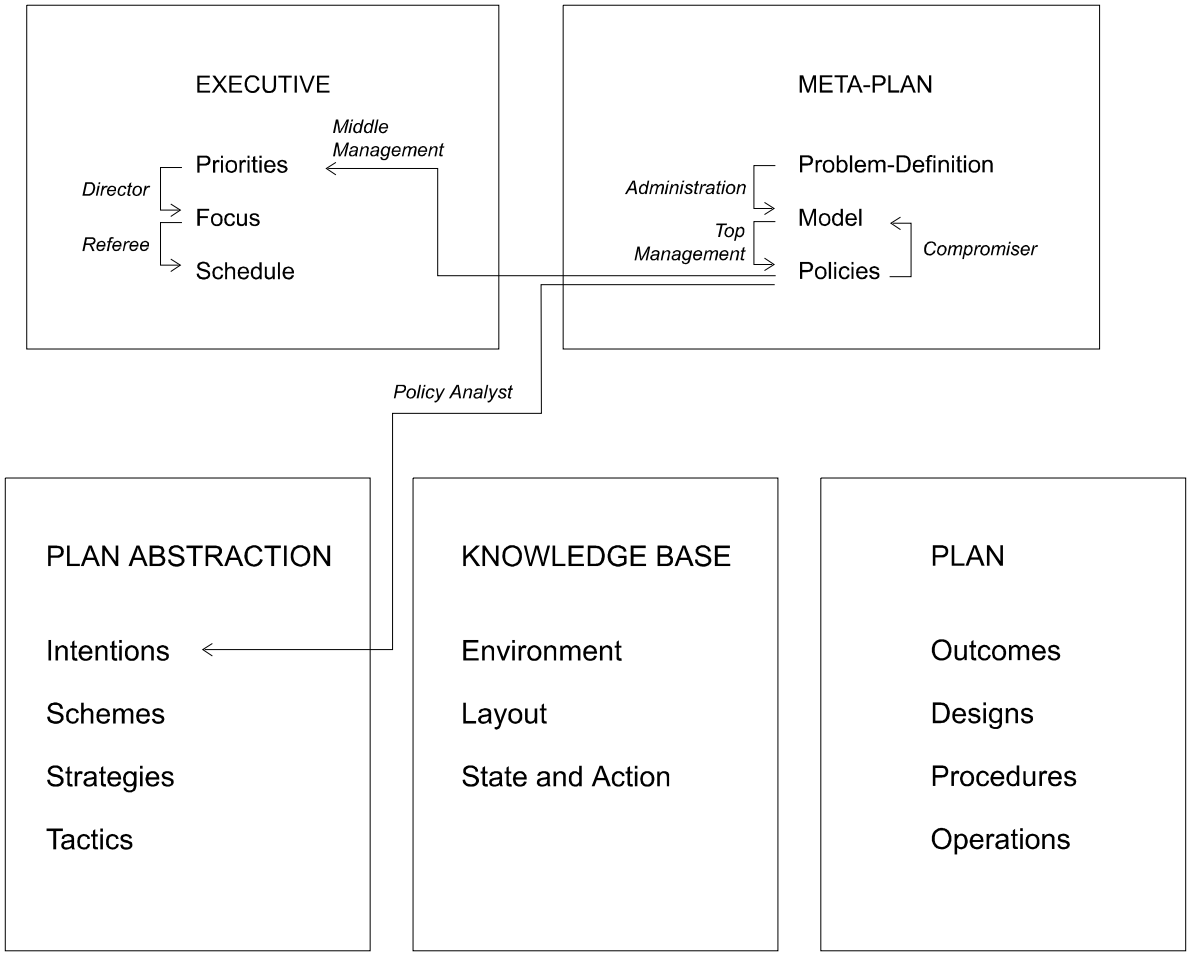

Figure 4.1

4.1 Administration

If we start a planning process there is normally only a problem definition. The first task of the planner is to find the best problem solving model for the given problem definition. In Figure 4.1 this process is called **Administration**.

4.1.1 The administration specialists for moving

If the robot has to move from one point to another point the planner has to find the best path. In our particular mobile robots world the problem definition can have three different attributes (and of course the opposite of this attribute).

- use the fastest way use the way that need the most time
- use the shortest way use the longest way
- use the safest way use the most unsafe way

The fastest and the shortest way are not at all the same. Sometimes a longer way is faster than a shorter way because of the possibility to drive faster. The shorter way could be slower because of a lot of corners and barricades.

In the focused world are safe areas and unsafe areas. Some barricades will never change. A wall will always be at the same place and a door will not change its position. Some barricades probably will not change. A cupboard for example does not change the position very often. At least there are a lot of things that change their position very often. For example a chair will never be at the same position after somebody used it.

In safe areas the possibility of changed barricades is very low. For example in a room with a table and chairs the place under the table is very safe because chairs are to high to stand under a table. The area around the table is obviously very unsafe.

If the robot has to find a safe route, the planner has to find a way with less areas with unexpected barricades.

4.1.1.1 Condition and Action

As described above a specialist is a condition action pair. Because of splitting the problems of the mobile robot into two groups the first condition and action is obvious.

Condition	Action
• Problem Definition: Move from one to one or various positions • No existing Problem Solving Model (Meta Plan)	Write Problem Solving Model (Meta Plan): Route finding problem

As described above the Route Finding Problem could have three different instances; fastest way, shortest way and safest way. This three instances are in the table below.

In the table below it is noticeable that if the robot has to move to various places, the problem is near to the travelling salesman problem. Only the instance is changeable. It seems to be a good idea to develop a general solution for the travelling salesman problem with the possibility of variable instances.

This could equally solve the problem of the opposite problem definition. If the blackboard could say that the algorithm has not to use min but max., the opposite problem definition is solved, too.

Condition	Action
• Problem Definition: Fastest way • Move to one place • Problem Solving Model: Route Finding Problem • In the map are path-time information	Write Problem Solving Model: Time limited scheduling problem
• Problem Definition: Fastest way • Move to different places • Problem Solving Model: Route Finding Problem • In the map are path-time information	Write Problem Solving Model: Time limited travelling salesman problem
• Problem Definition: Shortest way	Write Problem Solving Model:

• Move to one place • Problem Solving Model: Route Finding Problem • In the map are path-distance information	Shortest path problem
• Problem Definition: Shortest way • Move to different places • Problem Solving Model: Route Finding Problem • In the map are path-distance information	Write Problem Solving Model: Shortest path travelling salesman problem
• Problem Definition: Safest way • Move to one place • Problem Solving Model: Route Finding Problem • In the map are path-safe information	Write Problem Solving Model: Safest path problem
• Problem Definition: Safest way • Move to different places • Problem Solving Model: Route Finding Problem • In the map are path-safe information	Write Problem Solving Model: Safest path travelling salesman problem

4.1.1.2 Missing Information

A very interesting situation is if the blackboard has not access to the required map information. Sometimes it is possible to substitute another problem solving model, from which the needed information is accessible. However, the problem has to be very similar to the substituted problem.

As described the fastest and the shortest way are not at all the same. Nevertheless, it is a good idea if the robot uses the other problem solving model, if the information of the own problem solving model is not accessible. That means that if there is a ‘fastest way problem definition’ and there is no information about how long the robot needs to move from one place to another, we can also use the ‘shortest way problem solving model’. Of course we could also use the time ‘limited scheduling model’ as fallback strategy for the ‘shortest way problem’ (see Figure 4.2 and Figure 4.3).

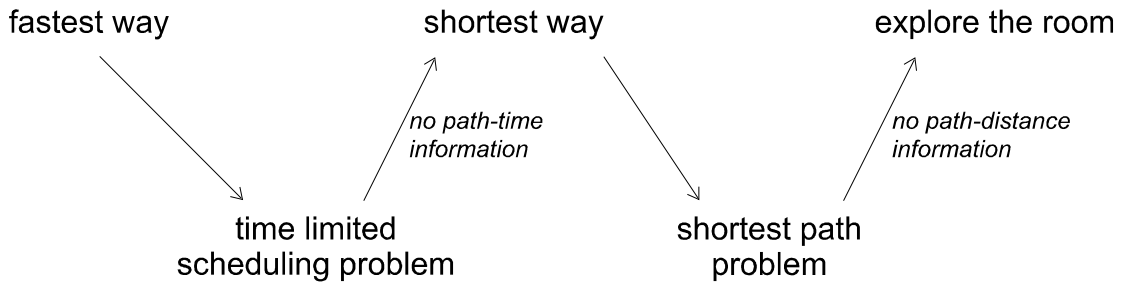

Figure 4.2

As you can see in Figure 4.2 and Figure 4.3 the last fallback strategy is to explore the room.

