View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repository@USM

AN ONTOLOGY AND CONSTRAINT-BASED APPROACH FOR DYNAMIC PERSONALISED PLANNING IN HEALTHCARE

NORMADIAH MAHIDDIN

UNIVERSITI SAINS MALAYSIA
2009

AN ONTOLOGY AND CONSTRAINT-BASED APPROACH FOR DYNAMIC PERSONALISED PLANNING IN HEALTHCARE

by

NORMADIAH MAHIDDIN

Thesis submitted in fulfillment of the requirements
for the degree of
Master of Science

June 2009

ACKNOWLEDGEMENTS

Alhamdulillah, thank to Allah for giving me a strength to pursue this research until the end.

Here, I would like to thank Universiti Sains Malaysia (USM) for the opportunity to pursue my postgraduate studies, for the generous financial assistance through the Academic Staff Training Scheme (July 2006 – July 2008), the School of Computer Sciences, and the Institute of Graduate Studies for the financial assistance through the Graduate (Teaching) Assistant Scheme (June 2004 – June 2006), for the use of the various facilities, and for their kind assistance.

I am highly appreciative of my main supervisor, Dr. Cheah Yu-N, and my co-supervisor, Dr. Fazilah Haron for their dedication, guidance, advice and ideas, moral support, and for checking and editing my thesis. The research experience that I have gained in the last four-and-a-half years is something that I will truly treasure.

I thank my friends: Nur Hana binti Shamsudin, Lim Lian Tze, Najwa binti Abu Bakar, Farizah Azmah binti Ridzuan, Maziah binti Salleh and also my other colleagues at the Postgraduate, Health Informatics Research, and Grid Computing Laboratories for their support and fellowship.

Finally, I am very thankful to my brother, En. Sulaiman bin Mahiddin for being a close friend and for the discussions, advice and ideas, and moral support through my journey in finishing this research, my mother, Puan Arbbayah binti Haji Ridzwan, my husband, En. Mohd Latiff bin Talib, my sister, Puan Norasiah binti Mahiddin and my whole family for their prayers, love, care, nurturing and support, and for having encouraged me always to accept challenges and to do my best.

TABLE OF CONTENTS

Page

Acknowledgements……...…...……...ii
Table of Contents…….....iii
List of Tables……..…….. ……vi
List of Figures……...……vii
Abstrak…..……..……x
Abstract…….....xii

CHAPTER 1: INTRODUCTION

1.1 Background of Planning……..……..1
1.2 Planning in Artificial Intelligence……...……...2
1.3 Planning in Our Selected Domain: Renal Health…….....2
1.4 Research Problems…….....3
1.5 Problem Solutions…… ……...…….4
1.6 Contributions of This Thesis…….5
1.7 Thesis Outline…….....7

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction…….....10
2.2 Planning System based on Ontologies and Generic Planning……11
2.2.1 O-Plan…….....11
2.2.2 Asgaard…….....13
2.2.3 PLANET…….....15
2.2.4 Blocks World Planning……..…….16
2.3 Plan Representation Language Evolution…….....16
2.4 Planning System…….....18
2.4.1 A Conceptual Model for Planning……..18
2.4.2 Planning System Evolution: Linear to Non-Linear ……...21
2.4.3 Other Computational Methods Suitable for Planning ……31
2.5 Grid Computing Environment…….....34
2.6 Discussion of the Various Approaches ……..36
2.7 Summary ……....37

CHAPTER 3: RESEARCH METHODOLOGY – MEANS TO AN END

3.1 Introduction …….....39
3.2 Research Methodology ……..39
3.2.1 Phase One: Plan Ontology Definition and Representation. .....……..41
3.2.2 Phase Two: Planning Algorithm Definition.....42
3.2.2.1 Sub-Phase One: Generic Plan Generation……..…...…...…..……44
3.2.2.2 Sub-Phase Two: Plan Personalisation……..…...……45
3.2.3 Phase Three: DP Planner in a Distributed System…….....45
3.2.4 Phase Four: Planning System Evaluation.…….....47

3.3 Justifications for Our Approach ……48
3.3.1 A New Plan Ontology……48
3.3.2 Generic Plan Generation……49
3.3.3 Plan Personalisation ……50
3.3.4 Planning System Architecture in Grid Environment ……50
3.4 Summary... ……51

CHAPTER 4: IMPLEMENTATION

4.1 Introduction ……52
4.2 Phase One: Defining a Plan Ontology and Plan Representation ……53
4.2.1 Features of Our Plan Ontology ……53
4.2.2 High-Level Components of our Plan Ontology ……54
4.2.3 Plan Fragment Ontology Details ……54
4.2.4 Representing Plans in XML……54
4.3 Phase Two: Defining a Dynamic Planning Engine ……57
4.3.1 Overview of Our Plan Algorithm……58
4.3.2 Development of Planning Engine……59
4.3.2.1 Generic Plan Generation ……60
4.3.2.2 Plan Personalisation ……63
4.4 Phase Three: Dynamic Planning in a Distributed System……67
4.4.1 Software Tools……69
4.4.1.1 Globus 4.0.3……69
4.4.1.2 OGSA-DAI WSRF 2.2……70
4.4.1.3 Client Tool Kit……71
4.4.1.4 XML DBMS – Xindice……71
4.5 Summary……72

CHAPTER 5: EXPERIMENTS AND RESULTS

5.1 Introduction……73
5.2 Generic Plan Generation……74
5.3 Plan Personalisation……82
5.3.1 Case 1: Inputs with “status is active”……82
5.3.2 Case 2: Inputs with “status is not active”……83
5.3.3 Case 3: User inputs do not fulfil conditions or constraints in the generated generic plan……85
5.3.3.1 Case 3.1: Inputs which do not fulfil conditions in the generic plan……85
5.3.3.2 Case 3.2: Inputs which do not fulfil some conditions but fulfil constraint in the generic plan……87
5.3.3.3 Case 3.3: Inputs which do not fulfil constraints in the generic plan……91
5.4 Comparison with Other Planning System in Planning Community……93
5.4.1 Plan Ontology and Representation……93
5.4.1.1 Qualitative Criteria: Adoption of a conceptual model for planning……95
5.4.1.2 Qualitative Criteria: Simplicity of the plan ontology……95
5.4.1.3 Qualitative Criteria: Simplicity of the plan representation……96
5.4.2 Methods of Planning Process for the Planning Engine……98
5.4.2.1 Qualitative Criteria: Planning approach……98
5.4.2.2 Qualitative Criteria: Genericity……99

