KMi Technical Report

Knowledge Media Institute

Intelligent scheduling - A Literature Review

Dnyanesh Gajanan Rajpathak
Knowledge Media Institute
The Open University,
Walton Hall, Milton Keynes,
MK7 6AA, UK

KMI-TR-119

August- 2001

Table of Contents

1. Introduction

The literature review builds the initial foundation in understanding the domain of our interest, i.e. knowledge-intensive approach to the intelligent scheduling. In order to comprehend the nature of scheduling problem it is equally important to know the planning paradigm which goes hand in hand with the scheduling problem.

Usually, the scheduling problem comes in a variety of flavours and hence it is vital to know the development of various aspects that are involved in constructing a good schedule. In generic terms the scheduling deals with the temporal assignment of jobs to the resources within the given time framework while maintaining the various constraints. The nature of the components involved in a schedule exhibits quite complicated inter-relation with each other. At the same time a good schedule can be seen as a key to success in variety of applications such as, manufacturing, resource allocation, project management etc. And it makes the scheduling problem very interesting to deal with.

The proposed literature review is outlined as follows. In the next section we will analytically compare the two main paradigms such as, planning and scheduling. During which we will try to find out the similarities and differences between them. In section 3, first we will look at the various techniques that can be applicable for dealing with the scheduling problem and then we present the most influential intelligent scheduling systems developed over the period of last two decades or so. In section 4, we present the various aspects in the field of knowledge-based engineering during which first we will acquaint ourselves with the basic notions in the field of knowledge engineering and then focus on the area of interest such as knowledge acquisition as a knowledge modelling. As a part of this section we are interesting in looking at the most influential knowledge modelling frameworks that are developed over the period of last two decades or so. In section 5, we will review the current status of the task ontologies that are developed exclusively for the scheduling task. In section 6, the main focus of the review will be on the libraries of problem solving methods developed for the scheduling task. As the ultimate aim of our research is to develop the library of problem solving methods for scheduling, and so this part of the research will be helpful in understanding the current status of the field. In section 7, we conclude the literature review along with the general discussion.

In sum the literature review will be helpful in making us aware of the current status of the research in the field of scheduling.

2. Differentiating between Planning and Scheduling – A Comparative Study

In this section we will analytically compare the two major paradigms of interest such as, planning and scheduling. The main purpose of this comparative study is first to establish the clear-cut distinction between planning and scheduling paradigms. After that we will analyse how these two paradigms are complementary to each other. During this part we are mainly interested in looking at the theoretical aspects of these two domains.

2.1 Planning

Nearly, over last 30 years or so much of the work have been done on planning which falls mainly into the category of what is called as *classical planning* paradigm. If we open any book on Artificial Intelligence one can find that at least one chapter is talking about planning, as it is one of the thoroughly studied area in AI. Still, it is difficult to find the succinct definition of planning, independently of the particular formalism of application. One can define a pure planning task by looking from the different viewpoints according to its applicability in the related fields.

"Planning is a task of finding a sequence of actions that will transfer the initial world into one in which the goal description is true." [3]

"The AI planning problem is to find the sequence of actions that will take the planning world from a defined initial condition to a given goal state." [73]

"The planning can be seen as a sequence of actions generator which are restricted by constraints describing the limitations on the world under view." [7]

There is another perspective looking at the planning task, which is somewhat different from the traditional viewpoints and which is analysing the planning task as a design or synthesis activity.

"Planning as the process of devising, designing or formulating something to be done, such as the arrangements of the parts of a thing or an action or proceedings to be carried out."[91]

Even though one can find that there is an overwhelming work has been done in planning research area these viewpoints often seems to differ from one other. Looking from the various definitions mentioned above there are few conclusions that can be drawn in order to characterise the nature of a planning task.

A) The planning a task is one in which one is mainly concern about generating the sequence of actions such that the required goal of the problem can be attained.
B) The generated actions are often restricted by the various types of constraints that limit the space of possible solutions.
C) The planning task can be seen as a synthesis activity in which the possible actions that compose the plan are known, but the time factor that determines the order or dependencies between possible actions are unknown.

D) Planning task normally emphasis towards achieving the *feasibility* aspect of the generated actions. [3, 29]. From the traditional viewpoint the main purpose of the planning task is to achieve the final goal and the time it takes to attain the goal is absent or somewhat a secondary issue. For example, one of the sample problems of the planning task is the famous *monkey and banana* problem in which the monkey has to get hold of the banana hanging in the middle of the room. The banana is high enough, as monkey can not reach there and has various other alternatives to reach the banana. Well, now the major goal of the problem is that the monkey finally gets hold of the banana and the main concern is to reduce the number of moves in which the monkey does it. The time it takes for the monkey to get hold of banana is irrelevant in this respect. The feasibility aspect of the problem is to reduce the number of moves of the monkey for getting hold of a banana.

Figure 1, depicts the planning task in its purest form in which planning takes the input from the external world based on some requirements or criteria and generates the sequence of actions so that the need from the external world is satisfied. In the figure the demands can either come from the marketing department describing the need of the market or directly from the customer, demanding the feasible plan that will fulfil the requirements of particular scenario. The main objective is to transfer these demands into rough plan to meet the demands. Then during the actual planning task the flow or sequence of actions is specified, which if carried out the demand in hand can be satisfied. Speaking more formally, it describes the transition from the initial state of the world to the goal state where demands are fulfilled.

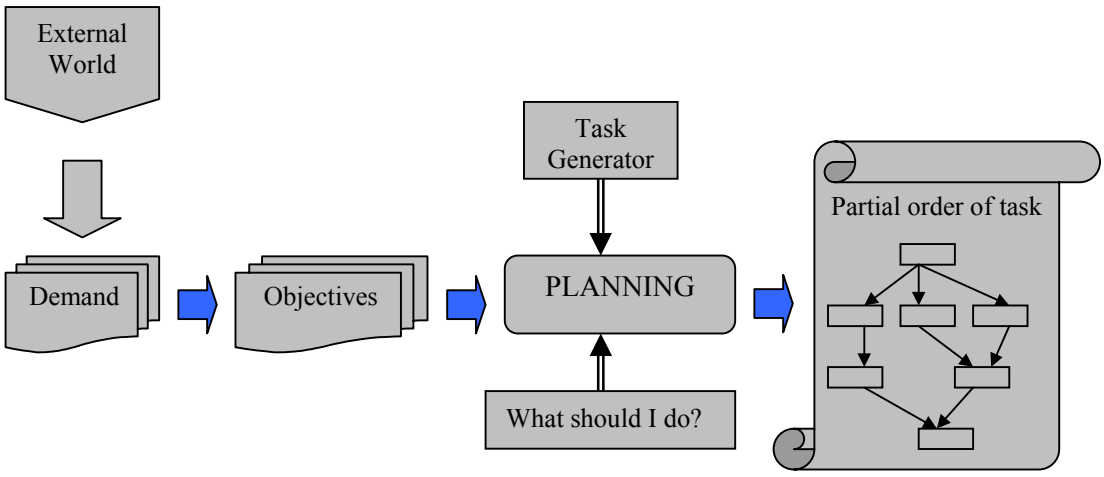


Figure 1. The overview of a planning task.

There are three major categories in which the planning task can be classified such as Classical planning paradigm, Hierarchical Task Network (HTN) planning and Decision-theoretical planning [110]. In the following subsections we will briefly review each of these planning systems.

2.1.2 Classical Planning Paradigm

In the classical planning, the main problem can be characterised in terms of achieving the given set of goals. The goal states are usually specified as a set of positive and negative literals in the propositional calculus. The given description of the set of world in the initial state is referred to as *initial conditions*, that expresses the initial state and is also expressed as a set of literals. The probable actions are characterised by using what is called as a STRIPS operator and algorithms in order to transform the states of the world [38] and this lies at the core of many planning problems. Most of the classical planning work has focussed on constructing the plans by searching backwards from the goal state where the goal state to be achieved is known beforehand. Usually, a STRIPS operator is seen as a parameterised template for all such set of possible actions. This could contain the set of *preconditions* that must be satisfied before a particular action can be executed and a set of *changes* or *effects* that the action will have on the world. Both the preconditions and effects can be positive or negative literals. However, a number of recent planning systems used the extended version of STRIPS language known as ADL [94]. Despite the fact that STRIPS and ADL has been used widely in the AI planning community over the period of time some serious limitations have been identified for their representations that are given in the following bullet points.

- **Atomic Time**. There is no explicit model of time in the representation. One can neither specify the duration of an action nor the constraints on either goals or actions.
- **Resources**. There is no provision for specifying the resource consumption or requirements for the execution of particular action.
- **Uncertainty**. There is no ability to model the uncertainty in the world under specific considerations. In other words it does not hold any accountability to handle the 'what-if' scenario while executing the classical planning paradigm.
- **Goals**. The only types of the alternative that can be specified are the goals of attainment [61, 33]. As it only specifies the final goal state or the number of goal states that needs to be attained by the plan.

2.1.2 Hierarchical Task Network (HTN)

In the Hierarchical Task Network (HTN) planning, the main objective is usually specified as a high-level task to be performed, rather than the conjunction of literals to be attained. The main difference between the classical planning paradigm and HTN is that, the HTN planning reduces the most *abstract task* down to the *primitive tasks*, whereas the classical planning mainly aims at assembling the several actions for the goal attainment. It can be found that virtually all the planning systems that are developed for the practical applicability has been developed by using the HTN planning [127]. In HTN, the planning procedure proceeds by

recursively expanding the high-level tasks into the networks of lower level tasks that eventually accomplishes the high-level tasks. The expansion procedure is described by transformation rules called *methods*. The methods can be seen as the mapping procedure from a task into a partially ordered network of tasks, along with the set of constraints imposed on the actions. A planning is completed when the resulting network contains only primitive tasks and the set of constraints is consistent with respect to the requirements or demands. As opposed to the classical planning approach the time and metric quantities representation do not cause much difficulty in HTN planning. One of the greatest strength of HTN planning is that the search process can be tightly controlled by the careful design of the methods. As in the classical planning the preconditions and effects of the actions simply specify when the action can be used and what could be used to achieve the actions.

2.1.3 Decision-theoretical Planning

The origins of the Decision-theoretical planning can be found in operations research and Decision Sciences. In both of these streams the planning problem have been modelled sequentially using Markov Decision Processes (MDPs) [99]. It was since last ten years, there has been increasing interest within AI community in using this technology for solving the planning problems that primarily involves the uncertainty. In this approach a state space transitional approach is probabilistic in nature. Traditionally, these kinds of problems are efficiently solved by using the powerful techniques such as value-iteration and policy-iteration techniques [99]. Usually these approaches find optimal policies, which amount to the conditional plans that specify which action should be taken in every possible state. The main trouble for using this kind of approach is that the size of the total state space grows exponentially. In order to overcome such a difficulty much of the AI work has concentrated on limiting the size of a state space by following approaches:

- By using the more compact representation, typically, many preconditions and actions are kept independent.
- Using an approximation technique it expands only that portions of a state-space that is most probable and seemingly useful.

Despite the problem with the state space there are few other limitations are observed for using this approach such as:

- **Complete Observability**. This framework assumes that only once an action is performed with an uncertain outcome, the agent can observe the resulting state.
- **Atomic Time**. There is no explicit model of time in the representation. Actions are modelled as if they are discrete, instantaneous and without pre-emption (i.e., once the action is started it cannot be disturbed until its completion).

- **Goals**. It rather seems to be difficult to express the goal attainment problems in this framework.

In sum, the planning task mainly deals with identifying about what actions need to be sequenced in order to attain the final goal state. But when to carry out these actions and how to carry these sets of generated actions remains unspecified. Also, the planning task can be seen as to attain the feasible solution rather than optimising the various criteria.

Slowly and steadily the important issue started raising its head in the community is about the optimisation of the tasks, as most of the problems started demanding about the optimal solution rather than only the feasible one. Although, the planing was successful for the long-term time horizon, on the short-term time horizon it started showing the limitations. This is when the scheduling technique came into the picture as a separate domain under the dominant shadow of planning.

2.2 Scheduling

It was still the time when most of the researchers were treating the problems from the planning and scheduling domains as a solely planning problem and the scheduling task was mainly considered as a sub-domain of planning. As it was mentioned earlier the real-life problems needed to achieve the optimal solution over some criteria such as, minimisation of cost etc. Although, the pure planning techniques were quite successful for dealing with the long-term time horizon but for the short-term time they have limited applicability.

The scheduling problem has a long history from the area of operations research where they are mainly refereed to as an assignment problem. Scheduling did not receive much attention in AI community since 1980, when Fox et al. began their work on the ISIS, which was constraint-directed scheduling system for the job-shop scheduling problem [39,40]. During that period growing number of researchers started working on scheduling by using the various techniques from artificial intelligence. More recently, it has garnered the attention of a significant number of AI researchers primarily in the application areas such as manufacturing, resource allocation, military transportation and space etc. Even today there is a conceptualisation about the scheduling task that it is a special case of planning in which the actions are already chosen and leaving only the question of allocating these orders for their assignment. This is an unfortunate trivialisation of the scheduling task. As opposed to the planning task the scheduling has found its well-defined boundary line for its definition. The scheduling task can be defined from the various viewpoints such as, operations research, artificial intelligence etc. So before going on talking more about the scheduling lets consider some of the few definitions that are widely accepted to describe the nature of scheduling task.

"Scheduling is the problem of assigning limited resources to tasks over time in order to optimise one or more objectives" [6, 95].

"Scheduling deals with the exact allocation of jobs over time, i.e., finding a resources that will process the job and time of processing" [17].

"Scheduling deals with the temporal assignment of jobs to the limited resources where a set of constraints has to be regarded" [102].

"Scheduling selects among the alternative plans and assigns resources and times for each job so that the assignment obey the temporal restrictions of jobs and the capacity limitations of a set of shared resources" [39].

It is worth mentioning the OR perspective looking at the scheduling problem. The Operations Research treats the scheduling as a class of assignment problem. The main difference between these two approaches is that scheduling normally works on the discrete time-line [8] where the assignment is based on the continuous time-line. The assignment is supposed to be more specific than the scheduling problem [98]. Due to standardisation of the continuous time range all the allocation problems are treated as working on the continuous time-line [104]. In scheduling one can jump from one time-point to another where as in the assignment problem such jumping from different time-points is not permitted. But looking from the practical point of view almost every time the time-line is discrete in its nature as the jobs may get interrupted in between its execution and can start at some other time etc. [67, 69]. From the Operations Research (OR) background the classical assignment problem can be formulated mathematically as follows [104].

Minimise the total cost:

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} Cij \ . Xij \ , i = 1,2,..., \ n; \ j = 1,2,..., \ n$$

Xij = 1 if ith job is assigned to jth resource and 0 if not

$\sum_{j=1}^{n} Xij$ = 1 (one job is done by the ith resource, $i = 1,2,...,n$)

$\sum_{j=1}^{n} Xij$ = 1 (only one resource should be assigned the jth job, $j = 1,2,...,n$

Where, Xij denotes that jth job is to be assigned to the ith resource.

The classical techniques from OR such as PERT/CPM deals with the assignment problems quite efficiently, but they do not consider the resources and mainly deals with finding the exact start time of each job. Also, another important criteria in PERT/CPM is to find the jobs on the critical path, that is the jobs that are more sensitive in terms of its processing time and trying to accomplish such jobs as early as possible as compared to rest of the jobs.

So having defined the scheduling problem from different perspectives we will draw the few conclusions for realising the important concepts that lie at the core of scheduling problem.

A) Scheduling is a process where one needs to reason about the resources and time for assigning the jobs. This lies at the very core of scheduling problems, and looking from the AI-community this issue has drawn very little attention [105].

B) As opposed to the planning the scheduling problems are in many cases seen as a problem of combinatorial optimisation [30], and normally finding the legal (or what is referred to as an valid) schedule.

C) The scheduling problem frequently involves various types of choices. These choices could be ordering among the jobs (job-precedence), dependency relation between them, choosing the available resources that satisfy the need of the job, selecting the proper time-slot for the execution of jobs in order to evenly accommodate the assigned job etc. [58].

D) Almost every time the scheduling problems are restricted by the various kinds of constraints that limit the space of assignment of jobs to the resources. The constraints are usually separated in two main categories such as, *hard-constraints* and *soft-constraints*. The constraints are characterised based on their nature in the scheduling process. The hard-constraints are the kind of constraints, which can not be violated under any circumstances, where as the soft-constraints are the type of constraints, which can be relaxed if necessary during the scheduling process. There is another class of constraints called *preferences* that are usually treated as user-specific choices and they can be seen as a desirable rather than the obligatory one. The application of preferences can affect the evaluation criterion (cost-function) to the greater extent [91, 108, 128]. The examples of hard-constraints in scheduling are the capacity of a particular resource, the duration of a job etc. As the examples of soft-constraints can be meeting the due-date, usage of a particular resource for the execution of job etc. [129]. The preferences can be explained by the following example. For example, if job j chooses the use resource r_1 with preference x and prefers to use the resource r_2 with preference y. These preference specific criteria can affect the cost related issues because the alternative resources might have the different functional characteristics as compared to the original choice of the resource [118]. For example, different speed and feed of the milling, drilling machines, different load carrying capacity of the vehicles etc. that could affect the throughput of a schedule.

As discussed above the following figure depicts the taxonomy of various types of constraints in the scheduling environment [40]. The hard constraints are referred to as a non-relaxable constraint whereas the soft constraints come under the category of relaxable ones.

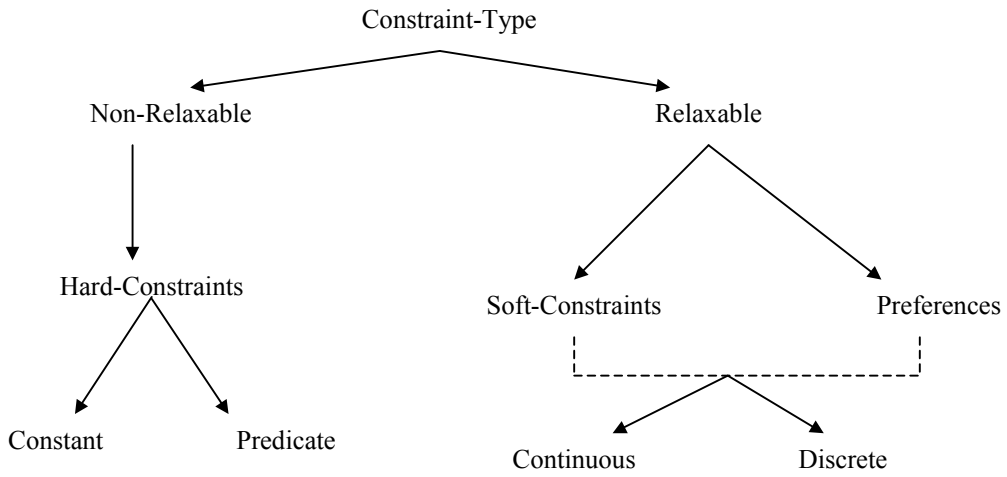


Figure 2. The Taxonomy of Constraints in Scheduling.