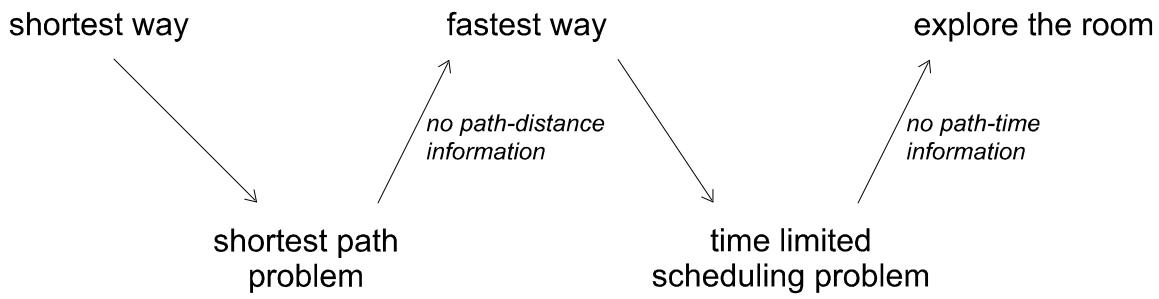

Figure 4.3

In our mobile robot world there is not a similar problem solving model to the safest path problem. The fallback strategy of the safest way problem is simply to explore the room. Of course it could be a good idea to use the fastest or the shortest way. This can change from environment to environment or from room to room. However, we define that in our environment the best fallback strategy is to explore the room.

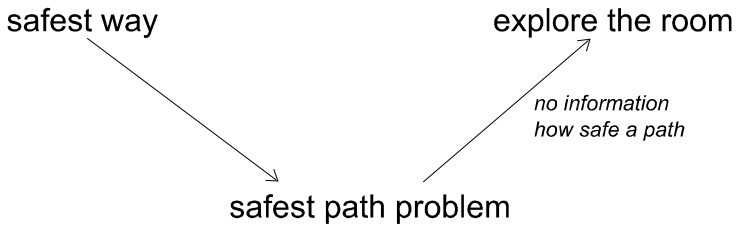

Figure 4.4

4.1.2 Stop the Exploration if has been found the Goal?

How should the robot behave if it has found the goal but has not much information about the environment? Imagine that the robot has to find the door and has not any information about the room. It explores the room and it finds the door very fast. Shall it explore the rest of the room or shall it stop? The reaction of mobile robots in such a situation could be different.

Perhaps it is helpful to have a look at humans behaviour because the reaction of humans in such a situation could be different, too. One approach is to study the problem domain only far so long as it takes to solve the given problem. One example is a man who tries to repair his car. He probably will not investigate how his car works. He probably will only fix the problem and after that he will stop. The other approach is to go on studying the problem domain until you have a complete overview of it, whatever the particular problem might be. One example of this approach is every student. Students try to get a complete overview of a problem domain (hopefully). After their studies they are able to solve problems of this domain very fast.

Humans behaviour depends on the human itself and whether he/she has to solve problems in this domain in the future.

Because our mobile robot exists only to solve very particular problems in a very particular world and this problems will obviously repeat, it is more sophisticated if the robot at first explores the room very well and after that it solves the given problem.

This strategy might be a disaster if the environment is so big that the robot needs a long time to explore it and the given problem was quickly solved. However, in our mobile robot world we assume that the robot moves only in an environment that can be explored in a reasonable time.

Of course the problem definition must be able to change this strategy. If the problem definition says that the robot should stop the exploration if it finds the searched place then it should stop the exploration. This feature is very important. Imagine that the room is burning and the robot gets the message to leave the room as fast as possible. In this situation it is not a good idea to explore the room after the robot finds the exit.
The problem definition should be: Leave the room as fast as possible. Do not explore it.

4.1.3 The administration specialists for exploring

We are still at the point to describe the administration specialists. After the investigation of the move-specialists and the discussion about the behaviour if the robot has found the goal, I will describe the explore-administration-specialists (see Figure 4.1).

The problem of exploring a room can arise in different situations. We distinguish between three cases.

- In the knowledge base there is not any information about the environment
- In the knowledge base there is some information about the environment
- In the knowledge base there is a lot of information about the environment

If there is not any information in the knowledge base, the task is to get an idea of the environment. If there is some information, the task could be to refine this data.

But does it make sense to explore a room where we have a lot of information about? Perhaps it is possible to find a pattern how changes happened. In this case the planner could embedded this information in future decisions. That means that in the case that there exists a lot of information about the room the task might be to explore unsafe areas.

However, one particular information is essential. It is essential that the robot knows the ‘end of the world’. The robot needs the information if it leaves its world and has to stop the exploration. If the robot is in a closed world this information is not necessary. But if the robot moves in a room with doors, it must know at which point it has to stop. Otherwise the mobile robot will not stop to explore the world.

There are a lot of other exploring parameters which have to be in the problem definition or as default parameters in the specialist.

- How long shall the robot explore the room (time or way)?
- Shall the robot explore the room coarse grained or fine grained?
- Shall the robot explore every location in the room?
- Does it matter where the robot stands after exploring the room?

More about the first three topics in the Chapter **4.1.4 When shall the robot stop the exploration?**
These parameters have not influence on the first specialist.

The administration specialist of the exploring task checks only if the problem definition fits with the problem definition "explore the room". In that case and if there does not exists a problem solving model, the problem can be categorised as an "exploring problem".

Condition	Action
• Problem definition: explore the room • No existing Problem Solving Model (Meta Plan)	Write Problem Solving Model (Meta Plan): Exploring problem

4.1.4 When shall the robot stop the exploration?

The question how long the robot shall explore the room has great influence on the exploring strategy. When the robot should stop the exploration can depend on the resource problem or on the problem how good is the information about the environment.

The robot could have limited resources. The exploring problem can be time restricted or way restricted. Time restricted means that the robot is able (or has) to explore the room for a limited time period. Way restricted means that the robot has the resource to move a fixed distance (for example 2 km). The task is to gather as much information as possible with this limited resource. The engine problem (limited battery) is either a time restricted or way restricted problem.

The second possibility is to stop the exploration if the information has a defined quality. The problem of this approach is how to qualify information. One approach is to classify information. Groups of this classification could be:
- How old is the information?
- How important is this information for our task?
- Will this information change very fast? (unsafe area)
- How refined is this information?

The problem of qualifying information depending on the given task is not easy (see also chapter 4.6.2 Value of information). There are a lot of questions and possibilities that can not be investigated in this final project.

4.2 Top management

The top management specialists find the meta plan policies. The policy specifies global constrains and desirable features for the developing plan. This task is very easy in our mobile robot world. Because this task is so easy I will not split it in moving and exploring.

Condition	Action
• Meta Plan Model: Time limited problem • No existing Policy	Write Policy: fastest way
• Meta Plan Model: Shortest path problem • No existing Policy	Write Policy: shortest way
• Meta Plan Model: Safest path problem • No existing Policy	Write Policy: safest way
• Meta Plan Model: Explore the room • No existing Policy	Write Policy: explore room

4.3 Middle management

The middle management specialists look for the best executive priorities. The executive priorities establish principles for allocating cognitive resources during the entire planning process. Because of that we have only to find different executive priorities for the moving task and the exploring task. The principles for allocating recourses during the entire planning process are independent of how the robot has to move from one place to an other place. They depend on whether the robot has to move or to explore the room.

4.3.1 Middle management specialists for moving

As in many other cases we can learn from humans behaviour. How does a human solve the problem if he has to move from one point to an other point?

At first he decides which route he wants to use. For example if he wants to reach the door and a table stands between him and the door; she/he will go around the table.