5.4.2.3 Qualitative Criteria: Application of case-based plan adaptation techniques….....100
5.4.2.4 Qualitative Criteria: Tolerance towards planning system execution failure……101
5.5 Summary.....103

CHAPTER 6: CONCLUSION

6.1 Planning – A Problem Solving Process……....105
6.2 Revisiting Our Contributions……106
6.2.1 Plan Ontology Design…..……107
6.2.2 Dynamic Planning…….....108
6.3 Dynamic Planning in Grid Environment.....108
6.4 Future Work……..……...109
6.4 Summary.....……...111

BIBLIOGRAPHY …….....……112

APPENDIX.....120

LIST OF PUBLICATION…….....132

LIST OF TABLES

Page

Table 2.1: Methods used during Asgaard's Design Time 27

Table 2.2: Methods used during Asgaard's Execution Time 27

Table 5.1: Comparison between DP Planner with other Planning Systems 94

LIST OF FIGURES

Page

Figure 2.1: Plan Ontology Structure (Tate, 1994) 12

Figure 2.2: Features in the <I-N-OVA> model (Tate, 2000) 12

Figure 2.3: The Asbru plan ontology structure (Miksch, Shahar & Johnson, 1996) 14

Figure 2.4: An overview of the PLANET ontology (Gil & Blythe, 2000) 15
(arrows pointing into space represent relations whose ranges are not fixed in the ontology)

Figure 2.5: A Framework of Components in a Planning/Scheduling System 23
(in *Agenda* form) (Tate, 1994)

Figure 2.6: O-Plan Planner Agent Components (Tate, 1994) 24

Figure 2.7: The Asgaard idea (Miksch & Hammermuller, 1999) 26

Figure 2.8: Asgaard Run-Time Modules (Miksch, Seyfang & Kosara, 2001) 28

Figure 2.9: The RAX-PS architecture (Jonsson et al., 2000) 30

Figure 2.10: The process flow for PHI composition (Abidi Sibte, Chong & 32
Abidi Samina, 2001)

Figure 2.11: The Globus Tol Kit Component (Globus Project Team, 2007) 35

Figure 3.1: Research methodology with techniques and approaches used 40

Figure 3.2: The sub-phases involved in dynamic planning 43

Figure 3.3: The overview of OGSADAI in a planning scenario 46

Figure 4.1: Proposed Plan Ontology 54

Figure 4.2: Plan representation in XML format 57

Figure 4.3: Overview of the planning system 59

Figure 4.4: Overview of generic plan generation 60

Figure 4.5: Detailed process of generic plan generation 61

Figure 4.6: Overview of personalised plan generation 63

Figure 4.7: Details of plan personalisation 64

Figure 4.8: The linking between plan fragments in a plan 65

Figure 4.9: The use of constraints in plan personalization 67

Figure 4.10: The OGSA-DAI Data Service Scenario (Mahiddin, Cheah & Haron, 2007) 68

Figure 4.11: The OGSA-DAI WSRF 2.2 with Globus in planning scenario (OGSA-DAI, 2005) 71

Figure 5.1a: The abridged plan repository it its original state 75

Figure 5.1b: The abridged plan repository it its original state (continued) 76

Figure 5.2: The System Interface 77

Figure 5.3: The process of finding the best-matched plan in the plan repository 78

Figure 5.4: Plan ID 4.0 80

Figure 5.5: An example of generic plan generated 81

Figure 5.6: Part of the system interface when status of the user is “active” 82

Figure 5.7: Part of the system output when status of the user is “active” 82

Figure 5.8: Part of the system interface when status of the user is “not active” 83

Figure 5.9: Part of the system output when status of the user is “not active” 84

Figure 5.10: System interface for case 3.1 85

Figure 5.11: Personalised plan for case 3.1 86

Figure 5.12: System interface for case 3.2 87

Figure 5.13: Generic plan for case 3.2 88

Figure 5.13a: Personalised plan for case 3.2 89

Figure 5.13b: Personalised plan for case 3.2 (continued) 90

Figure 5.14: System Interface for case 3.3 91

Figure 5.15: Personalised plan for case 3.3 92

Figure 5.16: Solving execution failures in DP Planner. 102

Figure 5.17: Solving execution failures in O-Plan Planner. 103

PENDEKATAN BERASASKAN ONTOLOGI DAN KEKANGAN UNTUK PERANCANGAN PERIBADI DINAMIK DALAM PENJAGAAN KESIHATAN

ABSTRAK

Para doktor atau pakar kesihatan semestinya mengambil berat berkenaan pengemaskinian rekod atau profil pesakit mereka, dan perancangan rawatan yang sesuai untuk pesakit mereka, untuk mendapatkan hasil perkhidmatan kesihatan yang berkesan. Walaubagaimanapun, tidak semua para doktor dapat melaksanakan aktiviti perancangan dengan berkesan, khasnya apabila perancangan ini dilakukan berasaskan komputer (perancangan diautomasi) disebabkan kelemahan-kelemahan dalam sistem perancangan sedia ada.

Secara umumnya, terdapat tiga masalah penyelidikan yang telah dikenalpasti. Masalah pertama ialah kebanyakan perwakilan atau ontologi rancangan sedia ada terlalu terperinci. Kita memerlukan perwakilan mudah alih dan mudah yang mana ia menyediakan kemudahan simpanan dan manipulasi rancangan generik (sebaka). Masalah kedua ialah kebanyakan sistem perancangan sedia ada seringkali berada dalam keadaan statik. Walaupun terdapat sistem-sistem perancangan sedia ada yang dinamik, ia lebih sesuai untuk bidang-bidang yang melibatkan pengetahuan yang luas dan kompleks tetapi agak kabur dan kurang lengkap.