Coming back to the comparative study between planning and scheduling, we have seen from figure 1 that planning often gets influenced by the external environmental factors and produces the partial order of task. These partial order of tasks serves as an input for the scheduling task. It can be seen from the same figure that the planning task is mainly concern with the question of "*what should I do?*" whereas scheduling mainly deals with the question of *"how should I do it?"* The following figure depicts the scheduling task in its purest form.

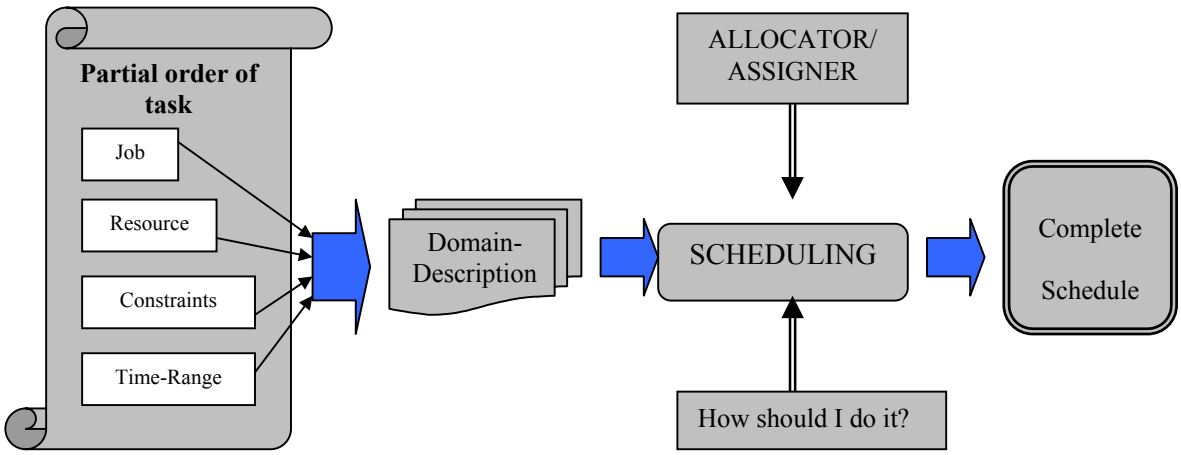


Figure 3. The overview of a scheduling task.

Up to this point we have treated the planning and scheduling tasks into two completely segregated spheres. However, in the real-life such distinction between planning and scheduling often gets blurred and to draw a distinct line between these two tasks is often difficult for several reasons.

1) Normally, the planning task can be seen as a *'long-horizon scheduling'*, or putting it in different words the scheduling can be often seen as a *'high-resolution planning'* [43, 8].
2) Planning is viewed as a generator of activities in the initial stage of a project during which the sequence of the activities are defined while the scheduling comes in the second stage

during which the generated activities are selected to allocate/assign over the resources and time. Thus, the planing deals towards the feasibility of actions as scheduling towards the optimisation of various aspects such as minimisation of cost or maximisation of resource usage etc. [101].

Despite the fact that planning and scheduling tasks have their separate existence as a research philosophy, they can be inter-linked with each other and thus forming the cohesive working environment. Such a type of environment is usually referred to as an *integrated planning and scheduling* environment [7]. The following diagram depicts the integrated planning and scheduling environment.

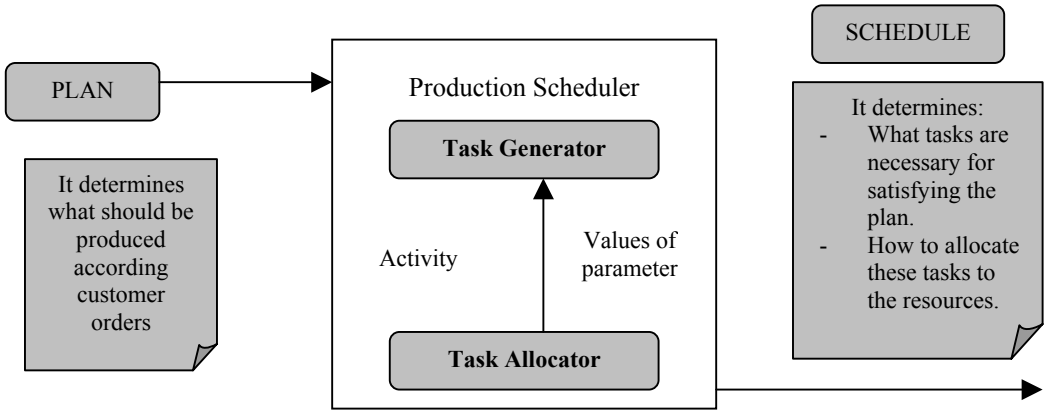


Figure 4. The integrated planning and scheduling environment

There is another dimension to the scheduling problem which is similar in nature as that of the planning task and during which it is seen as a *synthesis task activity* [46]. Talking more specifically, the scheduling problem comes under the category of *configuration task* [125]. In the configuration design process the task takes as input a set of goals that to be achieved and the interactions, and produces the complete specification of an artifact as an output. The configuration design process can be characterised as:

1) Usually the incomplete problem description is available in the initial stage and at this stage only the vague information is available about the task in hand.
2) Limited or delayed feedback from the world.
3) The cost is associated with the every action in the world. This specifically describes how the selection of particular actions could affect the cost related issues involved in the problem. Recalling the preference related criteria mentioned for the usage of particular resource and its impact on the cost. For instance, the square of the lateness of jobs could be treated as an aggregation of the cost-function.
4) An artifact usually functions independently of the planner, scheduler or designer.

There is one major difference between the configuration task and that of scheduling task is that almost every time the main building blocks of a scheduling task i.e., all the jobs, resources can be represented as a time-related entities [40]. To be more specific every job in

the scheduling problem is specified by the specific time interval during which the job must be executed. We'll discuss about the *time factor* in section 2.3 during which we will see the various time intervals that can be normally used for defining the jobs.

So looking from this perspective we can compare the design, planning and scheduling tasks saying that design task has its structure of components and connections. The components may be objects or processes, symbolic or physical. The planning task also has a structure but necessarily the sequencing one but not essentially a set of sequential structure. And if one needs to distribute the elements (actions, components) then it can be referred to as a assignment problem over a structure. If the structure is a plan then the assignment problem can be treated as a scheduling problem with added time element. If the structure is a design then the problem is a configuration task [16].

Also, it can be realised that the scheduling is a process in which the components are described as a dynamic set of elements and the requirements and constraints imposed on these dynamical set of elements is specified from either functional or from the global viewpoint. The global point of view of expressing the constraints specifies the need for optimising the solutions to the scheduling tasks. There has to be a mutual consistency between the local and global constraints in scheduling. To explain it in more detail, once the scheduler has satisfied all the local constraints then he/she has to be aware about the effect of local constraint satisfaction on the global constraints. As some of the locally satisfied constraints can violate the global constraints which might have even more adverse effect on the overall schedule performance.

There are various types of problems that can be efficiently tackled by the planning and scheduling task. The following Venn diagram depicts the possible problem types that can be tackled by the application of planning, scheduling, and design [91].

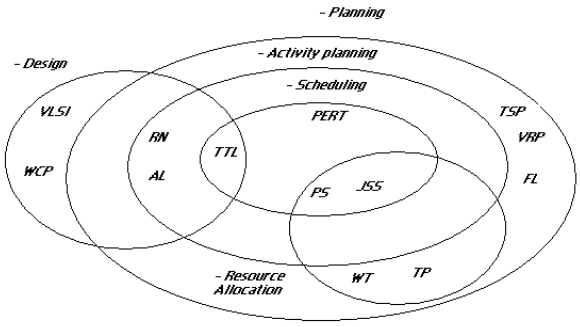


Figure 5. The possible application areas of planning and scheduling.

AL – planning an automobile assembly line.

FL – facilities location problem.

JSS – Job-shop scheduling.

PP – project planning.

PERT – unlimited project-resource scheduling.

PS – project scheduling.

RN – designing a new railway network.

TP – classical transportation problem.

TSP – travelling salesman problem.

TTL – time-tabling problem.

VRP – vehicle routing problem.

VLSI – doing a VLSI layout.

WCP – writing a computer program.

WT – allocation of weapons to targets.

It must be kept in mind that this classification is not crisp as the problem from one category may fall into more than one category. In particular the scheduling problem can be categorised into three broad categories such as, *the pure scheduling problem*, *the resource allocation problem* and finally *the joint scheduling and resource allocation problem*. The following points describe the each problem areas in more detail.

- The ***pure scheduling problem*** concentrates on the scheduling task in the manufacturing environment, e.g. classic job-shop scheduling and other variants of the job-shop scheduling such as open-shop and flow-shop scheduling. The flow-shop scheduling is stricter as compared to the job-shop scheduling where maintaining the sequence of jobs is quite critical. The pure scheduling problem can be further classified into three sub-groups indicated by $\alpha|\beta|\chi$, where α indicates the *machine environment*, β indicates the *job-characteristics* and χ denotes the *optimality* [96].
- The second category is the ***pure resource allocation*** problem [63] in which the demand for the resource is known beforehand and the problem is mainly to allocate the jobs to resources in time. Some of the typical examples of the resource allocation scheduling are aircrew to the flight gate assignment, the room allocation problem and sometimes a classical travelling salesman problem [110].
- Finally, the last one is the ***joint scheduling*** and ***resource allocation*** [63] problem in which the scheduler mainly decides which jobs to perform and when, and which resources needs to be available for satisfying the request of jobs.

Now in the next section we will discuss the significance of the *time element* in the scheduling environment, as it is one of the important components as far as the scheduling domain is concerned.

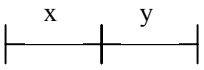
2.3 Time in Scheduling

Another important issue that raises the significant importance in scheduling is the representation of time. There are various time intervals that can be related with the basic

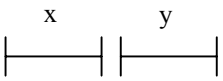
elements of scheduling task. The proper representation of the time is a key in constructing an efficient schedule.

In scheduling problem the time is usually related with the jobs or resources. It determines when a particular job can be carried out and with which resources. The jobs have a time slot, which determines when a particular job can start, finish and what could be the duration of that job. Simply assigning jobs to the resources is not sufficient but determining the time window in which it must be assigned by satisfying various constraints is also important. For instance, the precedence (job-precedence) relation between the time ranges of any two jobs, availability of a resource for the execution of jobs based on their time range overlap, the due date of jobs etc. are also equally important factors.

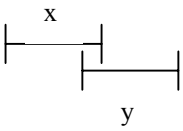
Another type of time range, which plays equally significant role in scheduling, is the time range of a schedule itself. The schedule time-range represents the time period for which the schedule is constructed. The schedule time-horizon imposes the constraints on the time-range of an individual job in terms of its start time and end time. There is some work that can be found in the literature for representing the intervals between any two elements [2, 89]. These standard time intervals can be adopted for representing the time relations between any two jobs such as J_1 and J_2. In the following figure the *x* and *y* represents the two time intervals.



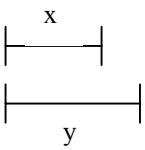
Time-range of J1 ***meets*** time-range of J2.



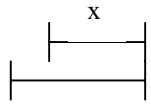
Time-range of J1 ***is before*** time-range of J2.



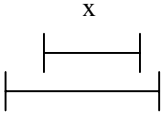
Time-range of J1 ***overlaps*** time-range of J2.



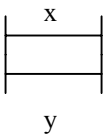
Time-range of J1 ***starts*** same as time-range of J2.



Time-range of J1 ***ends*** same as time-range of J2.



Time-Range of J1 is during the time-range of J2.



Time-range of J1 is equal to time-range of J2

Figure 6. The possible time-intervals between any two jobs.

Along with the representation of the time intervals we conclude the role of time in the scheduling task. Finally, table 1 summarises the main conceptual differences between the planning and scheduling tasks.

Table 1. Differentiating between planning and scheduling.

PLANNING	***SCHEDULING***
The planning task mainly deals with <u>WHAT</u> actions need to be carried out in order to achieve the final goal-state of the world.	The scheduling task mainly deals with finding out <u>WHEN/HOW</u> to carry out the actions to optimise the criteria.
It mainly concerns with reasoning the consequences of acting in order to choose among the set of possible courses of actions. E.g. A plan must consider a possible set of actions available and look for their consequences and choose one action that satisfies most of the requirements.	It is mainly concern with mapping of the various sets of tasks to the available resources for the specific time interval while satisfying the constraints. E.g. Assign *task A* to *machine A* and *task B* to *machine B*. The duration of *task A* is *x min.* and that of *task B* is *y min.* etc.
<u>*Input to the planning task:*</u> 1) A set of possible courses of actions. 2) A predictive model for underlying dynamics. 3) A performance measures for evaluating courses of action. <u>*Output to the planning task*</u>*:* One or more courses of actions that satisfy the specified set of requirements for performance. E.g. a travelling salesman problem. a) A set of possible courses of action is a set of travel options: air-flights, car, railway etc. b) Dynamics: say the travel dynamics, i.e., information regarding travel time, cost and the way travel time and cost affects	<u>*Input to the scheduling task:*</u> 1) A set of tasks to be assigned. 2) A set of available resources for the execution of task. 3) The capacity of the available resources. 4) The time intervals of the specified task. 5) The constraints imposed on the task. The constraints could be of two types such as hard-constraints and soft-constraints. The hard-constraints should not be violated under any circumstances whereas soft-constraints has to be satisfied by the completion of the schedule. <u>*Output to the scheduling task:*</u> The output to the scheduling task is a schedule that assigns the given set of tasks

each other. How external world conditions affect such actions. For example, weather conditions etc. c) A set of requirements specified by the external world. E.g. in city A on Monday, and Tuesday, in city B from Wednesday afternoon till Friday night. And in this case constraints on solution would be start no earlier than Sunday arrive before Saturday night etc., maximum expenditure should not exceed £1000.	on the available resources by maintaining their time intervals without violating the constraints. E.g. In job-shop scheduling set of inputs would be set of jobs such as drilling, milling etc. and available resources would be machines on which these jobs can be executed. The constraints can take the nature of no resource is assigned more than one job at a time, each job is completed before he starting of the next job (precedence relation among jobs) etc.
Planning can be stated as Satisfying (find some solutions satisfying the constraints) or finding the feasible set of solutions transferring the initial state of the world into the goal state.	The scheduling task is normally seen as a optimisation task over one or more objective functions such as minimisation of the cost or maximisation of the resource utilisation etc.

Now in the next section we'll see some of the most significant intelligent scheduling systems developed over the past two decades or so and the various techniques that can be useful for solving the scheduling problem.

3. Intelligent Scheduling

As it was mentioned earlier, the scheduling did not receive the serious attention till 1980 until Fox et al. started working on the first intelligent scheduling system named ISIS at Carnegie Mellon University. Since that period several researchers from AI community have developed various intelligent scheduling systems in order to deal with the problems from specific application domains. These scheduling systems have exploited various tools and techniques from artificial intelligence for successfully and efficiently solving the scheduling problem.

In the following section first we will review the various techniques that can be useful in order to deal with the scheduling problem. In the second part we will review the nine intelligent scheduling systems that have been influential over the period of last two decades and try to find out the differences in their underlying working philosophy. It includes ISIS [39], OPIS [93, 107], SONIA [25, 62], YAMS [122], FlyPast [79], S2 [32], DAS [15], REDS^2 [50] and ESCALAS [78].

3.1 Various Techniques for Scheduling Problem

The scheduling problem has been invested for a long time in Operations research and Artificial Intelligence. Various approaches have evolved over the period to solve the scheduling problems effectively. In the following section we will briefly see the various techniques developed for the scheduling task.

3.1.1 Heuristic Scheduling Approach

The heuristic scheduling approach is mainly based on the heuristic search techniques. The general heuristic knowledge as well as the problem (domain) specific knowledge is adopted from the scheduling experts in order to guide the search procedure. In the heuristic scheduling constraints play a major role in order to guide the search effectively and the knowledge representation schema used is normally rule based or frame based. The heuristic scheduling is either job-based or resource-based. In the job-based approach [58] usually the job is assigned to the resources and in the resource-based approach [68] the resource is selected for assigning the next potential job. Various strategies and rules are used in order to select the next potential unscheduled job, or in order to select the available resource and the time slot as well. For example, schedule the job that has earliest start-time and/or earliest due date first, job with least processing time etc. These kinds of rules are referred to as priority rules that are derived from the Operations Research. The algorithms differs essentially in the problem-specific heuristic knowledge that is applied for determining the appropriate decisions during the problem solving procedure and are specifically tailored to just one special problem scenario.

3.1.2 Constraint-Based Scheduling

In any scheduling problem the constraints are the most important part. The constraints normally limit the space of the valid solutions to the problem. Therefore, the constraint-based approaches have been rapidly become popular [9, 64]. The constraint-based scheduling has two important parts; one is the *constraint-formalism* [39] that is, representational aspect of the hard and soft constraints in the problem. The hard and soft constraints are useful in order to guide the search process effectively as it restricts the possible solutions to the problem.

The second part is the *general constraint programming* technique that actually solves the problem. The most widely used technique in constraint-based scheduling is the constraint-satisfaction problem (csp). In the constraint satisfaction problem the schedule is represented as a set of variables of certain domains and a set of constraints for restricting these domains. The generic framework for the constraint-directed scheduling is presented by Beck and Fox [9], in which they have analysed the various approaches for constraint-based scheduling. Talking particularly for the scheduling, the variables in constraint-based represent the jobs

and the values represent the resources or the time ranges assigned to the particular job. The feasible solution would be to satisfy all the constraint corresponds to the particular node in the constraint graph. The constraint-directed search explores the problem space based on relationships, dependencies and limitations among the various problem objects. The simple procedures used are Generate & Test or a Backtracking strategy without constraint propagation. The system stops when the first valid solution (solution that satisfy all the constraints) is found. Many traditional intelligent scheduling systems are based on the constraint-based approach for solving the scheduling problem such as, ISIS, SONIA, DAS etc.