At first he plans very abstractly. He does not think: turn left, go two steps, turn right, etc. He only thinks that he has to go around the table. He decides to use one particular route, perhaps go around the table on the left side. After he has decided the way in a very abstract form, he moves from one to an other place. He concentrated on what he is currently doing. He knows if he is at the corner that he has to turn right (abstract) and at the corner he moves his body in that way that he turns right (concrete).

As can be seen in this example, if humans move they plan on a very abstract level and after that they work on a very concrete level. The mobile robot should do this, too.
At first it should plan abstract: First decide the route.
Than it should plan concrete: than decide how.

Condition	Action
• Meta Plan Model: Route finding Problem • No existing Executive Priorities	Write Executive Priorities: First decide the route than decide how

4.3.2 Middle management specialists for exploring

After explaining the middle-management-move specialists I will now explain the middle-management specialists for exploring the environment.

The task of exploring a room is not like that of moving from one place to another. The approach for the route finding problem is to find a plan on an abstract level and to concrete it on a very low level. This approach is not suitable for the exploring problem. It is impossible to find a plan on the abstract level that solves the exploring problem.

If the robot explores a room, the planner has to plan again all the time. That does not mean that the planner has to change the plan abstraction (goal). The goal is and will be to explore the room. The planner can only reach this goal if it defines subgoals. These subgoals are by no means as abstract as the goal.
One example of the need to plan again is if the robot hits against a barricade. The new subgoal is to explore this barricade and to find out what it is. After exploring the barricade a while the robot has no clue what this barricade is. At this point it is a good idea to stop the exploration of the unknown barricade and to continue the exploration of the rest of the room.

Condition	Action
• Meta Plan Model: Explore Problem • No existing Executive Priorities	Write Executive Priorities: First decide the strategy (abstract) than define subgoals (less abstract) and the way to reach this subgoals. If the robot applies one of these ways to reach a subgoal, define how. If any problem appears check the strategy and the subgoals..

There are a lot of things that the robot has to take into account. Questions like the questions below are very important.

- Is there any idea what type of object the barricade is? (For example near a table, the barricade could be a chair)
- Does the barricade moves on its own? (So it is not really a barricade, only a temporary one)
- Did it move because the robot hit against it?

4.4 Director

The director specialists are looking for the executive focus. The executive focus changes during the planning time. It defines what kind of decision to make at a specific point in time. The executive focus depends on the executive priorities.

4.4.1 Director specialists for moving

The executive focus for the path finding problem is obvious. Deciding the route happens at a very abstract level. To decide how to move happens at a very low level.

Decide the route => Intentions and outcome levels
How to move => Lower levels

The interesting point of the described specialists below is that they act at different moments. As shown in the example (chapter 3.6 Example) the first specialist will appear very early. The second specialist, in contrast, will appear if the abstract problems are solved.

The planner is able to detect that the abstract problems are solved. If every plan abstraction (intention) has a corresponded plan (outcome) the abstract part of the planning process is finished.

Condition	Action
• Executive Priorities: First decide the route, than decide how • No existing Executive Focus • No existing Plan Intention and Outcome	Write Executive Focus: Intentions and Outcome levels
• Executive Priorities: First decide the route, than decide how • Executive Focus: Intentions and Outcome levels • Existing plan (outcome) for every plan abstraction (intention)	Write Executive Focus: Lower levels

4.4.2 Director specialists for exploring

The approach for the exploring problem is hierarchical. Finding the best strategy happens at a very abstract level. The subgoals will be defined at a less abstract level. The decisions about how to explore something will happen on a low level.

If the planning process starts, the specialists have to work out the most abstract level. After that they have to work out the subgoals and at least they have to work out how to explore. This approach is top down. The blackboard has to set the executive focus in this order.

The whole exploring process (including the controlling process) can be seen as a depth-first search. The planner has to refine a subgoal and has to execute what it has computed.

When does the specialist has to check the strategy? When does the specialist has to check the subgoals? These questions are very interesting.

Generally, we can say that if something unexpected happens the blackboard should set the focus at the next higher level. I will try to explain this mechanism with examples.

Imagine that the robot hits unexpectedly against something. This information should be given to the middle level because the lowest level is unable to see the context. Only the middle level is able to figure out what kind of barricade it could be. If there is not any subgoal like

- Explore the current barricade
- Move around the actual barricade

the middle level asks the highest level what the actual strategy is. If the strategy has not an answer the specialist should find a new strategy. Typical Strategies could be something like

- Explore everything you find until one specialist has an idea what it is
- If the robot hits against something explore it for two minutes. If there is no idea what it is after that time, move around.

We need specialists as described below. Notice that

- Strategy level = Intentions
- Subgoal level = Schemes and Designs
- How to explore = Strategies and Procedures

Condition	Action
• Executive Priorities: First decide the strategy than define subgoals. Define how to reach these subgoals. • No existing Executive Focus • No existing Intention	Write Executive Focus: Intentions level
• Executive Priorities: First decide the strategy than define subgoals. Define how to reach these subgoals. • Executive Focus: Intentions level • Existing Intention	Write Executive Focus: Schemes and Designs level
• Executive Priorities: First decide the strategy than define subgoals. Define how to reach these subgoals. • Executive Focus: Schemes and Designs level • Existing Schemes and Designs	Write Executive Focus: Strategies and Procedures levels
• Executive Priorities: First decide the strategy than define subgoals. Define how to reach these subgoals. • Executive Focus: Schemes and Designs level • Existing Design: Check Intention	Write Executive Focus: Intentions and Outcomes levels
• Executive Priorities: First decide the strategy than define subgoals. Define how to reach these subgoals. • Executive Focus: Strategies and Procedures level • Existing Procedure: Check Scheme	Write Executive Focus: Strategies and Procedures levels

Please notice that every specialist on the next lower level waits until the specialist on the next upper level finished.

4.5 Policy analyst

The task of the policy analyst is to analyse the meta plan policy, the knowledge base and other knowledge sources. The outcome is a suitable plan abstraction.

4.5.1 Policy analyst specialists for moving

The plan abstraction for the moving task is always the same. It does not depend on whether the robot has to use the shortest, fastest or safest path. As we saw in the executive focus the planner has at first a focus on the abstract level and after that on the lower level.

The structure of the abstract level is always the same.

- Which is the starting point?
- Which kind of route should the robot use (the fastest, the shortest or the safest)?
- Where is the goal point?

The order of the questions is not unimportant. At first the planner has to know where the robot is. There are two possibilities. The first possibility is that there is a note in the knowledge base about where the robot is. In this case the planner can compute the next question. The other possibility is that the knowledge base does not know where the robot is. In this case the robot has to find out where it is. Obviously it is impossible to find the route to the goal point without knowing where the robot is now.

The second information that the planner has to find is which kind of way the robot should use. Shall the robot use the fastest, the shortest or the safest way? The planner can find this information in the meta plan policy. It does not depend on the goal point. Nevertheless, the goal point depends on the kind of route the robot should use.
Consider our example to leave the room as fast as possible. In a room with two doors the goal point can change if the kind of route change. It is possible the route to door_1 is faster as the route to door_2 but the route to door_2 is safer than the route to door_1.

The plan abstraction of the lower level is very easy. The abstract plan is to find how the robot should move. That means the concrete movements like

- move 50 cm
- turn left 90°
- etc.

4.5.2 Policy analyst specialists for exploring

As described in chapter **4.3.2 Middle management specialists for exploring**, our chosen approach is to find the best strategy, to define subgoals and then to find out how to do it. This approach is also our plan abstraction.

Plan abstraction intentions: Find the best strategy
Plan abstraction schemes: Find the best subgoals

Plan abstraction strategy: Find how shall the robot do this

4.6 Introduction to the planning process

At this point the real planning process will start. Up to now the question was only how to plan. From this point the specialists plan indeed. There are a lot of questions that we should think about.

One question is whether it is best to choose one abstract approach and only to refine it or if it is an good idea to follow different abstract approaches (see chapter 4.6.1 One or several approaches).

Another important question is how the blackboard can find the best approach? In the moving task the solution of this question is obvious. If the planner has knowledge about the environment, every route has a value and the planner can choose the best route very easy. But how can the planner find the best explore strategy? One particular piece of information can be very important or not (see chapter 4.6.2 Value of information). The worth of information depends on the given task. Information that is very important for one task is useless for an other task. That means that the planner can estimate the value of the information only if it knows the task.