Hasilnya, kami menyediakan sistem perancangan kesihatan yang generik dan dinamik dipanggil Perancang DP (Dinamik dan Peribadi), yang mana sistem perancangan ini

padat dan sangat sesuai untuk pengetahuan yang jelas atau nyata. Oleh itu, Perancang DP menyediakan (1) perwakilan perancangan yang mudah dan generik berasaskan ontologi perancangan, dan (2) jentera peracangan dinamik berasaskan kekangan. Perancang DP juga boleh disokong oleh suatu seni bina perancangan berasaskan grid.

AN ONTOLOGY AND CONSTRAINT-BASED APPROACH FOR DYNAMIC PERSONALISED PLANNING IN HEALTHCARE

ABSTRACT

Healthcare service providers are undoubtedly concerned about updating their patients' health records or profiles, and the planning of their patients' treatments to support the effective delivery of healthcare services. However, not all healthcare service providers are carrying out planning activities effectively, especially when it comes to computer-based planning (automated planning) due to shortcomings in current planning systems.

Generally, three main research problems have been identified. The first problem is that most of the current plan representations or ontologies are too fine grained (detailed). We need to have a portable and intuitively easy representation that facilitates the storage and manipulation of generic plans. The second problem is that most of current planning systems are often static. Even though there are existing dynamic planning systems, they are however more-suited for domains with large and complex but partly vague and incomplete knowledge.

As the results, we present a generic and dynamic healthcare planning system called DP (Dynamic Personalised) Planner, which is a compact planning system that is more-suited for obvious knowledge. Therefore, DP Planner provides (1) a suitably light-weight and generic plan representation based on a plan ontology, and (2) a constraint-based dynamic planning engine. The DP Planner can also be supported by a grid-based planning architecture.

CHAPTER 1

INTRODUCTION

"If the Hereafter is about to occur and in the hands of one of you a plant (he is about to plant it in ground), he must do so as long as he has a chance."

[Hadith - Recorded by Bukhari]

1.1 Background of Planning

Cambridge International Dictionary of English (1997) defines "*plan*" as a "set of decisions about how to do something in the future". In human life, many different kinds of planning are carried out, such as project planning, urban planning, floor planning, and many others. Planning has also been used widely in many fields such as in business, medicine, administration, logistics, education, environment, and also in family matters. As people say: "if we fail to plan, we plan to fail"; such is the importance of planning in human life. Goal setting and planning is therefore very important to achieve the vision and goals of individuals and organizations. In other words, plans are needed in many different fields of human endeavor, and in some cases, it is desirable to create these plans automatically (Nau, 2007). Thus, automated or computer-based planning is an active area of research in computer science.

1.2 Planning in Artificial Intelligence

Artificial Intelligence (AI) is a “branch of computer science that is concerned with the automation of intelligent behaviour” (Luger, 2004). Thus, in automated-planning research, the word “plans” refers specifically to plans of actions. It is about the representation of future behavior, usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents. Theoretically, planning is an important component of rational behaviour. Therefore, planning plays an important role in modeling the computational aspects of intelligence (Nau, 2007; Traverso, Ghallab & Nau, 2004).

Classical planning mostly focused on robotic applications such as STRIPS and Blocks World Planning. However, current trends show that automated planning technology has abecome mature enough to be useful in applications that range from game playing to the control of space vehicles (Nau, 2007).

1.3 Planning in Our Selected Domain: Renal Health

Using computer applications in healthcare can improve the quality and effectiveness of healthcare services and reduce its cost. However, adoption of computerised information systems in healthcare lags behind the use of computers in most other sectors of the economy. The lives of many patients could be improved if they use computer technology to obtain information, make difficult decisions, and to contact experts and support groups.

As tele-health systems become one of the targeted areas for intensive development in the Multimedia Super Corridor (MSC) initiative, coupled with the Malaysian vision towards the rapid development and the eventual nation-wide deployment of such systems with the application of information technology (IT); this research project would be a step in that direction.

In this thesis, we study planning in the field of medicine in general and renal (kidney) health in particular. The complexities in monitoring kidney failure patients during dialysis have inspired this project. Most kidney failure patients need to undergo dialysis treatment three times a week. However, dialysis treatments have many side-effects such as fainting spells, loss of body calcium, low blood pressure, cramps, and heart problems. In order to avoid all of these, the patient's health records need to be monitored always. This task is quite difficult for medical staff to carry out in situations where there are not enough doctors at dialysis centers. Therefore, the need to improve medical services while reducing its cost has led us to propose a compact planning system called Dynamic Personalised (DP) Planner complete with the plan ontology or representation and the planning algorithms that come with it. The proposed planner is designed to be generic, hence making it applicable in any domain, e.g. business, education, and healthcare.

1.4 Research Problems

There are a number of problems that have been identified in the present state of affairs in planning systems. Primarily, most of the current plan representations or ontologies

are too fine grained (detailed). This means that the plan representations or onlotologies are not suited for all situations and for all levels. Therefore, we need to have a portable and intuitively easy representation that facilitates the storage and manipulation of generic plans.

The second problem is that most of the current planning systems are often static. This means planning is carried out once without taking into account changes that may take place as time passes. These plans also do not consider past events. Therefore, dynamic planning is required so that plans are updated as new situations arise. Dynamic planning systems are by no means non-existent. Dynamic planning systems that currently exist include Asgaard and O-Plan. However, these are more-suited for domains with large and complex but partly vague and incomplete knowledge. Therefore, we need to have a generic and dynamic planning system which is suited for obvious knowledge, i.e. data (human input) that are considered to be certain and complete. In the process, the issue of the distributed nature of planning resources is also considered.

1.5 Objectives

The first driving point in this thesis is to study various plan definitions and representations based on ontology. This is then followed by choosing or combining the best features from these plan representations for the DP Planner. A new plan representation that can facilitate the storage and manipulation of generic plans will be proposed.