3.1.3 Fuzzy-Approaches for Scheduling

The fuzzy approach provides the possibility in order to cope with the incompleteness and dynamic behaviour of scheduling. By using the different fuzzy sets, linguistic variables and the fuzzy rules (inferences) it allows to represent the vaguely formulated knowledge. Using the fuzzy approach the following type of information can be represented efficiently.

- Uncertain values for the scheduling parameters e.g. process time.
- Vague specifications of preferences, e.g. preferences between various kinds of alternatives.
- Uncertain definition of the due date and durations of the jobs etc

In order to represent the imprecise or uncertain information following steps are carried out.

1) First the scheduling data is transformed into a knowledge representation for the fuzzification. The uncertain knowledge is represented by the linguistic variables such as, very low, low, high, very high etc.
2) The actual processing is done based on the fuzzy scheduling knowledge by using the rules along with the integration of fuzzy arithmetic. These are stored in the knowledge base of fuzzy controller.
3) Final step is transforming the fuzzy scheduling decisions into the crisp scheduling data for determining the concrete knowledge.

3.1.4 Neural Network Approach for Scheduling

Traditionally the neural networks are used for forecasting, classification, pattern recognition and data mining and very small number of applicability have been found in the scheduling domain. In scheduling the neural networks can be used for the purpose of optimisation approach. Usually, it can be used for the predictive scheduling as in the reactive scheduling it is quite sensitive to changing situations. If the situation changes slightly then the new

network needs to be designed for addressing the changed situation. In scheduling it uses the following schema for problem solving by decomposing the main problem.

- Choose a variant for every job.
- Choose a resource for every job.
- Choose a time slot for jobs and resources.

Various kinds of networks can be used in the above-mentioned steps such as, Hopfield-Tank network can be used in the first step to represent the order of jobs. Each neurone represents the assignment of jobs to the resources in which only one job can be selected. Then in the last step the LP-network can be used in order to represent a linear programming approach for assigning the time slot to the job for minimising the tardiness. The neural network can give a good solution as compared to the heuristic approach but the sensitivity of the neural networks to the changing situation restricts its applicability.

3.1.5 Iterative Improvement Techniques for Scheduling

The major difference between the first four approaches and the iterative improvement technique is that the first four approaches work on the partial solution whereas the iterative improvement deals with the complete but flawed solution. Once the complete solution or set of solutions is constructed then iterative improvement technique selects one solution and improves it in order to find the optimal or sub-optimal solution by using iterative improvement technique [30]. The technique is based on the hill climbing approach that avoids getting trap in the local optima. Some of the examples of iterative improvement techniques are Genetic Algorithm (GA), Simulated Annealing, Taboo search, Threshold Acceptance, Iterative Deepening and Grand Deluge algorithms [102]. Most of these techniques except GA are using the quite straightforward operators to find the neighbourhood solutions for the modification e.g., exchange few assignments or try alternative resources. The procedure iterates until the specific stopping criteria is met. In the GAs the following three steps are carried out. First, the subset of each variable is elected for the *recombination*, then at the second step with the *crossover* operator new individuals are created and finally at the third step, the mutation operator selects the new individual with slight changes.

3.1.6 Distributed Scheduling

The main philosophy on which the distributed systems are based is the co-operative behaviour of the problem solving. In the distributed scheduling the systems are usually based on the co-operative intelligent agents. Each of the agents in the system is responsible for tackling the specific problem (task) in the complete scheduling system.

The main goal in the design of the distributed scheduling systems is that the division of responsibility of the complete system among the agents and at the same time deciding the responsibilities and facilities of an individual agent, the way each agent will communicate with each other etc. Several alternatives are possible to distribute the task among agents. One way of dealing with it is by representing the agents as a sub-system in the complete system's architecture (c.f. 3.2.7). Alternatively, the agents can be represented as a single object in the scheduling system that is, a job agent is responsible for performing all the tasks that are related to the specific job etc.

As mentioned before every agent needs to perform the certain set of tasks for which it is responsible. Usually, the control agents are necessary to control the overall performance of a system and they have the responsibility for achieving the global goals of the complete system. In order to establish the communication among different agents two types of architectures are commonly used, consist of, the blackboard system and the contract nets.

- The backboard system: it has all the data collected at the central place for the communication. Each agent reads the data from the blackboard and checks if the agent is capable of executing the task, if so then the agent records the result on the blackboard.
- The contract net: in the contract net all the agents are using the structural protocol to communicate with each other.

3.1.7 Meta Scheduling

Even though over the period vast number of algorithms or problem solvers have been developed still it is difficult to find the appropriate algorithms or problem solver for the given situation. One of the main reasons for this obstruction is that most of these approaches are developed to suite the particular problem type, e.g. job-shop scheduling, crew scheduling etc. As the nature of the problem area changes such type of problem solvers are no longer useful because of their brittle nature. It was the main motivation behind the development of meta-scheduling approach. Usually, in Meta scheduling the chucks of knowledge is identified that can be applicable to cover the wider problem area. It makes possible to explicitly represent the extracted rules from the different problem solvers. Extracted strategies can be described as a generic problem solver or skeleton. The dynamic combination of abstracted strategies/rules/skeletons is called as *dynamic scheduling knowledge*. These dynamic components can be used to form the libraries of generic components in order to deal with the various scheduling problems.

3.2 Intelligent Scheduling Systems

Here, we will review some of the most influential intelligent scheduling systems developed over the period of last two decades. The main focus of the review will be on the working

philosophy, problem-solving nature and general architecture of the intelligent scheduling systems.

3.2.1 ISIS

An ISIS [39] scheduling system is developed by Mark Fox and is the first reported expert scheduling system. An ISIS is developed particularly for the job-shop scheduling application by using the frames, also known as units and it is based on the constraint-based approach as a problem-solving strategy. In the job-shop scheduling various objects can be founds and the constraints imposed on them or among them is known as a *units*. Each of the constraints is represented as a separate distinct unit and has an associated utility. The utility factor is taken into consideration as a measure for analysing to what extent the particular constraint contributes in building the final schedule. The ISIS decomposes the scheduling task into four-tier hierarchy as shown in the figure 7.

At the first stage the order/lot is selected for the release on the shop floor. Once the lot is selected and is ready to be released on the shop floor then the capacity analyser determines the start time for each job within the selected lot. After this the actual scheduling is performed at the resource analysis stage. In which the resource on which the job is going to be assigned is checked based on various criteria such as, the availability, the capacity etc. And finally at the reservation stage the actual resource is reserved for the execution of the job. The reservation is made in such a way that it reduces the *work in process* inventory.

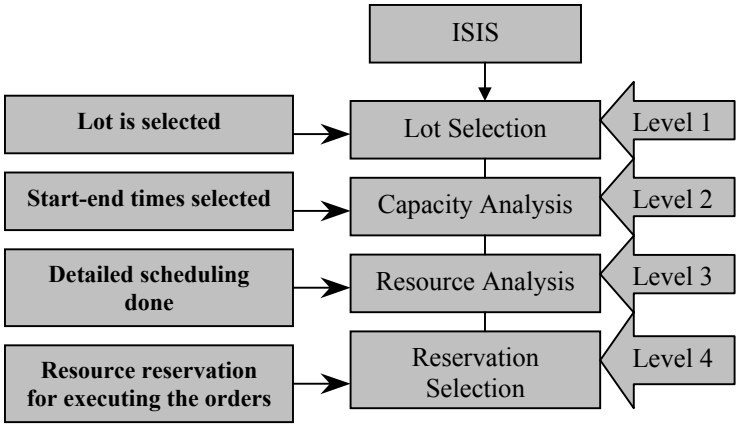


Figure 7. The four-tier hierarchy of ISIS

So based on these criteria the ISIS takes the job-based perspective i.e., the scheduling is considered as the procedure in which the job is assigned to the resource. The resources may remain unassigned by the end of the schedule if all the orders have been assigned. This helps to produce the schedule with acceptable levels of job tardiness (due-date violation by the job).

The ISIS takes the hierarchical constraint-directed search concept in which the evaluation function is dynamically constructed at each state in the search space. Every type of constraint contributes both an importance and the utility factor in the construction of a final schedule. The constraints are resolved by extracting them from the resources or jobs defined in that particular state. The performance of a schedule is determined by the two factors such as the tardiness of number of jobs, number of constraints that are satisfied by the schedule completion.

3.2.2 OPIS

An OPIS [93, 107] is a contribution of the ISIS initiative. An OPIS (OPportunistic Intelligent Scheduler) is implemented as blackboard architecture [26, 88] and uses multiple perspective as an assignment strategy. It does the assignment from both job-based and resource-based perspectives. The work to be scheduled and the state of the shop-floor are analysed to detect the areas of high resource contention (bottlenecks). The bottlenecks can be seen as resources that are always under high demand and it is not permitted to have these resources unavailable at any stage during scheduling otherwise it could make the other resources with increased load. The bottlenecks are then scheduled from the resource-based perspective in order to avoid the bottlenecks during the assignment process and this anchors the search space particularly based on the bottleneck areas. Once the critical resources, which are susceptible to the bottlenecks are determined then the search is guided based on the job-based perspective around the anchored points.

The OPIS employs an incremental and opportunistic strategy for solving the scheduling problem, the very nature of the blackboard architecture. The knowledge that can be applied to the scheduling task is distributed across the independent knowledge sources (KSs), the application of these knowledge sources is determined dynamically and opportunistically as the problem solving evolves. By decomposing the knowledge into independent module i.e., knowledge sources, the blackboard architecture allows the system to be executed and as the new knowledge becomes available, this new knowledge is added to the system in the form of knowledge sources. The top-level manager (TLM) is responsible for holding the co-ordination of scheduling task. The OPIS is designed mainly for the reactive scheduling rather than the predictive scheduling systems. In the reactive scheduling problem the schedule needs to be updated under the various real-life situation in the environment in which a schedule is executed. For example, in the steel-manufacturing first the predictive schedule is generated based on the known data in hand. This primary schedule is sent on the plat floor and then if required the initial schedule is undated (modified) due to unexpected occurrence of the events such as, the unavailability of the resources or changes in the due date of other orders etc. A predictive scheduling is more deterministic in nature as it depends on the static

data about the problem in advance. Following diagram depicts the scheduling architecture of OPIS.

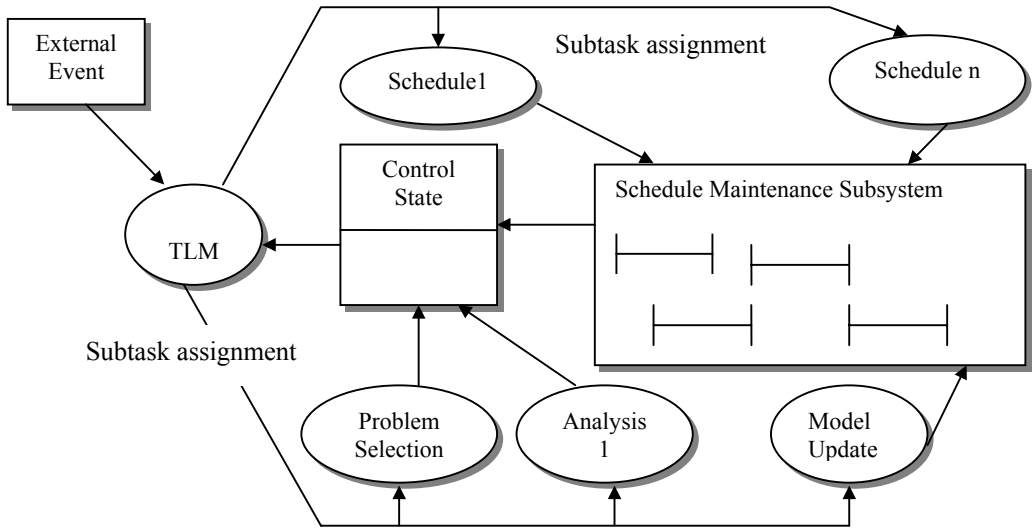


Figure 8. An architecture of OPIS [128].

3.2.3 SONIA

The SONIA [25] scheduling system is a continuation of the SOJA [62], work by Le Pape and Sauva. The main difference between SONIA and other scheduling systems is that SONIA integrates both predictive and reactive scheduling approach. SONIA works on the blackboard architecture similar to the OPIS. The predictive component of SONIA results from Le Pape's previous experience with SOJA system, and it comprises a selection component and an order component. The selection component selects the jobs to be scheduled and bind the selected jobs with the available resources. This approach is similar to the Norman Sadeh's work [100] on variable and value ordering heuristics for job-shop scheduling as a constraint satisfaction problem. The ordering component consists of an iterative constraint satisfaction process [63], for imposing the temporal constraints on selected jobs. The heuristic rules are used for making the various ordering decisions, which are then propagated through the schedule management system. If the ordering component encounters a failure it then enters into a reactive process and this is the place where SONIA scheduling system differs from OPIS. SONIA efficiently exploits the reactive component during the actual schedule creation. The scheduling decisions are first made with the predictive component, and then the backtracking takes place via the reactive component. Therefore, predictive and reactive components are viewed as being the same problem during schedule construction instead of the isolated one.

The blackboard system of SONIA is mainly consist of following three components:

- **The knowledge sources:** The knowledge that is needed to give the reasoning to the problem is partitioned into *knowledge sources*, which are kept separate and independent.

- **The blackboard data structure:** The compiled problem-solving state data is kept in the global database, the *blackboard*. The knowledge source produce changes to the blackboard that leads incrementally to a solution of the problem. The communication and interaction among the different knowledge sources takes place only through the blackboard system.
- **The control cycle:** There is no explicit control component specified and the knowledge sources are self-actively and opportunistically changes the blackboard. But it is often the case that control component is required to decide which instance of the knowledge source needs execution at each point in the problem-solving process.

The SONIA is based on the macro-opportunistic scheduling technique, i.e. each module makes a collection of decisions each time it applies. For example, an application of the module might consist of scheduling the complete manufacturing order or a complete set of resources [107]. In such case, the control problem consists of both ordering the contributions of the various modules and controlling the behaviour of each module every time it is applied. Finally, SONIA uses constraint-propagation as its problem-solving strategy.

3.2.4 YAMS

YAMS (Yet Another Manufacturing Systems) [122], is probably the first intelligent scheduling system that exploited the distributed artificial intelligence. YAMS's architecture is based on the Contract Net. In the Contract Net modules, the transfer of control is in a distributed system using the metaphor of negotiating among the agents. The agents within the Contract Nets are categorised as being *managers*, *bidders* or *contractors*. Each of these agents plays a certain role in the contract net. A manger identifies the work to be done and delegates that work to the agents via the process of negotiation. A bidder is an agent that offers to perform a task, and a contractor is a successful bidder who wins the contract. The agents in this system communicate by message passing. If there is any potential task to be performed, a manager agent makes the task announcement, and broadcast the task to all the agents in the network. The agents that have a potential to perform the announced task contact the manager with the bid message, and the highest bidder is awarded the contract and becomes a contractor. Similar to first approach the contractor agent then decompose the task in hand into further smaller subtasks and acts as a manager making its own announcement. Consequently, an agent can be a contractor and a manager at the same time. This style of problem solving has been referred as a *fractal* style of solving particular problem [122]. YAMS model the overall factory as a hierarchical workcells, where individual workcell in the factory is considered as an agent within that contract net. Each node in the level of hierarchy indicates the particular level of granularity, whereas the leaf agents correspond to discrete resources. Also, each node in the hierarchy has the information about its capabilities, and

represents it in a declarative style. Apart from this another important issue is that of a global schedule which is performed by the external system and then distributed across the net. Because of the dynamic nature of the scheduling system such a global schedule gets obsolete quickly. In such scenario the agents are allowed to perform a degree of local schedule. For creating a local schedule a *turnpike* [75] scheduling strategy is adopted, i.e. local perturbations to a local schedule is allowed, but the goal is always to return to the global schedule whenever possible. Also, the important issue worth mentioning here in the context of Contract Net is that the communication within the contract net takes place only for *superior-to-subordinate* and *peer-to-peer* communication is not allowed. But the peer-to-peer communication is required in order to propagate the constraints within the agents. The major problem with such architecture is that, when a local scheduling decision is made, there are chances that it may have the global consequences, and if the effects of the local decisions are not propagated to the global scheduling the global schedule can become obsolete quickly. Hence, this kind of Contract Net is only applicable when the problem in hand is already decomposed into the sub-problems, which is referred to as a completely accurate, nearly autonomous strategy [71].

3.2.5 FlyPast

The FlyPast [79] scheduling system was mainly designed for allocating the aircrew and aircraft to flights within the dynamic environment. FlyPast generates naval flying programs for a task force, i.e. air cover for a fleet of warships. And it is completely based on the reactive scheduling approach by using the constraint-based reasoning and the assumption-based truth maintenance system (ATMS) of de Kleer. The nature of generating a flying programme is simplified in the sense that system assigns the resources to flights after the timings of particular fights are completely specified. The temporary constraints are removed totally from the scheduling approach and the remaining decisions such as the allocation of resources that is, aircraft crew, to the jobs i.e., flights, are considered as a constraints. Characteristically, there are various constraints imposed on the number of flights. For instance, the flights can take off or land within the given time interval, constraints on the flying hours between maintenance operations, and the constraints on the number of flying hours a crewmember is supposed to perform with the given time interval etc. This problem typically can be seen as a constraint satisfaction and is represented by the constraint graph, where each node represents the flights, and the domain of a node as a possible resources that can be assigned to the node, whereas the arcs are the constraints acting between nodes. The forward checking [53] approach is taken in order to perform the search. As the forward-checking approach is applied, ATMS nodes are created for any domain reductions that take place, where the datum of an ATMS node is the domain reduction resulting from forward-

checking, and the justification of that particular ATMS node is the assignment that mainly caused the reduction. If the forward checking results in *total destruction* of the domain then a 'no good' solution can be derived and dependency-directed backtracking takes place. If no satisfactory results occurs then the 'no good' database is analysed to deliver an explanation to the user. Also, at the same time the user of the system is allowed to interact with the system for adding and retracting the constraints and forcing the allocations on the system.