4.6.1 One or several approaches

One of the most important questions is how the blackboard tend to work. Will the blackboard choose one specialist and let the specialist decide what the next step will be. The other possibility is that different specialists write their solution or the next move on the blackboard and a scheduler chooses the next move.

Before I discuss these possibilities I will give an example that shows the problem

A very interesting situation is if constrains for different actions are the same and only the action with the highest rating has different constrains. For example a robot with a camera and a microphone. The constrain for using the camera is that the robot stands still because of the movement. The constrain for using the microphone obviously is that the robot stands still too, because of the noise. If the rating of movement is the highest but the added benefit of using the camera and the microphone is higher than the movement, the better decision is to stop the robot. The constrains for the second and the third rated action are the same but the constrains conflict with the first rated action.
If one planner finds these actions it can handle this problem. If several actions are obtained from several specialists, the action with the highest rate will win.

As one can see from this example sometimes it is a good decision to follow different strategies or different ideas of movement, to consider all of them. Because one specialist sometimes is not able to compute all possibilities of movement, it seems to be a good idea if different specialists write their idea on the blackboard. Problem of this approach is that someone has to compare the different movements and choose one.

My first approach was to use the blackboard scheduler for this task. This approach is wrong because it changes the blackboard from a communication tool to a part of the planner.

A better idea is to have a specialist which chooses a movement if there are different movements.

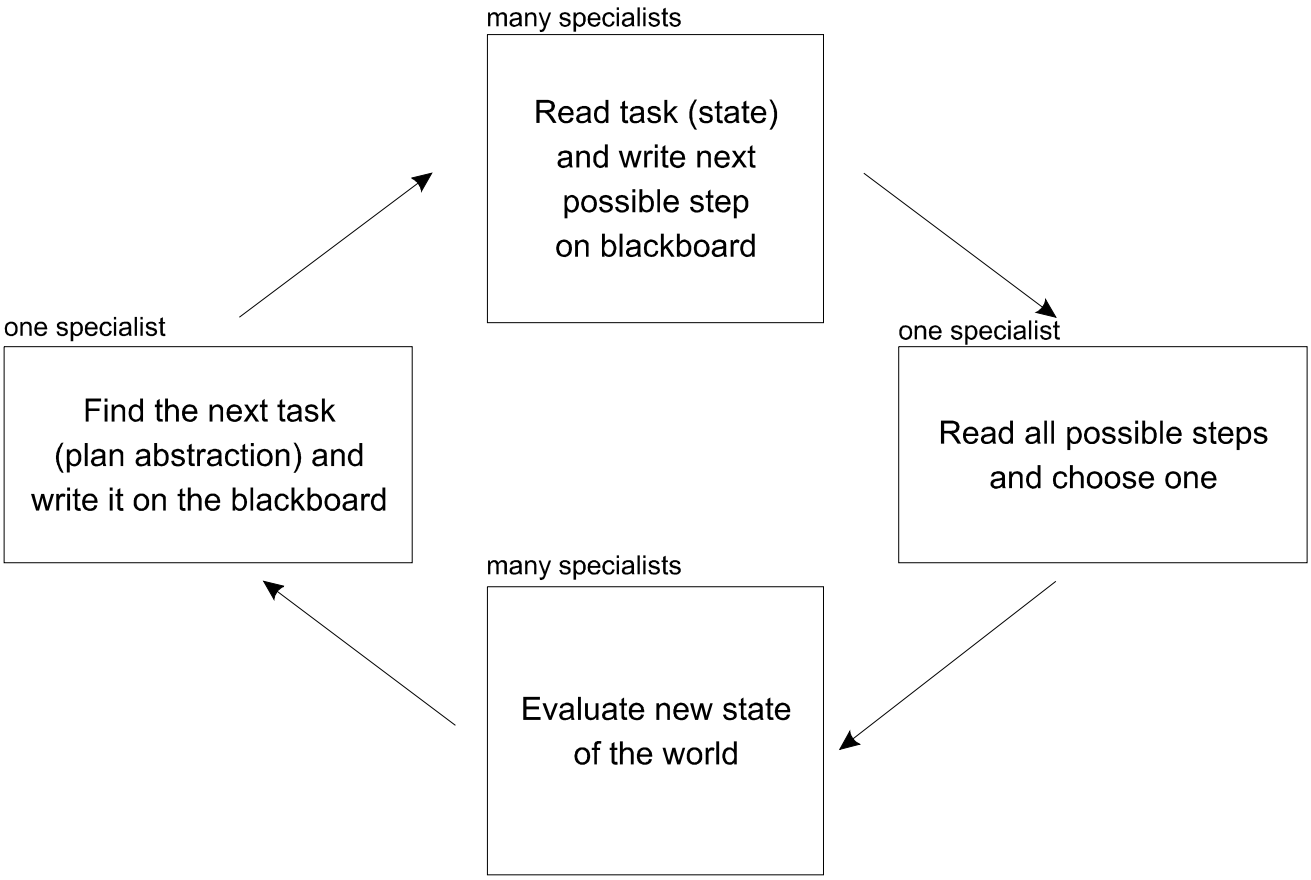

Figure 4.5

4.6.2 Value of information

A very important point is the question how to evaluate information. We distinguish between information that is stored and information that the robot will get.

A very easy solution to this problem is to find a general metric that finds a value for a piece of information. But this approach is to easy for us. If the robot only has to move in the right corner, information about the left corner is useless. The value of a piece of information depends on the task. Because of that reason it is very important to find a way to estimate the usability of a piece of information depending on one or more tasks.

If there exists information in the knowledge base about safe areas, the value of this information depends on how old this information is. Old information is more useless than new information. The worth of some (not all) information depends on the age.

The better this metric the better the behaviour of the mobile robot.

4.7 The planning process itself

In this chapter I will discuss some planning specialists. Because these specialists are used in the example, I will give only a brief description.

4.7.1 Strategy, subgoals and how to explore

As described in earlier chapters the process of exploring is at first hierarchical and at second less a planning process as more a controlling process.

Because the focus of this final project is the planning process it is not the task to work out how the specialist finds the best strategy or the best subgoals. Also to find the strategies and the subgoals itself is not the task of this final project. These are very huge and sophisticated tasks and depend on many parameters (see also **4.1.4 When shall the robot stop the exploration?** and **4.6.2 Value of information**).

4.7.2 Which is the starting point?

The first question of the moving task is to figure out the actual position of the robot. Either this information is in the knowledge base or the robot has to explore the environment until it knows its position.

If the information where the robot is, is in the Knowledge Base, one Specialist has to read the information and to write it on the blackboard.

If this information in not in the knowledge base the specialist has to define a new plan abstraction like:
Figure out where the robot is. Stop if you know the position of the robot.

4.7.3 Which kind of route should the robot use?

The second abstract task that the planner has to solve is the question of which route the robot should use (the fastest, the shortest, the safest or the opposite of one). This task is very easy because this information is in the meta plan policy.

It is important that this task is finished before the third task begins. The question of where the robot wants to go might depend on the character of the movement (fast, short or safe). The question how the robot has to move doesn't depend on the goal.

4.7.4 Where does the robot want to go?

The last task is to find out where the robot should move to. Sometimes the problem definition embodies this information.

In other cases there is only an abstract goal like *leave the room*. In such case the task is to figure out how the robot can do this. The specialist has to use the knowledge base to solve this problem.

4.8 Summary

This chapter shows that the task of finding and creating the specialists is not very easy. A lot of parameters have influence on the behaviour of the mobile robot.

If I started to create the specialists I supposed to create a lot of planning specialists as described in chapter **4.7 The planning process itself**. But the specialists that prepare the blackboard for the planning process were much more complex as I had thought.
However, this chapter shows that the preparation of the planning process is a very important part. This part embodies the translation of the task into the terminology of the blackboard.

Nevertheless, this chapter shows that the chosen structure of the blackboard, and as result the structure of the specialists, is suitable for the mobile robot world. It is possible to define the tasks of a group of specialists and the tasks of the specialists itself.

A topic that needs much more investigation is the topic described in chapter **4.6.2 Value of information**. It seems to be very important to be able to estimate how important one particular piece of information is, depending on a particular task.