The second driving point is to study various models of planning processes and, again, choose or combine the best amongst these for the purpose of the DP Planner. These will be utilized to develop a dynamic planning system that can update the plans once new conditions occur. An architecture will also be investigated in order to support the DP Planner in a distributed environment.

1.6 Contributions of this Thesis

This thesis contributes by taking a step forward in simplifying a plan definition based on an ontology which can adopt a XML representation, and personalising a plan dynamically.

More specifically, the contribution of this thesis can be measured along three dimensions:

1. **A simple hierarchical plan ontology with XML-based plan representation:** A hierarchical plan ontology has been designed through the adoption of a conceptual model called GOMS (Goals, Operators, Methods and Selections) for planning (Jones & Wray, 2006). In adhering to simplicity, the plan ontology has fewer elements and attributes that are necessary to represent a plan (results will be discussed in Chapter 5). The plan ontology is also directly implemented in XML.

2. **A Dynamic Personalised Planner:** A planning system based on constraints has been designed and developed with dynamic and personalised plan generation features. For this reason, the planning system has been named DP (Dynamic

Personalised) Planner. Furthermore, as an extension to the DP Planner, an architecture for the DP Planner has been provided in the grid computing environment. Grid computing is a type of distributed system which focus on large scale resource sharing, innovative applications, and also for some cases in high performance (Foster et al., 2001; Foster & Kesselman, 2004). In the case of DP Planner

3. Generally, in this thesis, we have looked into the following research issues in automated planning:
    1. classification of planners: domain-specific planners, domain-independent planners and domain-configurable planners (Nau, 2007),
    2. plan ontology,
    3. plan representation,
    4. planning algorithm,
    5. methods for planning engine development.

As a result, we have introduced the DP Planner which is classified as a domain-configurable planner. Domain-configurable planners are planning systems in which the planning engine is domain-independent but the input to the planner includes domain-specific knowledge to constrain or guide the planner's search. Hence, the planner searches only a small part of the search space. Consequently, the planning engine can be reconFigured to work in another problem domain by giving it a new domain description

(Nau, 2007). In contrast, classical planning requires complete knowledge about a deterministic, static, finite system with restricted goals and implicit time.

DP Planner was designed as a domain-configurable planner because:

1. it can be reconFigured easily for a new domain: one only needs to write a domain description but not the entire planner
2. it will have a medium level of performance in a given domain: It would be possible to program the domain description of the same problem solving techniques of a domain-specific planner into the domain-configurable planner.
3. it has a greater coverage across many domains: this is partly due to efficiency and partly due to expressive power
4. it has been proven to be best performer: they solved the most problems, solved them the fastest, usually found better solutions, and worked in many non-classical planning domains that were beyond the scope of the domain-independent planners (Nau, 2007).

1.7 Thesis Outline

This thesis is organised as follows:

Chapter 2 presents a literature review on computer-based or automated planning, and grid computing environment. Areas that would be covered in automated planning include a conceptual model of planning, plan ontology, and planning engine or planning process.

Chapter 3 outlines our research methodology. We present the research techniques and approaches used through four phases which are: (1) defining the plan ontology and plan representation, (2) defining a dynamic planning engine, (3) exploring the potential deployment of the DP Planner in a grid computing environment, and (4) defining criteria used for the evaluation of the DP Planner planning system. In this Chapter, we also provide some justifications as to the usage of the various approaches and techniques.

Chapter 4 presents the implementation of the DP Planner methodology. In the first phase, the plan ontology structure and a sample of the plan representation in XML format are presented. Then, in the second phase, details of the planning algorithm and the development of the planning engine are described. The third phase presents an architecture (with a sample scenario) of the DP Planner in a grid computing environment as well as the discussion on the suitable software and tools for implementation.

Chapter 5 described a number of case studies for the DP Planner. The case studies demonstrate the workings of the DP Planner with sample inputs and existing plans taken from the domain of renal health. As a result, the outputs of the case studies in the form of personalised plans were shown. In this Chapter, the details of the planning engine's processes, according to the planning algorithm, are also highlighted. To conclude this Chapter, a comparison is made with between DP Planner and other planning systems according to the evaluation criteria defined in Chapter 3.

Chapter 6 concludes the thesis with a re-visit to the contributions of this thesis and the identification of future research directions.

CHAPTER 2

LITERATURE REVIEW

"For your worldly affairs, construct your plans based on the assumption that you are going to live forever, and as for the work reserved for the Hereafter, construct your plans based on the assumption that you are going to die tomorrow."

[Umar Ibn Al-Khattab]

2.1 Introduction

Planning is a method to complete a particular task through finding a sequence of events. Classical planning mostly requires access and manipulation of large quantities of knowledge since plan creation needs the association of pieces of knowledge and partial plans into a solution process (Luger, 2002). Plans are needed in many different fields of human venture, and in some cases, the automatic generation of plans are desirable. Current trends show that automated planning research is moving away from the restrictions of classical planning (Nau, 2007).

Classical planning mostly starts with an empty plan. The plan develops without a plan ontology and is usually represented in a form that is similar to machine language. Thus, classic planning systems adopt a linear approach in their planning method. Classical planning is also deductive as it relies on the reasoning capabilities of description logics. However, with time, non-linear planning techniques have emerged.

In this Chapter, we present a review of various automated planning concepts that are related to our area of concern, i.e. plan ontology and plan representation, planning approaches and techniques, and planning in the grid computing environment. These concepts include a conceptual model of planning, types of planners, evolution of automated planning, as well as some current trends in automated planning, i.e. the methods and approaches used and how they have evolved over time based on certain selection criteria. We also discuss how plans can be resolved once execution failure happens.

2.2 Planning System based on Ontologies and Generic Planning

A popular approach for plan representation is via ontologies and plans are designed based on project specific ontologies and domain description languages.

2.2.1 O-Plan

Figure 2.1 shows the structure of the Plan Ontology proposed by Tate (Tate, 1994). This ontology contains:

- *Meta-Ontology* which explains the ontology itself and the assumptions behind the explanation.
- *Top-Level Ontology* contains the basic ontology or surface level of the ontology.
- *Library of Shared Ontological Elements* contains a collection of ontological elements that can be shared between the detailed ontological elements.
- *Detailed Ontology Sections* contain the specification of detailed ontological elements.