3.2.6 S2

S2 [32], scheduling system is similar in nature to that of FlyPast and is designed for the VLSI wafer fabrication. The problem domain of S2 is considered to be similar to that of job-shop-scheduling. S2 recognises that the problem domain is highly dynamic and uncertain in its characteristics, and so the designed scheduling system must address to its domain directly. Also, there is another explicit assumption on which the scheduling system is built is that there is no single performance measure on which the final schedule can the measured, especially so when that the schedule must exists within an uncertain problem domain which is as mentioned above. Thus, due to these factors the S2 sees the schedule as one of the problem of satisfaction within an open world. Consequently, the S2 is a reactive scheduling system and the final schedule delivered is satisfactory. S2 is composed as three main modules such as, *a constraint maintenance system*, *a schedule generator*, and *a request interpreter*. All the scheduling problems are represented within the constraint maintenance system (CMS), where the CMS perform the constraint propagation when the constraints are imposed or retracted. The final schedule generated within the CMS tries to satisfy the constraints within the CMS. As the schedule generator is reactive in nature it reacts immediately to the addition and retraction of the constraints by modifying the existing solution, instead of performing the complete rescheduling from scratch. All the process of determining a solution, a constraint satisfaction problem (CSP), changing the structure of the problem accordingly, and solving the modified problem is referred to as an *incremental constraint satisfaction*. S2 employs a depth-first search strategy and the dependency-directed backtracking. Also, it keeps the record of all the dead-end states encountered during the search along with the sources of their inconsistencies. A hard-wired ATMS is applied, and the 'no good' database is distributed across the soft-constraints.

As compared to FlyPast the S2 also delivers the satisfactory schedule and then allows the user to modify the schedule either by adding or retracting the constraints via the request interpreter. Again there is one pre-assumption on which the S2 works and is that the user of the system has the knowledge about how a satisfactory schedule looks like and thus the user of the system can guide the whole scheduling procedure towards a *good* schedule. So looking from this perspective the user can be seen as a major source of knowledge that is not implicit

in the S2 system and also the source of world dynamics. This kind of scheduling approach is referred as a *funk box* scheduling by Elleby et al. [32].

3.2.7 DAS

The DAS [15], which stands for Distributed Asynchronous Scheduler is developed at the University of Stracthclyde. The DAS distributes the scheduling problem on three-tier hierarchy of the problem-solving agents. At the lowest level of hierarchy, is the *operational level* at which the O-agents are responsible for scheduling the operations on individual resources for example, machines etc. Then the middle level is the, *tactical level*, in which the T-agents are attached with the aggregate resources and load balance operations among subordinate O-agents. Finally, the top level is the *strategic level*, in which the S-agents introduces the work into schedule and they have unlimited control over the conflict resolution. The S-agents are allowed to relax the problem if necessary according the environmental conditions. All these agents can act asynchronously and the constraint propagation takes place through message passing. Various mechanisms are existing in order to focus the control and to resolve the conflicts among the agents as they occur. The figure 9 depicts the three-tier hierarchy of DAS.

DAS scheduling system can be seen as one of functionally accurate communication architecture (FA/C) [71] system from the distributed AI perspective. The reason why it is accurate communication architecture is because the local decisions made by the O-agents are always locally accurate. However, these local decisions can have global effects so these local decisions are distributed among agents via communication passing by using the constraint propagation strategy. Like other scheduling systems such as, FlyPast and S2, DAS also works as a reactive scheduling system.

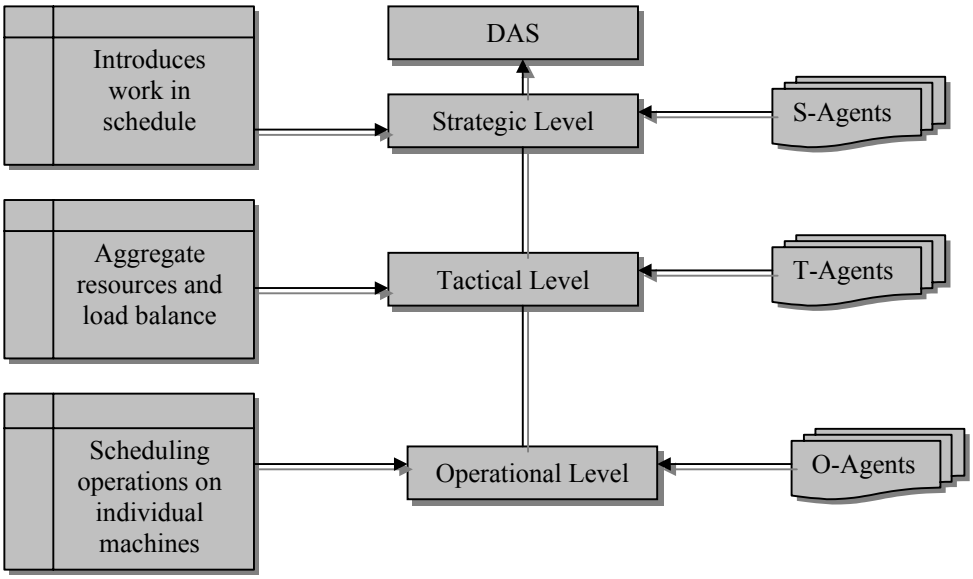


Figure 9. The three-tier architecture of DAS.

3.2.8 REDS^2

The REDS^2 scheduling system can be seen as a logical successor of DAS, which also works on distributed scheduling approach. The system is developed by Hadavi et al. [50] and it provides a sound reason why the scheduling system should be distributed if it needs to be. If we assume that the scheduling problem is of optimisation, still it is not clear what exactly needs to be optimised in this scheduling process. By distributing the scheduling problem across the group of agents, we might allow these agents to optimise on various different criteria. Most of the architectural behaviour of REDS^2 is similar in spirit to that of DAS. But the above-mentioned argument about the distribution of the scheduling problem among various agents is fully exploited here as opposed to DAS. In REDS^2 both predictive as well as reactive scheduling are incorporated along with the release control strategy named FORCE [51].

3.2.9 ESCALAS

ESCALAS [78] is a crew scheduling system developed for the Portuguese railway network. If the railway timetable for the railways is given to ESCALAS it distributes the crews to the railways in such a way that constraints associated with it does not get violated and the output schedule is *adequate* in some sense. The scheduling process works in various phases, the first phase of scheduling behaves as a pre-scheduling in which the unreasonable and undesirable schedules that subject to violate constraints are ruled out. This phase can be seen as a filtering of the domain in constraint satisfaction problem [9]. In the actual scheduling process the system searches for good solutions by using A* algorithm with heuristics to limit the number of successors generated and which is essentially a beam search.

The most interesting point about this system is that it can run in three different modes; *the manual mode*, *the automatic mode* and *the semi-automatic mode*.

In *the manual mode* the user of the systems makes all the necessary decisions about the schedule and then the system checks the legality of these decisions. In other words, the system makes sure that the decisions made by the user are not violating any of the predefined rules or constraints in the schedule. In *the automatic mode* the system itself explores all the search space using a scheduling strategy by using the above mentioned search strategy. And in the *semi-automatic mode*, it's a two-way communication between the system and the user, as the system generates the successor nodes and the user selects one of successor nodes generated by the system. Similar to the S2, here the user can be seen as a knowledge source of the heuristic knowledge. The developer of this systems claims that their system is really a decision support system in the domain of human resource management and the goal of the scheduling systems is not to replace the expert human schedulers but rather to help them in

making their decisions effectively. Also, they claim that their system enjoys the following three major advantages in having a scheduling system like this.

- This is a uniform scheduling process, as opposed to those produced solely by the human schedulers that may replicate different scheduling philosophies.
- The complete schedule can be generated within a fraction of time as opposed to the traditional trial and error process of producing the *feasible* schedule.
- The schedule generated by this way is an error free schedule.

They also claims some advantages from the end-user's point of view such as:

- Alternative schedules can be quickly generated, by using the different scheduling strategies or based on different scheduling criteria.
- The evaluation of a scheduling cost (i.e. the cost of the schedule solution), with a full set of statistics.
- The user of the system can also study the consequences of changes in labour rules etc. such as "what-if" scenarios.

3.3 Summary so far

If we examine the development of various scheduling systems that are evolved over the period of last two decades since the development of the first intelligent scheduling system, ISIS, it can be observed that it as an evolutionary trend of the development of these systems. ISIS that was the first truly intelligent scheduling system developed by using the techniques from artificial intelligence and knowledge-based engineering for tackling the job-shop scheduling problems. It has opened a wide stream of research in the domain of scheduling. Then OPIS explored the both order-based and resource-based scheduling strategy by using the blackboard architecture. But SONIA, which was the immediate successor of SOJA, truly integrated schedule creation with schedule maintenance by treating the two as a common problem. On the other hand FlyPast and S2 systems directly addressed the schedule maintenance for the already constructed schedule. Both of these systems are based on the assumption that the scheduling problem exists within an open world thus scheduling system must address the important issues from the open world. They are based on constraint-satisfaction and truth maintenance technologies as their problem-solving strategy respectively. YAMS was the first one to use the distributed scheduling system. It was based on the philosophy that scheduling system must model the organisation as being distributed and as YAMS is based on a distributed environment it does not have a predictive capability. Similar to YAMS, the DAS and REDS^2 are both distributed scheduling systems and are similar to FlyPast and S2 in terms of reactive scheduling and multi-perspective as that of

OPIS. Finally, ESCALAS scheduling system can be thought of as being technologically traditional, though it is significant in an unusual justification.

All the above mentioned systems are working efficiently in their own application areas. But these systems are hard-wired in nature as these are developed by keeping some specific targeted scheduling applications in mind. Obviously it was inefficient for using them in order to solve the problems from different applications because of the brittle nature. Slowly the need for generic system components started arising in the systems development for efficiently *reusing* in wider applications. The solution of the reusability question came in terms of developing the ontologies and libraries of problem solving methods that can be efficiently reused to tackle the problems from wider applications.

In the next section we will see some of the basic concepts in the field of knowledge engineering.

4. Knowledge (level)- Acquisition, Sharing and Reusability – a new trend

In this section we will understand the basic notions involved in the field of knowledge engineering especially looking at the knowledge acquisition and modelling perspective. First, we will see the various concepts that were introduced for the efficient representation of the knowledge and to increase the potential reusability. In section 4.5, we will turn our focus towards domain of interest i.e. knowledge acquisition, modelling etc. in which we will look at the various knowledge modelling frameworks that are influential over the period of last decades or so. While doing so we will opportunistically point out the strengths and weaknesses of each of these modelling frameworks.

4.1 Introduction

There are three major terminologies can be found in the area of knowledge engineering such as: knowledge, knowledge acquisition as knowledge modelling, knowledge sharing and reuse. In 1982, Allen Newell has introduced the notion of what is called as a knowledge level in his article 'The Knowledge Level' [87], the same article has identified the capabilities of a cognitive systems and revelled how the contents of the system are independent of its implementation. To put it in other words what the author has proposed is "*Knowledge is to be characterised entirely functionally, in terms of what it does and not structurally, in terms of physical objects with particular properties and relations*". In the same article the author has proposed the '*principle of rationality*' stating that, if the agent has the knowledge about choosing particular action among the several available actions, which can lead towards the solution or goal state then the agent will choose that particular action.

This article has removed all the pre-existing ambiguities by answering the fundamental questions in the field of knowledge engineering. First, it clearly states what is the basic

functional behaviour of a knowledge that is required by the particular task, at which level this knowledge can be represented and the way such a knowledge can be used by providing the reasoning in order to solve the problem.

Of the common ways to represent the knowledge is by modelling the associated chunks of knowledge with the task. An ontology acts as a common vocabulary for all such pieces of knowledge associated with the task. In the next section we will see in more detail the various facets in the ontological development.

4.2 Ontology: A Way to Enhance Knowledge Sharing and Reusability

Here we will see in more detail the notion of the commonly used word in the knowledge-acquisition community i.e. *Ontology* and try to find out the answer to some of the important questions such as, what is the meaning of term ontology? What are the various categories of ontologies one can find and what kind of libraries available for ontologies? What environments and languages can be used to build these ontologies? Etc.

One of the common trends in representing the knowledge that will have a common meaning for constructing the Knowledge-Level Models or knowledge models is by using the shared *Ontologies*. There are different viewpoints one can find in the literature that tries to explain the meaning of the term ontology. Gruber (1993) defines the notion of ontology as an '*explicit representation of conceptualisation'*. This definition suggests that an ontology is basically a conceptual vocabulary for understanding the entities and relations that exists in an *universe of discourse*. Another viewpoint suggests that '*an ontology has two purposes, that is, first it act as a represented vocabulary and as a conceptualisation that are the terms in the vocabulary is intend to capture'* [24]. This explanation also has the similar underlying philosophy as that of one proposed by Tom Gruber, but mainly focusing on the areas of application and intention for which the ontologies are built. Fundamentally, the ontologies can be useful in two ways (1) ontologies can support the human viewpoint on a particular domain and organisational communication and (2) they are machine-processable thus can support as a common understanding between machine and the human being.

Other researchers such as Motta, E. pointed out the more important issue that many of the earlier definitions failed to exploit is the *role of reusability* that ontologies serve. According to Motta, ontology can be defined as follows: *"An ontology is a partial specification of a conceptual vocabulary to be used for formulating knowledge-level theories about the domain of discourse. The fundamental role of an ontology is to support knowledge sharing and reuse"* [83]. The main context of this definition is not only to provide the common conceptualisation for the various aspects of knowledge but also to enhance the knowledge sharing and reusability of the existing knowledge.

The structure of an ontology is composed of two main parts such as, the *taxonomy* and the *axioms* [42]. The taxonomy of an ontology is a hierarchical organisation of the concepts and axioms are the established rules, principles or laws among the concepts. The axioms specify the competence of an ontology as well.

4.2.1 The Types of Ontologies

Broadly speaking the ontologies can be classified into four basic categories according to their type such as:

a) **Task ontology** – it is an ontology that formally specifies the terminology associated with the type of problem [77]. For example, task ontology of scheduling, planning, design etc.
b) **Method Ontology** – ontology that formally specify the definitions of the relevant concepts and relations used for specifying a reasoning process (problem solving) to accomplish a particular task.
c) **Domain ontology** – ontology for conceptualising the particular problem domain [24].
d) **Application ontology** – it contains the essential knowledge in order to model a particular application under consideration [81].

The ontologies can also be classified depending upon their generality issue. Various such types of ontologies are developed like, CYC [70], which mainly aims at modelling the generic notions so as to shape the foundation for knowledge representation across wider domains. In some cases these types of ontologies are referred to as a top-level ontologies [24] and one of the well-known example of such type of ontologies is Sowa's Ontology [111]. The domain specific ontologies are mainly developed for representing the concepts that are specific to particular domain. The examples of such type of ontologies are the PIF ontology [65], which is developed for modelling the business processes. The main aim of the PIF ontology is to develop the interchange format that enables in exchanging the process description among a variety of business modelling to support the systems such as planners, process simulation systems, flow charting tools and process repository. The Bibliographic Data ontology is residing on Ontolingua server [34] and is developed for representing the concepts and relationships that underlie a family of databases and tools in the domain of bibliographies, PhysSys ontology [13] that captures the knowledge related to the physical system process, the AIRCRAFT ontology [121] used for representing the air-campaign planning knowledge, the EngMath [49] ontology etc. The EngMath ontology is currently residing on the Ontolingua server as well [21]. As a part of TOVE project [130] developed at the University of Toronto various types of Enterprise ontologies are defined. One of them is particularly important from the scheduling viewpoint is a Generic Enterprise Resource Ontology [130], which mainly aims at emphasising on the nature of resource, its competence notion and the functionality

aspect. The most important aspect is the competence notion of an ontology that explains the types of tasks in which the ontology representation can be used.

Apart from this, another type of classification for ontologies is based on their development for supporting the representing of knowledge. The example of such a type is the Frame ontology [48] that captures the primitives used in various frame-based languages. The Frame ontologies make other types of ontologies to be specified by using frame-based conventions such as Knowledge Interchange Format (KIF) [44]. Following figure 10 [4] depicts the different types of ontologies.

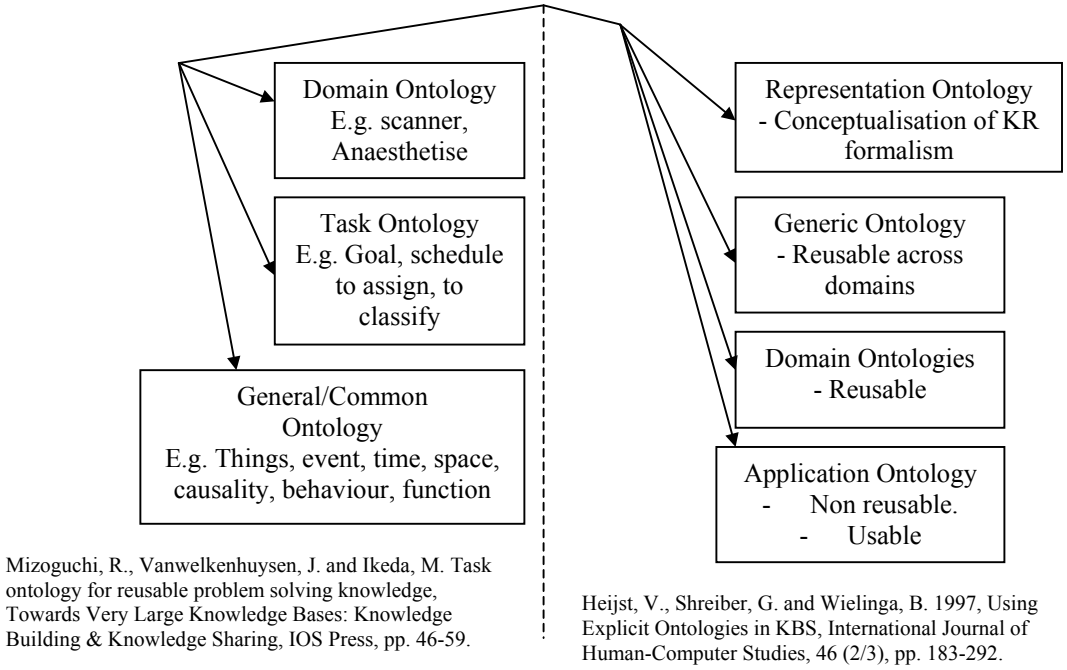


Figure 10. Various types of ontologies according to Mizoguchi and van Heijst and colleagues.