It might be helpful to read the example in chapter 3.6 again. Please compare the in this chapter described specialists with the in the example used specialists. The example will show that the described specialists are as a chain. The result of one specialist is the precondition of the next specialist.

5. Knowledge representation

In this chapter I will discuss knowledge representation focusing on the mobile robots world as described in chapter **2.2 Mobile robots** . Knowledge representation depends on the area where the robot moves but the problems in various areas are the same. In every area are fixed barricades and more or less movable barricades.
The place of focus in this chapter is a normal office. The global task of the robot is to pick up things and take them to other places.

The planner has to solve a lot of various problems. Because of that it seems to be a good idea if the knowledge base does not use only one approach to store the knowledge. It seems to be a better idea to use various ways of knowledge representation.

5.1 The hierarchical part

There is psychological evidence that in addition to their ability to associate concepts, humans also organise their knowledge hierarchically, with information kept at the highest appropriate levels of taxonomy. [Luger, Stubblefield, 1993]

Because we try to imitate human behaviour it seems to be a good idea to imitate how humans organise their knowledge.
With this in mind, a semantic net will be used. A semantic net involves everything we need. It supports inheritance and contains isa and instance relations, as well as other, more domain-specific relations.

The first task in building a semantic net was to find the objects. In the mobile robot world this task is not difficult. Imagine an office. Everything you can see is an object. But also things that are not obvious could be an object. For example the room itself is an object. An other example is the door. The door itself is an object (barricade) but the door way is also an object. With this object the robot can reach the goal "leave the room".

Figure 5.1 shows an example. In the first level I distinguish between barricades and non-barricades and the robot itself. This is very helpful but not necessary. The attribute ’barricade’ could also be a property of the objects in the next level. The robot is not a barricade for itself. If there are existing more than one robot, the other robots are barricades. The attributes change from the point of view.

The next level identifies the item; for example table, chair or cupboard. This level contains information that belong to the object itself. For instance the information ’close objects’ or the ’move circle’, described in chapter **5.2 Unsafe area**, belong to this level.

The next level depends on the item. In the case of items with different models we create a model layer. The model of an item describes it exact. It contains the three dimensional description (see chapter **5.4 The three-dimensional description**).
Sometimes we do not need this layer. For example if the item exist only once in the room or the instances are not different. In this case the described information can be stored in the identification level.

The next level is the instance itself.

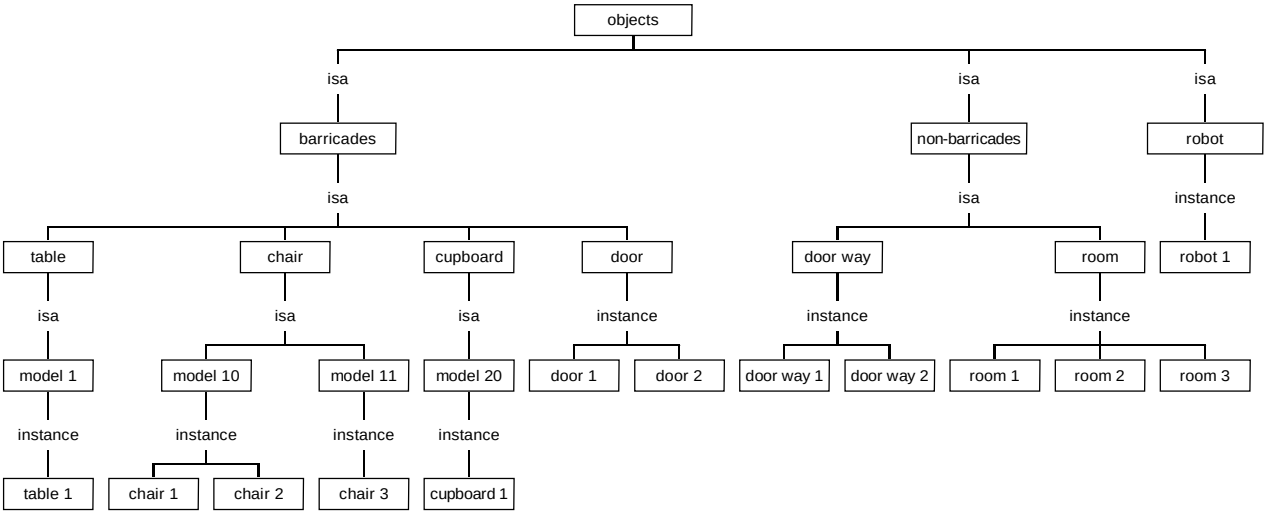

Figure 5.1

5.1.1 Location of items

A very interesting problem is how to store the positions of the items. One approach is to store the global position of every item. This approach contains the *frame problem*. One instance of the frame problem is a table with a book on it. If we move the table we also move the book. With a hierarchical organisation of items we can solve this problem as shown below.

Beside the semantic net with ’type’ or ’characteristic’ inheritance it seems to be a good idea to build a second hierarchy with ’move’ inheritance. Therefore we store the local position of every item. Every item on which another item can lie, gets its own co-ordinate system. For example a table gets its own co-ordinate system and we store only the position of the book on the table and that it lies on the table. If the table moves, everything on the table moves too without changing any co-ordinates of these items.
Of course this system allows inheritance over more than one level. Imagine a table with a plate on it and on the plate lies a fork. We store the position of the fork dependent on the plate and the plate dependent on the table.

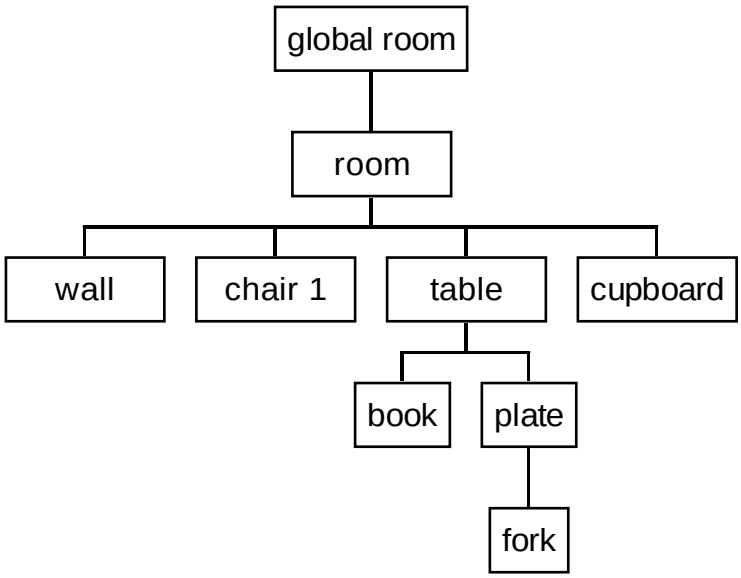

Figure 5.2

To build this tree we need only three values:

- x co-ordinate
- y co-ordinate
- lies on

Obviously these values will be stored at the instance of every item.

These values allow a tree to be built as shown in Figure 2. In the example in Figure 5.2 the fork lies on the plate and the plate lies on the table.

The interesting thing at this point is that we store only two dimensions. The third dimension will be inherited from the parent. For example the position of the book depends on the x- and y co-ordinates and the height of the table.

5.1.2 Movement facilities of items

Every item possesses its own movement facility. A chair, for example, can be moved in all directions of the co-ordinate system without limitation. Another example is a door. A door can only be moved in a restricted way.

A special item is the robot. It can move on its own in a restricted way and we can describe it. But of course it is always possible for somebody to take it and move it to another place. In our world we do not take this latest into account. The description of the possible movements only relates to the movements of the robot itself.

To describe the movement facilities we need three values; the two directions of the co-ordinate system and whether the item is able to turn. With this three atomic movements it is possible to build every complex movement.

I will give some examples to explain how this works.

A chair has no limitation in its movement. It can be moved in all directions in the co-ordinate system.

- x direction: no limits
- y direction: no limits
- turn facility : no limits

A door has a number of limitations. It can not be moved in the co-ordinate system. It can only be turned.

- x direction: not able
- y direction: not able
- turn facility : 0° to 170°

Obviously it is necessary for the item to get an initial angle and an initial state.