- *Encodings of the Ontology* are statements which describe the relationships between ontological entities using symbols.

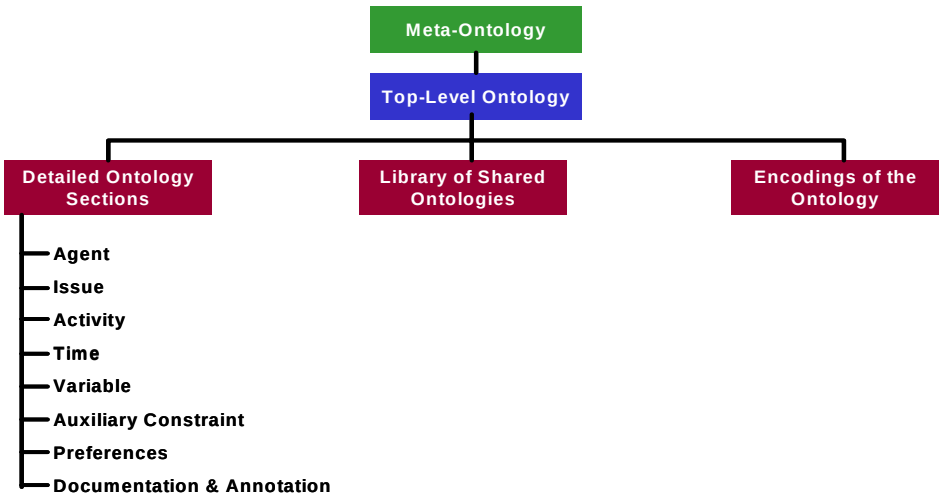

Figure 2.1: Plan Ontology Structure (Tate, 1994).

Complementing the Plan Ontology is the <I-N-OVA> model (Tate, 1995b) (I stands for Issues or Implied Constraints, N for Node Constraints, O for Ordering Constraints, V for Variables Constraints, and A for Auxiliary Constraints). It is an approach to represent and manipulate plans as a set of constraints. Figure 2.2 briefly shows the features in the <I-N-OVA> model.

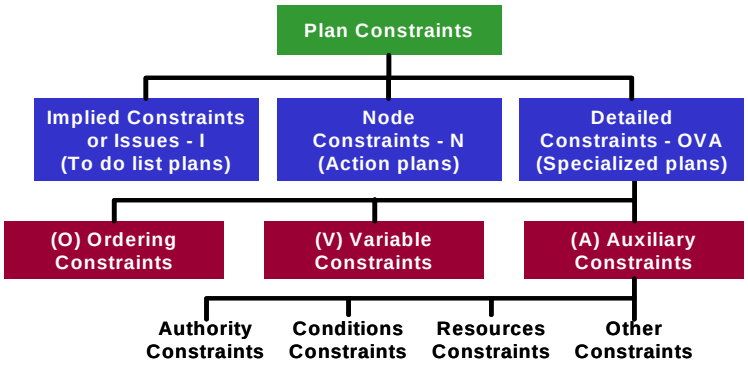

Figure 2.2: Features in the <I-N-OVA> model (Tate, 1995b).

The Issues, Ordering, Variables, and Auxiliary Constraints control the plans within that space which are valid. Ordering (temporal) and Variable Constraints are differentiated from all other auxiliary constraints since these acts as *cross-constraints*. These will be passed on to plan objects/variables and to time points or ranges in order to describe other constraints, e.g. resource constraints. This approach is applied in systems such as O-Plan, OPIS, DIPART and TOSCA (Tate, 1995b).

2.2.2 Asgaard

In the Asgaard project, they have developed a global ontology for guideline-application task called Asbru (Shahar, Miksch & Johnson, 1998). Asbru is a task-specific, intention-based and time-oriented language for representing skeletal plans (with plan schemata at different levels of detail, which captures the essence of the procedure but leaves enough room for execution-time flexibility in the achievement of particular goals) and it can be used to design specific plans. In other words, skeletal plans allow the reuse of existing plans.

In Asbru, a plan contains a name and a set of arguments. These arguments comprise of a time annotation and five knowledge roles:

- *Preferences* put constraints on the applicability of a plan and these guide the decisions in the plan selection process.
- *Intentions* are high-level goals which support tasks such as critiquing and re-planning.

- *Conditions* are temporal patterns which are taken at a specified frequency that leads to transitions between plan states.
- *Effects* describe the relationship between plan arguments and measurable parameter.
- *Plan layout* (or *plan body*) describes the order and frequency of the execution of sub-plans.

All plans and actions have a temporal dimension and the plan's execution is controlled by a number of conditions which are *filter*, *setup*, *suspend*, *reactivate*, *abort* and *complete*. Figure 2.3 shows the Asbru plan ontology structure.

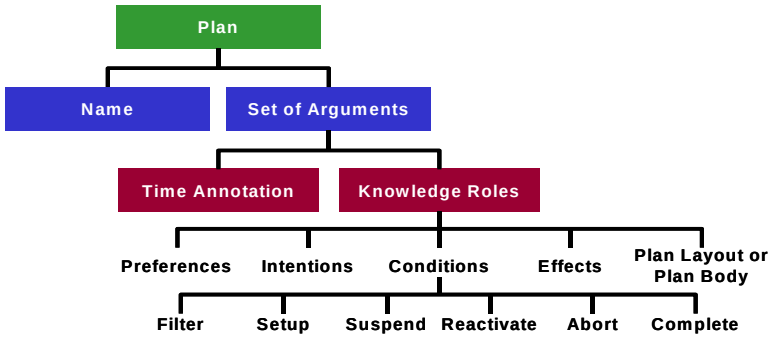

Figure 2.3: The Asbru plan ontology structure (Shahar, Miksch & Johnson, 1996).

Obviously, there are many generic planning approaches that use some form of hierarchical organization, thus giving rise to the popularity of ontologies and other taxonomical technologies. The hierarchical arrangement makes the plan ontology look simple even though it is complex.