Although the ontologies are quite helpful as a knowledge acquisition tool, there is another important issue raises importance during the development of ontologies and that is the *reusability and usability trade-off* problem [59]. It is a general conscience that the more reusable an ontology is less usable it is and vice versa. One has to be really careful in order to keep the exact balance between usable-reusable trade-off while developing the new ontologies. It has been observed that the nature of the components involved in building the application systems usually changes according to the target domain. Such a changing nature of the system components increases the overall cost and time required for building the applications systems. Such a brittle nature of the system's building blocks can be avoided by constructing them by using the reusable constructs (i.e. ontologies). The first step in avoiding such a risk is by deciding about are there any currently available ontologies that can be reused from the existing libraries? If the answer is yes, then the next step is to decide whether additional concepts needs to be added in it or refine the existing ones otherwise one needs to

start building the new concepts that are missing in the current ontologies and adding them into the existing library [4]. If the developer needs to add the additional concepts or in other case needs to build the new ontology right from scratch this is the point where the concept of *knowledge modelling* enters into the picture.

Talking about the task ontologies in particular the task ontology can be seen as one of the important basic building blocks i.e., a technology [22], in building the knowledge models. The task ontology can be referred as an associated ontology (terminology) for the targeted task organised from the viewpoint of problem solving [117]. On the other hand one can treat the task ontology in order to formalise the knowledge for problem solving in task specific but application independent viewpoint. So the term task ontology can be interpreted in two ways:

1) It can be seen as a task-subtask decomposition of the various tasks such as scheduling, design, diagnosis etc.
2) The task ontology is an ontology for specifying the problem solving process.

The systematic organisation of various concepts and relations of the task ontology may or may not be similar for the different problem-types[1], but usually changes according to the type of problem. For instance, the task ontology of a scheduling task could represent a set of modelling components for representing the complete facets of scheduling task e.g., job, resource, constraints, assigned-job, schedule-complete, schedule-violate etc. Generally speaking the task ontology represents the problem specification by taking the input parameters in order to produce the expected outcome to the problem. In section 5 we will review the existing task ontologies that are built exclusively for the scheduling task. In next section we will briefly see the various languages and environments that are built for developing ontologies.

4.3 The Languages and Environments for Ontology Building

There are several languages and environments, which are currently available for building the ontologies. Most of the languages use one of the following modelling approaches such as, a logic-based approach, a frame-based modelling approach or both. Some of the most commonly known languages that are widely in use are currently residing on Ontolingua [34]. It is one of the most widely accepted languages in the knowledge acquisition community, which is based on *Knowledge Interchange Format* (KIF) and on the Frame ontology. Ontolingua supports three different types of expressions for building ontology such as:

1) It supports the development of an ontology by using KIF expressions.

[1] By problem-type we mean that a scheduling, a planning, a design etc. as a different types of problems.

2) It supports the development of an ontology exclusively based on the Frame Ontology vocabulary.
3) And supports the development of an ontology by using the both languages at the same time.

CycL [70] is used as a Cyc's knowledge representation language. This language is similar to the first-order predicate calculus with some extensions. It also has an inference engine in order to execute several kinds of reasoning. CycL uses a form of circumscription that includes the exclusive name assumption, and can also make use of the closed world assumptions where it is appropriate. For more information on CycL please refer to [132].

LOOM [76] is a high-level programming environment that supports the language and environment for constructing the intelligent applications. It is based on first-order logic, which belongs to the KL-ONE family. The LOOM language supports various kinds of specifications such as an explicit and expressive declarative model specification language, a robust deductive support and several programming paradigms along with the knowledge-based services. LOOM uses a fully expressive, logic-based representation language (which is a variant of KIF). LOOM's inference engine is not a complete first-order theorem prover but it can handle complex rules, negation, equality reasoning, subsumption and restricted forms of higher order reasoning.

FLogic (Frame Logic) [60] specify most of the structural aspects of object-oriented and frame based languages in a clean and declarative fashion. For example, the object identity, the complex objects, the inheritance, the polymorphic types etc. In a sense FLogic shares quite a lot with object-oriented paradigm as a classical predicate calculus as that of the rational programming. FLogic has a model-theoretic semantics as well as a sound and complete resolution-based proof theory. FLogic's deductive system works with the theory of structural and behavioural inheritance and predicate calculus. One of the advantages of FLogic is its extensibility as it can be combined with a broad range of other specialised logic.

Operational Conceptual Modelling Language (OCML) [81], fundamentally focuses on the logical rather than implementation-level primitives. The main objective of OCML is to support the knowledge-level modelling. It provides the various mechanisms such as classes, relations, functions, rules etc. The OCML can be used in other modelling frameworks that are presently available such as CommonKADS [126], Components of Expertise [112] it mainly notifies the Task-Method-Domain-Application (TMDA) [81] modelling framework (c.f. section 4.5.6). Another important point worth mentioning about OCML is that it supports different styles of modelling such as, informal, formal or operational. Operational modelling style is supported in OCML in terms of evaluating control and functional terms as well as theorem proving facilities. Informal modelling can be visualised by creating a pseudo code for the formal modelling expressions. In sum, the OCML is meant to provide the useful

tool for knowledge modelling whose main aim is to support the operational modelling facilities. Such an aim is satisfied by including functional and controls terms, as well as a proof system by integrating the inheriting backward chaining, function evaluation and procedural attachments.

There are several software tools available for building the ontologies such as, Ontosaures [115]. It aims at producing the distributed large-scale ontologies for various applications. The Ontology Server [34] is a tool for constructing the collaborative ontology. ODE [35] is an ODE is the ontology construction tool, which facilitates the user interaction at knowledge level. It uses a set of intermediate representations to the languages that are used to specify the ontologies. Tadzebao and Webonto [31], the WebOnto is a web based application tool for collaborative browsing and editing the knowledge models and the underlying model is implemented in OCML as a knowledge modelling language.

4.4 Problem-Solving Methods

The Problem-Solving Methods (PSMs) can be seen as a one of the most valuable assets for knowledge acquisition. PSMs can be defined as a *"domain independent reasoning steps, which specify the behaviour that can be used across wider domains"* [36]. The more formal and complete definition of PSM can be defined as follows.

A problem solving method is a domain-independent knowledge-level specification of problem solving behaviour, which can be used to solve a class of problems, say C. A problem solving method can be characterised as a particular specialisation of the generic problem solving model associated with C, say Gen-PSM, and its method ontology associated with Gen-PSM [83].

Broadly speaking the work on PSMs varies in different areas such as, how to formalise the various types of PSMs, how to store and index PSMs efficiently and more importantly how to identify the generic components of the task-specific PSMs, such as PSMs for scheduling, design, classification etc.

The PSMs describes the control knowledge domain independently along with the reasoning steps and the knowledge requirement for the task (e.g. scheduling, planning, design etc.) in order to solve it. The control knowledge of a PSM specifies the order of execution for various components along with the rationale for their execution and the data flow. The knowledge requirement states the kind of knowledge required for giving the reasoning steps. Thus the use of PSMs can be identified in a number of ways:

- It can be seen as a guideline for acquiring the problem solving knowledge from an expert.
- It can be seen as a guideline for decomposing the complex task into manageable subtasks and the order of their execution.

- It can be seen as a template for describing the reasoning process of the knowledge-based system.
- It can be seen as a way of selecting the flexible reasoning pattern by selecting appropriate methods during the process of problem solving etc.

The PSMs holds the unique position in knowledge acquisition community as compared to the constraint satisfaction community. Unlike to that of constraint-satisfaction community the knowledge acquisition community mainly aims at achieving the solution to the task through the efficient use of application knowledge [37]. As one of the main objectives of the researchers in the knowledge acquisition community is to provide the rational in finding out how the particular problem has been tackled while generating the solution rather than the actual solution itself. So the generalised patterns of reasoning can be reused efficiently in order to solve the similar kind of problem in future.

Similar, to the task ontology the PSMs can be broadly categorised into two major classes such as, *task-specific* PSMs and *task-independent* PSMs. The former type of PSMs are generally known as a *strong methods* [74] and are developed in order to solve the specific tasks such as, planning, scheduling, design etc., whereas the later type can be seen as a *weak methods*. The term '*weak'* indicates that these PSMs do not exhibit any assumptions according to the type of task and can be used to solve any type of task independent to its knowledge requirements. The examples of strong PSMs are Propose and Revise [72], Propose and Exchange [97] etc. and that of task-independent PSMs are A*, Hill-Climbing, Depth-First search, Breadth-First search etc. The above mentioned strong PSMs that are developed for solving the synthesis tasks [125] such as, scheduling, planning, design etc. belong to the family of Propose-Critique-Modify-Revise (PCMR) [23] methods. Talking in particularly about the PCMR classes of methods the Propose role of these methods is similar in nature with each other. During the propose role the solution to the initial model is proposed. The propose role terminates if the constraints are violated while proposing the solution to the problem or if the propose phase is complete. For example, in scheduling the propose role is said to be complete if all the jobs are assigned to the available resources or if the constraints on the jobs are violated. And depending upon their constraint removal strategy the revise role, exchange role etc. of these methods differs from each other. Most of the PSMs from the PCMR family are based on the following scheme.

1) Propose the initial extension to the model.
2) Identify the constraints on the extended model.
3) If the extended model violates the constraints then try to remove the violated constraints or at least minimise it with further effort (during this stage the methods differs based on how they remove the constraint violation).

For example, in the Propose and Revise PSM the model is extended by using the procedures that defines the value for the different parameters of the model. If the constraints are violated then these constraints are removed by applying the appropriate *fixes*. In Propose and Exchange method during the exchange phase some of the earlier set of solutions (assignments) are exchanged at the same depth of search tree in order to remove the violated constraints.

In general the PSM can be realised based on its architecture which is described in the following three levels:

First Level – The competence notion of a PSM can be explicitly explained based on its input-output performance. The input-output performance of a PSM states that what can be achieved by the particular PSM.

Second Level – The operational specification of a PSM gives the inference structure that satisfies the competence of the PSM if the appropriate domain-knowledge is provided. This level of PSM consists of three main components such as, *the inference roles*, *the knowledge requirements* for providing the inferences and *the control-flow* among the knowledge-roles. The inference roles are specified in terms of the input-output relations that can be solved by two types of methods, *decomposition method* and *primitive method*. The decomposition method decomposes the top-level task into one or more manageable sub-tasks until they have enough knowledge in order to solve by the primitive methods. On the other hand the primitive methods solves the task directly without any further decomposition. The control-flow determines the order in which the inference steps must be executed.

Third Level – This level specifies the requirements and assumptions needed for a PSM in order to describe the domain knowledge requirements of PSM for achieving its competence.

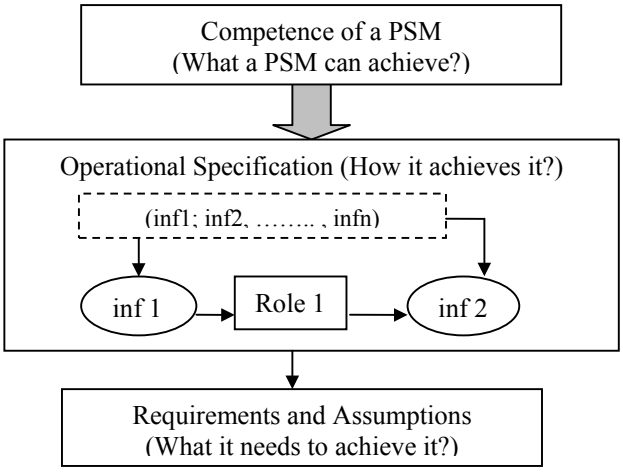


Figure 11. The basic architecture of a PSM [95].

There are various ways by which the PSM can be reused and stored in the library for their further applications. Once the PSM has been successfully developed then it is important to identify and abstract the generic components of all such PSMs. Once the generic components

are identified, collected and abstracted at the generic level then they can be stored in the library. The developer of a new PSM can be benefited from such libraries by using the pre-existing generic components by incorporating them in the new development. It saves the effort of a knowledge engineer to build the PSMs from scratch. Most of the libraries that are built for solving the particular type of task usually exhibit domain independent but task specific functionality. Currently several such types of libraries are available that are developed for different types of tasks such as, diagnosis, classification, parametric design etc.

The structure of such type of libraries can be identified based on various levels such as, *generality of a PSM*, *formality of a PSM*, *granularity* and *size of a PSM* [95].

- The generality of a PSM determines whether the PSM is developed for any particular task or it is task independent. A task-specific PSMs mainly deals with solving some specific tasks such as design, scheduling, diagnosis etc. The few examples include Planning [92], design [22], parametric design [82], diagnosis [10], Assessment [120]. The CommonKADS library [16] can be viewed as an extensive collection of most of the above-mentioned task-specific PSMs. To mention few the assessment, design, configuration, planning, assignment and scheduling etc.
- The formality notion of a PSM categorises the libraries in three main categories depending upon the library specification such as, *the formal library*, *the informal library* and *the implementation* oriented library. The formal specification of PSM libraries gives scope for the formal verification of it properties [1, 11]. The informal libraries of PSM offers structured textual representation of a PSM. Such representation can vary from simply textual representation to highly structured specification of a task, for instance, diagnosis [11]. The implemented libraries of PSM offer operational specifications that can be executed directly.
- The granularity of a PSM differentiates between the libraries of complex component such as, the parametric design library of Motta [80] and fine-grained PSMs concentrates on the small part of a task as opposed to complex component that focuses on the complete task.
- Finally, the size of libraries simply determines the number of PSMs that are included in the library. It does not raise important issue directly in terms of the construction of a PSM. The CommonKADS [16] library can be seen as an extensive collection of PSMs for solving various types of tasks as mentioned above.

The basic organisation of the knowledge modelling approach can be defined in terms of knowledge modelling frameworks. Hence, in the next section we will see some of the most influential knowledge modelling frameworks.

4.5 The Knowledge Modelling Frameworks

In this section we will look at some of the most influential knowledge modelling frameworks that are developed over the period of last two decades or so. Currently, there are wide number of knowledge modelling frameworks available such as, KADS/CommonKADS [124, 126], Generic Task [20], Components of Expertise [112], Protégé [85], Role-Limiting Methods [74] and Task-Method-Domain-Application (TMDA) [80, 81].

All these modelling frameworks are developed by keeping some specific aims and objectives and focusing on the different purposes. The knowledge modelling framework supports the construction of knowledge-based systems by defining the basic modelling organisation. Each knowledge modelling framework represents its own philosophy for developing the knowledge model. Here we will mainly focus on analysing the main purposes and aims that lie behind the development of these frameworks.

4.5.1 KADS to CommonKADS a Journey of a Decade

KADS/CommonKADS [124, 126] is one of the most influential knowledge modelling framework in the field of knowledge acquisition community. It can be seen as a comprehensive methodology for developing the knowledge-based systems. The work on the methodology was under development for nearly a decade or so. In the beginning the KADS was mainly focused on the knowledge acquisition [16], which was one of the major concerns in the knowledge-based technology then. Over the time period it has grown as a complete methodology that covers project management, organisational analysis, knowledge engineering and software engineering etc.

In CommonKADS the various roles can be captured into three basic categories of knowledge such as: task knowledge, domain knowledge and inference knowledge. All the categories of knowledge are identified at the *knowledge level* [87] by specifying *sufficient and necessary* conditions to define an expert system.

- *Task Knowledge*: task knowledge serves a dual purpose, first it specifies the iterative task-subtask decomposition of the top-level task. It expresses what it means to achieve these tasks and secondly it gives the order of execution of these subtasks (control structure over the execution) so that the top-level task can be achieved. The first part of the knowledge gives the task definition as the second part the task body. The task definition mentions the goal of a task in terms of the input and output roles. The task body of task knowledge gives the description about how that particular task can be achieved. In order to achieve the top-level (parent) task it is decomposed into various other categories such as: primitive task, composite task and goal specification task. The primitive task can be achieved directly without its further breakdown into other types of task. They have

enough knowledge associated with them so as to achieve them directly. The composite tasks needs further breakdown into low-level tasks so that by achieving these low-level tasks the solution for the composite task can be constructed. Finally, the goal specification task defines the goal specification for the associated task.

- *A Domain Knowledge*: a domain knowledge categorises into three basic knowledge elements that specifies; the structure of a knowledge (i.e. form), the way knowledge is organised (i.e. its structure) and the contents of the domain specific knowledge. For example, in the job-shop scheduling problem the domain knowledge shows the various properties associated with a particular job or a resource along with the constraints that restrict the space of their application and the way these properties are associated together for their execution etc. The form and structure of a knowledge is explicitly defined by the domain specific knowledge statements i.e. domain ontology.
- *The Inference Knowledge*: the inference knowledge gives the reasoning behaviour for the task subtask decomposition. It specifies the primitive steps in a particular application and the way domain knowledge statements are manipulated by these inferences. More importantly the inference knowledge specifies the data dependencies among different inference but it does not account for the order of execution of these primitive steps i.e. when and how such inferences can be made.

However, it is important to make the rationale that allows for constructing and making the explicit explanation about the system. In CommonKADS these rationales are specified by the meta-level knowledge that describes the structure of knowledge instead of giving the description of the structure itself. The CommonKADS framework supports the three types of metalevel knowledge such as: ontologies (that define the way to structure the knowledge), case model schemata (explains the way to conceptualise the task) and finally the problem solving methods (it explains the pragmatic approach adopted for solving the particular task).
In CommonKADS framework the three knowledge categories are interrelated with each other. And the knowledge roles are used as a *reference* among them. In order to enhance the transparency and maintenance for shaping the domain knowledge the notion of ontology [48] is adopted. An Ontology expresses the explicit viewpoint on particular domain application by specifying the common vocabulary.

The case model schemata are adopted in order to enhance the pragmatic rational behind the nature of problem solving. However, from the historical perspective the knowledge level [87] does not talk much about such behaviour except stressing on the selection of particular action for achieving the goal. Clancey [28] stressed that knowledge-based systems need to be built by the situation specific knowledge and similar argument was previously made by Steels [112] when talking about the case models. So in CommonKADS the case model specifies the

rationalisation about the solution and gives the explanation about why certain solution is considered as a solution. It also provides with the global structure of the argument that will support the conclusion.

The problem solving is a kind of black box in which the problem goes in and the solution comes out. The problem solving methods explains the inferences by constructing the task bodies. It explains the way to decompose the task definition in a task-subtask hierarchy along with how the solution to the subtask contributes in building the solution to the top-level tasks.

In summary the CommonKADS modelling framework can be seen as a comprehensive methodology for the knowledge modelling. It provides clean clarification on reusability aspect by the specification of ontologies, problem solving methods and the library organisation by giving its own language support such as, CML [103] and $(ML)^2$ [54]. The CommonKADS framework provides a distinct separation between application-specific knowledge and generic model. Though there are few limitations can be seen in the CommonKADS framework. Even though it gives the specific distinction between task, domain and inference knowledge categories the reusability focus is not explicit enough. Also, the ontological commitment associated with the problem solving level is missing which can be realised by developing the method ontology [45]. The notion of method ontology can be integrated in the CommonKADS as the schema specific domain viewpoint.