A very interesting item is the robot. Imagine a robot that has the same move facilities as a car. It can move forward and backward and it can turn left or right. But every robot has its turning-circle.

- x direction: no limits
- y direction: not able
- turn facility : 1° needs 0.5cm x direction

The turn-facility '1° needs 0.5cm x direction' means that the robot has to move to be able to turn (for instance 45 cm to turn 90°). The longer the distance the robot need to turn 1° the bigger the turn-circle.

Another group of items is the items with very limited movement facilities, for instance a drawer. A drawer can only be moved a limited way in one direction.
Please note that we define the x and y direction of every item which does not change if the item change the position.

- x direction: 50 cm
- y direction: not able
- turn facility : not able

5.2 Unsafe areas

As described in earlier chapters it is possible to distinguish between save and unsafe areas. Probably chairs will move a lot and are in the vicinity of a table. The space around the table is very unsafe.

To store this information we need three values:

1. probability the item will move
   - it will never move (for example a wall)
   - it will not move very often (for example cupboard, table)
   - it will move very often (for example chair)
   - it will never be at the same place (for example human)
2. move circle (for example 50 cm)
3. close objects (for example desk: desk-chair, wastebasket, computer, human, ...)

The *move circle* describes the circle the item probably will not leave. We need the *move circle* to build the two dimensional map with safe and unsafe areas. It is obvious how this information changes the two dimensional map, so I will not describe how it works.

The *close objects* value is very helpful if the robot hits against an unknown object and wants to find out what it could be. If the robot bumps against an unknown object, and is not in an unsafe area, the planner can find out which items are close to the robot and which of these items have *close objects*.

Please notice that safe and unsafe areas are a result of the position of a item and the move circle. The value close object is only to get a clue which item could be close.

Of course the *move circle* and the value *in the vicinity of* will not be stored for items that either will never or not very often move.

All these values can be stored in the hierarchy (see Figure 5.1) and can be inherited to the instance.

5.3 The direction

The given values are enough to build a tree as shown in Figure 5.2 but they are not enough to define a correct position. Therefore we need more values, the direction of the item.
Items do not lie right-angled in the world. Because of that we have to define the direction of every item. From Figure 5.3 to Figure 5.6 shows that an object with three dimensions can turn in three directions. To describe the position of every item exactly, we would need these three values.
The problem is that the side-view- and the front-view-angle are rarely used. It does not happen very often that a chair, a table or a cupboard needs these angles. Only items like books and other small items sometimes need these values. Furthermore all algorithms that are using these values become much more complex (see Chapter **Convert the information into a two-dimensional map**) . Because of that reason I decided to drop the side- and the front-view angle but it should be noticed that they do exist.

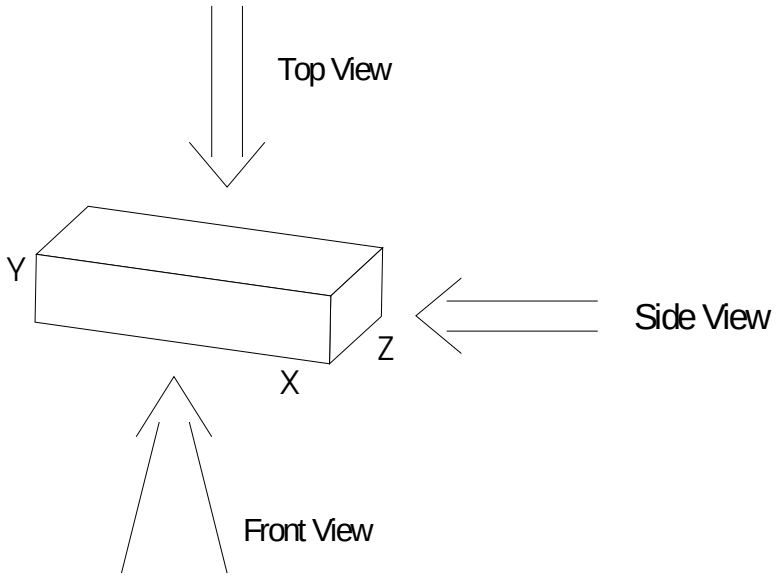

Figure 5.3

Front View (X,Y)

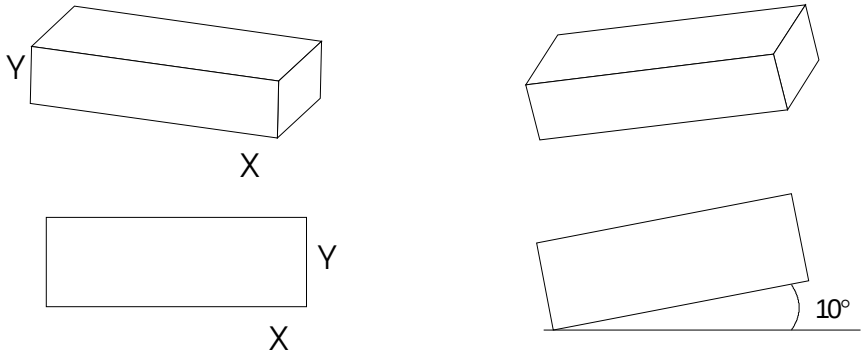

Figure 5.4

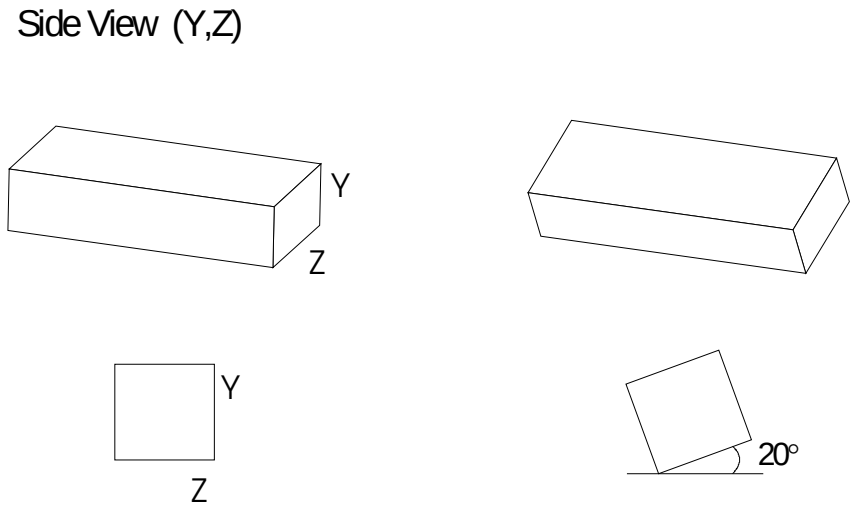

Figure 5.5

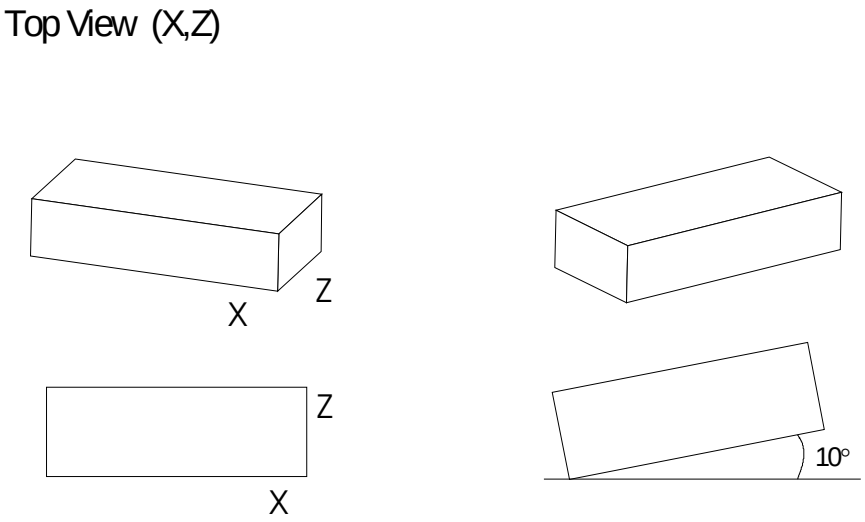

Figure 5.6

Of course the top-view-angle is very important in our mobile robot world.