2.2.3 PLANET

Another ontology to represent plans is PLANET. PLANET complements efforts on formalizing, organizing and unifying artificial intelligence-based planning algorithms. PLANET is developed for *knowledge modeling*, and can be a key instrument for *knowledge reuse* across planning applications and to simplify the integration of planning tools throughout *knowledge sharing*. Figure 2.4 shows an overview of the PLANET ontology.

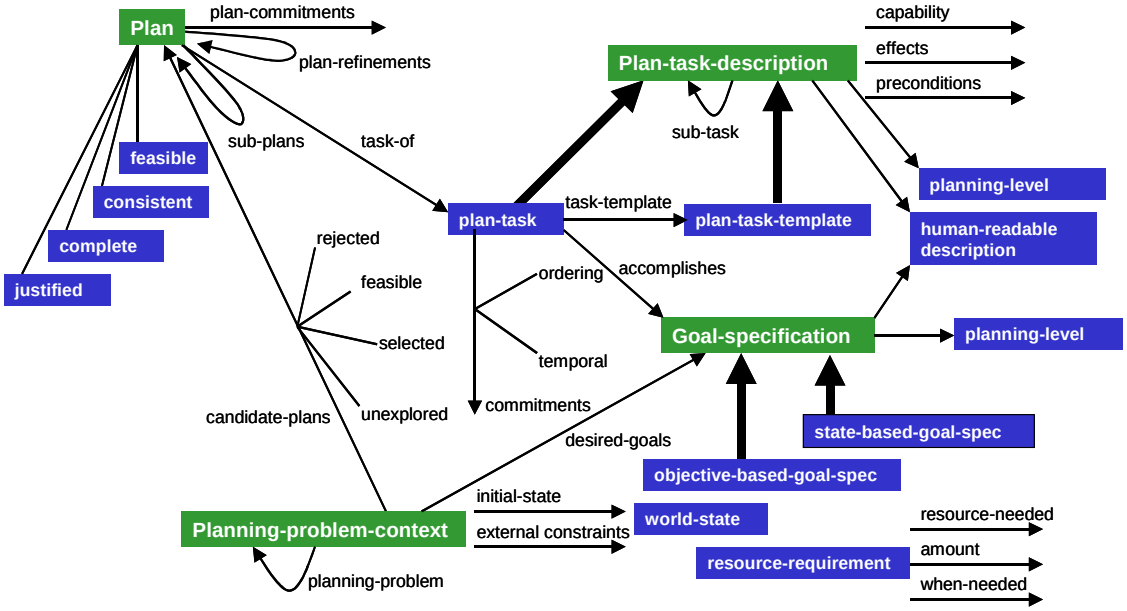

Figure 2.4: An overview of the PLANET ontology (Gil & Blythe, 2000) (arrows pointing into space represent relations whose ranges are not fixed in the ontology).

However, in our view this ontology is quite complex and may be difficult to adopt and implement.

2.2.4 Blocks World Planning

There are also research initiatives on generic planning (Lever & Richards, 1995) that are applied to the domains of Blocks World Planning and Flight Allocation. The Generic Planning Architecture contains elements such as *Basic Representation*, *Specifying Actions, Specifying Integrity Constraints, Resources, Labeling Strategies, Calling and Returning* and some *Miscellaneous Predicates.*

2.3 Plan Representation Language Evolution

One of the most important problems which need to be addressed in any planning system is that of plan representation. Classical plan representation is mostly written in the form of machine readable languages. However, later, most planning systems have adopted a plan ontology to allow knowledge sharing and to allow the reuse of the plan through formal and real-world semantics (Bruijn, 2003). The plan ontology makes it possible for plan representation and implementation to be carried out in many types of programming languages such as Lisp and Prolog. Presently, XML is gaining popularity as a plan representation language. XML, besides being a fairly easy formalism to work with, can be used to ensure that the plans adhere to the original plan structure and to facilitate the interchange of plan details from one platform/planning engine to another.

Plan representation languages usually follow the evolution of programming languages. Programming languages have evolved from binary machine code to powerful tools that create complex abstractions (Cook, 1999). The concept of abstraction is one of the keys to successful programming. Abstraction is required to allow the programmer to grasp

necessary concepts. It is helpful to show the topology or mapping of a language to the data structures and program modules that the language provides. Once we see the topology of early languages, we can better understand the problems and solutions. This influence the chosen of programming language type to form a plan representation.

Classic planning system such as STRIPS in 1970 had represented plans in the form of predicate calculus in which actions are represented as a set of preconditions, an add list and a delete list. The add list is a set of propositions that become true once the action is performed while the delete list is a set of propositions that become false once an action is performed (Eshner & Hartley, 1988). The popular planning system O-Plan, on the other hand, was written based on the classic planning system Nonlin. Thus, the O-Plan plan representation was in the form of Task Formalism. Task Formalism (TF) is a declarative language for expressing action schemata, for describing task requests and for representing the final plan. It allows time and resource constraints in the domain to be modeled (Tate, 1995a).

Another current and popular planning system called Asgaard had developed its own plan representation called the Asbru language. Asbru can be used to design specific plans as well as support the performance of different reasoning and executing tasks (Miksch, Shahar & Johnson, 1997). In its early development, Asbru (introduced in 1998) (Shahar, Miksch & Johnson, 1998) had been represented in a functional programming language such as Lisp. Later, Asbru had been redesigned using XML due to its flexibility and other advantages mentioned earlier in Section 2.2.

2.4 Planning System

Planning is an abstract, explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes. This deliberation aims at achieving as best as possible some prestated objectives. One motivation for automated planning is obvious: to design information processing tools that give access to affordable and efficient planning resources. Since there are various types of actions, there are also various forms of planning (Ghallab, Nau & Traverso, 2004). In this Section, we present several planning models that were explored. Of these, we focused more on O-Plan and Asgaard since we have found these approaches more suitable to solve the research problems of this thesis.