4.5.2 Generic Tasks

The Generic Tasks [20, 21] modelling approach was developed by B. Chandrasekaran and his group at The Ohio State University for nearly a decade in 1980s. It has started with learning about how to decompose the complex task into manageable component tasks. The main intuition of the work was based on that of *classification*, *data retrieval*, *plan selection* and *refinement*, *state abstraction* and *abductive assembly* since all of these are in one sense or other are re-usable subtasks. Another important driving factor that has lead the development of Generic Tasks approach was that of *interaction* problem. The interaction problem states that: *"Representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and by the inference strategy to be applied to the knowledge"* [18]. So the approach to this problem has been in identifying the "generic tasks" i.e. the basic combinations of *knowledge structure* and *inference strategies*. In one of the paper about Generic Task [23] the author has defended their viewpoint saying "generic tasks" is in a sense a misleading term. And what the author really meant is the following three basic concepts of the Generic Task framework consist of, *elementary generic combination of a problem*, *representation* and *inference strategy*. These three concepts can be described as follows:

- *Elementary generic combination of a problem:* it represents the input and output of the generic task along with the function of the generic task and what is the generic task is feasible for.
- *The representational knowledge:* basically explains the way knowledge should be organised and the structure that needs to be adopted for achieving the functionality of the task. It also represents the type of concepts that are involved in the generic task, the concepts explains the input and output of a task and the way knowledge is organised in terms of these concepts.
- *The Inference Strategy:* it represents the three basic elements such as, *the process*, *the reasoning behaviour* in order to achieve the task (problem-solving approach) and *the control regime*. In other words it gives the explanation about the inference strategy that should be applied to the knowledge such that it accomplishes the functional behaviour of a task.

In the Generic Task approach the tasks are rationalised as a "problem and goal" combination i.e. the goal of one problem is specified as the input to the next problem. The methods are characterised as a ways to accomplish the task and are of two types: *computational* or *situated*. The situated methods involve extracting information from the surroundings of the physical world. The concept of computational methods is pragmatically similar to the notion of problem space computational model of Newell [86]. In which the goal state is specified as a desired state that can be achieved by applying the operator. The operator transforms one state into the another state. Such a state transformation in a problem-space is controlled by the search control knowledge, which has the similar underlying philosophy as that of SOAR's architecture [66]. When there is more than one operator potential to apply on the current state then in such cases the search control knowledge determines which operator to be selected. Finally, the inference strategy is used as a part of method description. The inference terminology arises from the "reasoning metaphor" [23] in which the agent solves the current problem by making the inference from the available knowledge to generate a new knowledge. The model building procedure of Generic Task approach is similar to that of Components of Expertise [112]. In Generic Task the top level task is broken down into smaller manageable tasks until the task is solvable directly without its further breakdown.

In summary, the Generic Task approach exhibits various properties as it mentions that there is not only one-to-one relationship between generic task and problems. In other words the problem might match the function of more than one generic task hence several strategies can be applied for solving the problem. The Generic Task approach appeared to have computational advantages. It explains how the knowledge of right type can help in solving the problems in an acceptable time even though the general problem structure is NP-

complete. As mentioned earlier the Generic Task approach is applied on the range of problem types such as classification, design, diagnosis etc. Apart from all these advantages one of the limitation of Generic Task approach is that it does not provide task-independent domain models and hence it is difficult to achieve the reusability aspect. Also, there is no clear-cut distinction between the domain knowledge and the application knowledge as it assumes that the application knowledge is already absorbed in the specific computational categories. Thus, it cannot be seen as an efficient way to analyse the domain-specific knowledge requirements.

4.5.3 Components of Expertise

The Components of Expertise (CoE) [112] modelling framework was proposed by Luc Steels in 1990. This framework is mainly built for analysing what is called as the *knowledge-use level*. The knowledge-use level describes the way to decompose the overall task into manageable sub-tasks, it determines the ordering among these sub-tasks and decides what kind of access the knowledge is needed etc. In this sense the Generic Task and CoE shares a lot with each other. The work on CoE is mainly motivated by the various previous approaches in the field of knowledge acquisition such as, Generic Tasks [20], Clancy's Heuristic classification [27] and series of knowledge acquisition tools developed by McDermott [74]. In CoE three major types of components are proposed such as, *the task component*, *the method component* and *the domain component*. These are described in more detail in the following bullet points.

- *Task Component:* at this level of the modelling framework the detailed task analysis of the task is conducted. It explicitly mentions that each task can be decomposed into subtasks with input-output relations among them. It also emphasises on two viewpoints for analysing the task such as, *a conceptual* and *a pragmatic*. From the conceptual viewpoint a task can be characterised by stating the problem that should be solved, for instance, scheduling, planning etc. The pragmatic viewpoint focuses on the environmental constraints in which the task is executed or from the epistemological limitations of the environmental factors as such.
- *Method Component:* at this level of the modelling framework it emphasises on the problem solving methods which can be applied to the task by considering the appropriate domain knowledge. The CoE bifurcates the functionality of method components into two basic functions such as; it first decomposes the task into manageable number of subtasks or otherwise solves the task directly. In both the cases it emphasises on the strong coupling between methods component level and the domain level.

- *Domain Component:* at this level it emphasises on the particular viewpoint imposed by the task components on the particular application domain. It looks at the domain models from the domain theory viewpoint. The domain theories explain the principal theory that is underlying the problem solving in a particular domain. So the problem solving opportunistically selects the appropriate part of these theories in order to support the reasoning behaviour at the method component level.

In summary, the Components of Expertise modelling framework has combined and synthesised the various ideas from the previous modelling frameworks and refined them distinctly in a task, method and a domain component of the application development. Such a distinct categorisation of these components exhibits truly what is called as orthogonal dimensions for the component reuse and thus can be very helpful for organising the library of reusable components. Even though Steels's framework provides clear distinction between task-method-domain level it does not provide any accountability between domain and application level. As described in the second and third bullet points the problem solving level is strongly coupled with the domain level by domain theories. Thus this framework is very much method oriented. Still, Components of Expertise modelling framework has unique place in the history of knowledge modelling as it has introduced the neat separation between task, method and domain levels.

4.5.4 Role-Limiting Methods

Role Limiting Methods (RLM) [74] is one of the well-known approach for the generic models. The Role Limiting Methods reduce the roles of knowledge in the domain problem solving. The Role limiting Methods approach facilitates knowledge acquisition, knowledge representation, and efficient inference by representing the control regimes separately. The RLM requires a certain domain model organisation and provides a control mechanisms to apply on the domain knowledge for making certain types of inferences in order to reach the goal state. The RLM approach is based on three basic claims, which are listed as follows.

1) It emphasises that there exists a family of tasks which can be adequately solved by the methods and whose control knowledge can be abstracted from the peculiarities of any of the family members.
2) It says that if any of the methods whose control knowledge is not task specific then in such cases the method will use the task-specific knowledge in order to achieve the identification, selection and implementation of actions for achieving the goal states.
3) Finally, it claims that by separately representing the control regimes it increases the strength of the guidance provided by a method.

The RLM approach is mainly based on problem solving as the basis of identifying, selecting and implementing the sequences of actions that will accomplish some task within a particular domain. A selected problem solving method provides a means for identifying the selection of a potential action at each stage. It also provides one or more mechanisms in order to select among the potential candidate actions and finally ensures that the selected action is implemented properly. In RLM the method's control knowledge consist of two parts: 1) An algorithm specifying when to use what type of knowledge for achieving the above mentioned criteria of identification, selection and implementation of the action. 2) It emphasises that whatever knowledge the method possesses for selecting among the candidate actions that knowledge is not control knowledge. Thus, it provides the separate control knowledge apart from the problem solving knowledge itself. Hence, potentially there is only one component in RLM that is the complete problem solving method in itself.

In summary, the Role Limiting Methods approach is important historically because it was one the approach, which for the first time implemented Clancey's *role differentiation principle* [28]. The principle states that the problem solving agents can be described as a generic models in such a way that problem solving roles can be imposed on the domain knowledge. Even though RLM looks quite basic as compared to the other modelling frameworks developed in the later years one should not forget that RLM was developed when the knowledge modelling was still in its early age.

4.5.5 Protégé

The Protégé modelling approach was developed at Standford University's Medical Informatics Laboratory by Mark Musen. His early work on the OPAL [84] knowledge acquisition system has demonstrated how the domain-specific tools can potentially accelerate the encoding of a medical knowledge that was used for the ONCOCINE expert system. The OPAL system was originally developed for providing the treatment to the cancer patients thus had very limited applicability because of the limited focused domain. In order to spread the wider domain applicability the work on Protégé [85] acquisition tool started. Later the Protégé was used to handle the problems from another domain area i.e., planning and more specifically from the Episodic Skeletal Plan Refinement [119]. But the users faced the problems while using the Protégé environment because the problem specification at the task level hardly matches with the requirements at the problem solving level. In order to overcome this drawback the Protégé-II environment was developed. The Protégé-II environment is based on Protégé environment. The Protégé-II environment is a knowledge-acquisition shell that efficiently provides a task-modelling environment for the knowledge engineers and knowledge-editing environment for application users. In contrast to the Protégé, the Protégé-II is independent of the task, methods and domains. In Protégé-II one

can specify the definitions of various knowledge roles at the knowledge level from both domain-dependent and domain-independent knowledge perspective. The problem solving methods can be generated by using mechanism [45] as a building block and models the application task in terms of the constructed methods. The complete model is constructed by the library of methods and mechanisms. Similar to the Generic Tasks approach the model construction process is based on generating a task model through the method driven task decomposition processes.

In summary, the Protégé is one of the most accomplished knowledge modelling tools that provides the knowledge acquisition as well as knowledge editing tools. It supports the construction of a library of task and methods as well as editing tool for selection and configuration of methods.

4.5.6 Task/Method/Domain/Application Modelling Framework

The Task/Method/Domain/Application (TMDA) [80, 81] modelling framework takes as an input the initial methodology of task-method-domain partition proposed by Components of Expertise [112] and Protégé [85] and from there it refines it on various dimensions. In TMDA modelling framework there are in all four types of generic components such as, task knowledge, method knowledge, domain knowledge and application configuration knowledge or application knowledge.

- *Task Knowledge:* at this level the generic task is specified by means of the task ontology. Ideally, the task ontology should be generic in the sense that it should be domain as well as application independent. Task ontology can be seen as a problem specification of a task by specifying its goal statement to be achieved based on the input parameters. For example, if we take the scheduling as a task then its input parameters can be jobs, resources, constraints, time ranges, cost and preferences etc. and the goal of a task is to construct a schedule.
- *Method knowledge:* at this level the method knowledge gives the domain-independent specification of the reasoning components. In order to enhance the communication between these knowledge types the concepts from the task knowledge level are mapped to the problem solving level. The method knowledge is further subdivided into two levels such as generic problem solving model and the problem solving methods. First, the generic problem solving model can be developed by abstracting the generic components from the various problem solving methods. The generic problem solving model provides the common basis for comparing and contrasting the knowledge requirements of other problem solving methods. And at the next level other problem solving methods can be constructed as a refinement of generic problem solving model. The generic problem solving model takes as an input the task specification from the task

ontological level and the problem solving paradigm that can be adopted for solving the problem and produces the method independent but task specific model of problem solver and generic method ontology as an output.

- *Domain Knowledge:* at this level of the methodological development the reusable domain components are defined which serve as specifying the multi-functional [82], reusable domain models. Hence, it avoids the knowledge interaction mismatch between the domain layer and the concepts defined at the method layer. In TMDA framework the gap between these two layers is bridged by defining the appropriate mapping mechanism [45].
- *Application (configuration) knowledge:* finally at this level of methodological development the application specific knowledge is defined. As the application-specific knowledge is strongly application related hence it cannot be a part of multi-functional domain knowledge base and thus cannot be reusable.

In summary, as mentioned earlier in the initial stage the TMDA modelling framework adopts the similar underlying philosophy as that of the previous modelling frameworks such as Components of Expertise and Protégé but refines it on several dimensions. The major strength of TMDA is that it adds the fourth type of component i.e., the application configuration knowledge by formulating the relevant mapping mechanisms. The mapping mechanism also helps in bridging the gap between knowledge interaction mismatch. Also, TMDA tries to bring together the important usability and reusability trade-off. It can be seen from the first two bullet points that it provides a strong coupling between generic tasks and problem solving methods by mapping the concepts from task ontological level to problem solving level. In order to increase the reusability of domain knowledge it adopts a weak coupling between problem solving knowledge and domain knowledge; as the domain knowledge is characterises the task-independent nature of the domain.

So along with the description of Task/Method/Domain/Application framework we conclude the review of the most influential modelling frameworks during last two decades or so.

4.6 Summary so far

It can be realised from the various modelling frameworks that each framework has its unique place in the history of knowledge modelling. The first few frameworks such as Role Limiting Methods, Generic Task and Components of Expertise can be seen as the basic foundation for the work on later frameworks such as CommonKADS and task/method/domain/application framework. The CommonKADS and Protégé were one of the few to exploit the knowledge interaction between the domain and problem solving level by defining various types of ontologies such as method and domain ontology at the problem solving and domain level. In

order to bridge the gap between domain and application level for avoiding the knowledge interaction mismatch the task/method/domain/application framework has refined it by using the mapping mechanisms. Thus, it has distinctly identified the clear boundaries between task level, problem-solving level, domain level and application level. At each of the four levels various types of ontologies are defined such as task ontology, method ontology, domain ontology and application ontology.

Now in the next section we will review the currently available task ontologies for scheduling. In the following review we will try to analyse the conceptualisation behind the development of these task ontologies.

5. Existing Task Ontologies for Scheduling

In this section we will review the presently available task ontologies developed for the scheduling task. This section of the review will be helpful for the proposed work of the research in the sense that it will make us aware in understanding the position of other researchers on developing the scheduling task ontology. We will mainly concentrate on finding out the concepts involved in their task ontologies, the various input parameters included in the task ontology etc. In the following review we will analyse the work done by the three research group for developing the task ontologies which are OZONE [109], Job-Assignment Ontology [52, 55], MULTIS [77].

5.1. Job-Assignment Task Ontology

The Job-Assignment task ontology [52, 55] is developed under the project named CAKE since 1987 by the group of Hama, T., Hori, M. and Nakamura, Y. and is presently residing on the Ontolingua server [34]. This task ontology gives the knowledge level description for various concepts involved in the scheduling task and their relations with each other. The job assignment task, which is a class of scheduling problems of, "assigning all given jobs to the available resources within the time range, while satisfying various constraints" [131]. The main input parameters that comprise Job-Assignment task ontology are jobs, resources, time-ranges and constraints. These input parameters are categorised into three fundamental *entities* and two basic types of *relations*. The entities are *jobs*, *resources* and *time-ranges* whereas the relations among them are *assignment* and *constraints*. The fundamental entities are described in terms of their attributes. For example, the job is described in terms of its time range, duration and type. In this task ontology the relation 'assignment' can be seen as a mapping of jobs to the resources or time ranges. The final output (i.e. goal) is the total mapping of all the jobs to the resources or time ranges by satisfying the various constraints. The constraints can be seen as a restriction on the space of assignment and are mainly defined in terms of the relation among attributes of jobs, resources and time ranges. For example, the constraints can

be specified in such a way that only certain jobs can be assigned to the resources within certain time slots, which can be interpreted as a job specific constraint. The constraints impose the conditional requirements on the attribute values of jobs, resources or time ranges. These conditions with some additional requirements are considered as a special type of constraints. The constraints are mainly used as an evaluation criterion in the task ontology. In other words, the final schedule is validated in terms of number of constraints that are satisfied by the completion of an assignment.
Thus, the job-assignment task ontology exhibits quite straightforward framework for the class of job assignment class of scheduling problems.

Even though all the main concepts are described at the knowledge level in the job-assignment task ontology their definitions are informally illustrated at the same length and the properties of the important relations and classes are not characterised with the required level of detail and formalism. For example, the definition of a job is characterised in terms of the source object (where source object denotes an object to be assigned on the target object) and similar characterisation holds for the resources, which is represented in terms of the target object. The only evaluation criterion that is considered in this task ontology is the number of constraints that are satisfied by the completion of a schedule. It is important to keep in mind that the scheduling task is usually seen as a problem of finding the optimal or sub-optimal solution. By using the evaluation criteria considered in the job-assignment one can only get the feasible schedule solution but the optimality criteria remains unanswered. In order to achieve the optimal solution one needs to consider the cost as an optimality criterion in order to choose the cheaper scheduling solutions. Also, there is no clear indication about separation of a constraint into hard constraints and soft constraints. The constraints are treated simply as hard constraints throughout the generation of a schedule that can act as an over-constrained approach for a scheduling problem. One way of dealing with the different constraints in scheduling is to divide them into two basic categories such as hard constraints and soft constraints. The hard constraints can be used as minimum satisfactory condition for the schedule to be valid and then satisfying the soft constraints can be seen as a full-proof criterion on the schedule solution.

5.2. MULTIS Task Ontology

The MULTIS task ontology [77] is the outcome of the MULTIS project conducted at Osaka University since 1987. This task ontology is developed through a task analysis interview system for a general class of scheduling problem. The MULTIS task ontology is called as a system of "generic vocabulary" [77] it mainly consists of generic nouns, generic verbs, and generic adjectives along with other task related concepts. The *generic process* is mainly a

combination of generic verbs and nouns that occur in the domain. More than one generic processes creates the network of these processes which is called as a "generic process network" (GPN). The GPN is a knowledge level representation/skeleton for representing the problem structure under scrutiny i.e. scheduling. The lexical level task ontology consist of four main types of concepts such as:

- *Generic Nouns:* that represents the objects reflecting their roles appearing in the problem solving process. For example, Schedule Recipient (job, order etc.), Schedule Resource (line, machine etc.), Schedule representation, Constraints, Goal and priority etc.
- *Generic Verbs:* that represents unit activities appearing in the problem solving process. For example, Assign job, classify job, select, pick up, neglect etc.
- *Generic Adjectives:* that modifies the pre-existing objects. For example, unassigned job, the last etc.
- *Other related words that are specific to the task.* For example, Strong/weak constraints, constraint predicates and attributes etc.