As described above every item on which an other item can lie has its own co-ordinate system. Obviously this co-ordinate system turns with the item as you can see in Figure 7.

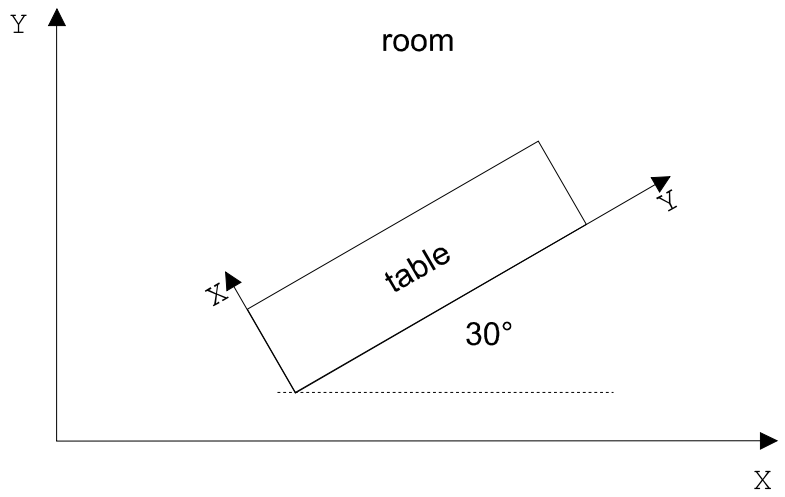

Figure 5.7

It is not the task of this Final Project to describe the benefits and problems if the front-view- and the side-view-angle were stored.

5.4 The three-dimensional description

It is necessary to store information on how big the item is. The robot needs that information to compute if a item is a barricade or not. A table for example can be a barricade if the robot is higher than the table. It also can be free space except the legs if the robot fits under the table. However, this might be changed if the robot has to transport an item that is higher than the table.

Obviously there are a lot of ways of storing this information. One way is to describe every part of the item by itself.

For example the table 'model 1':

- Four legs
- Cylinder
- Diameter 15 cm
- Table-top is a rectangle
- 150 cm long
- 110 cm wide
- 90 cm high
- Pass-high 80 cm

This kind of description would require very sophisticated algorithms to transform it into a two-dimensional map. Furthermore it would require defining all kinds of forms (cylinder, cube, etc.).
It is easier to describe the items with small cubes. This kind of description is not that exact but it is very easy and the algorithms transform it into a two-dimensional map very easily (see also Chapter **Convert the information into a two-dimensional map**).

Imagine a cube that is a little bit bigger than our table. This cube consists of a lot of small cubes. One small cube represents for example the space of 1 cm^3. The value '0' means that there is no barricade. The value '1' means that there is a barricade.
In the first moment it seems that this kind of description needs a lot of hard disc space. But to describe a table with 3 m^3 there is only a need of 375 kilo byte
(3 * 100 cm * 100 cm * 100 cm / 8). If this is too much hard disc space it is possible to store the items in coarser grained or to compress the stored data. A data compression will be very effective because of the structure of the data.

5.5 Convert the information into a two dimensional map

Above I described how to store three dimensional information about the room. This is all very well and important but useless for planning algorithms. Most planning algorithms are developed for two dimensional maps. In this Chapter I will discuss how to convert the given three dimensional information into a two dimensional map.

Because the robot moves in the room we have only to take into account those items lying in the room. This seems obvious but in our knowledge representation it means something special. It contains that we do not have to take into account these items that are lying on one of the items in the room. For example the book on the table has no influence on the two dimensional map. Of course it is possible that an item is a barricade although it lies on an other item. For example an item that is bigger than a table and lies on it will be a barricade. This case will not be taken into account. Nevertheless this case is not really a problem and can be solved very easy.

The chosen representation offers the possibility to have a view on every item in the room that is a barricade. We have to look only on these items that have the value "lie-on: *room*". With the three-dimensional map from these items and the three dimensional description of the robot it is easy to build the two dimensional map. Below I will describe one simple approach of how to do this.

The first step is to look at how high the robot is. After that take the three dimensional map from the room and take the few centimetres on the bottom of the map of the high of the robot and throw the rest away. Now we have a map of the room as high as the robot. The last step is to have a look at every column. Every column in the three dimensional map will be represented from one frame in the two dimensional map. If all cubes of one column say that there is no barricade the frame gets the value "free". If one cube in the column says that there is a barricade then the frame will get the value "barricade".

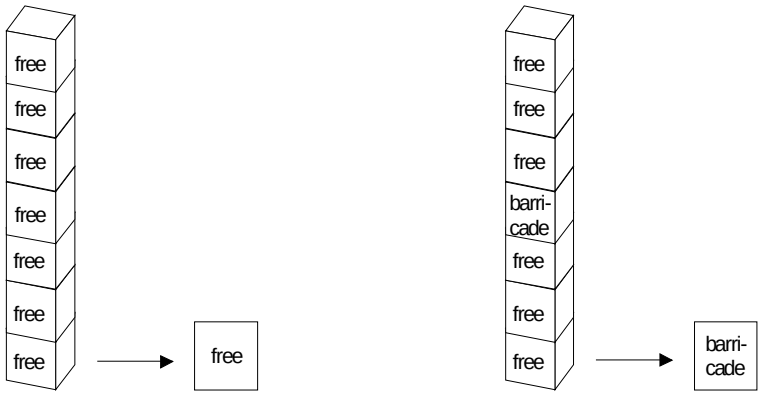

Figure 5.8

This approach produces a very fine grained map from the room. If this map is to fine grained it is easy to produce a more coarse grained map.

5.6 Convert fine grained map into coarse grained map

Sometimes there is the need of a very coarse grained map to get a first idea which way the robot can be used. In this chapter I will describe one way of many ways how to produce a coarse grained map from a fine grained map.

One simple way to do it is to produce groups of frames. In the given example one group is four frames. Every group of frames produces a new frame in the new map. If every frame in the group has the value *free* then the representing frame in the new map has the value *free*, too. If only one frame in the group of the old map has the value *barricade* the representing frame in the new map gets the value *barricade*.

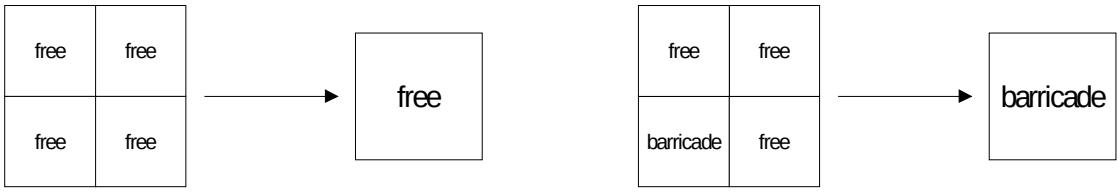

Figure 5.9

The advantages of this algorithm are that it is very easy and fast. The disadvantage is that free areas shrink. It can happen that a free area shrink that much that it look like a barricade. That means that this algorithm is only suitable if there is a lot of free space.

5.7 The procedural representation

In this chapter I will describe procedural knowledge and its representation.

The most common technique for representing procedural knowledge in artificial intelligence programs is the use of production rules.

Very often a place has characteristics. One example for a characteristic is that the robot can **leave the room** if it **moves through the door way**. To store this information production rules are suitable. An other characteristic is if a place is suitable to park the robot. In the table below are some examples how to store it.

if ...	then ...
problem definition "leave the room"	move through the door way
problem definition "park"	move to position X
barricade is computer	do not bump against it
barricade is a cup of tea	do not bump against it
...	...

It is not the aim of this Final Project to build a written English understanding application. Nevertheless the table above seems to solve such problems. But it should be clear that this is a very poor solution. To build a written English understanding application it needs a lot more development.

Characteristics of items are not the only rules to store. Every specialist has a lot of condition action pairs with the same structure as production rules. Of course it is possible that specialists use the knowledge base to store these condition action pairs (see Chapter **Specialist**).

5.8 Inexact data

It is very common in computer science to develop something without having to look at the technical abilities in the first steps. The reason is that the technical abilities can change very fast.
Indeed the technical abilities of cheap mobile robots are very poor. Digital cameras for example are very expensive. Cheap phototransistors or photodiodes do not give the best results.