2.4.1 A Conceptual Model for Planning

Foundational software systems that tightly integrate some number of representations and processes are always taken as a basis for the development of human-level intelligent systems. This has been considered sufficient for the purpose of generating automated intelligent behavior. The design of these foundational software systems, which include both cognitive and agent architectures, have generally been based on some small set of theoretical principles (Jones & Wray, 2006). Here, we review two different popular conceptual models for intelligent system (like planning system): BDI (Beliefs, Desires, Intentions) and GOMS (Goals, Operators, Methods and Selection).

Beliefs, Desires, Intentions (BDI)

The BDI model or framework is based on Bratman's theory of human practical reasoning (Bratman, 1987). Currently, BDI is a famous logic-based methodology for building competent agents (Georgeff & Lansky, 1987; Rao & Georgeff, 1995; Wooldridge, 2000; Jones & Wray, 2006). The basic theory in BDI is that intelligent agents should be rational and sensible. "Intentions" refers to actions that arise from internal constructs. An intelligent agent can only make decisions about its intentions after it has at least some representation of its beliefs regarding its condition. This means, the agent must maintain a set of beliefs about what is true in the world. Thus, there might be many different situations that the agent may consider as desirable when given a particular set of beliefs. However, the agent can only act on some subsets of these desires to continue performing by selecting a subset and its intentions when the agent is given limited resources. In summary, based on BDI terminology, the whole set of applicable activities represent the agent's *desires*, while the set of currently selected actions that address some subset of those desires represent *intentions* (Jones & Wray, 2006).

Goals, Operators, Methods and Selections (GOMS)

GOMS is a methodology based on psychology and human computer interaction (Card, Moran & Newell, 1983; Jones & Wray, 2006). GOMS formalizes many details of high-level human reasoning and interaction. It is often used in knowledge-intensive agents to simulate human behaviour and not strictly as an agent framework. GOMS also has been used to represent the human knowledge necessary for performing many tasks but has

not been used to develop large-scale systems. Other than that, the improvements in GOMS in terms of efficiency allow executable cognitive models to compete with AI architectures in certain application areas (John, Vera & Newell, 1994). GOMS also identifies the representation and process regularities for knowledge-intensive agents that will encode the type of knowledge such as hierarchical task decompositions, invoking subgoals or primitive actions to complete the goal and selection rules that provide conditional logic for choosing between plans (Jones & Wray, 2006).

A key feature of GOMS is its support for hierarchical task decomposition. GOMS starts with a top-level task goal plus a number of methods, or plans, for achieving various types of goals and sub-goals. Each goal's plan specifies a series of actions (called operators by the GOMS community). These actions will invoke sub-goals or primitive actions to complete the goal. Thus, the selection rules provide conditional logic for choosing between plans based on the agent's current set of beliefs (Jones & Wray, 2006).

2.4.2 Planning System Evolution: Linear to Non-Linear

The first planning systems were essentially linear planners, i.e. those that work on one goal until it is completely solved before moving on to the next goal. Later on, these were complemented with non-linear planners. In contrast to linear planners, non-linear planners use a set of goals instead of one goal. Non-linear planners include all possible sub-goal orderings in the search space which handles goal interactions by the process of interleaving (Simmons, 2001).

Classical Planning System: STRIPS and Prodigy

The classical planning system STRIPS (introduced around 1972) is an example of a linear planner. The STRIPS's Planner language was written in predicate calculus form. Subsequently, the Prodigy plan representation or language is similar to that of STRIPS. Prodigy is an architecture that integrates planning with multiple learning mechanisms. Prodigy's language for describing operators is based on the STRIPS domain language, and extended to express disjunctive and negated preconditions, universal and existential quantification, and conditional effects. Prodigy was a planning system which generates a plan from an empty plan. The first version of Prodigy was developed as a linear planner. However, Prodigy version 4.0 has transformed into a completely nonlinear planner.

O-Plan

Another popular planning system is O-Plan (non-linear). It was designed to be a domain-independent, general planning and control framework with the ability to embed

detailed knowledge of the domain. O-Plan is a planning system based on agents. It employs three agents: Task Assignment, Planner, and Execution System. The O-Plan's agents have a representation of the plan in the form of a plan state consisting of plan agenda, plan entities and plan constraints (see Figure 2.5). So basically, the plan state is a complete description of a plan at some level of abstraction. The plan state also holds the current errors in the plan which could relate to abstract actions. However, these actions must be expanded before the plan is valid for execution. Other than that, the plan state also holds unsatisfied conditions, unresolved interactions, over commitments of resource and time constraint faults.

The Planner Agent employs a task formalism that is suited to the representation of a plan state and hence acts as a basis for communication between the Task Assignment and Execution System agents. The actual plan state inside the Task Assignment and Execution System agents is likely to differ from that within the Planner agent. The Planner agent's plan state holds information about decisions made during planning and information about decisions which are still to be made in the form of an agenda (see Figure 2.5), (Tate, 1995a).