The knowledge about the generic verbs, nouns etc. is acquired by the task analysis interview with the domain experts that establishes the interview system. The main framework of the MULTIS task ontology is based on three basic terminologies for representing the final schedule such as Schedule Recipient i.e., a job, Schedule Resource that is, a resource e.g. line, machine, track etc., and the time slots which can be allocated to the job. A final solution to the scheduling task is represented as a *generic noun* such as schedule. The solution to the schedule is represented as a *generic verb*, which represents the assignment of Schedule Recipient to the Schedule Resource. The goal of a scheduling task in MULTIS is based on optimising the priority of jobs or data information according to specific domains. In MULTIS the constraint related vocabulary is divided into two parts such as strong constraints (hard) and weak constraints (soft) and they are subject to satisfaction for the jobs by using the *generic adjectives* such as satisfying or violating the constraints for the assigned/unassigned job etc.

The main optimisation criterion considered in the MULTIS is based on optimisation of a priority of jobs or a data information based on domain specific situation. The task ontology is developed in such a way that it provides the required vocabulary for the problem solvers.

5.3. OZONE Ontology for Scheduling

The OZONE [108] scheduling ontology was developed at the Carnegie Mellon University for developing a toolkit for configuring the constraint-based scheduling system. It is built as a part of project named Ontology Development Environment (ODO) [12]. The OZONE scheduling ontology is partly available through the WebPages of Enterprise Integration

Laboratory of University of Toronto [130]. It is used by uniting constraint-directed search under a foundation of assertion and retraction of commitments (with propagation of the commitments to the related problem variables). The OZONE ontology lies at the core of this framework to provide the necessary base of concepts in order to describe and represent the scheduling problem. The OZONE ontology adopts the job centric scheduling viewpoint. The ontology is based on the concept that in scheduling the jobs can be assigned to the resources and on the other hand the resources may remain assigned once the complete assignment of all the jobs is done.

The main base concepts considered for building OZONE ontology is Demand, Product, Activity, Resource and Constraint. Where activity is considered to be a non-deterministic selection of an action with respect to the available resources. The activity has a same meaning as that of the concept job from Job-Assignment task ontology and the concept Schedule Recipient from the MULTIS task ontology. The concept Resource is the central to the definition of this scheduling ontology. The main concept resource is considered as a set of homogeneous resources with respect to some activity. The capacity of the resources is determined by the cardinality value of the associated set of resources and is further divided into Capacitated-Resource, Reusable-Resource, Consumable-Resource and Renewable-Resource. The precedence relations have been defined in terms of time intervals among any two activities such as, before, after etc.

The OZONE scheduling ontology considers various external environmental factors as well such as Demand, Order etc. (more specifically it considers the planning phase as well). The main scheduling task triggers once the *Demand* for the particular *Product* arises. The demands as well as order are user-defined concepts in which the user has to specify the Demand for the particular Product. The final goal in the OZONE ontology is considered to be achieved once the Demand for the Product is satisfied by assigning all the Activities to the Resources while satisfying all the constraints.

In sum, the OZONE ontology is mainly concerned with modelling the entities and constraints of particular domain. It can be realised from the available literature on OZONE that it is mainly developed for the constraint-based scheduling as a prime problem solver. And usually the constraint-based solvers do not reason about the cost criteria. It might be one of the reasons that there is no indication about any optimisation criteria such as cost and preferences in OZONE ontology.

6. The Libraries of Problem-Solving Methods for Scheduling

In this section we will review the existing libraries of problem-solving methods for the scheduling task. As mentioned earlier the ultimate aim of our research work is to construct the generic library of problem-solving methods for scheduling. In order to fulfil the objective

the current status of the libraries will be helpful in realising the kind of technology and approaches that other researchers have used. While reviewing the libraries we will mainly concentrate towards the organisation of these libraries and the types of problem solving methods incorporated in the library.

6.1.Hori and Yoshida's Domain-Oriented Library for Scheduling

Hori and Yoshida [56] proposed a domain-oriented perspective for constructing the library of production scheduling, which is one of the classes of scheduling problem. The library consists of three loosely coupled sub-systems such as, a manufacturing domain, scheduling method and graphical user interface (GUI). The domain model of library provides the fundamental concepts that needs to be employed by the remaining two sub-systems.

The overall principle of problem-solving knowledge of the library is influenced by Chandrasekaran and Musen's philosophy of hierarchical task-methods decomposition. In which the top-level task in the library i.e., a production scheduling is iteratively decomposed into smaller tasks based on their functionality. The sub-tasks are in turn solved by the sub-methods. The sub-methods in the library are organised on the basis of their data flow i.e. a control knowledge. One aspect of this library is that the control knowledge is clarified in terms of the domain model. In other words the control between the execution of any two sub-tasks depends on the domain specific requirements for achieving the corresponding top-level tasks. The domain model in this library corresponds to a model that includes concepts employed by the problem-solving processes. In order to bridge the gap between domain model and problem solving knowledge the mapping mechanisms are added.

Mainly the library can be divided in terms of three basic dimensions such as the task level, problem-solving level and the domain level. At the task level the task is specified in terms of its task ontology. At the problem solving level the problem solving inferences are specified by integrating with the domain level concepts. As it is not always possible to relate the top-level task in terms of its control knowledge, the collection of such methods is organised in a *family* of method. As mentioned earlier the problem solving and domain knowledge are separated with each other in such a way that it could be possible to change the problem solving knowledge even during the run time. In the library three main problem solving patterns are analysed such as, *Divide & Merge*, *Transform & Restore* and *Check & Modify* [55]. In Divide & Merge, the given problem is divided into sub-problems and by invoking another component the solution to the sub-problem is achieved. The solution of all these sub-problems is merged into the schedule hypothesis. The merging is not always necessary if each sub-problem shares the same assignment space. In Transform & Restore, the problem structure is reformulated that invokes another component and the schedule hypothesis is restored obtained by the original problem. Check & Modify, is mainly responsible for finding

the unexpected situation such as constraint violation in a schedule hypothesis and then for modifying the hypothesis. All the three patterns have same set of input/output parameters, that is, jobs, resources, time ranges, constraints and assignments.

In our point of view the library has limited scope of applicability because primarily the library is reusable only in a single domain that is production scheduling. Talking specifically about the underlying principle of the library it is unclear the way various components are executed. The assignment method of the library is executed in terms of three basic operations such as, 'reset', 'isDone' and 'doNext'. Some of these operations are low-level actions as 'doNext' simply performs single step iteration for assigning a unit to an appropriate resource. On the other hand the Dispatching method uses completely different vocabulary such as 'isEmpty' that checks if there is any lot in the queue to be dispatched. The same task is carried out by 'isDone' in the assignment method. Hence, it is difficult to compare and contrast their knowledge requirements. Even though three problem-solving methods are collected together in a family of methods, such a library organisation does not provide any accountability for realising how the generality of these components is achieved. The reusability of the library is investigated in terms of the ratios of code reuse in a real case.

6.2. CommonKADS Library of Assignment and Scheduling

The CommonKADS [114] provides the library of problem-solving methods for assignment and scheduling. The assignment is a kind of synthesis task that operates on two sets such as demands and supplies. Each member of one set must be assigned to the each member from another set. The scheduling is considered as a special case of assignment problem type. The problem-solving methods in the CommonKADS library are based on the Propose and Revise [72]. The Propose and Revise class of methods is the general approach for solving the synthesis task e.g. scheduling, design, configuration etc. The CommonKADS library of assignment and scheduling takes the 'top-down' approach for tackling the problem. In the proposed library of assignment and scheduling two variants of Propose and Revise are discussed such as, *simple propose and revise* and *hierarchical propose and revise*. In simple propose and revise the complete problem is solved by considering the three basic steps of Propose and Revise consisting of, proposing the solution to the initial problem, then identifying the violation of constraints on the proposed solution and finally revising the solution in order to remove the constraint violation if there are any. In hierarchical propose and revise first large chucks of components (jobs) are assigned to high level resources. This abstract solution is specialised in order to obtain a solution to the initial problem. For every rule types in Propose and Revise such as propose rule, constraints violation identification rule and revise rule various 'functions' are discussed in order to accomplish that rule type. For

each of the variants of Propose and Revise problem-solving method the knowledge requirements and data flow is discussed explicitly.

The CommonKADS library of problem solving methods directly associates each problem solving method with the problem type i.e., assignment/scheduling. It is difficult to realise the generic components of these problem solving methods in order to specialise for the other types of problem solving methods. Similar to the Hori's library based on such organisation it is difficult to understand the generic components of these problem solving methods. Nevertheless, the basic principle applied in CommonKADS is obviously quite useful for achieving the hierarchical decomposition of task, sub-task in order to solve the top-level task.

6.3. Le Pape's Library of Resource Constraint Scheduling (ILOG)

The scheduling problem has a long history from constraint satisfaction literature [63]. Many researchers have tackled the scheduling problem successfully by using constraint-directed search. The ILOG library of constraint satisfaction mainly concentrates on the *resource scheduling* as a class of scheduling problems. ILOG library of constraint scheduling supports three types of programming models of constraint propagation that consist of,

- The constraint satisfaction problem such as a graph, in which the nodes in the graph represents the problem variables, to which finite domains are associated and the edges represents the constraints the needs to be satisfied. The constraints are typical binary-constraints that are associated with each of the node.
- The second type of model is logic programming, in which constraint satisfaction is viewed as an extension of unification.
- And the third type of model consists of distinguishing the concepts of a generic constraint and the concept of an instance for the constraint application to the specific variables of the problem to be solved.

Various timetables in constraint satisfaction are proposed by using the Resource Time-Table. The resource-time table represents a constraint variable for the value to which a function is associated a value v(t) for representing the time at which the resources can be available for the execution. The semantic model proposed for the implementation of a timetable is quite straightforward. First, it assumes the discrete representation of a time (which is usually the case for all scheduling problems) and then memorises the status of the variable for any instance of time (t) during the interval of a time. The second implementation of a timetable does not make assumptions about the nature of the time whether discrete or dense and simply memorises the instants of time during which the status of the variable changes. These two types of implementation referred to as a *Discrete Array* and *Sequential Table* respectively. Apart from this one of the assumption for the resources in ILOG scheduling is that the

resources are assumed to be a Unary capacitated Resources i.e. resources that can handle only one job at any time. Such kind of capacity representation is particularly useful in the scheduling domains for representing the finite capacitated resources.

The problem solving model of ILOG library does not take into account the existence of domain knowledge, which is quite common case in the constraints satisfaction problem solving. It mainly depends on developing the efficient algorithms for solving the problem type. The envisaged library of problem solvers during this research work will be mainly based on identifying the knowledge-intensive tasks and the various types of application specific knowledge, which can be brought to bear while developing the reasoning process for the problem solving methods.

Nevertheless, some of the techniques from the constraint-satisfaction can be useful while developing the generic problem solver for scheduling in the proposed research work to support domain-independent least commitment strategies such as Dynamic Search Rearrangement techniques, some variant of variable and value ordering heuristics etc.

Along with this we conclude the review of currently existed task ontologies and the libraries of scheduling task. This literature review has set the initial foundation in understanding the basic concepts involved in the development of task ontologies and the different approaches adopted for developing the libraries of problem solver. Also, while reviewing the various task ontologies and libraries we have opportunistically pointed out the strength and weaknesses of those approaches along with the scope for the further development which will be helpful while developing the envisaged library of problem-solving methods.

7. Discussion and Summary

In the literature review we reviewed some of the important research activities involved in three major disciplines, consisting of, planning and scheduling, the intelligent scheduling approach and finally from the knowledge acquisition perspective. During the first part of the review the main focus was on informing the main concepts involved in two different paradigms such as planning and scheduling and their basic philosophy. During which we have identified the basic differences between planning task and scheduling task. The former deals with finding the sequence of actions in order to transform the initial world state into goal state whereas the latter mainly deals with allocation or assigning the jobs over the available resources or time while satisfying the different constraints. Also, we have seen the various kinds of problems that can be tackled by the synthesis tasks such as planning, scheduling etc. This phase of the review has set the initial foundation in understanding the basic nature of the problem that has lead in finding the status of scheduling task particularly from the artificial intelligence background.

The review in section 3 has presented different approaches that can be applicable to the scheduling task and then various intelligent scheduling systems have been reviewed that are evolved over the period of last two decades or so in order to solve the scheduling problem. These intelligent systems differed in their working principle and architecture for dealing with the problem in their respective domains. Over the time period it has been realised that even though the currently existed intelligent scheduling systems were working efficiently to tackle the problem, they are hard-wired in nature. The main reason for this inflexibility was that all these systems were built by keeping a targeted application domain thus could not be used in different domains. In order to overcome such a serious bottleneck of inflexibility the need for the reusable components arisen in the systems development.

The answer to such a serious issue came from the knowledge acquisition community. The knowledge acquisition field has tried to acquire the experience and expertise from the specific technical areas and experts. This acquired knowledge was represented in the form of ontologies that can be seen as a 'common vocabulary' for expressing the viewpoint on particular scenarios. Various types of ontologies have been developed such as, task ontologies, generic ontologies, domain ontologies etc. The task ontologies are helpful in representing the vocabulary associated with the problem type whereas the method ontology for expressing the behaviour of the problem solving procedure for solving the particular problem type etc. The generic representation of such type of ontologies can be very useful in order to enhance the reusability aspect of knowledge. For instance, the generic task ontologies of scheduling can be very useful for representing the problem specification of scheduling without subscribing to any specific application. The effort of generic representation of various types of ontologies and problem solving methods has shown the way to build the libraries of such technology for the further reuse.

As mentioned during the literature review opportunistically, the main aim of our research is to build the library of problem solving methods for scheduling applications. By keeping such an objective in mind the last section of the literature review was focused on finding out the current status of the task ontologies and libraries of problem solving methods for scheduling. As it can be seen from section 5 and 6 various researchers have put the efforts in building the libraries of problem solving methods and task ontologies for the scheduling task. Based on this understanding of a field we have realised that there are still some issues that are found to be restrictive or partially uncovered in the current approaches in building the libraries of problem solving methods or in developing the task ontologies for scheduling. It can act as a main motivation in the proposed research for developing the library of problem solvers for the scheduling applications.

References

[1] Aben, M. 1993, Formally specifying re-usable knowledge model components, Knowledge Acquisition, 5, pp. 119-141.

[2] Allen, J. F. 1983, Maintaining Knowledge about Temporal Intervals. Communications of the Association for Computing Machinery, 26 (11).

[3] Allen, J. and Koomen, J. 1983, Planning using a temporal world models, Proc. 8th Int. Joint Conf. On AI, pp. 741-747.

[4] Asuncion Gomez Perez and Benjamins, V. R. 1998, Applications of Ontologies and Problem-Solving Methods, Report on ECAI'98 Workshop. in NVKI Nieuwsbrief, 15(5), pp. 146-150.

[5] Asuncion Gomez Perez and Benjamins, V. R., 1999, Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99), http://sunsite.informatikrwth-aachen.de/Publications/CEUR-WS/Vol-18/.

[6] Barker, K. R. 1974, Elements of sequencing and scheduling, John Wiley and Sons, New York.

[7] Barták, R. 1999, On the Boundary of Planning and Scheduling: A Study, Proceedings of the Eighteenth Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG), Manchester, UK, pp. 28-39.

[8] Bartak, B. 2000, Slot Models for Schedulers Enhanced by Planning Capabilities, 19th Workshop of the UK Planning and Scheduling Special Interest Group, pp. 11-24.

[9] Beck, J. C. and Fox, M. S., 1998, A Generic Framework for Constraint-Directed Search and Scheduling, AI-Magazine, 19 (4) pp. 101-130.

[10] Benjamins, V. R. 1993, Problem Solving Methods for Diagnosis, PhD thesis, University of Amsterdam, Amsterdam, The Netherlands.

[11] Benjamins, V. R. and Aben, M. 1997, Structure-preserving KBS development through reusable libraries: a case-study in diagnosis, International Journal of Human-Computer Studies, 47, pp. 259-188.

[12] Blazquez, M., Fernadez, M., Garcia-Pinar, J. M., Gomez-Perez, A., 1998, Building Ontologies at the Knowledge Level using the Ontology Design Environment, in Proceedings

of the 11th Knowledge Acquisition, Modelling and Management Workshop, KAW98, Banff, Canada. Available at http://ksi.cpsc.ucalgary.ca/KAW/KAW98/KAW98Proc.html

[13] Borst, P., Akkermans, H. and Top, J. 1997, Engineering Ontologies, International Journal of Human-Computer Studies, 46, pp. 365-406.

[14] Brazier, Frances M. T. and Wijngaards, Niek J. E. 1998, A Purpose Driven Method for the Comparison of Modelling Frameworks, in Knowledge Acquisition Workshop, (KAW-98), Banff, Canada. Available at http://ksi.cpsc.ucalgary.ca/KAW/KAW98/brazier1/

[15] Burke, P. and Prosser, P. 1991, A distributed asynchronous system for predictive and reactive scheduling, Artificial Intelligence in Engineering, 6(3), pp. 106-124.

[16] Bruker, J. and Van de Velde (eds.), CommonKADS Library for Expertise Modelling: Reusable problem solving components, IOS Press, Netherlands.

[17] Brusoni, V., L. Console, Lamma, E., Mello, P., Milano, M. and Terenziani, P. 1996, Resource-based Vs Task-based Approaches for scheduling problems, 9th ISMIS96, LNCS series, Springer-Verlag.

[18] Bylander, T. and Chandrasekaran, B. 1987, Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition, International Journal of Man-Machine Studies, 26, pp. 231-243.

[19] Chaib-Draa, B., Moulin, B., Mandiau, R., and Millot, R. P. 1992, Trends in Distributed Artificial Intelligence, Artificial Intelligence Review, 6, pp. 35-66.

[20] Chandarasekaran, B. 1983, Towards a taxonomy of problem solving types, AI Magazine, 11(4), pp. 59-71.

[21] Chandrasekaran, B. 1986, Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design, IEEE Expert, 1, pp. 23-30.

[22] Chandrasekaran. B., 1990. Design problem solving: A task analysis, AI Mag., 11, pp. 59-71.

[23] Chandrasekaran, B., Johnson T. R., Smith J. W. 1992, Task Structure Analysis for Knowledge modelling, Communications of the ACM, 35 (9), pp. 124-137.

[24] Chandrasekaran, B., Josephson, J. R. and Benjamins, V. R. 1999, Ontologies: What are they? Why do we need them? IEEE Intelligent Systems and Their Applications, 14 (1), pp. 20-26.

[25] Collinot, A., LePape, C. and Pinoteau, C. 1988, SONIA: a knowledge based scheduling system, Artificial Intelligence in Engineering, 7 (2), pp. 38-53.

[26] Corkill, D. 1991, Blackboard systems, AI Experts, 6 (9), pp. 40-47.