The knowledge base is able to store the information about the room very refined. How can a robot exactly explore a room?
The answer is that it is not possible. We have to explore the room once and to transmit the data to the robot. Obviously we can not do this all the time but it must be done at least once if the robot is start to the first time.
It seems strange that we have to explore the room once. But it is the same way as humans behave. Humans do not explore the whole world by itself. They go to school and somebody tells them about the world and how it works.
If we expect human behaviour from the mobile robot we have to teach it. Especially knowledge that it can not reach on its own.

Another problem is if the robot hits against something and do not know exactly where it is. In this case the robot has to store that information that it knows. The next step will be orientation of the robot. To orientate and to know the position is always a task with high priority. Because of the way that the robot moves the planner can compute where it hits against something.

The problem of inexact data and how to store it is a very difficult problem. Further ideas should be developed.

5.9 Unknown object

A very important object in our hierarchy is the unknown object. If the mobile robot hits against something and has no idea what it is, it must be able to store it. In this case it needs an unknown object in our knowledge base.
The task of this unknown object is to represent a barricade as long as the planner has no idea what it is. If the planner gets an idea what kind of barricade it is, the instance of the unknown object will be deleted and the information will be stored in the instance of the realised object (position, direction, etc.).
Obviously the unknown object has to have the ability to grow.

5.10 Summary

Obviously there is a need of a mixture of different kinds of knowledge representation. In this chapter I have shown that it is possible to build a knowledge base with well known methods of artificial intelligence.

Another very important thing that I learned was that it is not possible to build a knowledge base with little space. Abstraction offers that the knowledge base is more understandable but needs a lot of overhead. Therefore we need a lot of space.

6. Conclusion

6.1 Evaluation of the objectives

The difference between the common planning systems like NOAH and a planner built with the blackboard architecture is that most of the common planning systems work with only one problem solving approach. They work either hierarchically or with the A* algorithm or with other approaches.

Every problem can be solved in different ways, although some approaches are more suitable and some approaches are less suitable. The blackboard architecture tries to find the best approach for every problem. Either a problem will be solved with one approach or it will be solved with various approaches.

On the other hand the planning task of the mobile robot world is a mixture of several problems. Obviously the task to plan an errand needs another approach to the task to pick up an item with the goal to move it. Even the task to plan an errand itself involves different approaches.

Theoretically the blackboard seems to be a good tool to solve the planning problem.

As written in the summaries of the chapters 3 **The data structure of the blackboard** and chapter 4 **Specialist** this final project shows that the blackboard and the presented structure of the blackboard are suitable for mobile robot's planner. It offers the possibility of opportunistic planning.

The advantages of a planner constructed with a blackboard are
- Specialists can solve one particular problem optimally
- Specialists are easily exchangeable
- The structure of the blackboard is well defined
- Possibility of changing behaviour in same situations
- Possibility of learning specialists

The disadvantages of a planner constructed with a blackboard are
- The blackboard and its structure is not easy to understand
- Especially the design of specialists can be a hard job
- Developers have less experience with blackboards

6.2 Evaluation of project management

At the beginning of the final project I started with a lot of reading. Sacerdoti (1977) was a good introduction in the domain of planning and its problems.

During my research about the blackboard I saw that Barbara Hayes-Roth has done research in this domain. I sent her an email with a brief description of my project. One of her students (Karl Pfleger) mailed back a list with very good references and papers, sometimes only

available on the Internet. This was a very good start and should encourage everybody to use the Internet (especially email) and its possibilities.

One problem was that the frame of this final project was not defined. Sometimes I had the problem that I did not know if one particular investigation should be continued or not. There was always the danger of either getting lost in one particular problem or being superficial.

Sometimes my expectations of the capabilities of the blackboard were too big. This also is a result of a project without defined borders. I learned that it is quite important to check the expectations all the time. If the expectations are to high you will not finish the project. If they are to low the result of the project is not as it could be.

To handle the problem of working without clear defined description of the project can be seen as one part of the project. I learned to toggle between project work and management work.

One part of this final project, that I did not describe, was to install and to test one implementation of the blackboard. I installed the C++ version of BB1, a blackboard implementation of Stanford University, California. This helped me to become more familiar with blackboards. One problem was that the C++ version is only one part of the original implementation of BB1. I tried to install the original version of BB1, written in Lucid Lisp, but the Computer Centre was not able to offer me Lucid Lisp.

However, the installation problem was not the main reason that I did not implement some of my ideas as described in the advanced objectives of this final project. The reason that I did not implement anything was that the core objectives where so complex and needed so much time that I did not have enough space to do anything beside those.

If I did a research project again, I would try to define a smaller project including implementation. Firstly an implementation encourages one to investigate problems until they are really solved and stops investigation if they are unimportant for the implementation. Secondly doing only conceptual and theoretical work can be boring. Thirdly an implementation checks whether the ideas are feasible or not. On the other hand some of the investigation of this final project would not be done if I had to implement everything. However, in summary I think it is better to research and to implement.

Of course one theoretical possibility is to shorten the core objectives and to implement some of the ideas. I think this was not possible in this project. I needed the whole time to get a clear picture of the whole domain.

Another problem was the foreign language. As a German native speaker it was part of the project management to improve my English as fast as possible. Very helpful was that I wrote a lot accompanying my investigations. I tried to explain the things that I learned in my own words. This needs a lot of time but it improves the knowledge about the language.

6.3 Discussion of further work

I have the feeling that this final project sometimes stops at a very early point of research. A lot of interesting questions would be worth investigating much more deeply. Below I name a few examples.

In chapter **2.2.5 Unsolved problems in the mobile robots world** I described the task of building a blackboard that co-ordinates a lot of robots. To build this blackboard would be a attractive task.

A very interesting problem is how to qualify and compare the value of information (Chapter **5.6.2 Value of Information**).

Another problem is the general behaviour in problem situations (For example **5.4.2 Strategy, Subgoals and How explore**).

In chapter **5.8 Inexact data** I described the problem of how to store inexact data. This needs to be investigated.

6.4 Final conclusion

Finally it can be said that the blackboard architecture is suitable as the basis for planning mobile robot movements. I will give this result to the research group ICS which will use this result for further work.

7. References

Erman, Lee D.; Hayes-Roth, Frederick; Lesser, Victor R.; Reddy, Ray D. (1980).*The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty* Computing Surveys, 12(2), pp153-167

Hayes-Roth, Barbara (1993). *Intelligent control.* Artificial Intelligence 59 (1993), Elsevier Science Publishers B.V., pp 213-220

Hayes-Roth, Barbara; Hayes-Roth, Frederick (1979a).*A Cognitive Model of Planning* Cognitive Science 3 (1979), pp 275-310

Hayes-Roth, Barbara; Hayes-Roth, Frederick; Rosenschein, Stan and Cammarata, Stephanie(1979b)*Modeling Planning as an Incremental, Opportunistic Process*In: Proceedings of the Sixth International Join Conference on Artficial Intelligence Tokyo, August 20-23, 1979, Volume One, pp375-383

Hayes-Roth, Barbara; Pfleger Karl; Lalanda, Philippe; Morignot, Philippe; Balabanovic, Marko (1995). *A Domain-Specific Software Architecture for Adaptive Intelligent Systems* Knowledge Systems Laboratory, Computer Science Department, Stanford Universtity, 701 Welch Rd., Stanford, CA 94304

Jones, Joseph L. and Flynn, Anita M.(1993). *Mobile Robots. Inspiration to Implementation*Artificicial Intelligence Laboratory Massachusetts Institute of Technology: A K Peters Wellesley

Luck, Kai von; Owsnicki-Klewe, Bernd (1995). *Integrating Cognitive Systems. First intermediate report: Project outline.* Fachhochschule Hamburg, Fachbereich Elektrotechnik und Informatik. Hamburg

Luger, George F. and Stubblefield William A.(1993)*Artificial Intelligence*2nd ed. University of New Mexico, Albuquerque: The Benjamin/Cummings Publishing Company, Inc., pp 308-393

Sacerdoti, Earl D. (1977). *A Structure for Plans and Behaviour.* Elsevier, North-Holland, Inc.