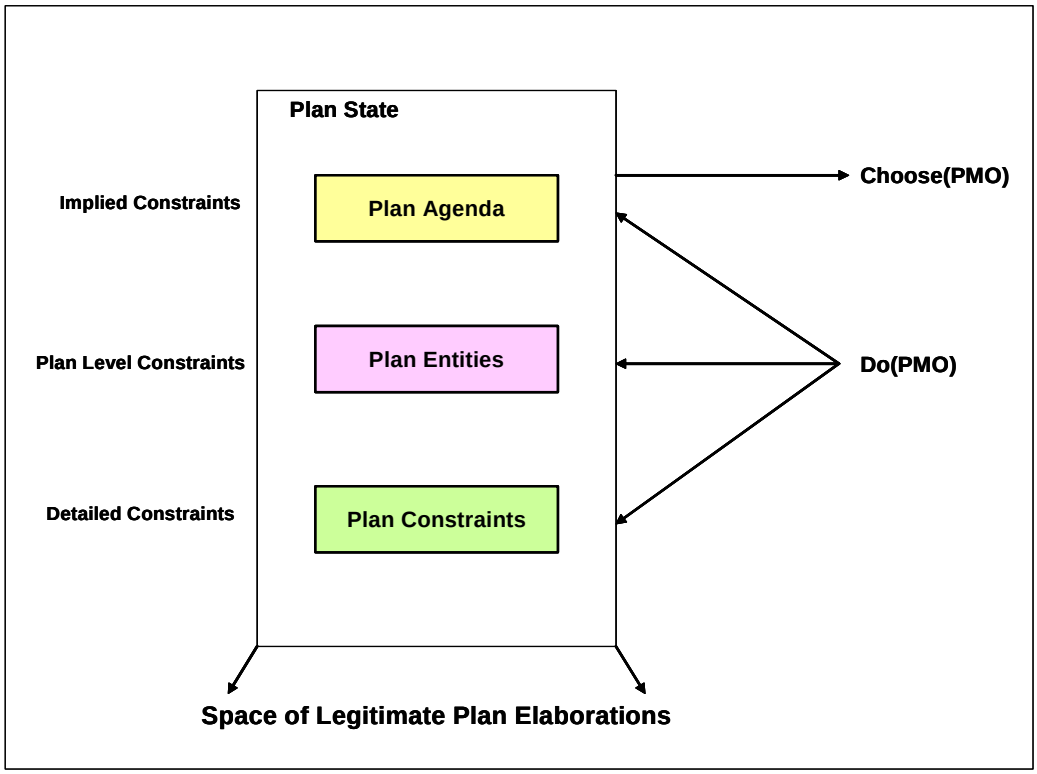

Figure 2.5: A Framework of Components in a Planning/Scheduling System (in *Agenda* form) (Tate, 1994).

Figure 2.6 shows the O-Plan Planner agent's components (Tate, 1994). The TOME (Table of Multiple Effects) and GOST (Goal Structure Table) Managers play a main role of repairing plans to account for execution failures and changes in the execution situation (Drabble at al., 1997). The TOME and GOST have been implemented as tables in the O-Plan plan representation to be used by the plan repair algorithms to determine the consequences of failures. Here, failure refers to execution failures and changes in the execution situation. An execution failure occurs when one or more of the expected effects at a node-end fail to be asserted. Each effect is recorded in the TOME and when an action depends on an effect asserted earlier, that is recorded in the GOST.

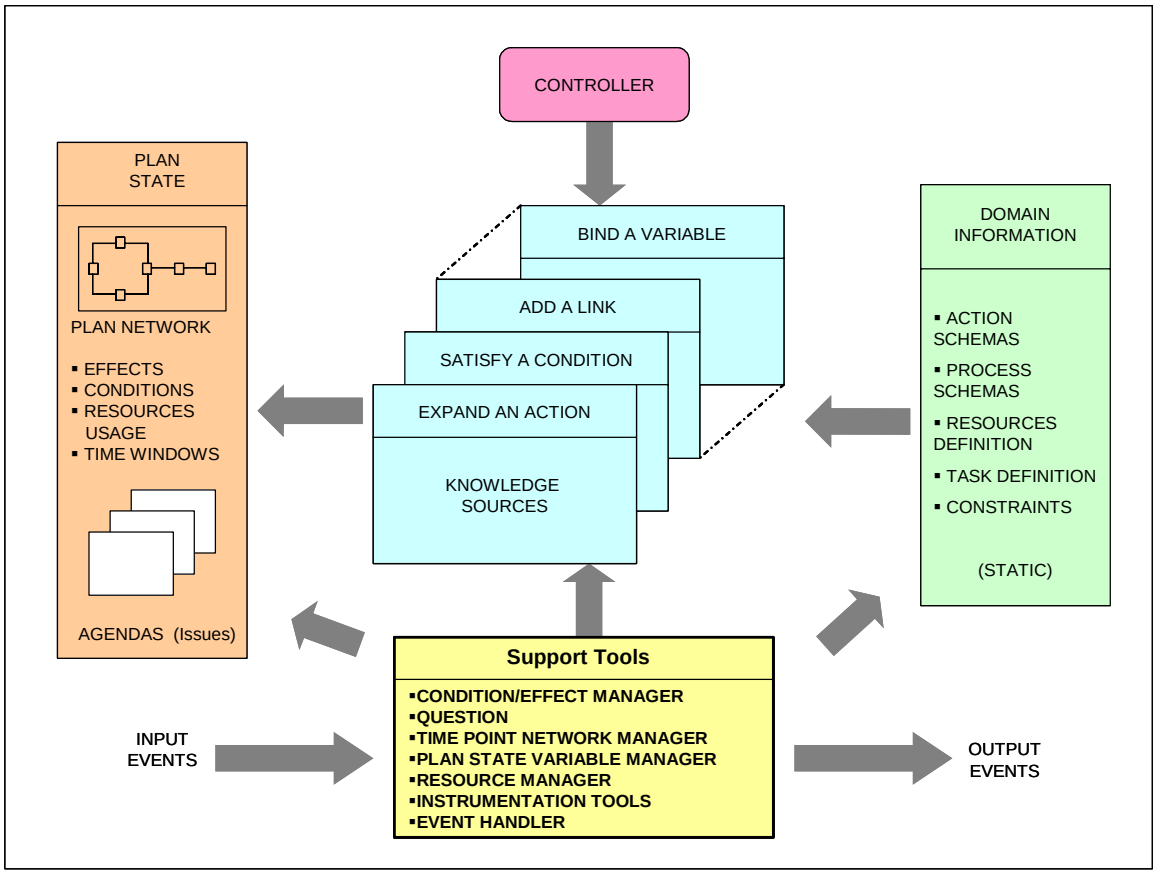

Figure 2.6: O-Plan Planner Agent Components (Tate, 1994).

When an execution failure occurs, the TOME will be updated and its related GOST entries will be found. If the related GOST entries are found, then the appropriate repair is carried out using the knowledge sources. The knowledge sources are responsible for determining the consequences of unexpected events or of actions that do not execute as intended, for deciding what action to take when a problem is detected, and for making repairs to the effected plan (Drabble at al., 1997). This is because each knowledge source in O-Plan encodes a piece of planning knowledge such as how to expand an action, bind a variable, add a link to satisfy a condition, check a resource and many others as shown in Figure 2.6 (Tate, 1994).