[27] Clancey, 1985, Heuristic Classification, Artificial Intelligence, 27, pp. 289-350.

[28] Clancey, W. J. 1992, Model construction operators, Artificial Intelligence, 53 (1), pp. 1-135.

[29] Dean, T. and Kambhampati, S. 1996, Planning and Scheduling, In CRC Handbook on Computer Science and Engineering.

[30] Dorn, J. 1995, Iterative improvement methods from knowledge-based scheduling, AI communications, 8 (1), pp. 20-34.

[31] Domingue, J. 1998, Tadzebo and Webonto: Discussing, browsing, and editing ontologies on the web, 11th Workshop on Knowledge Acquisition, Modeling and management, KAW'98, Banff, Canada.

[32] Elleby, P., Fargher, H. E. and Addis, T. R. 1988, Reactive constraint-based scheduling, IEE Colloquium on Artificial Intelligence in Planning of Production Control, London.

[33] Etzioni, O., Hanks, S., Weld, D. Draper, D. Lesh, N. and Williamson, M. 1992, An approach to planning with incomplete information, 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning, pp. 115-125.

[34] Farquhar, A., Fikes, R. and Rice, J. 1997, The ontolingua server: a tool for collaborative ontology construction, International Journal of Human-Computer Studies, 46 (6), pp. 707-728.

[35] Fernandez, M., Gomez-Perez, A., Pazos, J. and Pazos, A. 1999, Building a chemical ontology using methontology and the ontology design environment, IEEE Intelligent Systems, pp. 37-46.

[36] Fensel, D., Benjamins, V. R. 1998, Key Issues for Automated Problem Solving Methods Reuse, in Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), pp. 63-67.

[37] Fensel, D., Straatman, A. 1998, The essence of problem-solving methods: making assumptions to gain efficiency. International Journal of Human-Computer Studies, 48, pp. 181-215.

[38] Fikes, R. E. and Nilsson, N. J. STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence, Vol. No. 3-4, pp. 189-208.

[39] Fox, M. S. 1983, Constraint–directed search: a case study of job-shop scheduling. Technical Report, CMU-RI-TR-83-22, Robotics Institute, Carnegie Mellon University.

[40] Fox, M. S. 1994, ISIS: A Retrospective, in M. Zweben and M. Fox (eds.), Intelligent Scheduling, pp. 3-28.

[41] Friedland, P. and Iwasaki, Y. 1985, The concepts and implementation of skeletal plans, Journal of Automated Reasoning, 1, pp. 161-208.

[42] Gangemi, A., Pisanelli, D. and Steve, G. 1998, Ontology Integration: Experiences with Medical Terminologies, in Proc. 1st International Conference on Formal Ontology in Information Systems, FOIS'98, pp. 163-178.

[43] Garrido, A., Salido, M.A., and Barber, F. 2000, Scheduling in a Planning Environment, ECAI-2000, New Results in Planning, scheduling and design.

[44] Genesereth, R. and Fikes, R. 1992, Knowledge Interchange Format, Computer Science Dept., Stanford University, 3.0 edition, Technical Report, Logic-92-1.

[45] Gennari, J. H., Tu, S. W., Rothenfluh, T. E. and Musen, M. A. 1994, Mapping Domains to Methods in Support of Reuse, in Proceedings of the 8th Banff Knowledge Acquisition Workshop, Banff, Canada.

[46] Goel, V. and Pirolli, P. 1989, Motivating the notion of generic design within information processing theory: The design problem space, AI-Magazine, pp. 19-36.

[47] Grant, T. J. 1986, Lessons for OR from AI: A scheduling case study, Journal of Operations Research Society, pp. 41-57.

[48] Gruber, T. R. 1993, A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, Knowledge Acquisition. 5 (2), pp. 199-221.

[49] Gruber, T. R., & Olsen, G. R. 1994, An Ontology for Engineering Mathematics. In J. Doyle, P. Torasso, and E. Sandewall (Eds.), Fourth International Conference on Principles of Knowledge Representation and Reasoning, Gustav Stresemann Institute, Bonn, Germany, Morgan Kaufmann.

[50] Hadavi, K., Hsu, W-L., Chen, T. and Lee, C-N. 1992, An architecture for real-time distributed scheduling, AI Magazine, 14(3), pp. 46-56.

[51] Hadavi, K. and Shahraray, M. 1989, Release No Job Before Its Time, 3rd International Conference on Expert Systems and the Leading Edge in Production and Operations Management, Production and Operations Management Society.

[52] Hama, T.; Hori, M.; and Nakamura, Y. 1992, Task-Specific Language Constructs for Describing Constraints *in* Job-Assignment Problems. Technical Report IBM Research Report RT0084.

[53] Haralick, R. M. and Elliott, G. L. 1980, Increasing tree search efficiency for constraint satisfaction problems, Artificial Intelligence, 14, pp. 263-313.

[54] Harmelen, F. VAN. and Balder, J. 1992, $(ML)^2$: A formal language for KADS models of expertise, Knowledge Acquisition, 4, pp. 127-161.

[55] Hori, M., Nakamura, Y., Satoh, H., Maruyama, K., Hama, T., Honda, S., Takenaka, T. & Sekine, F. 1995, Knowledge-level analysis for eliciting composable scheduling knowledge, Artificial Intelligence in Engineering. 9 (4), pp. 253-264.

[56] Hori, M. and Yoshida, T. 1998, Domain-oriented library of scheduling methods: design principles and real-life applications, International Journal of Human-Computer Studies, 49(4) pp. 601-626.

[57] Kempf, K., Le Pape, C., Smith, S. F. and Fox, B. R. 1991, Issues in the Design of AI-Based Schedulers: A Workshop Report, AI Magazine, 11(5), pp. 37-46.

[58] Keng, N. P., Yun, D. Y. and Rossi, M. 1988, Interaction Sensitive Planning Systems for Job-Shop Scheduling, in M. D. Oliff (ed.) "Expert Systems and Intelligent Manufacturing", Elsevier.

[59] Klinker, G., Bhola, C., Dallemagne, G., Marques, D. and McDermott, J. 1991, Usable and reusable programming constructs, Knowledge Acquisition, 3, pp. 117-136.

[60] Kifer, M., Lausen, G. and Wu, J., 1995. Logic foundations of object-oriented and frame-based languages, Journal of the ACM, pp. 741 - 843

[61] Koehler, J. 1988, Planning under resource constraints, 13th European Conf. on AI, pp. 489-493.

[62] Le Pape, C. and Sauve, B. 1985, SOJA: an expert system for workshop daily planning, Proc. Avignon-85, Avignon, France, pp. 849-867.

[63] Le Pape, C. 1994, Implementation of Resource Constraints in ILOG SCHEDULE: A library for the development of Constraint-Based Scheduling Systems, Intelligent Systems Engineering, 3(2), pp. 55-66.

[64] Le Pape, C. 1994, Scheduling as Intelligent Control of Decision-Making and Constraint Propogation, in M Zweben and M Fox (eds.), Intelligent Scheduling, pp. 67-98.

[65] Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost, G. and members of the PIF working group. 1998, The PIF Process Interchange Format and framework, Knowledge Engineering Review, 13 (1), pp. 91-120.

[66] Liard, J. E., Newell, A., Rosenbloom, P. S. 1987, SOAR: An Architecture for general intelligence, AI, 33 (1), pp. 1-64.

[67] Liu, B. 1988,Scheduling via reinforcement, Artificial Intelligence in Engineering, 3(2), pp. 76-85.

[68] Liu, B. 1991, Resource Allocation and Constraint Satisfaction in Production Scheduling. The World Congress on Expert Systems.

[69] Lloyd, E. L. 1982, Critical Path Scheduling with Resource and Processor Constraints, Journal of ACM, 29 (3), pp. 781-811.

[70] Lenat, D.B. and Guha, R.V. 1990, Building large knowledge-based systems. Representation and inference in the Cyc Project. Addison-Wesley, Reading, Massachusetts.

[71] Lesser, V. R. and Corkill, D. 1981, Functionally accurate, cooperative distributed systems, IEEE Trans. SMC-11(1), pp. 81-96.

[72] Marcus, S. and McDermott, J. 1989, SALT: A Knowledge Acquisition Language for Propose-and-Revise Systems, Artificial Intelligence, 39 (1), pp. 1-37.

[73] Marsay, D. J. 2000, Uncertainty in Planning: Adapting the framework of Game Theory, Proceedings of the 19th Workshop of the UK Planning and Scheduling Special Interest Group, pp. 101-107.

[74] McDermott, J. 1988, Preliminary steps towards a taxonomy of problem-solving methods. In S. Marcus (ed.), Automating Knowledge Acquisition for Knowledge-Based Systems. Boston: Kluwer Academic Publishers.

[75] McKenzie, L. W. 1976, Turnpike theory, Econometrica, 44 (5), pp. 841-865.

[76] MacGregor, R. 1991, Inside the LOOM classifier. SIGART Bulletin, 2 (3), pp. 70-76.

[77] Mizoguchi, R., Vanwelkenhuysen, J. and Ikeda, M. Task ontology for reusable problem solving knowledge, Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing, IOS Press, pp. 46-59.

[78] Morgado, E. M. and Martins, J. P. 1993, An AI based approach to crew scheduling, in Proc. of CAIA-93, Orlando, pp. 71-77.

[79] Mott, D. H., Cunnigham, J., Kelleher, G. and Gadsen, J. A. 1988, Constraint-based reasoning for generating naval flying programmes, Expert Systems, 5 (3), pp. 226-246.

[80] Motta, E. and Zdrahal, Z. 1997, An approach to the organisation of a library of problem solving methods which integrates the search paradigm with task and method ontologies, International Journal of Human-Computer Studies, 49(4), pp. 437-470.

[81] Motta, E. 1998 An Overview of the OCML Modelling Language. Paper presented at the 8th Workshop on Knowledge Engineering Methods and Languages (KEML '98).

[82] Motta, E., and Zdrahal, Z. 1998, A library of problem-solving components based on the integration of the search paradigm with task and method ontologies, International Journal of Human-Computer Studies, 49 (4), pp. 437-470.

[83] Motta, E. 1999, Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press, Amsterdam, The Netherlands. ISBN: 1 58603 003 5.

[84] Musen, M. A., Fagan, L. M., Combs, D. M. and Shortliffe, E. H. 1987, Use of a Domain Model to Drive an Interactive Knowledge-Editing Tool, International Journal of Man-Machine Studies, 26, pp. 105-121.

[85] Musen, M. A. 1989, Automated Generation of Model-Based Knowledge Acquisition Tools, Research Notes in Artificial Intelligence, Pitman, London.

[86] Newell, A. 1980, Reasoning, Problem Solving, and Decision Processes: The Problem Space as a Fundamental Category, in The Soar Papers: Research on Integrated Intelligence, Vol. 1, Lawrance Erlbaum Associates, Hillsdale, pp. 693-718.

[87] Newell, A. 1982, The Knowledge Level, Artificial Intelligence, 18, pp. 87-127.

[88] NII, H. P. 1986, Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures, AI Magazine, 7(2), pp. 38-53.

[89] Nilsson, N. J. 1998, Artificial Intelligence: A New Synthesis, Morgan Kaufmann.

[90] Nilsson, N. J. 1993, Principles of Artificial Intelligence, Springer-Verlag, ISBN- 3-540-11340-1

[91] Noronha, S.J. and Sarma, V.V.S. 1991, Knowledge-Based Approaches for Scheduling Problems: A Survey, IEEE Transaction on Knowledge and Data Engineering, 3(2), pp. 160-171.

[92] Nunes de, B. L., Valente, A. and Benjamins, V. R. 1996, Modelling planning tasks, 3rd International Conference on Artificial Intelligence Planning Systems, pp. 11-18.

[93] Ow, P. S. and Smith, S. F. 1988, Viewing scheduling as an opportunistic problem solving process, Annals of Operations Research 12, pp. 85-108.

[94] Pednault, E. 1989. "ADL: Exploring the middle ground between STRIPS and the situation calculus", 1st Int. Conf. On Principles of Knowledge Representation and Reasoning, pp.324-332.

[95] Perez, A. G. and Benjamins, V. R. 1999. Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods, in Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden. Available at http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/

[96] Pinedo, M. 1995, Scheduling Theory, algorithms and Systems, Prentice-Hall, ISBN 0-13-706757-7.

[97] Poeck, K. and Puppe, F. 1992, COKE: Efficient solving of complex assignment problems with the propose-and-exchange method, 5th International Conference on Tools with Artificial Intelligence, pp. 136-143.

[98] Poeck, P., Gappa, U. 1993, Making Role-Limiting Shells More Flexible, 7th Workshop of Knowledge Acquisition for Knowledge-Based Systems, pp. 103-122.

[99] Puterman, M. 1994, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley.

[100] Sadeh, N. and Fox, M. S. 1996, Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem, Artificial Intelligence, 86, pp. 1-41.

[101] Sathi, A., Fox, M. S. and Greenberg, M. 1985, Representation of activity knowledge for project management, IEEE Transaction Pattern Anal. Machine Intelligence, 7 (5), pp. 531-552.

[102] Saucer, J. 1997, Knowledge-Based Systems Techniques and Applications in Scheduling, In T. L. Leondes (ed.), Knowledge-Based Systems Techniques and Applications, Vol. 1, Academic Press.

[103] Schreiber, A. T., Wielinga, B. J., Akkermans, H., Van de Velde, W. and Anjewierden, A. 1994, CML: The CommonKADS Conceptual Modelling Language, in L Steels, A. T. Schreiber and Van de Velde, W. (Eds.), "A future for knowledge acquisition, in Proceedings of the 8th European Knowledge.

[104] Sharma, S. D. 2000, Operations Research, Kedar Nath Ram Nath & Co.

[105] Silcock, J. and Kutti, S. 1993, Taxonomy of Real-Time Scheduling, TR C93/32, Deakin University; school of computing and mathematics.

[106] Smith, S. F. e al. 1986, Constructing and Maintaining Detailed Production Plans: Investigation into the Development of Knowledge-Based Factory Scheduling, 7(4), pp. 45-61.

[107] Smith, S. F., Fox, M. S., and Ow, P. S. 1987,Constructing and maintaining detailed production plans: Investigations into the development of knowledge based factory scheduling system, AI-Magazine, 7(4), pp. 45-61.

[108] Smith, D. and Goodwin, S. D. 1995, Constraint-based Intelligent Scheduling, CS-95-02, University of Regina, pp. 1-26.

[109] Smith, S. and Becker, M. A., 1997, An Ontology for Constructing Scheduling Systems, Working Notes of 1997 AAAI Symposium on Ontological Engineering, AAAI Press, March, 1997.

[110] Smith, D., Frank J. and Jonsson, A. K. 2000, Bridging the gap between planning and scheduling, Knowledge Engineering Review, 15 (1), pp. 47-83.

[111] Sowa, J. 2000, Knowledge Representation: Logical, Philosophical and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, USA, ISBN: 0-534-94965-7.

[112] Steels, L. 1990, Components of Expertise, AI Magazine, 11 (2), pp. 29-49.

[113] Stefik, M. 1995, Introduction to Knowledge Systems, Morgan Kaufmann, ISBN: 1-55860-166-X.

[114] Sundin, U. 1994, Assignment and Scheduling, in J Breuker and W Van de Velde (eds.) CommonKADS LIBRARY FOR EXPERTISE MODELLING Reusable problem solving components, chapter 11, pp. 231-263.

[115] Swartout, W., Patil, R., Knight, K. and Russ, T. 1997, Toward distributed large-scale ontologies. In Spring Symposium Series on Ontological Engineering, pp. 33-40.

[116] Tate, A., Drabble, B. and Kirby, R. 1994, O-plan2: an open architecture for command, planning and control, in M Zweben and M Fox (eds.), Intelligent Scheduling, Morgan Kaufmann, pp. 213-239.

[117] Tijerino, Y. A., Ikeda, M., Kitahashi, T. and Mizoguchi, R. 1993, A Methodology for Building Expert Systems Based on Task Ontology and Reuse of Knowledge, Journal of Japanese Society for Artificial Intelligence, 8 (4), pp. 476-487.

[118] Tsang, E. P. K. 1995, Scheduling techniques - a comparative study, British Telecom Technology Journal, 13 (1), 16-28.

[119] Tu, S. W., Shahar, Y., Dawes, J., Winkles, J., Puerta, A. R. and Musen, M. A. 1992, A problem solving model for episodic-skeletal-plan refinement, Knowledge Acquisition, 4(2), pp. 197-216.

[120] Valente, A. and Lockenhoff, C. 1993, Organisation as guidance: A library of assessment models. 7th European Knowledge Acquisition Workshop, pp. 243-262.

[121] Valente, A., Russ, T., MacGrecor, R. and Swartout, W. 1999, Building and (Re)Using an Ontology for Air Campaign Planning, IEEE Intelligent Systems, 14(1), pp. 27-36.

[122] Van Dyke Parunak, H., Irish, W., Kindrick, J. and Lozo, P. W. 1985, Fractors actors for districuted manufacturing, Proc. CAIA-85, Stanford, California, pp. 653-660.

[123] Vere, S. 1983, Planning in time: windows and durations for activities and goals, Pattern Analysis and Machine Intelligence, 5 (3), pp.246-267.

[124] Wielinga, B. J., Schreiber, A. T. and Bruker, J. 1992, KADS: A Modelling Approach to Knowledge Engineering, Knowledge Acquisition, 4(1), pp. 5-53.

[125] Wielinga, B. and Schreiber, A. TH. 1997, Configuration-design problem solving, IEEE Expert. 12 (2), pp. 49-56.

[126] Wielinga, B., Van de Velde, W., Schreiber, G. and Akkermans, H. 1992, The CommonKADS Framework for Knowledge Modelling, in Proceedings of the 7th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada.

[127] Wilkins, D. 1990, Can AI planners solve practical problems?, Computational Intelligence, 6 (4), pp. 232-246.

[128] Zweben, M., Davis E., Daun B., Drascher E., Deale M. and Eskey M. 1992, Learning to improve constraints-based scheduling, Artificial Intelligence, 58, pp. 271-296.

[129] Zweben, M. and Fox, M. S. (eds.) 1994, Intelligent Scheduling, Morgan Kaufmann.

[130] Enterprise Integration Laboratory. EIL, TOVE Project, University of Toronto, Canada, available from http://www.ie.utoronto.ca/EIL/tove/ontoTOC.html

[131] http://www.ksl.stanford.edu/knowledge-sharing/ontologies/html/job-assignment-task/job-assignment-task.lisp.html

[132] http://www.cyc.com/cycl.html